PHILIPS

Philips Media

Technical Note #105

Seamless Branching with CD-i Digital Video

Jean-Pierre Abello
Advanced Development
Philips Interactive Media

June 13, 1994

Abstract

This document explains the low-level seamless branching mechanisms that allow a CD-i player with the Digital Video cartridge to play non-linear interactive Digital Video. Seamless branching provides the ability to play MPEG data from different streams or different portions of streams without breaking the delivery and presentation continuity of video or audio. The build-time and run-time requirements of seamless branching are discussed, and the discussion is followed in the appendices by practical methods and examples. The buffering mechanisms necessary to ensure continuous data access from the storage medium are considered part of a different problem domain and are not covered in this paper.

Publication History:
No revisions.

Note that TN numbers 91 through 100 are reserved for conversion of Philips TSA numbers 1 through 10 to the technical note catalog.

To receive Philips Interactive Media technical notes and other publications, or for more information, please contact the person below who is designated for your area.

From Europe and the Middle East:

Hein Zegers
Developer Support
Philips Interactive Media Centre
Maastrichterstraat 63
B-3500 Hasselt
Belgium
Fax: +32 11 242168
Internet ID: hein@pimc.be
CompuServe ID: 73544,1206

From the USA and Asia:

Lucy Lediaev
Technical Support
Philips Media
11050 Santa Monica Boulevard
Los Angeles, CA 90025
USA
Fax: +1 310 477 4953
Internet ID: lucy@aimla.com
CompuServe ID: 72056,1130

Table of Contents

SEAMLESS BRANCHING WITH CD-I DIGITAL VIDEO

Overview

Digital Video

The CD-i Digital Video (DV) system adds to CD-i the ability to play full screen video and audio of near Compact Disc quality, using the MPEG-1 (Motion Picture Expert Group) data compression standard. Newcomers to CD-i DV should start by reading References (6) and (7) before moving on to the more advanced topics addressed in this paper.

Seamless Branching

Seamless branching can be described as a technique that allows CD-i applications to jump from place to place in one or more DV streams while compensating for any discontinuity in the MPEG data before delivering it to the decoder. The goal of this method is to create the illusion of a continuous piece of video and audio that can be forced to take arbitrary paths or branches specified by user interaction or program control.

Applications

Seamless branching can be applied at different levels in the MPEG streams to produce different kinds of non-linear DV effects, such as those that follow:

- Follow a path inside a branching network or create a linear presentation from a database of video clips. This technique can be applied to interactive movies, adventure games, exercise videos, etc. Each branching point connects entire, but separate, MPEG streams at the sequence level. (See the section below entitled "Seamless Branching at the Sequence Level.")
- Insert MPEG streams at arbitrary points inside the current stream. This effect can be used for creating short visual effects (such as explosions) in interactive DV action games. Seamless branching occurs at the Group of Pictures (GOP) level between portions of the main stream and the alternate stream to be inserted. (See the section below entitled "Seamless Branching at the Group of Pictures Level.")
- Play DV loops without the delay incurred when playing from host. This effect can be used to allow looping of DV backgrounds in game-style applications. Seamless branching can be done at the sequence level between the beginning and the end of the entire stream.
- Play DV pictures in reverse order, or at various fast or slow speeds, by skipping or repeating pictures. This method can be used for controlling the direction of a scrolling DV background in a fighting game, for moving at various speeds in a driving game, or for doing random access in a database of DV pictures. Seamless branching can be done at the GOP level between portions of streams the size of only one picture.

For the purpose of the applications above, the low-level mechanisms of seamless branching can be separated into two main categories: seamless branching at the sequence level (usually on entire streams), and seamless

branching at the GOP level (usually on small portions of streams). See the section entitled "Build-Time Requirements" for a complete description.

Limitations of the ISO/IEC 11172 Standard

CD-i DV complies with the MPEG ISO/IEC 11172 international standard[1,2,3], which unfortunately was not designed for doing seamless branching. Limitations are present at every link of the chain, from the encoding tools to the MPEG drivers in the CD-i DV cartridge[4].

One major implication of this is that the CD-i MPEG system decoders cannot be stopped and restarted seamlessly on branching points due to the start-up delay inherent to the ISO/IEC 11172-1 system layer between the first SCR and PTS. Thus, in the case of play from host, video and audio map loops are never seamless. Consequently, with the current implementation of the DV drivers, seamless branching can be performed only while maintaining the decoders in an up and running state with a one-time call to **mv_cdplay()** and **ma_cdplay()**.

Basic Method

The basic idea behind seamless branching is to use the CD-i player's CPU and RAM to reconstruct in real-time a valid ISO/IEC 11172 system stream from an indefinite number of system stream "pieces" before delivering it to the MPEG system decoder.

Not everything, however, can be the done at run time. The bulk of the work can be done only at build-time to prepare the MPEG data so that it is possible for the CD-i application to re-organize it in real-time into a valid ISO/IEC 11172 stream acceptable to the system decoder. In particular, VBV (Video Buffer Verifier) and STD (System Target Decoder) buffer regulation issues critical to the proper functioning of the decoder can be addressed only at build-time as explained in the section entitled "Build-Time Requirements."

If the time and direction of branches can be known at build-time, then the correct system layer time stamps can be generated at the same time; otherwise, the system layer time stamps must be patched at run time, as described in the section entitled "Time Stamp Requirements."

Implementation

The above requirements cannot be implemented without suitable build-time tools and methods and run-time libraries, which are covered in the appendices.

- Appendix A lists all the features required from the encoding, multiplexing and disc building tools, for having a valid build-time pathway for seamless branching.
- Appendix B describes two existing run-time time-stamp patching techniques using either the **iso_patch.l** library, or the DV ROM's **mv_jump()** and **ma_jump()** CD-RTOS functions.
- Appendix C briefly presents the **cdi_dvx** OS-9 utility, which implements all the techniques discussed in this paper. The utility **cdi_dvx** was designed for experimenting and providing example source code for playing linear or non-linear DV on CD-i.

- Appendix D helps troubleshoot some common problems with seamless branching.
- All references are listed in Appendix E.

Build Time Requirements

At run time, a seamless branching CD-i application must be able to concatenate separately multiplexed streams to rebuild a valid system stream that can be decoded continuously by **mv_cdplay()** and **ma_cdplay()**. Therefore, it is necessary to prepare the MPEG data at build-time so that the streams can be easily assembled at run time without bringing the decoders into illegal situations.

