Name: *SMDIPROT*

Description: *Documentation of SCSI Musical Data Interchange Protocol*

Document Version: *0.03*

Contents

SCSI Musical Data Interchange Protocol (SMDI)

Overview

SMDI has been designed primarily for the fast transfer of digitally-sampled audio information between SCSI-equipped devices using a non-device specific format. The message exchange system used to perform such transfers is based loosely upon the MIDI Sample Dump Standard (SDS), but is structured to take advantage of the much higher data throughput capabilities of SCSI as compared to MIDI. The sample transfer protocol goes beyond the limitations which MIDI SDS imposed, while remedying some of its specific deficiencies.

SMDI is not intended as a general-purpose information interchange platform. However, ample room is available for the definition of new messages not directly associated with the transfer of sample files. In addition to the actual sample transfer protocol, SMDI provides for the transmission of arbitrary MIDI messages via SMDI, and sets aside certain message code ranges for use with manufacturer- or device-specific messages. This allows SMDI to fully support applications which deal with other types of data in addition to generic sample files (e.g., sample editor/librarians which offer device-specific support) without resorting to the use of a simultaneous MIDI connection. It also provides a convenient path for the migration of existing MIDI data-transfer functions to SMDI with minimal modification, as well as a means whereby a single application can transparently use either MIDI or SMDI as the communications interface.

It may be worthwhile to note two more things SMDI is not: 1) an attempt to find a successor to MIDI - SMDI, and SCSI in general, while far superior to MIDI as a medium for the transfer of large amounts of data, are poorly suited to the type of real-time event communication for which MIDI is commonly used, and 2) an attempt to create an audio or MIDI network - SMDI addresses the data-transfer aspects of digital audio and MIDI devices, and makes no attempt to address any of their other aspects.

SCSI Background

SMDI employs a limited set of SCSI commands available to devices which conform to the standard SCSI definition for a processor device. Processor devices communicate by transferring data packets using the SCSI SEND and RECEIVE commands, which are defined only for the SCSI processor device type. The SEND command is used by the initiator to send a packet of information to the target, while the RECEIVE command is used by the initiator to obtain a data packet from the target (the data packet is that which is transferred during the data-out phase of the SEND command or the data-in phase of the RECEIVE command). Transfers using these commands are sized to arbitrary byte lengths, rather than in blocks as per the more-familiar READ and WRITE commands defined for direct-access devices (e.g., disk drives). Data packet content is not defined by the SCSI spec, but is established by consensus among the processor devices involved. This makes the SEND and RECEIVE commands ideal as a basis for SMDI, which operates by transmitting SMDI messages as SEND/RECEIVE data packets. Note that SMDI messages are completely unrelated to SCSI messages (i.e., message-in and message-out), which are not utilized by SMDI.

The specification for a SCSI processor device includes a small number of “mandatory” SCSI commands in addition to SEND and RECEIVE. Notable among these are TEST UNIT READY, INQUIRY and REQUEST SENSE, all of which are utilized by SMDI devices at certain times. The INQUIRY command is the one whereby target device identity information is obtained, most significantly the SCSI device type, which should be 03h for SMDI slave devices, indicating that they are SCSI processor devices which implement the SEND and RECEIVE commands. A SMDI master should not issue these commands to any target device unless it has obtained this device type value from the target using INQUIRY. SMDI provides additional means for determining whether or not a SCSI processor device is also SMDI-capable. The uses of TEST UNIT READY and REQUEST SENSE are detailed in later sections. As with any SCSI target mode device, SMDI slave devices must be able to smoothly intercept and reject unrecognized SCSI commands without causing SCSI bus hang-up.

Because the computers typically used in a music environment usually do not readily support SCSI target mode operation, SMDI places the entire burden of conducting communications in both directions upon the SMDI master device. The master device always assumes the role of SCSI initiator, regardless of the direction of message transfer. SMDI slave devices are passive - in order for a SMDI slave device to transfer a message to a master, the master must issue a RECEIVE command to the slave, which responds as a SCSI target.

It should be mentioned that there is no prohibition against a single device being able to assume both SMDI master and SMDI slave roles at different times, provided that it maintains any given role for the duration of any SMDI procedure in which it becomes engaged.

General Principles of SMDI Messages and Transactions

This section contains a significant amount of detail pertaining to SMDI at the level of an individual message and its component parts, as well as some discussion of the rationale behind the design of the message format, and of conditions which are considered as errors. The reader may find it helpful at first to read quickly through this section (and the following one, which focuses upon message error handling) and proceed to the discussion of SMDI procedures which appears in a later section in order to obtain a clearer and more high-level understanding of the way in which SMDI messages normally work in the context of procedures, before attempting to fully absorb the low-level details presented here.

SMDI messages consist of strings of bytes which are transferred during the data-out phase of a SEND command or the data-in phase of a RECEIVE command, and which conform to the SMDI message format. SMDI procedures (such as those for transferring sample files), generally consist entirely of sequenced exchanges of such messages. SCSI commands other than SEND and RECEIVE are not usually involved in SMDI procedures.

The SEND and RECEIVE commands, in and of themselves, do not convey any SMDI information and should be viewed as merely the vehicle of SMDI messages. These commands are quite generic, in essence specifying only the length of the packet which an initiator is about to send or the maximum data packet length which an initiator is prepared to receive. (In SCSI terminology, this maximum length is referred to as Allocation Length. A SCSI target device can legally return any number of bytes up to the specified Allocation Length when responding to a RECEIVE command or to any other command which specifies an Allocation Length without violating SCSI rules, although it is generally expected to return all of the bytes it has, and no more, when it is not limited by the Allocation Length.) A SMDI master device therefore cannot use the RECEIVE command alone to request any specific message from a slave device, nor can it know in advance the type or precise length of the message with which the SMDI slave will respond to the RECEIVE command. Likewise, although SMDI specifies prescribed sequences for multi-message procedures, a SMDI slave device cannot make any assumptions about the next message which will be sent to it by a SMDI master device, especially if it is not currently engaged in such a procedure. Finally, any SMDI device is likely to be called upon to engage in SCSI transactions with non-SMDI devices with which it will inevitably find itself sharing the SCSI bus, and must be able to respond appropriately.

For these reasons, there are three central principles which apply to all SMDI activity:

1. All SMDI messages fully identify themselves via their message content.
2. All SMDI transactions consist of a SEND followed by a RECEIVE.
3. SMDI messages must always be transmitted in their entirety.

The full meaning of each of these principles, and their implications, are elaborated upon in the following discussion.

