INTERNATIONAL STANDARD

ISO/IEC 14776-232

First edition
2001-11

Information technology – Small computer system interface (SCSI) – Part 232: Serial Bus Protocol 2 (SBP-2)

Reference number
ISO/IEC 14776-232:2001(E)

INTERNATIONAL STANDARD

ISO/IEC 14776-232

First edition
2001-11

Information technology – Small computer system interface (SCSI) – Part 232: Serial Bus Protocol 2 (SBP-2)

PRICE CODE X

For price, see current catalogue

CONTENTS

INFORMATION TECHNOLOGY – SMALL COMPUTER SYSTEM INTERFACE (SCSI) –

Part 232: Serial bus protocol 2 (SBP-2)

FOREWORD

1) ISO (International Organization for Standardization) and IEC (International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work.

2) In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC1. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

3) Attention is drawn to the possibility that some of the elements of this International Standard may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

International Standard ISO/IEC 14776-232 was prepared by subcommittee 25: Interconnection of information technology equipment, of ISO/IEC joint technical committee 1: Information technology.

This publication has been drafted in accordance with the ISO/IEC Directives, Part 3.

Annexes A, B and C form an integral part of this International Standard.
Annexes D, E and F are for information only.

INFORMATION TECHNOLOGY – SMALL COMPUTER SYSTEM INTERFACE (SCSI) –

Part 232: Serial bus protocol 2 (SBP-2)

1 Scope and object

1.1 Scope

This part of ISO/IEC 14776 defines a protocol for the transport of commands and data over High Performance Serial Bus. The transport protocol, Serial Bus Protocol 2 or SBP-2, requires implementations to conform to the requirements of this standard as well as to ISO/IEC 13213:1994 and permits the exchange of commands, data and status between initiators and targets connected to Serial Bus.

1.2 Object

Original development work for Serial Bus Protocol (SBP) was initiated out of a desire to adapt SCSI capabilities and facilities to a particular serial environment IEEE 1394. Serial interconnects offer a migration path for SCSI into the future because they may be better suited to cost reduction and speed increases than the parallel interconnects first utilized by SCSI.

As development of the standard progressed, it became evident that the solutions provided by SBP-2 were of general applicability to large classes of Serial Bus peripheral devices. With this in mind, the development work was redirected to provide mechanisms for the delivery of commands, date and status independent of the command set or device class of the peripheral. SBP-2 provides a generic framework that may be referenced by other documents or standards that address the unique requirements of a particular class of devices. The enhanced goals set for the design of SBP-2 are ranked below:

– the protocol should permit the encapsulation of commands, data and status from a diversity of command sets, legacy as well as future, in order to preserve the investment in an existing application and operating system software base;

– the protocol should allow the initiator to dynamically add tasks to this set while the target is active in execution of earlier tasks. The addition of new tasks should not interfere with the target's processing of tasks currently active;

– although the protocol should enable varying levels of features and performance in target implementations, strong focus should be kept on a minimal set deemed adequate for entry-level environments;

– within the constraints posed by the preceding goal, the hardware and software design of the initiator should not be unduly affected by variations in target capabilities;

– in order to promote the scalability of aggregate system performance, the protocol should distribute the DMA context from the initiator adapter to the target devices.

Although SBP-2 has been designed for Serial Bus as currently specified by IEEE 1394, it is believed that it will be appropriate for use with future extensions to Serial Bus as they are standardized.

2 Normative references

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO/IEC 9899:1999, *Programming Languages – C*

ISO/IEC 13213:1994, *Information technology – Microprocessor systems – Control and Status Register (CSR) Architecture for Microcomputer Buses*

ANSI/IEEE 1394:1995, *IEEE Standard for High Perfomance Serial Bus*

IEEE P1394a, *Draft Standard for High Perfomance Serial Bus (Supplement)*[1)]

BSR X3 PN 1157-D, *Information technology – SCSI Architecture Model 2 (SAM-2)*[2)]

BSR NCITS PN 1236-D, *Information technology – SCSI Primary Commands 2 (SPC-2)*[2)]

1) Under development. Available from the Institute of Electrical and Electronics Engineers, 445 Hoes Lane, P.O. Box 1331, Piscataway, NJ 08855-1331

2) Under development. Available from the National Committee for Information Technology Standards, 1250 Eye Street, NW, Suite 200, Washington, DC 20005-3922

3 Definitions and notation

3.1 Definitions

3.1.1 Conformance

Several keywords are used to differentiate levels of requirements and optionality, as follows:

3.1.1.1 expected: A keyword used to describe the behavior of the hardware or software in the design models assumed by this standard. Other hardware and software design models may also be implemented.