If the requirements listed below are not observed, the concatenated stream will not be compliant with the CD-i specification (Green Book), and the decoder may either lose synchronization with the system layer time stamps or run into STD buffer underflow or overflow problems.

All problems can be avoided by observing a few restrictions when encoding and multiplexing the different streams. However, these requirements are different depending on whether seamless branching is applied at the sequence level or at the GOP level, as described below.

Seamless Branching at the Sequence Level

Video sequences always start with a sequence header, and can range in size from a few GOPs up to an entire elementary stream. Seamless branching at this level is designed to work with streams made of one of more sequences, where seamless branching points can occur only at the boundaries of entire streams encoded and multiplexed separately.

This technique allows for a great variety of separately encoded and multiplexed ISO/IEC 11172 streams to be seamlessly concatenated at run time, in order to give the impression of a single, continuous, stream built from the rearrangement of many individual streams.

In some cases, buffer regulation problems may occur at the beginning of the individual streams. These problems can be solved by targeting a VBV state with the MPEG encoder and an STD buffer state with the multiplexer, as explained in the encoding and multiplexing sections below.

Some alterations to the sequence and system layers are necessary to allow for their concatenation into valid ISO/IEC 11172 streams. In particular, to ensure that the resulting concatenated multiplexed stream is a valid ISO/IEC 11172-1 stream, two types of streams with different encoding and multiplexing requirements must be differentiated:

- Introductory stream — This stream forms the beginning of the concatenated multiplexed stream and takes care of initializing the system parameters. In the case of video, it also takes care of handling start-up boundary conditions of an empty STD buffer.
- Body streams — These streams form the rest of the concatenated multiplexed stream and can be used for doing seamless branching at the sequence or GOP level.

They typically must be able to seamlessly continue the decoding state of the previous stream, and they can also be used as end streams.

Encoding Requirements

The MPEG encoders generate the video and audio elementary streams. To allow seamless branching between these entire elementary streams, the video sequence layer[2] and the audio frame layer[3] of each stream must be compatible with its concatenation into a single valid elementary stream that can be continuously fed to the video and audio decoders. For that reason, the introductory and body elementary streams must satisfy the following requirements:

Introductory Stream Video Sequence Layer

- The introductory elementary video stream must begin with a **sequence header** (starting with the field **0x000001b3**), containing parameters for the image size and aspect ratio, the picture and bit rates, the VBV size, the constrained parameters flag, and the quantization matrices.
- Inside the introductory video sequence, **sequence headers** may be repeated to allow decoding to start on selected GOPs, but all sequence header parameters, except the quantization matrices, must remain the same. If this capability is not needed, it is better to avoid repeating sequence headers, because they uselessly consume disc bandwidth and memory.
- If the sequence header parameters, other than the quantization matrices, change after a group of pictures, then a **sequence_end_code** (**0x000001b7**) is required before the new **sequence header**. This is necessary in cases where, for example, the image size changes.
- A **sequence_end_code** can be present only before a sequence header whose parameters (other than the quantization matrices) change, or at the very end of the last stream to be played. In all other cases, the CD-i DV drivers unfortunately misbehave when unnecessary sequence_end_codes are reached, resulting in decoding problems or even system crashes (ISO/IEC 11172-1 is unclear about delays involved in altering the sequence parameters). Thus, all of these **sequence_end_codes** must be padded to 0 (by zeroing the **0x000001b7** fields or by using the **patcheos** tool described in Technical Note #102 by Maris[11]). Most video encoders usually put a **sequence_end_code** in the last 32-bits of the elementary stream.
- Inside the sequence header of the introductory stream, the **VBV size** should be set to the maximum allowed by the video constrained parameters[2], which is **40kbytes**. This will ensure the best encoding quality and the highest

average image size. The maximum VBV size is constrained to a size of about 6Kb smaller than the size of the STD buffer; this allows space for the system layer[1] data.

Body Stream Video Sequence Layer

- The elementary body stream does not need to start with a **sequence header** unless the previous introductory or body stream ends with a **sequence_end_code**, or the sequence parameters (other than the quantization matrices) have changed, in which case a **sequence_end_code** must always precede this **sequence header.**
- The last GOP of a body elementary stream must not be followed by a **sequence_end_code**, unless the sequence parameters (other than the quantization matrices) of the next video sequence change, or unless the play ends there.
- As for the introductory stream, all unnecessary **sequence_end_codes** should be padded out of the elementary video stream, to avoid possible problems with the current CD-i DV cartridges.

VBV Regulation

- At a jump, the VBV end state of a prior elementary video stream and the VBV start state of the next elementary video stream must be identical. Otherwise, buffer regulation errors can be introduced during stream transitions, possibly compounding the problem sufficiently to cause VBV overflow or underflow, and leading to a video decoder crash. To avoid this problem, the video encoder must be able to encode elementary streams ending with an arbitrary VBV end state and starting with an arbitrary VBV start state, identical to the VBV end state of the prior elementary stream.

  WARNING: Failure to achieve identical VBV start and end states during a stream transition will NOT cause any problems with the current implementation of the DV drivers, even though this method clearly is not green. The current speculation is that the DV drivers are gracefully compensating for VBV regulation errors up to a relatively large percentage of the total VBV size, and therefore avoiding fatal accumulation of VBV regulation errors. Discussions about how this is being performed are still under way at the time of this writing. In any case, VBV state errors are illegal and should be avoided.

- The length of the introductory stream must be at least equal to the time it takes to reach the first VBV end state, starting from an initially empty VBV state. This duration depends on the bitrate, the size of the images, and the target VBV end state for the introductory stream.

Audio Frame Layer

- An **audio sequence** is composed of an integer number of audio frames, composed of an integer number of samples (384 in layer I, 1152 in layers II and III). This means that the granularity of MPEG audio is the **audio frame.**
- Because of the difference between the temporal granularity of audio frames and video pictures, an audio stream will not have exactly the same

temporal length as the associated video stream of supposedly identical length. Therefore, on each successive seamless jump, if nothing is done to compensate for the difference, the audio may drift increasingly out of synchronization with the video. This is a production issue that has to be taken into account when encoding audio, as explained in Appendix A.

- Unlike video pictures, audio frames are presented instantaneously, and there are no decoder buffer regulation problems associated with audio frames.