All SMDI messages fully identify themselves via their message content. Every SMDI message begins with a SMDI message header which is followed by additional bytes as appropriate to the particular message. Some SMDI messages consist only of a header. The SMDI message header is always 11 bytes long and consists of three distinct parts, presented here in order of appearance:

1. The SMDI message tag is a fixed four-byte string (“SMDI”) serving as an identifying tag for all

SMDI messages.

2. The Message Code identifies the type of message. It consists of a two-byte Message ID code, followed by a two-byte Message Sub-ID code.
3. The Additional Message Length is a three-byte field which indicates the number of bytes following the header in the current message.

The SMDI message tag is the primary means whereby a SMDI device can distinguish between SEND/RECEIVE data packets which are SMDI messages and those which are not. SEND/RECEIVE data packets not beginning with this tag are not to be recognized as SMDI messages, and should generally be taken as an indication that the device from which the data packet originated is not a SMDI device.

The message ID code is the primary identification of a message type. The message sub-ID code is normally zero unless the particular message type has defined variations which do not warrant a completely different ID. The message sub-ID may also be used to indicate revised/extended versions of a particular message.

The main purpose of the Additional Message Length parameter is to provide a SMDI master device with a specific length value for a message being received from a slave, to assist it in correctly completing the SCSI RECEIVE command. Without this parameter, the SMDI master has no direct indication of the actual length of an incoming message. Although most SMDI messages have a fixed length which is implied by the Message Code, there are some messages whose length may vary. More importantly, the inclusion of the Additional Message Length parameter permits a master device to correctly complete a RECEIVE command in which an unknown or unsupported message is being returned, thereby making it unnecessary for the master to have specific knowledge of each and every SMDI Message Code, and allowing compatibility with unanticipated future extensions to the Message Code list. An additional function of this parameter is to enable a SMDI master to detect the truncation of an incoming message owing to an inadequate Allocation Length (more on this in the next section).

As used in a message sent to a slave, the Additional Message Length parameter is somewhat redundant, since it should always be equal to the SEND command Transfer Length minus eleven (the size of the SMDI message header). As such, however, it serves as a further message integrity indicator. Slave devices which detect a conflict between the two numbers should always defer to the SCSI Transfer Length value in order to complete the SEND command. Master devices should also perform such an integrity check when receiving known messages, especially those which have a fixed length, and should consider conflicts as cause for termination of a SMDI procedure.

All SMDI transactions consist of a SEND followed by a RECEIVE. A SMDI master issues a RECEIVE command in order to obtain the slave's response to the most recent SEND command (referred to as a "reply"). The message which a SMDI slave presents during a RECEIVE command is always its reply to the message sent via the most recent SEND command. This alternation must be adhered to strictly, with only rare exceptions. Deviations, which include spurious RECEIVE commands not preceded by a SEND, as well as SEND commands issued to a slave device which is still holding the reply to a previous SEND command, are flagged as SCSI errors. The slave device thus enforces the SEND/RECEIVE alternation by refusing to accept SEND and RECEIVE commands which are issued out of sequence.

SMDI messages must always be transmitted in their entirety. In order to successfully receive a slave message, a SMDI master must issue its RECEIVE command with an Allocation Length adequate to fully

receive the message which the slave wishes to return. Although the master cannot predict with absolute certainty the message which it will receive, the number of appropriate slave replies to any given message from a master device is usually very small - the master should set its Allocation Length to accommodate the largest of the anticipated messages.

The slave device is always required to return its entire message if the presented Allocation Length is large enough to permit this. However, the slave device is not permitted to exceed the Allocation Length limitation set by the RECEIVE command. If presented with an inadequate Allocation Length, the slave must retain its reply (message) in anticipation of a subsequent attempt by the master to correct the problem and receive the complete message. In contrast to the usual handling of Allocation Length in commands such as INQUIRY and REQUEST SENSE, where a target is expected to return bytes up to the Allocation Length limit if this is smaller than the total number of bytes available, a SMDI slave in this situation is expected to return only a SMDI message header if the Allocation Length permits, or no data at all if the Allocation Length is too small to permit the complete transfer of a header. This situation is discussed in detail in the next section.

As well-behaved SCSI target devices, SMDI slave devices are required to correctly complete any SEND command to which they respond according to standard SCSI rules. The SEND command Transfer Length parameter indicates precisely the number of bytes which must be transferred to the slave during the data-out phase. In order to guarantee avoidance of SCSI problems such as possible bus hang-ups, the slave must transfer this number of bytes during the data-out phase, even if the transferred bytes are discarded upon receipt, unless it skips the data-out phase altogether. The latter response is appropriate only under very exceptional conditions which are discussed in the next section. The discarded bytes may represent a valid SMDI message which the slave is for any reason unable to accept at the time that it appears (this would be reported as a SMDI error, using a SMDI message reply), or may be a SEND data packet in which no recognizable SMDI message appears. SMDI procedures are structured so that a slave should never need to handle a very large incoming message unexpectedly.

SMDI Message and Transaction Errors

It should be emphasized here that most of the error and exceptional conditions mentioned in the foregoing discussion of SMDI messages and discussed here in detail are not features of normal SMDI operation, which is typically quite straightforward and streamlined. Nevertheless, SMDI devices must be able to detect such conditions and respond to them in a consistent manner. Most of this burden falls to the slave device, which must report all errors back to the master. The master should be able to detect errors in messages from a slave and take appropriate action, but this action does not include reporting of errors to the slave.

SMDI error conditions fall into two main categories: those which are reportable using SMDI messages, and those which require SCSI-level handling. SMDI strives to keep communication at the SMDI level as far as possible, using only normally-terminated SEND and RECEIVE commands at all times. SCSI-level error handling and reporting (i.e., CHECK CONDITION status and the use of the REQUEST SENSE command) is considerably more disruptive and is employed only where the use of SMDI messages is made impossible by the circumstances of an error, or where the error condition is clearly outside the proper domain of SMDI.

Errors in the first category are related to valid SMDI message exchanges in which the message content is at issue - for example, the sending of a message with an unrecognized Message Code to a slave, or one which

is inappropriate in the context of a SMDI procedure currently in progress, or which contains an unexpected, incorrect or inappropriate parameter value in some part of the message beyond the message header. SMDI provides the means for a slave to report all such errors at the SMDI message level.

