AUTOLOADER SCSI GATEWAY PROTOCOL:
SERIAL BUS IMPLEMENTATION
This document is a development standard adopted by Quarter-Inch Cartridge Drive Standards, Inc. (QIC). This document may be revised several times during the development cycle. It is intended solely as a guide for companies interested in developing products which can be compatible with other products developed using this document. QIC makes no representation or warranty regarding this document, and any company using this document shall do so at its sole risk, including specifically the risks that a product developed will not be compatible with any other product or that any particular performance will not be achieved. QIC shall not be liable for any exemplary, incidental, proximate or consequential damages or expenses arising from the use of this document. This development standard defines only one approach to the product. Other approaches may be available in the industry.

This development standard is an authorized and approved publication of QIC. The underlying information and materials contained herein are the exclusive property of QIC but may be referred to and utilized by the general public for any legitimate purpose, particularly in the design and development of quarter-inch tape cartridge drive subsystems. This development standard may be copied in whole or in part provided that no revisions, alterations or changes of any kind are made to the materials contained herein. Only QIC has the right and authority to revise or change the material contained in this development standard, and any revisions by any party other than QIC are totally unauthorized and specifically prohibited.

Compliance with this development standard may require use of one or more features covered by proprietary rights (such as features which are the subject of a patent, patent application, copyright, mask work right or trade secret right). By publication of this development standard, no position is taken by QIC with respect to the validity or infringement of any patent or other proprietary right, whether owned by a Member or Associate of QIC, or otherwise. QIC hereby expressly disclaims any liability for infringement of intellectual property rights of others by virtue of the use of this development standard. QIC has not and does not investigate any notices or allegations of infringement prompted by publication of any QIC development standard, nor does QIC undertake a duty to advise users or potential users of QIC development standards of such notices or allegations. QIC hereby expressly advises all users or potential users of this development standard to investigate and analyze any potential infringement situation, seek the advice of intellectual property counsel, and, if indicated, obtain a license under any applicable intellectual property right or take the necessary steps to avoid infringement of any intellectual property right by virtue of the evolution, adoption, or publication of any QIC development standard.
1.0 INTRODUCTION

The objective of the SCSI Gateway Protocol is to allow a SCSI-II media changer and tape drive to exist on the SCSI bus at a single SCSI ID with each device occupying one logical unit number of the device ID.

One method of achieving this functionality dictates that only one device will be physically connected to the SCSI bus. This primary device would be responsible for providing SCSI pass-through functionality for the secondary device. This protocol adopts this philosophy and requires the tape drive to be physically connected to the SCSI bus and the media changer to be physically connected to the tape drive through a serial communications link. The tape drive will be responsible for managing the SCSI bus communications and parsing the appropriate information across the serial bus for host-to-changer communications. In addition, the protocol supports direct operations between the tape drive and media changer.

This protocol defines Master and Slave roles for devices on the serial bus. In a typical system, there will be a tape drive and a tape changer device. The tape drive is designated as the Master in the system. The Master is responsible for passing SCSI commands through to the tape changer, designated the Slave device.

The protocol attempts to establish a method for the Master device to manage SCSI communications between a SCSI Host and the Slave device. In doing so, certain requirements are placed on the Master device in terms of SCSI bus interface management. This implementation requires the Master device to manage all SCSI bus interface for the Slave device. For instance, the Master device must manage all SCSI bus messaging protocols.
2.0 PHYSICAL LAYER

The physical layer of the interface defines separate transmit and receive data lines, yielding a minimum two-wire serial bus configuration and allowing for full-duplex operations. Serial baud rate support includes 1200, 2400, 4800 and 9600 baud with transmission characters consisting of 1 start bit, 8 data bits and 1 stop bit.

NOTE:
Due to serial bus overhead requirements, it is recommended that the maximum baud rate possible be utilized to minimize SCSI overhead.

2.1 Serial Bus Device Addressing

Each device located on the serial bus will be assigned a unique serial bus device address. It is through this unique address that devices communicate on the bus. For a simple two-device Master/Slave configuration, the Master device will always be assigned an address of ‘0’. The Slave device will assume an address of ‘1’. 
3.0 PACKET LAYER

All serial bus communications take place in the form of packets and packet handshakes. The Packet Layer defines the serial bus packet-level communications protocol. Although support is provided for multiple packet types, this layer does not attempt to identify the contents of the packet data field but merely defines the serial packet transmission protocol.

3.1 Packet Format

The serial communications packet format is defined consistently for all type of communications. All packets are formatted in ASCII or ASCII-hex in order to delineate packet data from asynchronous messages and packet handshakes. Packets consist of a packet START character, serial device ADDRESS field, packet TYPE field, packet data COUNT field, packet DATA field, packet CHECKSUM field and a packet STOP character.