Multiplexing Requirements

The role of the multiplexer is to add the system layer[1] to combine the elementary video and audio streams into a single, synchronized, ISO/IEC 11172-1 stream. The system layer consists of a pack layer and a packet layer. To allow the run-time concatenation of multiplexed introductory and body streams into a single valid ISO/IEC 11172-1 system stream, these pack and packet layers must comply with the following requirements:

Introductory Stream System Layer

- The pack size must be **2324 bytes** for video packs, and **2304 bytes** for audio packs. This requirement is specific to CD-i, and applies to both introductory and body multiplexed streams.
- The first pack of the multiplexed introductory stream must begin with a **system header** (starting with the field **0x000001BB**).
- The **iso_11172_end_code** (**0x000001B9**) marks the end of the ISO/IEC 11172-1 stream and hence cannot appear at the end of the introductory stream. It usually forms the last 32-bytes of the multiplexed stream and must either be removed by the multiplexer or padded to 0, as described in Appendix A.
- The **STD buffer size** must be **46kbytes**, as specified in ISO/IEC 11172-1[1] for a constrained parameter system stream. The initial STD buffer state for the multiplexed introductory stream must be empty to correspond to the initial VBV state for the elementary introductory stream.

Body Stream System Layer

- The system header at the beginning of a multiplexed body stream must be identical to the system header of the multiplexed introductory stream. However, it is preferable that a multiplexed body stream not start with a **system header** (including the **STD sub-header**), because there is already at a system header at the beginning of the multiplexed introductory stream.
- The last pack of a multiplexed body stream must not end with an **iso_11172_end_code**, unless the play ends there.

  Note: If an iso_11172_end_code is present at the end of an intermediate body stream, as in the case of loops, the decoder will not be able to continue decoding.

- The **stream ID** of the multiplexed body stream must be the same as the ID for the multiplexed introductory stream. If they are different, they can eventually be patched at run time by the CD-i application.

STD Buffer Regulation

- To prevent STD buffer underflow or overflow during a jump due to the accumulation of buffer regulation errors, the multiplexer must be able to generate the same STD buffer state at the end of a prior multiplexed video stream, and at the beginning of the next multiplexed video stream. In the case of loops, or to gain random access to the body streams, these STD end and start states should all be exactly identical.

  WARNING: As with the VBV states, failure to achieve identical STD start and end states during a jump does not cause problems with the current implementation of the DV drivers. However, this method is clearly not compliant with the Green Book and should be avoided.

Seamless Branching at the GOP Level.

Since a video sequence is made of one or more GOPs, this section deals with the extension of the seamless branching technique between sub-sections of MPEG body streams, which can be very short, even the size of a single picture. The requirements of the section "Seamless Branching at the Sequence Level" above still apply and are extended to the GOP level here. At this new level, the encoding and buffer regulation granularity is lower and require more flexible MPEG encoders and multiplexers. In particular, it is necessary to be able to control the characteristics of individual groups of pictures (GOPs) within sequences. If such feature are not available, then it is always possible to do seamless branching at the sequence level on shorter streams instead and to concatenate them as explained in Appendix A.

To do seamless branching at the GOP level, the multiplexed body streams must be compatible with the re-arrangement of some their sub-sections into a new valid multiplexed stream. This goal can be reached only by following the additional encoding and multiplexing requirements described below.

Additional Encoding Requirements

Group of Pictures Layer

- The video sequence layer[2] of the elementary body streams must still comply with the "Encoding Requirements" presented above.
- Seamless branching at the GOP level can be done only towards portions of streams starting with a **closed GOP with no broken link**. The reason for this results from the fact that GOPs begin, in display order, with an I-picture or a B-picture, and end with an I-picture or a P-picture. Closed GOPs, identified by a flag in the GOP header, start with I-pictures or B-pictures and use backward-only motion compensation. These closed GOPs do not make any reference to the previous group of pictures; nor do they have the broken link bit set. If the GOP has a broken link and starts with B-pictures with forward motion compensation, the B-pictures cannot be correctly decoded. Therefore, only a closed GOP with no broken link is completely independent from the previous GOP.

Audio Frame Layer

- The audio frame layer[3] of the elementary body streams must still comply with the requirements of the section "Encoding Requirements."

- Because of the difference in temporal granularity between audio frames and video pictures, audio frames are unlikely to be time-aligned with group-of-pictures boundaries. Therefore, it is necessary to take extra steps at production time to avoid the loss of audio/video synchronization on seamless jumps. For a discussion of possible solutions for pre-defined jump orders, see Appendix A.

VBV Regulation

- The VBV requirements presented in the section "Encoding Requirements" above still apply here.
- The VBV end state after the last GOP of a portion of an elementary video stream and the VBV start state at the beginning of the first GOP of the next portion of stream must be identical; otherwise, VBV regulation errors can occur and be compounded during successive seamless jumps.

Additional Multiplexing Requirements

The resulting multiplexed stream, made up of sub-sections of multiplexed streams and possibly of the size of a group of pictures, which is delivered to the decoders must still comply with the ISO/IEC 11172-1 specifications.

This is possible, however, only by following the additional requirements laid out below:

Group of Pictures Layer

- Because video or audio MPEG data is transferred to the system decoder in packs the size of a CD-i video or audio sector, the exit GOP must be padded to the end of the last pack of the prior stream and the entry GOPs must be aligned to the beginning of the first pack of the next stream.

STD buffer regulation

- The requirements presented in section "Encoding Requirements" still apply here.
- As with the VBV state, the STD buffer states at the end of the last GOP of the prior portion of video stream and at the beginning of the first GOP of the next portion of video stream must be identical; otherwise, STD buffer regulation errors can occur and accumulate during successive seamless jumps.

Run-Time Requirements

The above build-time requirements ensure that multiplexed streams that can be concatenated or re-organized at run time can be produced. If these requirements are followed, then the run-time system has only to patch the time stamps to re-create a valid ISO/IEC 11172-1 stream.

The system decoder uses these time stamps for buffer management, time identification, and playback synchronization; the time stamps must be contiguous throughout the life of the MPEG play.

Because a seamless branching application jumps between multiplexed streams or portions of multiplexed streams, these time stamps must, in general, be patched at run time before being sent to the decoder It is sometimes possible to prepare the multiplexed streams so that time stamps are continuous at build time, but doing this is likely to greatly reduce the choice of possible run-time paths. See Appendix B for a description of scheduling techniques for patching time stamps in real-time with the iso_patch.l library and for an explanation of delta value calculations.

Time Stamps Requirements

The system layer time stamps (SCR, PTS, DTS) are divided between the pack layer and the packet layer for the purposes described below. They can be visualized with the **muxInfo** utility (15).