Errors in the second category are typically related to incorrect SEND/RECEIVE sequencing, or to problems in the SMDI message header itself. Such errors do not normally arise when properly-functioning SMDI devices are communicating with one another, and generally indicate a significant software error on the part of one of the devices, or that one of the devices is not a SMDI-capable device. In either case, the use of SMDI messages to report errors is unworkable. A SMDI slave reports the occurrence of such errors by returning CHECK CONDITION status during the status phase at the end of a SEND or RECEIVE command. A SMDI master (or other initiator) which obtains this status from a SMDI slave (or other target) must issue a REQUEST SENSE command and read extended sense data in order to obtain more specific information regarding the error. Within this category are two classes of errors: those which involve execution of a data-out or data-in phase, and those in which this phase is skipped.

Errors in the first class are those which are not detectable until data-in or data-out transfer has gotten underway, because the error is related to the content of the SEND/RECEIVE data packet. This includes data packets which do not begin with the "SMDI" message tag, those in which the Additional Message Length parameter in the SMDI message header is in conflict with the Transfer Length parameter of the SEND command, and those in which the Additional Message Length parameter deviates from the expected value for a known, fixed-length SMDI message. A SMDI slave handles such cases by continuing to accept (and, typically, discard) bytes from the initiator until the data-out phase is complete, based upon the value of the Transfer Length parameter of the SEND command, after which it proceeds to report the error in the status phase.

Errors in the second class are detectable at the SCSI command level and are related either to incorrect SEND/RECEIVE sequencing or to unacceptable Transfer/Allocation Length values (specifically, values smaller than 11, which preclude the possibility of transferring a complete SMDI message header). A SMDI slave responds to errors in this class by bypassing the data-in/out phase and proceeding directly from command phase to status phase to report the error.

There is one exceptional condition which does not fit neatly into any of the above categories. This condition arises when a master attempts to receive a slave's pending message using an Allocation Length which is equal to or greater than 11, but not large enough to permit the complete transfer of the slave's message. The correct slave response in this case is to return only a SMDI message header corresponding to the message it was prepared to reply with, and to retain this message as a pending reply. The header should indicate the Additional Message Length as though the entire message was being returned - this is referred to as a truncated receive, and provides the master with the information it needs to allow the complete message to be received without a violation of SCSI rules. The slave, in this situation, does not explicitly signal an error condition in either the SCSI or SMDI domain.

The master device is expected to detect a truncated receive when it occurs by noting that the Allocation Length which it used was not sufficient to accommodate a message with a length as indicated in the returned header, and is expected to immediately (i.e., without an intervening SEND command) make another attempt to receive the pending reply, using a larger Allocation Length based upon the Additional Message Length value in the returned header. Note that this is an exception to the SEND/RECEIVE alternation rule, as well as to the usual definition of the Additional Message Length parameter (since no additional message bytes will follow in this case). As is true at any other time when a SMDI slave has a reply message pending, the reply will remain pending until it can be completely transferred to the master, and any SEND command issued during this time

will be rejected.

Because the SMDI error conditions which are reported using SCSI-level error handling are nonetheless unique to SMDI, the sense data which a SMDI slave will return in response to the REQUEST SENSE command after such an error will consist of codes in the vendor-specific code ranges. The sense key will be 09h, the catch-all sense key for vendor-specific errors. The additional sense codes should also be read, as these will indicate precisely which SMDI error occurred (see table *xxxx*). SMDI devices should not use the SCSI 1 non-extended sense data format. SMDI slaves should be capable of returning the mandatory (SCSI 2) minimum of 18 bytes of sense information, and SMDI masters should use this number as the Allocation Length value in the REQUEST SENSE command.

SMDI Procedures

A SMDI procedure consists of one or more interlocked (SEND/RECEIVE) SMDI message exchanges between a SMDI master and a SMDI slave.

SMDI procedures provide the master device with the ability to learn the current state of the slave device and of its sample memory, to adapt itself to the protocol capabilities or limitations of the slave, and to initiate, perform and prematurely terminate sample or other data transfers in either direction between master and slave. The master may also be able to command the execution of specific tasks which are carried out entirely by the slave, with no further transfer of information between master and slave, such as deleting a sample from slave memory or directing the slave to save its memory contents to its own mass storage, if appropriate procedures have been defined. Other standard procedures will likely be defined subsequently to the release of this as-yet preliminary version of the SMDI protocol document, some using messages which also will be defined at a later time. Manufacturer-specific procedures are also possible, provided that they adhere to all of the protocol rules.

Both single- and multi-exchange procedures follow prescribed sequences which allow for little or no deviation, thus making the task of implementing these procedures in software a fairly simple one. There is usually but a single "expected" next message from either a master or a slave at any given point in a procedure, and the appearance of any inappropriate or unexpected message is always considered cause for immediate termination of the procedure and return to an idle or disengaged state from which another procedure may be initiated.

The SMDI Message Reject message may be presented at almost any time as a slave's reply to a master's message if conditions warrant it. This message incorporates rejection codes which permit a slave to notify the master of the specific reasons for the rejection of a message, and always results in termination of a procedure when it appears. A SMDI slave uses this message to terminate a procedure if the master sends it an inappropriate message (a slave must never abandon an incomplete procedure without completing the last SEND/RECEIVE exchange).

Most other SMDI messages have very specific uses and are considered inappropriate outside of their limited areas of applicability. While in the idle state, a SMDI slave is expected to reject any message from a master which is not appropriate to the idle state - i.e., which is not a message used to initiate a procedure.

The SCSI commands TEST UNIT READY, INQUIRY and REQUEST SENSE are not components of SMDI procedures. SMDI slave devices should be able to field these commands at any time without effect

upon a SMDI procedure in progress, including during the interval between a SEND command and its corresponding RECEIVE command. However, when a procedure is in progress, it is acceptable for a SMDI slave to terminate these commands with BUSY status instead of responding normally.

A reading of the following procedures should make clear the possibilities for confusion and errors which are posed by inadvertent interference between multiple master devices. A slave device should guard against this by keeping track of the SCSI ID of the master device with which it is communicating, by means of the SCSI ID information presented during the selection phase. When a multi-exchange SMDI procedure is in progress with a given master/initiator, the slave should respond to any SCSI command issued by any other initiator by terminating it with BUSY status. If the master/initiator does not identify itself during the selection phase, the slave must assume a single-initiator system with the initiator on SCSI id 7 - this is most likely the same assumption made by the initiator in such cases. It is then the user's/master's responsibility to avoid any situations in which multiple-master interference might occur.