3.1.1.2 ignored: A keyword that describes bits, bytes, quadlets, octlets or fields whose values are not checked by the recipient.

3.1.1.3 may: A keyword that indicates flexibility of choice with no implied preference.

3.1.1.4 reserved: A keyword used to describe objects—bits, bytes, quadlets, octlets and fields—or the code values assigned to these objects in cases where either the object or the code value is set aside for future standardization. Usage and interpretation may be specified by future extensions to this or other standards. A reserved object shall be zeroed or, upon development of a future standard, set to a value specified by such a standard. The recipient of a reserved object shall not check its value. The recipient of an object defined by this standard other than reserved shall check its value and reject reserved code values.

3.1.1.5 shall: A keyword that indicates a mandatory requirement. Designers are required to implement all such mandatory requirements to assure interoperability with other products conforming to this standard.

3.1.1.6 should: A keyword that denotes flexibility of choice with a strongly preferred alternative. Equivalent to the phrase "is recommended."

3.1.2 Glossary

The following terms are used in this standard:

3.1.2.1 byte: Eight bits of data.

3.1.2.2 command block: Space reserved within an ORB to describe a command intended for a logical unit that controls device functions or the transfer of data to or from device medium. The format and meaning of command blocks are outside the scope of SBP-2 and are command set- or device-dependent.

3.1.2.3 device server: A component of a logical unit responsible to execute tasks initiated by command blocks that specify data transfer or other device operations.

3.1.2.4 initial node space: The 256 terabytes of Serial Bus address space that may be available to each node. Addresses within initial node space are 48 bits and are based at zero. The initial node space includes initial memory space, private space, initial register space and initial units space. See either ISO/IEC 13213 or ANSI/IEEE 1394 for more information on address spaces.

3.1.2.5 initial register space: A two kilobyte portion of initial node space with a base address of FFFF F000 0000_{16}. Core registers defined by ISO/IEC 13213 are located within initial register space as are Serial Bus-dependent registers defined by ANSI/IEEE 1394.

3.1.2.6 initial units space: A portion of initial node space with a base address of FFFF F000 0800_{16}. This places initial units space adjacent to and above initial register space. The CSRs and other facilities defined by unit architectures are expected to lie within this space.

3.1.2.7 initiator: A node that originates device service or management requests and signals these requests to a target for processing.

3.1.2.8 kilobyte: A quantity of data equal to 2^{10} bytes.

3.1.2.9 logical unit: The part of the unit architecture that is an instance of a device model, e.g., disk, CD-ROM or printer. Targets implement one or more logical units; the device type of the logical units may differ.

3.1.2.10 login: The process by which an initiator obtains access to a set of target fetch agents. The target fetch agents and their control and status registers provide a mechanism for an initiator to signal ORBs to the target.

3.1.2.11 login ID: A value assigned by the target during the login process. The login ID establishes a relationship between an initiator and a task set. The login ID is used to identify subsequent requests from an initiator; in some cases the login ID is not present in the operation request block and its value is implicit.

3.1.2.12 node: An addressable device attached to Serial Bus.

3.1.2.13 node ID: The 16-bit node identifier defined by ANSI/IEEE 1394 that is composed of a bus ID portion and a physical ID portion. The physical ID is uniquely assigned as a consequence of Serial Bus initialization.

3.1.2.14 octlet: Eight bytes, or 64 bits, of data.

3.1.2.15 operation request block: A data structure fetched from system memory by a target in order to execute the command encapsulated within it.

3.1.2.16 quadlet: Four bytes, or 32 bits, of data.

3.1.2.17 receive: When any form of this verb is used in the context of Serial Bus primary packets, it indicates that the packet is made available to the transaction or application layers, *i.e.*, layers above the link layer. Neither a packet repeated by the PHY nor a packet examined by the link is "received" by the node unless the preceding is also true.