Pack Layer Time Stamps

Every video or audio pack header contains a System Clock Reference (SCR) time stamp specifying a byte arrival schedule for the STD buffer.

The SCR is a 33-bit number, in units of 90kHz, that indicates the intended time of arrival of the last byte of the **system_clock_reference** field at the input of the system target decoder[1].

The SCR must be present at intervals not exceeding 0.7 second as measured by the values contained in the SCR fields[1]. In CD-i, this sets a limit for the maximum possible number of skipped CDFM sectors between consecutive MPEG sectors, assuming just-in-time delivery.

Packet Layer Time Stamps

Some video or audio packet headers can contain a Presentation Time Stamp (PTS) and a Decoding Time Stamp (DTS) that specify a byte removal schedule from the STD buffer.

The PTS is a 33-bit number, in units of 90kHz, that indicates the intended time of presentation in the system target decoder of the first video picture or audio frame commencing in that packet.

The DTS is a 33-bit number, in units of 90kHz, that indicates the intended time of decoding of the first video picture commencing in that packet. The DTS can be present only in the packet header of I-pictures and P-pictures, which also contain a PTS and has different decoding times and presentation times. B-pictures and audio frames are always presented as soon as they are decoded, and, therefore, do not have a DTS.

The PTS and DTS fields must be encoded in the packets where an access unit starts, but they are not necessarily encoded for each picture or audio frame and can occur with intervals not exceeding 0.7 second[1]. For those presentation units for which a PTS is not encoded, the decoder can approximate the correct value from the most recent PTS value and an incremental value.

Time Stamp Discontinuities

The ISO/IEC 11172-1 specifications[1] do not allow the values in the time stamps to increase by more than 0.7 second between two consecutive pack SCRs of the same system-layer stream. Furthermore, to maintain a proper video and audio frame presentation rate, the PTS increment from packet to packet must correspond exactly to the number of frames passed since the previous PTS. Otherwise, there could be significant variations in the video picture rate, or the audio could crash.

A time stamp discontinuity occurs when the PTS reaches a value inconsistent with the presentation rate and the number of elapsed frames, or when the SCR does not increase within the above constraints.

Therefore, it is necessary to patch the SCR, PTS and DTS at run time to maintain the expected time-stamp continuity in the concatenated stream delivered to the system decoder and to obtain a normal presentation rate.

Delta Value

Since the SCR, PTS and DTS are all relative to the same System Time Clock (STC), they can be incremented by the same delta value for a particular system stream when a time-stamp discontinuity is reached. The delta value measures the temporal gap of the time stamp discontinuity and is equal to the error in the PTS values. For that reason, the delta value must be calculated from the PTS's in the system layer data, either at build-time or at run time.

If the multiplexer encodes a PTS for every presentation unit, then it is possible to calculate the delta value as the difference between the value of the PTS of the first presentation unit after the discontinuity and the value that the PTS should have had according to the presentation rate alone (picture rate for video, frame rate for audio). See Appendix B for a description on how to do this with the **iso_patch.l** library.

If the multiplexer skips encoding PTS's for some frames in gaps of up to 0.7 second as allowed in ISO/IEC 11172-1[1], then the delta value can be calculated as the difference between the value of the first PTS following the discontinuity and the value it should have had, based on knowledge of the presentation rate and the number of presentation units passed since the last PTS. This solution requires more stream parsing, and is, in general, more difficult to implement. However, since seamless branching points are most likely to occur on access units in CD-i, this case is expected to happen infrequently.

> *Note: Audio and video delta values are likely to be different in most cases because of the frame granularity problems mentioned in sections earlier and explained in Appendix A-2.*

If the multiplexer outputs information about the value of the PTS of the presentation unit following the last frame of a multiplexed stream, then the

delta value can be calculated as the difference between these PTS values and what they should have been. See Appendix A for a description of multiplexer features.

Green Book Bug

It is also important to be aware that it is incorrect to calculate the delta value as a difference in the SCRs, as specified in sections 8.2.3.1.12 and 8.2.4.17 of the DV extension to the Green Book[4]. The SCRs are used for scheduling data arrival (based on the storage medium), rather than scheduling data presentation (based on the presentation rate), and are dependent on the data delivery strategy adopted by the multiplexer, rather than the actual presentation of MPEG frames.

Patching Time Stamps

Using mv_jump() and ma_jump()

The DV cartridge ROMs contain two functions, **mv_jump()** for video and **ma_jump()** for audio, to patch the system layer time stamps to their correct values at run time.

The functions **mv_jump()** and **ma_jump()** have the following advantages:

- They are non-destructive to the system layer data. The time-stamp fields in the MPEG data in memory is not modified by **mv_jump()** and **ma_jump()**.
- They run in system state, therefore consuming a minimal amount of CPU.
- They are in ROM, therefore reducing the application code size.

They unfortunately also have the following problems:

- In the early DV cartridges, both **mv_jump()** and **ma_jump()** contain bugs. (See Appendix B for a detailed discussion.)
- The technique for calculating the delta value is incorrectly documented in the DV extension to the Green Book[4]. (See the section entitled "Time Stamp Requirements" above).

Using iso_patch.l

Since the above problems have to be taken into account in all current and future titles using seamless branching, it is recommended that you use the **iso_patch.l** library of time-stamp patching functions instead of the ROM functions discussed in the preceding section. These can be obtained from PIMA Developer's Services, and are documented in Appendix B.

The **iso_patch.l** functions have the following advantages:

- They have been thoroughly tested and have been proven to work reliably.
- They provide complete control over the values of the different time stamps in the system layer (SCR, PTS, DTS).
- They provide a run-time mechanism for calculating the delta values from the PTS values, as described in Appendix B.
- The CPU overhead remains low.
- The assembly language source code is available.

These functions also have some minor disadvantages:

- They are destructive to the system layer time-stamp data. After patching, the time-stamp data fields differ from their original values in the multiplexed stream.
- They require more complex buffering mechanisms. The decoder requires dedicated PCLs controlled by the application; the application decides when to set the **PCL_Ctrl** bit to 1. If data is to be loaded from disc, then a different set of PCLs must be used for CDFM.
- The memory consumption and the signal overhead are higher than for **mv_jump()** and **ma_jump()**.
- The application code is more complex.

However, the routines in the **iso_patch.l** library currently represent the only robust solution for patching time stamps to perform seamless branching. Therefore, it is recommended that you use the **iso_patch.l** library instead of the calls **mv_jump()** and **ma_jump()**.