SMDI Capability Test

The master sends an SMDI Master Identify message to slave, then expects to receive a SMDI Slave Identify message. This procedure should be performed after a master has identified a SCSI target as a SCSI processor device using the INQUIRY command, and before attempting other SMDI procedures. The master must send no further SMDI messages to this target if it fails to reply properly to this one. There are no restrictions on the use of this procedure when SMDI devices are in an idle state. It is permissible to use this procedure as a preliminary step whenever preparing to perform more extensive procedures. Note that the messages which are exchanged are entirely generic and do not convey any device-specific information.

Obtaining A Slave Sample Header

This procedure is used as a first step when the master wishes to select a particular sample in the slave for transfer to the master, or is used on its own when the master merely wishes to obtain information about the contents of a particular sample location in the slave.

The master sends a Sample Header Request message to the slave, then expects to receive either a Sample Header message or a SMDI Message Reject message as reply. The master must use an Allocation Length adequate to receive the Sample Header reply, which is the larger of the two.

The Sample Header reply is used if the specified sample exists and if it can be transmitted to the master if requested. The Sample Header is analogous to a MIDI SDS dump header - it presents all relevant information about the sample. If the master is preparing to obtain this sample from the slave, the Sample Header provides the information which allows the master to decide whether or not it can actually accommodate the sample.

The SMDI Message Reject reply is used if the specified sample does not exist, including cases in which the specified sample number is beyond the range used by the slave device. A SMDI Message Reject reply should also be used if the slave device's normal capabilities do not include the transfer of sample information to another device.

The slave device must not assume the liberty of interpreting sample numbers in a loose or inconsistent fashion. While the slave can use any desired method to map SMDI sample numbers to its own internal wave

numbers, this mapping must always yield the same result for a given SMDI sample number (i.e., it must always point to the same sample location in the slave regardless of circumstances such as addition/deletion of other samples, etc). Furthermore, the mapping must be unique - it must not permit more than one SMDI sample number to point to a given sample location. Sample numbers should not be aliased by the slave.

Transferring A Sample From Slave To Master

The master executes the procedure for obtaining a slave Sample Header. This provides the master with confirmation that the sample exists and can be transferred to the master, as well as with the information which the master needs to determine whether it can accommodate the sample, and which it will use both to enact the transfer and to maintain the sample after it has been transferred. It also serves to notify the slave that the specified sample may presently be requested for transfer to the master.

Following the successful transfer of a Sample Header to the master, the master must send a Begin Sample Transfer message to the slave. This message specifies both the requested sample number and the maximum Data Packet Length which the master can accommodate.

Note: In this discussion and those which follow, "Data Packet" refers to a specific SMDI message used to transfer blocks of file or other data, and should not be confused with SCSI SEND/RECEIVE data packets which are the vehicle for all SMDI messages, as referred to in preceding sections).

The master must receive a reply from the slave which will be either Begin Sample Transfer Acknowledge or SMDI Message Reject.

The Begin Sample Transfer Acknowledge reply is given when the slave is prepared to proceed with the transfer. It specifies the Data Packet Length which will be used by the slave in transferring the sample to the master. Normally, the slave device should be able to match the Data Packet Length (maximum) value sent by the master in the Begin Sample Transfer message, and in general it should do so unless it is unable to, even if the sample to be transferred is smaller than a single packet. Otherwise, it should specify the largest Data Packet Length it is able to support - under no circumstances should the slave specify a Data Packet Length larger than the maximum length indicated by the master. The Data Packet Length chosen must not result in the splitting of individual sample words across packets - eg., for a 16-bit format, the Data Packet Length must be an even number.

The SMDI Message Reject reply will be given if a Sample Header with the same Sample Number has not been successfully transferred to the master in the exchange immediately preceding that of the Begin Sample Transfer message. A SMDI Message Reject reply should also be used if the slave device's normal capabilities do not include the transfer of sample information to another device.

Assuming that the Begin Sample Transfer Acknowledge reply has been given, the transfer now proceeds as a series of slave-to-master packet transfer exchanges, each of which is as follows:

The master sends a Send Next Packet message to the slave. This message includes the number of the packet currently expected.

The master must then receive a reply from the slave. The reply is either a Data Packet message, a SMDI Message Reject message, or an Abort Procedure message.

Note: That the master, when receiving the reply, must use an Allocation Length which is large enough to accommodate a complete Data Packet message (usually much larger than the other possible replies) in order to ensure that the entire reply will be received from the slave. This length is equal to the Data Packet Length plus the Data Packet message overhead (SMDI message header plus Packet Number).

The Data Packet reply is the expected one.

The SMDI Message Reject reply is used when the Packet Number in the master's Send Next Packet message is different from the number of the packet which the slave is prepared to transfer.

The Abort Procedure message may be returned by the slave if for any other reason it is unable to continue the transfer. This may occur if the slave encounters an unrecoverable internal hardware failure or software/data error while the procedure is underway. Another possible reason for this is that the user has exercised the "manual override" option of hitting the designated stop button on the slave device's front panel (assuming that one exists) as a means of forcing an immediate halt to the transfer. It should be noted that it is considered preferable (and desirable) to implement this sort of manual stop capability on the master device, rather than on the slave. If a master-side manual stop is used, the master sends an Abort Procedure message to the slave instead of the next Send Next Packet message. The slave replies to the Abort Procedure message with an Ack message. The Abort Procedure message should not be used under normal conditions.

Following the successful completion of each Data Packet exchange, the master is expected to initiate the next packet exchange. Each Data Packet is expected to contain precisely the number of data bytes which the slave indicated would be used for these messages. The Data Packet message exchange continues until the number of bytes remaining to be transferred is equal to or less than the stated Data Packet Length, at which point a final Data Packet which contains only the remaining bytes (and is therefore probably smaller than the others) is transferred. The master and the slave are expected to independently anticipate both the time of appearance of the final Data Packet message and its data length. Once the final Data Packet is transferred, the procedure is complete, and both master and slave are disengaged.

Transferring A Sample From Master To Slave

The master sends a Sample Header message to the slave. This provides the slave with all of the sample information which it needs to determine whether it can accommodate the sample, and which it will use both to enact the transfer and to maintain the sample after it has been transferred. The master must receive a reply from the slave which will be either Begin Sample Transfer Acknowledge or SMDI Message Reject.

The Begin Sample Transfer Acknowledge reply is given to indicate that the slave is able to accept a transfer of the indicated sample from the master, and that it is ready for the master to initiate the transfer. This reply also informs the master of the maximum Data Packet Length which can be accommodated by the slave for the current transfer.