<table>
<thead>
<tr>
<th>Start Field</th>
<th>The PACKET START field contains a single ASCII character (STX 02h) marking the start of the current packet.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Address Field</td>
<td>The PACKET ADDRESS field contains a single hexadecimal byte expressed in two ASCII-hex characters indicating the serial device ID of the destination device.</td>
</tr>
<tr>
<td>Type Field</td>
<td>The PACKET TYPE field contains a single hexadecimal byte expressed in ASCII-hex format representing the packet type of the current packet.</td>
</tr>
<tr>
<td>Count Field</td>
<td>The PACKET COUNT field consists of a single hexadecimal byte expressed in ASCII-hex indicating the number of hexadecimal bytes contained in the PACKET DATA field.</td>
</tr>
<tr>
<td>Data Field</td>
<td>The PACKET DATA field contains from 0 to 255 bytes of packet data each expressed as two ASCII-hex characters. The contents of the packet data field are not defined by the packet definition.</td>
</tr>
<tr>
<td>Checksum Field</td>
<td>The PACKET CHECKSUM field contains a single hexadecimal byte expressed in two ASCII-hex characters representing a sum-of-bytes for all packet bytes starting with the packet ADDRESS field, ending with and including the last byte of the PACKET DATA field.</td>
</tr>
<tr>
<td>Stop Field</td>
<td>The PACKET STOP field contains a single ASCII character (ETX 03h) marking the end of the current packet.</td>
</tr>
</tbody>
</table>

3.2 Packet Handshake
All properly-received packets are positively confirmed by the receiving device in order to manage potential serial communication errors. Following the reception of a transmitted packet, the receiving device will confirm packet integrity and respond with an appropriate packet handshake.

Packet integrity is determined by performing a cumulative eight-bit addition of all received bytes beginning with the PACKET ADDRESS field and continuing through the PACKET DATA field. This cumulative eight-bit value is then compared against the value found in the PACKET CHECKSUM field. A compare failure indicates a packetization error or packet transmission error.

In order to minimize serial bus overhead, properly-received packets may be acknowledged by a Slave device by continuing with the next logical packet transmission. This shall be considered to be an appropriate packet handshake and acknowledgement of packet integrity.

A command packet (SCSI_COMMAND or SERIAL_COMMAND) which is not received correctly by a Slave device will NOT be acknowledged. An erroneously-received command packet may have experienced an error in the PACKET ADDRESS field, in which case the Slave device has not been properly identified for the associated operation. A time-out will be required for this situation.

ACK  Acknowledge (06h)

The Master device may transmit this packet handshake byte to indicate successful reception of the current packet. The Slave device will positively acknowledge successful reception of a packet by transmitting the next logically-appropriate packet.

NAK  Negative Acknowledge (15h)

A receiving device may transmit this packet handshake byte to indicate unsuccessful reception of the current packet. The current packet should be retransmitted in its entirety. Either the Master or Slave device may transmit this packet handshake.

ESC  Escape (1Bh)

A device may transmit this control character asynchronously to terminate the current serial bus operation. This control character may be issued by the Master device to terminate the current operation and connection (usually following the reception of a STATUS packet). This control character may also be transmitted by the Slave device asynchronously to terminate the current operation similar to a SCSI bus-free transition.

CAN  Cancel (18h)

The Master device may transmit this control character asynchronously to indicate the occurrence of a SCSI bus reset or the receipt of a Bus Device Reset message. The Slave device should terminate the current operation and connection and perform a hard-reset.
4.0 PACKET PROTOCOL

The Master device initiates activity on the serial bus with the Slave device by means of a
SCSI_COMMAND or SERIAL_COMMAND packet transmission. These packets will never be
rejected by the Slave device as unsuccessful reception has not positively identified the Slave
device.

A SCSI_COMMAND packet is used to initiate a SCSI connection between the SCSI Host and the
target Slave device. For SCSI operations, the Master device serves simply as a gateway between
the SCSI Host and the target Slave device which represents the SCSI Target/LUN. No restriction
is placed on the master device. It may intercept, interpret, buffer, or queue SCSI commands from
the SCSI Host if desired.

A SERIAL_COMMAND packet is used to initiate a serial connection between the Master device and
the target Slave device. For serial operations, the Master device is the host of the operation.

Following the successful initiation of an operation, the sequence of operations on the serial bus
is controlled by the Slave device. Three types of commands exist: (1) simple command/status
commands, (2) command/data out/status commands, and (3) command/data in/status
commands. For each type of command, protocol sequence will follow a logical fashion (i.e.,
command packet/status packet, etc.).