Summary

Although seamless branching is not a feature designed into ISO/IEC 11172, it is still possible to achieve that effect by observing strict build-time and run-time restrictions; these restrictions allow the delivery of a valid contiguous ISO/IEC 11172 data stream to the decoder across branching points. They make possible various interactive digital video techniques that can be used to create a wide variety of non-linear digital video effects.

The build-time requirements emphasize the encoding of the elementary streams, and, most importantly, their proper multiplexing into system streams using special multiplexers with features especially designed to allow seamless branching.

The run-time requirements focus on patching the system layer time stamps to completely conform to ISO/IEC 11172-1 just before delivery to the system layer decoder.

The practical aspects of implementation of seamless jumps in real-life projects is further explained in the following appendices.

Appendix A: Build-Time Tools

Encoders

Video encoding and audio encoding are generally done separately and, for that reason, are covered separately in the following sections.

Video Encoders

The purpose of this section is to list the features required in a video encoder to allow seamless branching on ISO/IEC 11172 video streams.

Encoder features needed for doing seamless branching at the sequence level

The elementary stream video encoder must provide the following features to allow seamless branching at the sequence level:

- Ability to encode for Constrained Parameters Streams, with a VBV size of 40kbytes.
- Ability to start encoding from an arbitrary initial VBV state.
- Ability to target a VBV end state after the last picture is removed from the VBV buffer.
- Ability to set the placement of video sequences (if more than one) in the elementary video stream.
- Ability to disable the generation of the **sequence_end_code** at the end of the elementary video stream, or wherever it is not necessary.

> *Note: The ANDROX encoder and the IMS software encoder provide the first four features. The last feature can be worked around by padding sequence_end_codes to 0 after the encoding is finished.*

Additional encoder features needed for seamless branching at the GOP level

The elementary stream video encoder must provide all the features listed above plus the additional features listed below:

- Ability to set the position and size of individual GOPs in the elementary video stream.
- Ability to target a VBV state after the last picture of an individual GOP is removed from the VBV buffer.

> *Note: The ANDROX encoder (but not the IMS software encoder) provides both of these features.*

Audio Encoding

The purpose of this section is not to list features required from MPEG audio encoders, but to point out a granularity limitation of MPEG audio that can seriously complicate the production phase of synchronous video and audio MPEG assets to be used with seamless branching.

Limitations

In MPEG audio, the duration of audio frames is constant for a given sampling rate. The number of frames per coded sequence must be an integral number; thus, the total length of the coded sequence has a time granularity of one audio frame period. However, in the case of video, the time granularity is the video frame period, which can be 1/23.976, 1/24, 1/25, 1/29.97 or 1/30. If the video and audio frame periods do not have a relatively small common multiplier, then the video and audio sequences will have different temporal lengths.

In the case of CD-i, MPEG audio is sampled at 44.1kHz and is encoded into an audio layer II with 1152 samples per frame. The table below compares the audio and video frame durations and the corresponding smallest common multiplier, which gives the smallest common duration for coding audio and video:

Source	Audio	23.976 Hz	24 Hz	25 Hz (PAL)	29.97 Hz (NTSC)	30 Hz
Frame length	1,152/44,100	1,001/24,000	1/24	1/25	1,001/30,000	1/30
Prime factors	$2^5/(3^2 \times 49)$	$1,001/(2^6 \times 3 \times 5^3)$	$1/(2^3 \times 3)$	$1/5^2$	$1001/(2^4 \times 3 \times 5^4)$	$1/(2 \times 3 \times 5)$

Resulting minimal common lengths of audio and video coded streams:

Source	23.976 Hz	24 Hz	25 Hz (PAL)	29.97 Hz (NTSC)	30 Hz
In audio frames	1,506,785,280	940800	1568	1,883,481,600	11760
In video frames:	943,718,400	589824	1024	147,456,000	9216
In seconds:	39,360,921.6	24576	40.96	4,920,1152	307.2

All other lengths must be multiples of the figures in the table above.

In most cases, it is clearly not possible to encode short video and audio sequences to the same temporal length. The best case is with 25Hz video (PAL), which allows synchronization of sequences that are multiples of 40.96 seconds, and the worst case is with 29.97Hz (NTSC) in which sequences shorter than 49,201,152 seconds (more than a 1 year and 194 days) cannot be synchronized.

The maximum possible error between the length of the video and the audio coded streams is one audio frame (1152/44100 =.0261 second).

Since the audio encoder has to encode in an integral number of audio frames, an encoded audio stream is likely to last longer than an encoded video stream of identical length. This means that in the case of synchronous loops, after an average of 2/(.0261)=76.6 loops (or in the worst case after 1/.0261=38.3 loops), the decoded audio will lag by one full second behind the decoded video.

Asset Production

This problem is inherent in the MPEG standard, but can be minimized by implementing audio partitioning strategies at production time.

If, in the case of loops, the number of loops is small, audio sequence lengths can be chosen to keep their lengths as close as possible to multiples of the numbers of audio frames listed above in order to reduce the video/audio drift as much as possible. If a large number of loops is to be played, then one solution can be to skip the last audio frame when the drift has reached the duration of a whole frame.

In the case of seamless branching at the sequence level between streams, the lengths of the audio streams should be corrected with knowledge of the number of audio samples in the preceding audio stream that extend beyond the end of the preceding video stream.

Finally, in the case of seamless branching at the GOP level, in which the lengths of the audio blocks may be shorter or longer than the lengths of the video GOPs by up to one-half of an audio frame, it may be necessary to skip play of an audio frame when the drift is compounded up to the duration of a full audio frame.

The above two cases are difficult to implement along with the arbitrary ordering of individual streams.

Multiplexers

The multiplexer "pink" was not designed for doing seamless branching, and it has many limitations that make it unusable for non-linear applications. An extension of "pink" had been proposed by Pieter de Visser[9,10] but, unfortunately, the extension has never been implemented. This section lists the features that must be provided by the multiplexer for being able to do seamless branching.

Multiplexer features seamless branching at the sequence level

The multiplexer must provide the following features for being able to do seamless branching at the sequence level:

- Ability to omit the **iso_11172_end_code** at the end of a multiplexed stream. This is absolutely necessary in both introductory streams and body streams, but can be ignored for end streams.
- Ability to multiplex video streams with an arbitrary **STD buffer start state**. This feature is essential to ensure safe STD buffer regulation at the beginning of body streams, which never start with an empty STD buffer state, as explained in the section entitled "Additional Multiplexing Requirements" above.
- Ability to **target an STD buffer end state** at the end of a multiplexed video stream. This feature, used along with the above feature, can ensure the continuity of STD buffer states across seamless branches, and avoid the possible accumulation of STD buffer regulation errors leading to buffer underflow problems.

