OXFORD SEMICONDUCTOR

Proposal to modify isochronous recording format

Andy Green, 30 May 2001

Introduction

The isochronous recording format as currently described in SBP3 revision 1b defines a recording format which includes all the data needed to recreate a 1394 stream at a later time. However, the inclusion of much of this data onto the medium presents a number of problems for applications which directly access this data asynchronously, e.g. video editing. This proposal describes extensions to the proposed format (or preferably an alternative to the proposed format) which will make such applications much simpler.

For a market application which just records the stream and plays it back later, the current scheme is likely to be sufficient. However, for more intelligent A/V solutions which require direct access for editing, trick play etc. it would be far more appropriate to record in a format appropriate to the CONTENT, rather than the 1394 traffic. Every application except record/play of a single stream requires the drive to have some understanding of the content.

Background

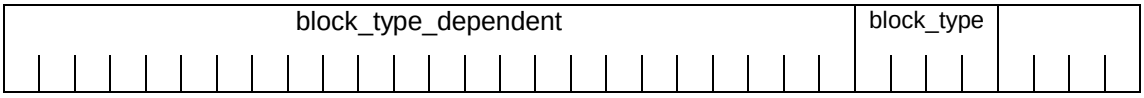
The current format records data to the medium on every isochronous period, which may be an isochronous data packet, an empty packet (just CIP headers) or a cycle mark. In addition to these, section 11.4 describes a cycle mark index scheme to enable software to find cycle marks in hard disk sectors. A number of problems and inefficiencies have been identified with this recording format:-

- **Recorded Cycle Marks are not useful** for two reasons – (1)discontinuities may occur on the bus during record, therefore cycle marks are not ideal for pacing the data. (2) Also, if trick-play is required on playback, the cycle marks will probably need to be discarded / manipulated.
- **Multiple streams.** If stream A is recorded to disk, and then at some later date stream B is recorded to a different track, the cycle marks in each track will almost certainly be completely out of sync. To play them back simultaneously, one or both will have to be transformed by the data anyway, so why store them at all?
- **DV images (normally fixed size) will be stored in variable-length structures.** Depending on the speed of the frame clock in a camcorder, a variable number of null packets will be recorded to the medium. Therefore a video-editing application would need to parse the data, discarding some of the data to reconstruct every frame. In addition to this, to jump forward n frames, every intervening frame must be read and parsed just to find the required frame start.
- **Video files created on a PC/Mac.** If a user creates a video file, and then wishes to store it on an SBP3 A/V disk to be played back autonomously, the host computer must generate all the cycle mark / empty / null packets and insert them into the data stream at an appropriate rate. Similarly, if an editing package replaces a frame in a stream, the edited data will also likely include timestamp discontinuities and rate discontinuities.
- **MPEG2-TS data.** The lowest useful quantity in MPEG data is a transport stream source packet. However in low bit-rate MPEG, the source packets are split into multiple 1394 packets. Similarly for high bit rate, several source packets can be sent in one 1394 packet. Therefore, a single source packet may end up being stored with 1,2,4 or 8 cycle mark packets in the middle of it. In addition to this, if one of the packets fails (e.g. data CRC error), the whole source packet should be discarded, but in the current scheme, some of it may already have been recorded to media. High bit-rate streams will contain multiple source packets in each 1394 block.

Proposal

Extend (or preferably replace) the isochronous recording format as follows:

Isochronous data may be stored in raw format, but where the content is understood by the drive, it shall be stored in blocks appropriate to the media content being recorded. Every block, raw or formatted, shall begin with a header quadlet and be followed by the recorded data. The header quadlet is shown below.



The *block_type* field is used to identify the format of the block, as encoded below

Value	Name	Description
1	DV	DV data block
2	MPEG2-TS	MPEG2-TS source packet
3	AUDIO	61883-6 compliant audio stream blocks
8	CYCLE MARK	Mark time of a cycle start event
A	RAW DATA	Raw data recorded from stream
E	NULL	Null/filler packet
All other values	-	Reserved

……..
Cycle marks, raw data and null packets are as currently defined in SBP3r1b.
……..…

The new block types are described below. Note: In all cases, empty packets are discarded. Only valid data is recorded to the medium. This ensures that data structures have a predictable format for application software to access.

The general intention is to only store the actual content data, with enough out of band information for the drive to play back the data autonomously, and recover from errors while recording.

DV blocks

When the block_type field has a value of 1_{16}, the data stored on the medium shall be of the format shown below:-

data_length								tag	reserved	1	sy
s	c	e						reserved for use by disk implementor			
CIP header 0											
CIP header 1											
480 bytes data											

s = sy valid
c = CIP valid
e = error in data

MPEG2-TS block

When the block_type field has a value of 2_{16}, the data block shall represent a single MPEG2-TS source packet, regardless of the data rate at which the packet was received. For low bit-rate data, the intervening CIP headers are discarded. For high bit-rate data, the CIP headers are replicated for all source packets included in the 1394 packet. The data shall be stored in the format shown below:-

data_length								tag	reserved	2	sy
s	c	e						reserved for use by disk implementor			
CIP header 0											
CIP header 1											
192 bytes source packet											

s = sy valid
c = CIP valid
e = error in data

Note: The *s* and *c* bits are used for recording A/V from non-1394 sources, such as a webcam on the host PC. In this case, the host PC can write the data with the *s* and *c* bits set to zero, and the drive can generate this data when the data is played back isochronously.
The *e* bit signifies that there is some error in the data block, probably because it was not received correctly during recording. On playback the disk will probably not send a packet at all

AUDIO block

Undefined until I or someone else reads and understands the content format!

Why not just record the data?

It would be possible to record just the data payload from any stream, and discard the 1394 headers and CIP headers entirely. This would make it possible for video-editing applications to read the data directly without parsing it to remove headers etc. However such an approach has the following limitations:-

The possibility of errors in the iso transport. When recording, a packet could be missed, received in error, or a cycle start could be missed along with iso data during that period. In this case, the most valid action for the disk to take would be to discard the entire video frame. However, chances are that most of the frame data is OK, so in this proposal the drive will create a block with the error bit set; then the application should be able to implement some more advanced error recovery (displaying most of the frame correctly)

The CIP headers provide the drive with ready information describing the content (SD-DV, HD-DV, MPEG etc), and therefore it can play the track without parsing the data stream to ascertain the format.