The SMDI Message Reject reply indicates that the slave cannot currently accept a transfer of the sample as it was represented by the Sample Header. Possible reasons for rejection include insufficient memory space available or a sample number which is beyond the slave device's range. A slave which can accommodate only monaural sample data files may reject a Sample Header indicating multiple interleaved channels of sampled audio data. A SMDI Message Reject reply should also be used if the slave device's normal capabilities do not include accepting sample information from another device.

A slave device which can respond to SCSI commands while engaged in an operation which would conflict with sample transfers may also give a SMDI Message Reject reply indicating that it is currently busy.

Note: That the existence of a sample in the slave at the specified sample number is not a valid reason for a reject reply - the slave is expected to delete any existing sample at the specified location once the transfer is actually initiated. Also note that the slave is not at liberty to assign incoming samples to sample numbers other than the one specified by the master. These stipulations are necessary in order to ensure that the master will have the assumed high degree of control over the slave's memory.

Following receipt of the Begin Sample Transfer Acknowledge reply, the master must issue the Begin Sample Transfer message. This message reiterates the target sample number and specifies the Data Packet Length which the master will use during the transfer. The master must receive a reply from the slave which will be one of Send Next Packet, Wait, or SMDI Message Reject.

Send Next Packet is the expected reply, and kicks off the series of master-to-slave Data Packet transfer exchanges. Each exchange consists of the sending of a Data Packet message by the master, followed by the receipt of a reply from the slave. Again, the typical replies are Send Next Packet, Wait and SMDI Message Reject.

Wait is given as a reply whenever the slave is unable to immediately accept the next Data Packet. Typically, the reason for this is that it must delete an existing sample at the specified destination sample location before it can begin transferring the new sample to that location. During the time that it keeps the master waiting, the slave device should assume a busy state - one in which it responds to a TEST UNIT READY command with BUSY status. The master device must issue the TEST UNIT READY command periodically to the slave until it responds with READY/GOOD status. The frequency of issue of these commands is left to the discretion of the master device, but should not be less than once per second in order to avoid excessive delays. Upon obtaining a READY/GOOD response from the slave, the master must receive a reply from the slave, which is expected to be Send Next Packet

Note: That this is an exception to the SEND/RECEIVE alternation rule. The master should then resume the Data Packet exchange process. If the slave is not able to maintain normal SCSI communications during the time it is busy, it can simply go completely off-line until it is able to resume the transfer.

A SMDI Message Reject reply may be obtained in response to the Begin Sample Transfer message if the sample number in this message is different from the one which was contained in the preceding Sample Header message, or if no Sample Header message was sent by the master immediately prior to the Begin Sample Transfer message. During the packet exchange process, a reject reply will be obtained if the Packet Number contained in a Data Packet is different from the packet number expected by the slave at that point. A SMDI Message Reject reply may also be obtained if the slave device's normal capabilities do not include

accepting sample information from another device, although if such is the case, the slave device will most likely reject the preceding Sample Header message and thereby prevent the procedure from proceeding any further.

Data Packet exchanges proceed in a manner similar to those in the procedure for transferring a sample from a slave to a master. After accepting the final Data Packet from the master, the slave should reply with an End Of Procedure message, at which point the procedure is complete, and both master and slave are disengaged.

As with sample transfers from the slave to the master, transfers from master to slave may be halted by either the master or the slave at any time by means of the Abort Procedure message. If this message is sent by the master, the slave replies with an Ack message to complete the message exchange and signal its acceptance of the Abort Procedure message.

Deleting A Sample From Slave Memory

This procedure is, in effect, a subset of the one for transferring a sample from a master to a slave. The master sends a Delete Sample From Slave Memory message to the slave. This message specifies the number of the sample which the master wishes to have deleted.

The expected slave reply is End Of Procedure. If the process of deleting a sample involves long time delays on the part of the slave, the slave may instead reply with the Wait message. This reply is handled in exactly the same fashion as was described in the procedure for transferring a sample from a master to a slave, and the End Of Procedure reply is expected after the Wait period ends.

The SMDI Message Reject Reply will be used if the specified sample number is out of range for the slave device, if the sample location specified is already empty, or if the slave device does not support this operation.

SMDI Messages

This section defines specific SMDI messages and provides relevant details on how each is to be used.

Command Descriptor Blocks (CDBs) for the SCSI SEND and RECEIVE commands used to transmit SMDI messages are shown immediately below for reference. For the purposes of SMDI, all fields in these CDBs other than the operation code (byte 0) and the Transfer/Allocation Length (bytes 2-4) should always be set to zero. A slave device which receives these commands with non-zero values in bytes 1 or 5 is required to reject them according to the standard SCSI method for responding to an Illegal Request condition.

Each message diagram indicates only the actual message text - i.e., the data transferred during the the data-out phase of a SEND command or the data-in phase of a RECEIVE command. The command descriptor block for the associated SCSI SEND or RECEIVE command is not shown with each message description, since it is always the same except for the Transfer/Allocation Length value.

SMDI Message: No Message

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB)							
1								
2				SMDI message tag (“SMDI”)				
3								(LSB)
4	(MSB)							
5				Message ID Code (0000H)				(LSB)
6	(MSB)							
7				Message sub ID Code (0000H)				(LSB)
8	(MSB)							
9				Additional Message Length (000000H				
10								(LSB)

No defined uses at present.

SMDI Message: SMDI Master Identify

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB)							
1								
2				SMDI message tag (“SMDI”)				
3								(LSB)
4	(MSB)							
5				Message ID Code (0001H)				(LSB)
6	(MSB)							
7				Message Sub-ID Code (0000H)				(LSB)
8	(MSB)							
9				Additional Message Length (000000				
10								(LSB)

Sent by a master, usually in order to test an unknown target SCSI processor device for its ability to reply as a SMDI slave. The slave is expected to reply with the SMDI Slave Identify message.

SMDI Message: SMDI Slave Identify

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB)							
1								
2				SMDI message tag("SMDI")				
3								(LSB
4	(MSB)							
5				Message ID Code (0001H)				(LSB
6	(MSB)							
7				Message sub ID Code (0001H)				(LSB
8	(MSB)							
9				Additional Message Length (000000				
10								(LSB

Standard reply to the SMDI Master Identify message. Identifies a target SCSI processor device as a SMDI-capable slave device.