The multiplexer can also provide the features below for additional convenience:

- Ability to omit the **system header** and **STD sub-header** in a multiplexed body stream. This feature is useful for reducing the bandwidth usage of body streams.

- Ability to start multiplexing video and audio streams at arbitrary values for the SCR, PTS and DTS time stamps. This feature is very useful for avoiding unnecessary patching of the time stamp after some transitions. It also allows continuity in the system time stamps for possible concatenation of the multiplexed streams at build-time.

Other multiplexer features needed for seamless branching at the GOP level

The multiplexer must provide all the features listed above, plus the additional features listed below, to be able to handle seamless branching at the GOP level:

- Ability to **align individual GOPs** so that they start at the beginning of the first packet in a pack.
- Ability to **align individual audio frames** so that they start at the beginning of the first packet in a pack.
- Ability to **target an STD buffer state at the end of individual GOPs or audio frames**. This feature is likely to be used only with body streams.

Computing The Delta Values

As described in the section entitled "Time Stamp Requirements," the delta value should be calculated as the difference between the PTS of the first presentation unit of the next stream and the value it would have had if no time-stamp discontinuity had occurred during the transition.

The value of the PTS of the first presentation unit of the next stream can be calculated incrementally from the PTS of the last presentation unit of the current stream, if the presentation rate is known. For example, in the case of video, if the presentation rate is 25 pictures per second, then the next PTS will be the last PTS plus 90000*1/25.

Sometimes the multiplexer has a feature that outputs the value of the PTS of the next presentation unit. This value can be used at run time to calculate the delta value as described above.

Analyzing and Editing Multiplexed Streams.

Verification of Multiplexed Streams

A parsing utility, called **muxInfo**, has been developed for examining the contents of multiplexed system streams of the above types. Among its features are the ability to display header parameters, including time stamps, and to number packs and packet.

If available, an MPEG verifier should also be run on the multiplexed stream to ensure conformity with the ISO/IEC 11172 standard across and beyond seamless jumps.

Concatenation of Multiplexed Streams

To allow seamless branching, it must be easy to concatenate the introductory and body system streams at run time . However, it is sometimes desirable to be able to concatenate them at build-time into a single system stream with contiguous time stamps that can be built into a real-time file. See the section below entitled "Real-Time Files" for more details.

The advantages of this approach are a simpler disc image structure and the ability to play one of the paths without having to perform disk seeks, nor having to patch the time stamps at run time. In addition to this, the concatenated stream can feature the same pack-aligned structure as a multiplexed stream created for doing seamless branching at the GOP level, as described above.

Real-Time Files

Two kinds of MPEG files are desirable in the real-time file builder. They follow:

- Normal real-time MPEG files that contain video and audio sectors laid out in real-time format on the disc. These files can be multiplexed with other kinds of sectors so that the MPEG sectors are loaded when needed in the decoder buffers.
- Non real-time MPEG files that contain video and audio sectors laid out in packed file format on the disc. These files can also be multiplexed with other kinds of sectors, but are really meant to be loaded as fast as possible by the application.

The normal real-time MPEG files can be used for playing linear DV, while the non-real-time MPEG files can be useful for doing seamless branching from disc, where a bridge sequence must be loaded in memory while still playing the current clip, in order to be able cover up the next disc seek.

Below are example master scripts for generating master MPEG files:

```
record real_time mpeg                    ! Real-Time MPEG
in channel 0                             ! In channel 0
   from "system_file.mps"                ! Multiplexed input
   ep_list off                           ! No entry point lists
   at 0      ! Start at sector 0


record non_real_time mpeg                ! Non-Real-Time MPEG
in channel 0                             ! In channel 0
   from "system_file.mps"                ! Multiplexed input
   ep_list off                           ! No entry point lists
   at 0                                  ! Start at sector 0
```

These real-time files can be used with various disc builders, such as "master', or even "bd," in conjunction with the real-time file format converter "mpegtorfd".

SSome of these issues were covered in a note on programming the FMV system[9].

Appendix B: Patching Time Stamps

mv_jump() and ma_jump()

The ROMs in the Philips DV cartridges contain two functions, **mv_jump()** and **ma_jump()**, for patching the system layer time stamps of multiplexed video and audio streams.

These functions were designed to be as fast and as transparent to the application as possible, which only needs to provide a 22.5 kHz delta value.

Unfortunately, the early cartridges that were shipped contained flawed versions of **mv_jump()** and **ma_jump()**, resulting in the following problems:

- **mv_jump()** does not patch the PTS when a DTS is not present in the first packet of a video picture. This is the case for all B-pictures, which are displayed as soon as they are decoded. The consequence of this omission is incoherent SCRs and PTSs on B-pictures. This causes the CD-i decoder to play these video stream at an erratic picture rate.
- The sign of the delta parameter expected by **mv_jump()** is the opposite of the sign specified in the Green Book.
- When the 33rd bit in the values of the time stamps is set by **mv_jump()** or **ma_jump()** (this occurs after playing for a maximum of 13 hours 15 minutes and 21.86 seconds), the decoder crashes. This is caused by the fact that, in converting 90kHz values for the time stamps into 22.5kHz values, arithmetic shift right instructions instead of logical shift right instructions, are used in **fmvdrv** and **madriv**. This has the effect of leaving the leftmost bit set on positive values.
- In addition, the method for calculating the delta value is incorrectly documented in the Full Motion Video extension to the Green Book. The delta value must calculated from the PTSs, as explained in section above, and not from the SCRs.

We do not recommend use of **mv_jump()** and **ma_jump()**, until they are fixed. This is especially the case with **mv_jump()**, which does not work at all.

iso_patch.l Library

To provide an alternative to using **mv_jump()** and **ma_jump()**, as described above, a library of functions, **iso_patch.l**, has been developed for manipulating and patching the system layer time stamps.

This library, along with documentation and example code, is available from PIMA Developer Services.