For either type of connection, a STATUS packet issued by the Slave device terminates the
operation. The Master device will respond to the STATUS packet with either a NAK (requesting
retransmission) or an ESC (requesting termination) handshake. Either the Master or Slave
device may also terminate an operation at any time by means of a ESC handshake.

4.1 Packet Types

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Byte 0</td>
<td>SCSI Host ID</td>
</tr>
<tr>
<td>Data Byte 1</td>
<td>SCSI Target ID</td>
</tr>
<tr>
<td>Data Byte 2</td>
<td>SCSI Identify Message Byte</td>
</tr>
<tr>
<td>Data Byte 3..n</td>
<td>SCSI Command Data</td>
</tr>
</tbody>
</table>

This packet is issued by the Master device to direct the Slave device
to execute the accompanying command. The source of the command
data is the SCSI Host.

Normally, the Master transmits the correct number of bytes. If,
however, the Master transmits six bytes of SCSI command data and
the Slave device determines that the SCSI command is larger than six
bytes, the Slave device will issue a GET_SCSI_COMMAND control
packet to the Master device with a byte count equal to the balance of
the command length.
**SERIAL COMMAND PACKET**

Data Byte 0..n = Serial Command Data

This packet is issued by the Master device to direct the Slave device to execute the accompanying command. The source of the command data is the Master device. The entire serial command must be transmitted in a single SERIAL COMMAND packet transfer.

**DATA PACKET**

Data Bytes 0..n-1 = Data

This packet will be issued by the Master device in response to a GET_DATA packet from the Slave. Data-out data may be transmitted to the Slave device in one or more DATA packet transfers from the Master device. In the case of a SCSI connection, the source of these data bytes will be the SCSI Host. In the case of a serial connection, the data source will be the Master device.

This packet will also be issued by the Slave device to transfer data to the Host device. The data-in data may be transmitted to the Master device in one or more DATA packet transfers from the Slave device. In the case of a SCSI connection, the destination of these data bytes will be the SCSI Host. In the case of a serial connection, the data destination will be the Master device.

**STATUS PACKET**

Data Byte 0 = Status Data

This packet is issued by the Slave device to complete the current command operation. For a SCSI operation, the destination for the Status byte will be the SCSI Host. For a serial operation, the destination will be the Master device.

**CONTROL PACKET**

Data Byte 0 = Control
Data Bytes 1..2 = Byte Count

This packet will be issued by the Slave device to request the Master device to perform a particular operation as defined by the Control byte.

GET_SCSI_COMMAND = Get SCSI Command Bytes
GET_SCSI_DATA = Get SCSI Data Bytes
5.0 Command/Status Definitions

5.1 SCSI Commands

Refer to SCSI specification for details on SCSI commands.

5.2 SCSI Status

Refer to SCSI specification for details on SCSI status.

5.3 Serial Commands

To be determined. As an example, a Slave Autoloader and Master Drive may want to communicate drive activity light information so that it may be displayed on the Autoloader front panel.

5.4 Serial Status

To be determined.
6.0 Example Sequences

![Diagram of SCSI Test Unit Ready Operation]

**Figure 1. SCSI Test Unit Ready Operation**
Figure 2. Simple Command Operation

<table>
<thead>
<tr>
<th>Drive</th>
<th>Changer</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCSI COMMAND PACKET</td>
<td>GET SCSI DATA PACKET</td>
</tr>
<tr>
<td>SCSI DATA PACKET</td>
<td>SCSI STATUS PACKET</td>
</tr>
<tr>
<td>ESC</td>
<td>ESC</td>
</tr>
</tbody>
</table>

Figure 3. Data Out Command Operation

<table>
<thead>
<tr>
<th>Drive</th>
<th>Changer</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCSI COMMAND PACKET</td>
<td>SCSI DATA PACKET</td>
</tr>
<tr>
<td>ACK</td>
<td>SCSI STATUS PACKET</td>
</tr>
<tr>
<td>ESC</td>
<td>ESC</td>
</tr>
</tbody>
</table>

Figure 4. Data In Command Operation

<table>
<thead>
<tr>
<th>Drive</th>
<th>Changer</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCSI COMMAND PACKET</td>
<td>SCSI STATUS PACKET</td>
</tr>
<tr>
<td>NAK</td>
<td>SCSI STATUS PACKET</td>
</tr>
<tr>
<td>ESC</td>
<td>ESC</td>
</tr>
</tbody>
</table>

Figure 5. Status Packet Retransmission

<table>
<thead>
<tr>
<th>Drive</th>
<th>Changer</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCSI COMMAND PACKET</td>
<td>ESC</td>
</tr>
</tbody>
</table>
Figure 6. Unexpected BUS FREE Transition

![Diagram showing CAN transitions]

Figure 7. SCSI Bus Reset

![Diagram showing SCSI command packet with no response]

Figure 8. Slave Unavailable