SMDI Message: SMDI Message Reject

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB)							
1								
2				SMDI message tag (“SMDI”)				
3								(LSB
4	(MSB)							
5				Message ID Code (0002H)				(LSB
6	(MSB)							
7				Message Sub-ID Code (0000H)				(LSB
8	(MSB)							
9				Additional Message Length (000004				
10								(LSE
11	(MSB)							
12				Rejection Code				(LSE
13	(MSB)							
14				Rejection Sub-Code				(LSE

Slave reply to a any message from a master requesting information which is not available or an operation which the slave is not able to perform, or containing invalid or out-of-range parameters, or arriving at an inappropriate time, or otherwise presenting a situation in which the slave cannot provide the reply normally expected. The Message Reject message contains specific rejection codes which indicate the reason for its use as a reply. Defined rejection codes are listed elsewhere in this document.

SMDI Message: ACK

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB)							
1								
2				SMDI message tag (“SMDI”)				
3								(LSB
4	(MSB)							
5				Message ID Code (0100H)				(LSB
6	(MSB)							
7				Message Sub-ID Code (0000H)				(LSB
8	(MSB)							
9				Additional Message Length (00000				
10								(LSB

Reply given by a slave to any message from a master which signals an unexpected change in the course of a procedure - e.g., Abort Procedure - and which demands a recognizable reply in order to complete the exchange.

SMDI Message: NAK

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB)							
1								
2				SMDI message tag (“SMDI”)				
3								(LSB
4	(MSB)							
5				Message ID Code (0101H)				(LSB
6	(MSB)							
7				Message Sub-ID Code (0000H)				(LSB
8	(MSB)							
9				Additional Message Length (000000				
10								(LSB

No defined uses at present.

SMDI Message: Wait

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB)							
1								
2				SMDI message tag (“SMDI”)				
3								(LSB)
4	(MSB)							
5				Message ID Code (0102H)				(LSB)
6	(MSB)							
7				Message Sub-ID Code (0000H)				(LSB)
8	(MSB)							
9				Additional Message Length (000000H				
10								(LSB)

This message may be used as a reply a slave at any time during a procedure to notify the master that an abnormally long or unknown delay will be experienced before the procedure can continue. This is typically related to a need on the part of the slave to access local mass storage or to rearrange memory contents in order to continue the procedure (for example, deleting a sample from memory when the master is sending a new one to the same location). The Wait message is appropriate whenever the anticipated delay has some likelihood of approaching the standard 250 millisecond SCSI Selection Response Time and thereby raising the possibility that the slave will miss the master’s next SCSI selection attempt.

The Wait message is never an expected reply, but always preempts a different expected reply. At the end of the delay signalled by the Wait message, during which the slave device may or may not go completely off-line, the slave is expected to have a reply message pending for the master, which receives the pending reply in order to resume the procedure. The master must not attempt to send any further messages to the slave before it has received this reply. The message with which the slave replies at this time depends upon the procedure being performed, as well as upon current device conditions. Typically, however, it will be the reply which was originally expected at the time that the Wait message was received in its place. In this sense, the Wait message and its associated delay can be viewed, from a procedure perspective, as having been inserted transparently between a message SEND from a master and its RECEIVE of an expected reply.

SMDI Message: Send Next Packet

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB)							
1								
2				SMDI message tag (“SMDI”)				
3								(LSB
4	(MSB)							
5				Message ID Code (0103H)				(LSB
6	(MSB)							
7				Message Sub-ID Code (0000H)				(LSB
8	(MSB)							
9				Additional Message Length (00000				
10								(LSB
11	(MSB)							
12				Packet Number				
13								(LSB

Packet handshake message during a data transfer procedure. Sent by the master during slave-to-master transfers - received from the slave during master-to-slave transfers. The Packet Number is that of the next packet to be sent.

SMDI Message: End of Procedure

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB)							
1								
2				SMDI message tag (“SMDI”)				
3								(LSB)
4	(MSB)							
5				Message ID Code (0104H)				(LSB)
6	(MSB)							
7				Message Sub-ID Code (0000H)				(LSB)
8	(MSB)							
9				Additional Message Length (000000				
10								(LSB)

May be used by slave or master to signal or acknowledge the normal completion of a procedure. The uses of this message are defined specifically for each procedure - it is not applicable to all procedures.

SMDI Message: Abort Procedure

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB)							
1								
2				SMDI message tag (“SMDI”)				
3								(LSB
4	(MSB)							
5				Message ID Code (0105H)				(LSB
6	(MSB)							
7				Message Sub-ID Code (0000H)				(LSB
8	(MSB)							
9				Additional Message Length (000000				
10								(LSB

May be transmitted by either master or slave at any time during any procedure to halt the procedure immediately. If this message is sent by the master, the slave should reply with an Ack message to complete the message exchange and signal its acceptance of the Abort Procedure message.

SMDI Message: Data Packet

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB)							
1								
2				SMDI message tag (“SMDI”)				
3								(LSB)
4	(MSB)							
5				Message ID Code (0110H)				(LSB)
6	(MSB)							
7				Message Sub-ID Code (0000H)				(LSB)
8	(MSB)							
9				Additional Message Length (n+3)				
10								(LSB)
11	(MSB)							
12				Packet Number				
13								(LSB)
14 to				packet data (n bytes)				
n+13								

This message is used to transfer a packet of data in either direction between master and slave. The content of the data section of the Data Packet message depends upon the particular transfer currently in progress. It may be raw sample data, device-specific control information, or any other type of information which, because of its nature, is more convenient to transmit as a series of packets than by way of specific messages defined for that type of data. The Data Packet message format shown above is used in all cases.

Raw sample data transfers use a data format identical to the Sound Designer II data-fork format. Each packet contains a number of sample data words represented as a string of bytes. Sample data is encoded in two's complement format. 16-bit sample data is sent two bytes per word in little endian format - i.e., MS byte first, followed by LS byte. Data of other precision is handled by adjusting the format accordingly. Two bytes per word are used for samples with resolution from 9-15 bits, with the significant bits of each sample left-justified in each 16-bit word formed by an ms/ls pair of packet bytes, and all non-significant bits set to zero. Samples of 8-bit or lower resolution require only a single byte per sample, with sample data again left-justified in each byte. Samples with resolution beyond 16 bits require three or more bytes per sample word - the left-justification rule applies here as well.

The data byte count of a Data Packet message may not be such as to split a single sample data word across consecutive packets. The data byte count of a Data Packet message must never be zero. There is no other specific lower limit on the allowable size of a Data Packet except for the need to satisfy requirements (such as the one regarding word-splitting) imposed by the packet contents.

