TO: T10 Membership, ADI Working Group
FROM: Rod Wideman, ADIC; rod.wideman@adic.com
DATE: July 10, 2002
SUBJECT: ADI ADT Frame Format Proposal (document T10/02-274r0)

Introduction

This document is an attempt to describe an actual ADT frame format as a follow-up to Paul Suhler's document, T10/02-233r0 (Frame Format Notes). The goal of this document is to facilitate the discussion regarding the frame definition and lead towards a final definition to be included with the actual ADT specification. Complete transaction descriptions are beyond the scope of this document. Intentional points of discussion are shown in *italics*.

ADI ADT Frame Format

The general layout of the ADI frame is shown in Figure 1. It consists of a Start of Frame character, followed by a frame header, the frame payload, a checksum field, and concludes with an End of Frame character.

SOF	Header	Payload	Checksum	EOF

Figure 1 General ADT Frame Format

Special Characters

To guarantee that the Start of Frame and End of Frame characters are unique to the data stream, special characters are reserved to represent them. These are:

- Start of Frame (SOF) 02h
- End of Frame (EOF) 03h

To ensure that these are unique to the data stream, a technique known as "byte stuffing" is utilized to encode any other occurrence of these values outside of indicating the start of end of a frame. This is accomplished by using an "escape" character to indicate that the very next byte in the stream has been modified from its original value.

- Escape 7Dh

Occurrences of the Escape character value are also encoded. When a data byte having the value of 02h, 03h, or 7Dh is encountered in the data stream, a 7Dh is inserted before it and the data byte itself is modified by an OR operation with 80h, which results in the values 82h, 83h, or FDh respectively to represent it.

Byte stuffing does not affect the actual usable header or payload sizes, as the Escape sequence encoding and decoding only happens as the data is being sent or received. The checksum is also calculated before the encoding occurs or after the decoding occurs.

[Actual values for SOF, EOF, Escape, and encoding are TBD]

[I didn't choose a Hamming distance > 1 because with a limited number of special characters it didn't seem overly necessary]

Frame Header

The frame contains a seven-byte header as shown in Figure 2. The fields are described below.
[Should more of the header be contained within the payload (addresses, LUNs)?]

Frame Type
Frame Number
Source Address
Target Address
Target LUN
Target LUN Task ID
Reserved

Figure 2 Frame Header

Frame Type – This field indicates the type of frame, which really indicates the type of payload. The different frame types are shown in Figure 3.

Type	Value
Command	0
Data	1
Response	2
Acknowledgement	3
Alert	4
Abort	5

Figure 3 Frame Types

[What are the other types needed?]

Frame Number – This is a continuously incrementing number that uniquely identifies a frame from other frames. It ranges in value from 0 to 255, and repeats. Acknowledgement frames return the same frame number as the one they are acknowledging.

Source Address – This is the address of the initiator.

Target Address – This is the address of the target device.
[Do we really need these? In a point-to-point configuration, these seem to be unnecessary. For a multi-drop configuration they would be needed.]

Target LUN – This is the destination Logical Unit Number within the Target. This is used to identify either the SSC device server or the ADC device server over the same serial port.

Target LUN Task ID – This is the specific task within the Logical Unit for which the frame is intended. This can be used to communicate to special diagnostic tasks within the ADC device server for example.

Reserved – A reserved field is included to pad the beginning of the frame to an eight-byte boundary prior to the payload.

Frame Payload

The frame payload contains data defined by the frame type. For a Command type frame, the payload contains an encapsulated SCSI Command Descriptor Block.

(etc., etc.)

[I didn't want to get into the details of the payload yet.]

Frame Checksum

The checksum field is across all the data contained within the header and payload. It excludes the SOF, EOF, and itself. *The checksum algorithm is TBD*.

Complete Frame

The complete ADT frame is shown in Figure 4.

Bit / Byte	7	6	5	4	3	2	1	0
0	SOF							
1	Frame Type							
2	Frame Number							
3	Source Address							
4	Target Address							
5	Target LUN							
6	Target LUN Task ID							
7	Reserved							
8	Payload Length (n-12)							
9								
	[Payload]							
n-2	Checksum							
n-1								
n	EOF							

Figure 4 Complete ADT Frame