The functions of **iso_patch.l** and their scheduling in a real-time environment are explained below.

iso_patch.l Functions

The iso_patch.l library functions treat all time stamps as unsigned integers (in units of 45kHz) consisting of the higher 32-bits of the 33-bit ISO/IEC 11172-1 values. Since the 90000Hz system layer time stamps have an accuracy of 4Hz, dropping the least significant bit of the 33-bit values does not hurt the precision of the time stamps and has the advantages of simplicity and speed.

iso_patch.l provides different sets of functions for different levels of control of the system layer time stamps, as specified below:

Higher Level User Functions

```
- void iso_patch_video_sector (unsigned char *pack_ptr, unsigned long delta)
- void iso_patch_audio_sector (unsigned char *pack_ptr, unsigned long delta)
```

The routines iso_patch_video_sector() and iso_patch_video_sector() are fast routines for patching all the time stamps contained in a pack and its packets. The pointer pack_ptr points to the pack, and delta is the 32-bit delta value to be added to the SCRs and PTSs and DTSs. All packets inside the pack, regardless of their stream IDs, are patched with that same delta value. The assembly language source code for these functions have been optimized for speed, and are available for modification or further improvements.

The functions iso_patch_video_sector() and iso_patch_audio_sector() can be considered as a generic replacement for mv_jump() and ma_jump().

Utility Functions

```
-unsigned char *iso_find_packet (unsigned char *pack_ptr)
```

The function iso_find_packet() returns a pointer to the next valid packet inside a pack, skipping system headers if necessary. If no packet is found then -1 is returned.

```
-void iso_patch_scr (unsigned char *pack_ptr, unsigned long delta);
-void iso_patch_pts (unsigned char *packet_ptr, unsigned long delta);
-void iso_patch_dts (unsigned char *packet_ptr, unsigned long delta);
```

The above functions, iso_patch_scr(), iso_patch_pts() and iso_patch_dts(), patch the SCR, PTS and DTS time stamps with a 45kHz delta value. These functions extract the time stamps, add the delta value, and re-insert the time stamps back into the pack or packet structure.

Lower Level Functions

```
-unsigned long iso_extract_scr (unsigned char *pack_ptr)
-unsigned long iso_extract_pts (unsigned char *packet_ptr)
-unsigned long iso_extract_dts (unsigned char *packet_ptr);
```

The above functions extract the most significant 32 bits of the SCR, PTS or DTS time stamps from their pack or packet headers.

The lower level function iso_extract_pts() may be used for computing the delta value at run time from the PTS's in different packets, as explained further below.

```
-void iso_inject_scr (unsigned long tc, unsigned char *pack_ptr)
-void iso_inject_pts (unsigned long tc, unsigned char *packet_ptr)
-void iso_inject_dts (unsigned long tc, unsigned char *packet_ptr)
```

The above functions insert SCR, PTS or DTS time stamp values into pack or packet headers.

Run-Time Scheduling Methods

This section assumes that the reader is familiar with CDFM and that the multiplexed MPEG data is loaded in memory via an independent set of PCLs.

A DV decoder needs to have its own dedicated sets of PCLs controlled by the application. The buffer-full bit in the PCL_Ctrl field must be set by the application to make the sector available to decoder only after the MPEG time stamps have been properly patched.

This time-stamp patching and PCL controlling process can be done at different points in the DV play, based on the type of application, as follows:

With High Signal Overhead

If signal overhead is not a major preoccupation, then the easiest scheduling technique is to patch the time stamps and set the buffer-full bit in the PCL_Ctrl field as soon as a sector is read by the decoder. This implies on time receiving and processing of one signal per sector read by the decoder.

Two different PCL approaches are valid for doing this:

- With one PCL: When the decoder has finished reading in a sector, the PCL_Sig signal from that PCL is sent to the application and the buffer-full bit in the PCL_Ctrl field is reset. At this time, the application can modify the PCL_Buf pointer in the unique PCL to point to the next sector in memory and patch the time stamps contained in that new sector. When data in that sector is ready, the application sets the buffer full bit in the PCL_Ctrl field to make it available to the decoder.
- With two or more PCLs: On reception of the PCL_Sig signal, the application patches the time stamps contained in the sector pointed to by the next PCL, and sets the buffer-full bit in the PCL_Ctrl field of the next PCL to make it available to the decoder.

With Low Signal Overhead

If signal overhead is a major concern, then it becomes desirable to have as many sectors as possible loaded into memory. On reception of the first PCL_Sig signal, as many sectors as possible should be patched all together, and all the buffer full bits in the PCL_Ctrl fields of the PCL pointing to all of these sectors should be set simultaneously. The decoder then grabs them at a rate based on the SCR values in the pack header so there is no risk of getting STD buffer overflow by using this technique. However, there is a danger of underflow if the processing overhead prevents the timely completion of the time-stamp patching.

Computing The Delta Values

With both of the above scheduling techniques, if a PTS is encoded for each video picture, it is possible to determine at run time if the delta value has changed. This determination can be made by reading the PTS of the next picture with iso_extract_pts() and comparing it with the PTS of the current picture, after both have been patched with the current delta value.

If the difference is inconsistent with the presentation rate of the stream being decoded, then there is a discontinuity in the time stamps, and the delta value must be updated with the new difference.

For example, the frame presentation rate for PAL video is 25 pictures per second; which corresponds to 1800 units of 45kHz per picture. Therefore, if there is no video time-stamp discontinuity between two pictures, the PTS should increase by 1800 units of 45kHz between successive video pictures.

When a discontinuity is detected, the delta value must be increased by the difference between the expected next PTS value and the actual next PTS value. This way, a contiguous stream of time stamps will be maintained across and beyond the discontinuity.

Appendix C: Experimentation with cdi_dvx

Overview

The **cdi_dvx 3.0** utility has been designed to meet the following goals:

- To provide a command-line tool for playing Digital Video linear real-time files with full control of the different DV parameters.
- To provide a tool for testing seamless branching at the sequence level or at the GOP level with DV memory loops. The introductory stream and any number of body streams are first loaded in memory and then looped seamlessly.
- To provide a tool for testing DV flying mattes (covered in PIM TN#103), in conjunction with linear DV or seamless branching DV memory loops.
- To provide example source code for learning the basics of DV play and the more advanced features of seamless branching.
- The program **cdi_dvx**, including its source code, the **iso_patch.l** library and sample assets are available from PIMA Developer Services.

> *NOTE: **cdi_dvx** is distributed on an "as is" basis by PIMA Developer's Services and will not receive any maintenance or support. I will, however, try to answer most questions that I will receive at **jp@aimla.com**.*

Seamless Branching Techniques Implemented

Encoding

cdi_dvx 3.0 can perform seamless branching at the sequence level or at the GOP level, which both require special encoding capabilities.

See the section in the body of this document entitled "Encoding Requirements" for details.

Multiplexing

The system streams must be created with a multiplexer of the type described in Appendix A.