3.1.2.18 register: A term used to describe quadlet aligned addresses that may be read or written by Serial Bus transactions. In the context of this standard, the use of the term register does not imply a specific hardware implementation. For example, in the case of split transactions that permit sufficient time between the request and response subactions, the behavior of the register may be emulated by a processor.

3.1.2.19 request subaction: A packet transmitted by a node (the requester) that communicates a transaction code and optional data to another node (the responder) or nodes.

3.1.2.20 response subaction: A packet transmitted by a node (the responder) that communicates a response code and optional data to another node (the requester). A response subaction may consist of either an acknowledge packet or a response packet.

3.1.2.21 split transaction: A transaction that consists of a request subaction followed by a separate response subaction. Subactions are considered separate if ownership of the bus is relinquished between the two.

3.1.2.22 status block: A data structure that may be written to system memory by a target when an operation request block has been completed.

3.1.2.23 store: When any form of this verb is used in the context of data transferred by the target to the system memory of either an initiator or other device, it indicates both the use of Serial Bus write request subaction(s), quadlet or block, to place the data in system memory and the corresponding response subaction(s) that complete the write(s).

3.1.2.24 system memory: The portions of any node's memory that are directly addressable by a Serial Bus address and that accepts, at a minimum, quadlet read and write access. Computers are the most common example of nodes that might make system memory addressable from Serial Bus, but any node, including those usually thought of as peripheral devices, may have system memory.

3.1.2.25 target: A node that receives device service or management requests from an initiator. In the case of device service requests, the commands are directed to one of the target's logical units to be executed. Management requests are serviced by the target. A CSR Architecture unit is synonymous with a target.

3.1.2.26 task: A task is an organizing concept that represents the work to be done by a target to carry out a command encapsulated by an ORB. In order to perform a task, a target maintains context information for the task, which includes (but is not limited to) the command, parameters such as data transfer addresses and lengths, completion status and ordering relationships to other tasks. A task has a lifetime, which commences when the task is entered into the target's task set, proceeds through a period of execution by the target and finishes either when completion status is stored at the initiator or when completion may be deduced from other information. While a task is active, it makes use of both target resources and initiator resources.

3.1.2.27 task set: A group of tasks available for execution by a logical unit of a target. This standard specifies some dependencies between individual tasks within the task set but there may be others not specified by this standard.

3.1.2.28 terabyte: A quantity of data equal to 2^{40} bytes.

3.1.2.29 transaction: A Serial Bus request subaction and the corresponding response subaction. The request subaction transmits a transaction code (such as quadlet read, block write or lock); some request subactions include data as well as transaction codes. The response subaction is null for transactions with broadcast destination addresses or broadcast transaction codes; otherwise it returns completion status and possibly data.

3.1.2.30 unit: A component of a Serial Bus node that provides processing, memory, I/O or some other functionality. Once the node is initialized, the unit provides a CSR interface that is typically accessed by device driver software at an initiator. A node may have multiple units, which normally operate independently of each other. Within this standard, a unit is equivalent to a target.

3.1.2.31 unit architecture: The specification of the interface to and the services provided by a unit implemented within a Serial Bus node. This standard is a unit architecture for SBP-2 targets.

3.1.2.32 unit attention: A state that a logical unit maintains while it has unsolicited status information to report to one or more logged-in initiators. A unit attention condition shall be created as described elsewhere in this standard or in the applicable command-set- and device-dependent documents. A unit attention condition shall persist for a logged-in initiator until a) unsolicited status that reports the unit attention

condition is successfully stored at the initiator or b) the initiator's login becomes invalid or is released. Logical units may queue unit attention conditions; after the first unit attention condition is cleared, another unit attention condition may exist.

3.1.2.33 working set: The part of a task set that has been fetched from the initiator by the target and is available to the target in its local storage.

3.1.3 Abbreviations

The following abbreviations are used in this standard:

CSR	Control and status register
CRC	Cyclical redundancy checksum
EUI-64	Extended Unique Identifier, 64-bits
LUN	Logical unit number
ORB	Operation request block
SAM-2	SCSI Architecture Model 2
SBP-2	Serial Bus Protocol 2 (this standard itself)
SPC-2	SCSI Primary Commands 2

3.2 Notation

The following conventions should be understood by the reader in order to comprehend this standard.