Multi-channel sounds are transmitted in an interleaved format of ascending channel number according to the number of channels involved. For example, a stereo sample is sent as alternating left and right sample words (left channel is channel 1, right channel is channel 2). A four-channel sound would rotate through channels 1-4 in order, presenting one sample data word from each channel in turn. It is permissible for multi-channel sounds to be divided into packets without regard for this rotation, provided that the prohibition against splitting of individual sample words is observed.

In a multi-packet transfer of n packets, transfers start with packet number 0 and proceed through packet 1, 2, 3, ..., n-1. The total number of packets required to complete any transfer is determined completely by the size of the file being transferred, the data format used, and the negotiated Data Packet Length. All packets are expected to be the same length, with the exception of the final packet, which will usually be shorter than the others, since it contains only the bytes which remain to be transferred. No filler bytes are to be inserted at the end of the final packet.

SMDI Message: Sample Header Request

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB)							
1								
2				SMDI message tag (“SMDI”)				
3								(LSB)
4	(MSB)							
5				Message ID Code (0120H)				(LSB)
6	(MSB)							
7				Message Sub-ID Code (0000H)				(LSB)
8	(MSB)							
9				Additional Message Length (000003				
10								(LSB
11	(MSB)							
12				Sample Number				
13								(LSB

Sent by a master device to request a sample header. The slave replies with a Sample Headerssage, or with Message Reject. The sample number is used to indicate a specific sample location in the slave device about which the master wishes to obtain information.

SMDI Message: Sample Header

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB)							
1	SMDI message tag (“SMDI”)							
2								
3								(LSB)
4	(MSB)							
5	Message ID Code (0121H)							(LSB)
6	(MSB)							
7	Message Sub-ID Code (0000H)							(LSB)
8	(MSB)							
9	Additional Message Length (00001							
10								(LSB
11	(MSB)							
12	Sample Number							
13								(LSB
14	Sample format - Bits Per Word							
15	Sample format - Number Of Channe							

16	(MSB)
17	Sample Period (nanoseconds)
18	(LSB)
19	(MSB)
20	Sample Length (words)
21	
22	(LSB)
23	(MSB)
24	Sample Loop Start Word Number
25	
26	(LSB)
27	(MSB)
28	Sample Loop End Word Number
29	
30	(LSB)
31	Sample Loop Control
32	(MSB)
33	Sample Pitch Integer (LSB)
34	(MSB)
35	Sample Pitch Fraction (LSB)
36	Sample Name Lenght (n)
37 to n + 36	Sample Name (n bytes)

The sample header message transmits essential sample control information.

In general, parameters are as per the MIDI SDS Sample Dump Header. however, that multi-byte values are transmitted in MS-byte-to-LS-byte order, in contrast to MIDI SDS, but in keeping with general SCSI practice.

Sample Loop Start and Loop End word numbers may range from zero (the start of the sample) to [Sample Length minus one].

Sample Format - Number Of Channels refers to possible interleaving of sample data for two or more audio channels into a single file. This follows the Sound Designer II method of sample interleaving. When more than one channel of sample data is indicated, the Sample Length, Loop Start and Loop End values apply to each of the interleaved channels individually, and it is understood that the actual (word) length of the file when it is transferred will be the specified sample length multiplied by the specified number of channels.

Sample Pitch indicates the absolute musical pitch of the sound represented by the sample when played at the specified sample rate. It is presented in an integer/fraction format using a pitch numbering system borrowed from the MIDI Tuning Dump Standard. In this system, the integer designates a specific semitone (eg., 60 decimal, or 3CH, for middle C) and the fraction indicates a fraction of a semitone upward from the specified semitone (thus, 003C.0000 is exactly middle C, 003C.028F is one cent above middle C, 003C.8000 is 50 cents above middle C, etc). In cases where the sample pitch value is unavailable or not applicable, a default value of 003C.0000 should be used).

The Sample Name is an ASCII string which may contain any combination of character codes, subject to these restrictions:

Non-printing (i.e., control) characters should not be used.

Leading and trailing blanks should not be used - embedded blanks are permissible.

The length of the string may anywhere from 0 to 255 bytes and must exactly match the Sample Name Length parameter.

A device which receives a Sample Header message containing a name string may alter the received string as required to conform to the above restrictions or, more importantly, to meet constraints imposed by the receiving system (especially on the length of the name string). If appropriate, the sample name may be ignored altogether. A Sample Header message should never be rejected on the basis of the Sample Name Length or the content of the Sample Name field. Naming conventions and restrictions differ widely among systems which make use of sampled audio information - therefore, it should be assumed that a sample name which is relayed to another device may be altered to some degree upon arriving there. If the device generating the Sample Header message does not maintain name information, or does not have name information for the specified sample, it may specify a Sample Name Length of zero, and omit the Sample Name string entirely.

SMDI Message: Sample Name

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB)							
1								
2				SMDI message tag (“SMDI”)				
3								(LSB)
4	(MSB)							
5				Message ID Code (0123H)				(LSB)
6	(MSB)							
7				Message Sub-ID Code (0001H)				(LSB)
8	(MSB)							
9				Additional Message Length (000004H+n)				
10								(LSB)
11	(MSB)							
12				Sample Number				
13								(LSB)
14				Sample Name Length (n)				
15 to								
n + 14				Sample Name (n bytes)				

The Sample Name message is used by a master device to assign a new name to an existing sample in a slave device. Comments regarding the Sample Name parameter are the same as those for the Sample Name portions of the Sample Header message - refer to the description of that message for further details.

The expected reply is End Of Procedure. The Sample Name message should be rejected only if the specified Sample Number is out of range or corresponds to an unused sample location in the slave, or if the slave does not support this operation.

SMDI Message: Delete Sample From Memory

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB)							
1								
2				SMDI message tag (“SMDI”)				
3								(LSB
4	(MSB)							
5				Message ID Code (0124H)				(LSB
6	(MSB)							
7				Message Sub-ID Code (0000H)				(LSB
8	(MSB)							
9				Additional Message Length (000003)				
10								(LSB
11	(MSB)							
12				Sample Number				
13								(LSB

Used by a master device to command a slave to delete a specific sample from its memory. The expected reply is End Of Procedure - a Wait reply and associated delay may be inserted ahead of this reply. The Delete Sample From Memory message should be rejected only if the specified Sample Number is out of range or corresponds to an unused sample location in the slave, or if the slave does not support this operation.

SMDI Message: Begin Sample Transfer

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB)							
1								
2								
3								(LSB)
4	(MSB)							
5								(LSB)
6	(MSB)							
7								(LSB)
8	(MSB)							
9								
10								(LSB)
11	(MSB)							
12								
13								(LSB)
14	(MSB)							
15								
16								(LSB)

SMDI message tag (“SMDI”)

Message ID Code (0122H)

Message Sub-ID Code (0001H)

Additional Message Length (00000

Sample Number

Data packet length

Used in conjunction with Begin Sample Transfer Acknowledge (see below).