Run-Time

Both the introductory and the body system streams are loaded into memory. The decoders are started with **mv_cdplay()** and **ma_cdplay()**. Only one PCL needs to be allocated for decoding video or audio.

Each time the decoder reads a sector from these PCLs, a **PCL_Sig** signal is received, and the time stamps in the next sector are patched using the **iso_patch.l** library functions. The **PCL_Buf** field in the PCL is then updated to point to the next sector, and the buffer full bit in the **PCL_Ctrl** field is set to let the system decoder grab it.

From the point of view of signals, **cdi_dvx** implements a version of the more signal intensive method described in Appendix B.

With iso_patch.l, cdi_dvx implements a memory loop using a destructive time-stamp patching method, and the delta values do not need to change from iteration to iteration.

If mv_jump() and ma_jump() are used, the MPEG data loaded in memory is not modified by the time-stamp patching process, and the delta values have to increase from iteration to iteration as time flows.

Command Line Options

Here is an edited printout of the command-line help page for cdi_dvx 3.0:

```
Digital Video Explorer v3.0 by Jean-Pierre Abello - July 14, 1994
Copyright 1994, Philips Interactive Media
Usage: cdi_dvx <mpeg_files> [linear opts] [non-linear opts] [flying matte opts]
<mpeg_files>:
    -r linear.rtf                          To play a linear RTF straight from disc
    -r intro.rtf body1.rtf [.. bodyN.rtf]  To seamlessly loop N body RTFs from mem
[linear DV options]:
    -c cdfm_channel                        Cdfm channel for loading MPEG data
    -s sector_offset                       Cdfm sector offset to lseek() to
    -v video_stream_number                 Stream number for MPEG video
    -a audio_stream_number                 Stream number for MPEG audio
    -b vid_byte_offset aud_byte_offset     Video and audio entry point offsets
    -p vid_pcts_sectors aud_pcts_sectors   Video and audio pcts and sectors (>1)
    -m video_mem_color audio_mem_color     A = PlaneA, B = PlaneB, S = SysRam
    -o x_org y_org                         DV UCM x and y origin offsets
    -M scroll_flag                         Thumbstick moves screen using mv_pos()
    -u                                     Unsynchronized audio/video decoding
    -T [sampling_window_in_pictures]       Display frame rate (window optional)
    -X                                     Freeze on the last frame
    -V[V]                                  Verbosity levels (normal or extreme)
[non-linear DV options]:
    -P vid_mem_sectors aud_mem_sectors     Nbr of video and audio memory sectors
    -dv vintro.std vbody1.std..vbodyN.std  Video state .std sink files
    -da aintro.std abody1.std..abodyN.std  Audio state .std sink files
    -DV video_delta1 .. video_deltaN       Video delta values in units of 45kHz
    -DA audio_delta1 .. audio_deltaN       Audio delta values in units of 45kHz
    -S                                     Notify when streams and loops begin
    -j                                     Use mv/s_jump() following the green book
    -J                                     Use mv/s_jump() w/ opposite delta sign
    -O body1.offsets .. bodyN.offsets      Video offsets files (for counting GOPS)
    -g total_GOPs_in_body_streams          Number of GOPs in all body streams
    -G pics_per_GOP [sectors_per_GOP]      Number of pictures and sectors per GOP
[flying mattes options]:
    -f mattes.ops width height frames      Matte opcodefile, width, height, frames
    -i planeB_PAL_image.dyuv               Background DYUV image for the mattes
```

Documentation

A UNIX man page is is distributed with the program by Developer Services.

Verification

The best way to verify that the video time stamps are patched correctly is to look at the instantaneous frame rate (command line option -T) to make sure that it does not vary more than 3 or 4%.

See Appendix D for trouble shooting most problems related to seamless branching.

Appendix D: Troubleshooting

Symptoms	Probable Causes
The presentation of pictures halts for an undefined amount of time across a video seamless jump.	An **iso_11172_end_code** was reached just before the branching point.
The presentation rate of video pictures changes just after a video seamless jump.	The video delta value is wrong, causing incorrect presentation time stamps to be decoded.
Bad video data randomly appears in the decoded video pictures	The video delta value is wrong, causing incorrect presentation time stamps to be decoded. STD buffer underflow can also be occurring (check video decoder signals received).
A bad video picture is displayed just after a video seamless jump	The first group of pictures after the branch is not closed, or the group of pictures across the branching point is not aligned to its first pack.
The audio decoder mutes or crashes after an audio seamless jump.	The audio delta value is too large or too short by a large amount.
The audio mutes temporarily across an audio seamless jump.	The audio delta value is slightly too large.
The audio is advanced on the video after an audio seamless branch.	The audio delta value is slightly too short.
An audio glitch is heard across an audio seamless jump.	An audio frame is split between packets from different streams at the branching point.

Appendix E: References

(1) "Information technology - Coding of moving pictures and associated audio for digital storage media at up to about 1.5 Mbit/s - Part 1: Systems," ISO/IEC 11172-1, First Edition, August 1, 1993.

(2) "Information technology - Coding of moving pictures and associated audio for digital storage media at up to about 1.5 Mbit/s - Part 2: Video," ISO/IEC 11172-2, First Edition, August 1, 1993.

(3) "Information technology - Coding of moving pictures and associated audio for digital storage media at up to about 1.5 Mbit/s - Part 3: Audio," ISO/IEC 11172-3, First Edition, August 1, 1993.

(4) "CD-I Full Functional Specification: Full Motion Extension," Philips Consumer Electronics BV, March 1993.

(5) "Disc Building for Sun Workstations," Optimage Interactive Services Company, L.P., March 3, 1993.

(6) "The Full Motion System for CD-I," Jan van der Meer, TSA Application Notes NR. TSA-007, February 15, 1993.

(7) "Introduction to Programming the FMV System," Ken Ellinwood, TSA Application Notes NR. TSA-008, May 05, 1993.

(8) "MPEG Data Structures" A Pictorial Guide to the ISO 11172 Spec," Andrew Duncan. (To be published August, 1994.)

(9) "Seamless Switching," Pieter de Visser, Interactive Media Systems, Philips Consumer Electronics B.V., July 21, 1993.

(10) "Seamless Switching and pink," Pieter de Visser, Interactive Media Systems, Philips Consumer Electronics B.V., July 22, 1993.

(11) "EOS Problem in Current DV Cartridges," Technical Note #102, Stefan Maris, Philips Interactive Media, March 21, 1994.