3.2.1 Numeric values

Decimal, hexadecimal and, occasionally, binary numbers are used within this standard. By editorial convention, decimal numbers are most frequently used to represent quantities or counts. Addresses are uniformly represented by hexadecimal numbers. Hexadecimal numbers are also used when the value represented has an underlying structure that is more apparent in a hexadecimal format than in a decimal format. Binary numbers are used infrequently and generally limited to the representation of bit patterns within a field.

Decimal numbers are represented by Arabic numerals without subscripts or by their English names. Hexadecimal numbers are represented by digits from the character set 0 – 9 and A – F followed by the subscript 16. Binary numbers are represented by digits from the character set 0 and 1 followed by the subscript 2. When the subscript is unnecessary to disambiguate the base of the number it may be omitted. For the sake of legibility, binary and hexadecimal numbers are separated into groups of four digits separated by spaces.

As an example, 42, $2A_{16}$ and $0010\ 1010_2$ all represent the same numeric value.

3.2.2 Bit, byte and quadlet ordering

SBP-2 is defined to use the facilities of Serial Bus, ANSI/IEEE 1394, and therefore uses the ordering conventions of Serial Bus in the representation of data structures. In order to promote interoperability with memory buses that may have different ordering conventions, this standard defines the order and signifi-

cance of bits within bytes, bytes within quadlets and quadlets within octlets in terms of their relative position and not their physically addressed position.

Within a byte, the most significant bit, *msb*, is that which is transmitted first and the least significant bit, *lsb*, is that which is transmitted last on Serial Bus, as illustrated below. The significance of the interior bits uniformly decreases in progression from *msb* to *lsb,* see Figure 1.

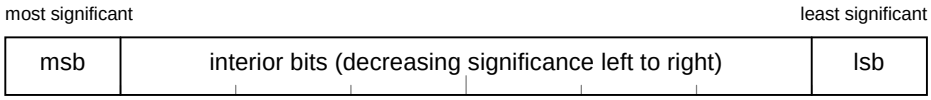

Figure 1 – Bit ordering within a byte

Within a quadlet, the most significant byte is that which is transmitted first and the least significant byte is that which is transmitted last on Serial Bus, see Figure 2.

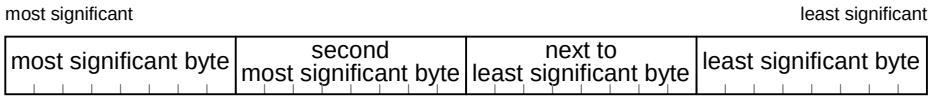

Figure 2 – Byte ordering within a quadlet

Within an octlet, which is frequently used to contain 64-bit Serial Bus addresses, the most significant quadlet is that which is transmitted first and the least significant quadlet is that which is transmitted last on Serial Bus, see Figure 3.

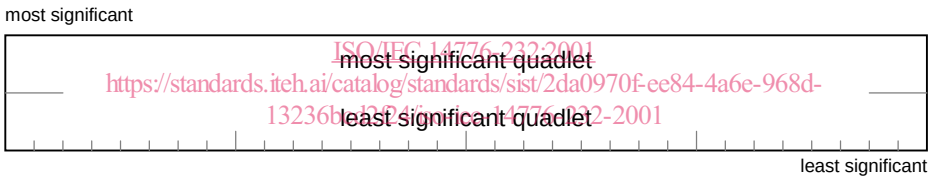

Figure 3 – Quadlet ordering within an octlet

When block transfers take place that are not quadlet aligned or not an integral number of quadlets, no assumptions can be made about the ordering (significance within a quadlet) of bytes at the unaligned beginning or fractional quadlet end of such a block transfer, unless an application has knowledge (outside of the scope of this standard) of the ordering conventions of the other bus.

3.2.3 Register specifications

This standard defines the format and function of control and status registers, CSRs. Some of these registers are read-only, some are both readable and writable and some generate special side effects subsequent to a write.

In order to define CSRs, their bit fields, their initial values and the effects of read, write or other transactions, the format illustrated by Figure 4 is used.