Sent by a master to a slave to initiate the transfer of a sample in either direction. Assumes that a sample header has been successfully transferred in the immediately-preceding message exchange. The Sample Number must match that of the preceding Sample Header message. A slave should reject this message if either

of these conditions is not met.

In transfers from slave to master, the Data Packet Length value specifies the maximum Data Packet length (in bytes) that the master can accommodate. The expected slave reply is Begin Sample Transfer Acknowledge.

In transfers from master to slave, the Data Packet Length value specifies the Data Packet length (in bytes) that the master will use. The expected slave reply is Send Next Packet.

SMDI Message: Begin Sample Transfer Acknowled

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB)							
1								
2				SMDI message tag (“SMDI”)				
3								(LSB)
4	(MSB)							
5				Message ID Code (0122H)				(LSB)
6	(MSB)							
7				Message Sub-ID Code (0001H)				(LSB)
8	(MSB)							
9				Additional Message Length (000006H				
10								(LSB)
11	(MSB)							
12				Sample Number				
13								(LSB)
14	(MSB)							
15				Data Packet Length				
16								(LSB)

Used in conjunction with Begin Sample Transfer (see above).

Received from a slave as a reply to a master’s Begin Sample Transfer message when initiating a sample transfer from slave to master. In this case, the Data Packet Length value indicates the number of data bytes per Data Packet message that will be used by the slave in carrying out the transfer.

Received from a slave as a reply to a master's Sample Header message when initiating a sample transfer from master to slave. In this case, the Data Packet Length value indicates the maximum number of data bytes per Data Packet message that can be accommodated by the slave in carrying out the transfer.

SMDI Message: Transmit MIDI Message

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
1								
2				SMDI message tag (“SMDI”)				
3								(LSB)
4	(MSB)							
5				Message ID Code (0200H)				(LSB)
6	(MSB)							
7				Message Sub-ID Code (0000H)				(LSB)
8	(MSB)							
9				Additional Message Length (n+3)				
10								(LSB)
11 to	(MSB)							
n+10				MIDI Message String (n bytes)				

Used to transfer a MIDI message via the SMDI protocol.

Any legal MIDI message may be transmitted by this message. The MIDI message string is a literal MIDI message complete with status and data bytes, presented exactly as it would be if transmitted via a MIDI interface. The string should contain one and only one complete MIDI message. The SMDI protocol does not support MIDI running status - status bytes must be presented explicitly.

The use of SMDI for MIDI message transmission is mainly envisioned as a method of transferring system exclusive and other non-real-time information. SMDI is not appropriate for transmission of time-critical performance or timing event messages. Nonetheless, there is no prohibition against using SMDI to transmit such MIDI event messages if the inherent limitations of doing so are not objectionable (e.g., transmission of test note events from within sample editor/librarian applications).

Also, SMDI capability does not imply any specific type or degree of MIDI implementation, nor does it define SMDI slave replies to MIDI messages transmitted by a master using this protocol, other than a default reply of No Message to a MIDI message sent by a master device. This may often be the most appropriate reply to such a message. more specific replies, such as MIDI messages returned by the slave, are also possible, but are not defined by the SMDI protocol.

Table 1 - SMDI Message Codes

Note: Message ID Codes F000H-FFFFH are reserved for manufacturer or device specific messages.

Message ID Code	Message Sub-ID Code	Message Name
0000H	0000H	No Message
0001H	0000H	SMDI Master Identify
0001H	0001H	SMDI Slave Identify
0002H	0000H	SMDI Message Reject
0100H	0000H	Ack
0101H	0000H	Nak
0102H	0000H	Wait
0103H	0000H	Send Next Packet
0104H	0000H	End Of Procedure
0105H	0000H	Abort Procedure
0110H	0000H	Data Packet
0120H	0000H	Sample Header Request
0121H	0000H	Sample Header
0122H	0000H	Begin Sample Transfer
0122H	0001H	Begin Sample Transfer Acknowledge
0123H	0000H	Sample Name
0124H	0000H	Delete Sample From Memory
0200H	0000H	Transmit MIDI Message

Table 2 - SMDI Message Rejection Codes

Message Rejection Code	Message Rejection Sub-Code	Message Rejection Cause
0002H		Last message rejected (general):
	0000H	- message/procedure is not supported
	0002H	- message is inappropriate
0005H	0000H	Device is busy
0011H		Send Next Packet message rejected:
	0000H	- Packet Number mismatch
0020H		Sample Header, Sample Header Request or other sample operation message rejected:
	0000H	- Sample Number is out of range
	0002H	- no sample at this Sample Number
	0004H	- insufficient sample memory available
	0005H	- insufficient param memory available
	0006H	- can't accommodate Sample Format - Bits Per Word
	0007H	- can't accommodate Sample Format - Number Of Channels
0022H		Begin Sample Transfer message rejected:
	0001H	- Sample Number does not match that of previously transferred Sample Header
	0002H	- can't accommodate Data Packet Length

Table 3 - SMDI Sense Codes (for use with REQUEST SENSE command)

These codes are to be used as the Additional Sense Code in byte 12 (of bytes 0 thru 17) of the 18-byte Extended Sense Data format. They are appropriate only when reporting one of the SMDI-specific error conditions listed below, and should be used only with a Sense Key (byte 2) value of 09H (Vendor-Specific). All of the Sense Codes values listed here are within the range reserved by the SCSI specification for vendor-specific use and do not coincide with any standard SCSI Sense Codes. The Additional Sense Code Qualifier (byte 13) is zero for all errors.

SMDI slave devices are required by their nature to support various Sense Keys and Additional Sense Codes other than the SMDI-specific ones listed here. These others are standard ones defined by the SCSI specification and are not reiterated here.

Sense Code Value	Sense Code Meaning
80H	SEND rejected - slave reply message is pending.
81H	RECEIVE rejected - no slave reply message is pending.
82H	SEND rejected - initiator's Transfer Length is smaller than 11.
83H	RECEIVE rejected - initiator's Allocation Length is smaller than 11.
84H	SEND rejected - initiator is sending a non-SMDI message.
85H	SEND rejected - initiator's Transfer Length and Additional Message Length parameters are in conflict.
86H	SEND rejected - initiator's Additional Message Length parameter is incorrect for the current message.