PHILIPS

Philips Interactive Media

Technical Note #102

EOS Problem in Current DV Cartridges

Stefan Maris
Philips Interactive Media Centre
Hasselt, Belgium

March 21, 1994

Abstract

Parallel processing of video material on several computers to create MPEG data streams may result in the creation of extraneous end of sequence (EOS) codes. Occasionally, the DV cartridge loses count of its frames after encountering an EOS code, which may result in read errors or system crashes. This note recommends removal of unnecessary EOS codes from the MPEG stream and provides source code for the program "patcheos," which performs this removal operation.

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 242546
Internet ID: hein@pimc.be

From the USA and Asia:

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

EOS Problem in Current DV Cartridges

Currently, most production sites for Digital Video (DV) still have to deal with the non real-time aspects of video preprocessing, grabbing, and MPEG encoding. Given the quality level of the MPEG video output, they cannot produce MPEG streams from a real-time video source. The production process is simply not fast enough.

Depending on the equipment used, the production process throughput can vary from a few times real time up to several hundred times real time.

The main bottleneck in the process is the MPEG encoding step. Until now, this part of the process has been done mostly by high performance computers. This is due to the fact that the MPEG encoding algorithm is done in software and is very CPU demanding.

To increase the production throughput, most of DV production sites started with the parallelization of several high performance computers. This is especially needed for long-play DV sequences, such as movies. In this parallelization technique, the DV sequence is chopped into smaller fragments. These fragments are then supplied to every single computer involved in encoding. Now more computers can work on the same DV sequence at the same time; of course, this reduces the time required to encode the entire sequence.

When all fragments are encoded to MPEG, they are again glued together into one large MPEG stream. (These fragments can vary from a few minutes to up to more than ten minutes)

Because every single computer encodes its own fragment(s), the output of these encoding operations results in an EOS (end of sequence) for every fragment produced.

When normal playback of these DV streams (containing EOS codes) is done with the current DV cartridges there are occasional read errors and crashes in the system.

After an EOS code, the cartridge seems to become "out of rhythm" and loses count of its frames. This results in an instruction being given to a frame that is not yet available. The fault does not happen at every EOS, but may happen on an any EOS.

The smaller the fragments, the more EOS codes there will be in the stream, and he higher the chance that such an error will occur.

To prevent this kind of trouble, it is recommended that those who are using this kind of parallelization technique remove the unnecessary EOS codes from the MPEG stream. This can be done by using a program running on SUN called 'patcheos'.

The source code for this program is shown below. For those who would like to have the executable and/or source code please contact one of the Developer Services representatives cited on the inside cover page.

```
/*************************************************************/
/* NAME   : patcheos                                         */
/*                                                           */
/* FUNCTION: main for Patching EOS in system stream          */
/*                                                           */
/* AUTHOR  : Stefan Maris                                    */
/*        Philips Interactive Media Centre                   */
/*                                                           */
/* Initial date: 940111                                      */
/*                                                           */
/*************************************************************/
/*
EDITION HISTORY
===============
|date  | change                          | edition | by |
|------------------------------------------------------|
|940111 | Birth                          | 1.0   | sm |
|------------------------------------------------------|
|940117 | Check for video pack           | 1.1   | sm |
|------------------------------------------------------|
|940224 | New software layout            | 2.0   | sm |
|      | bugs solved : - not all EOS found |      |    |
|      |             - patch after new SOS |      |    |
|      | Scan mode provided             |       |    |
|------------------------------------------------------|
|940301 | In video packet FIFO scan      | 2.1   | sm |
|      |                                |       |    |
|      |                                |       |    |
|      |                                |       |    |
|      |                                |       |    |
|      |                                |       |    |
|      |                                |       |    |
|      |                                |       |    |
|      |                                |       |    |
|      |                                |       |    |
|      |                                |       |    |
|------------------------------------------------------|
|      |                                |       |    |
|      |                                |       |    |
*/
#include <stdio.h>
#include <sys/types.h>
#include <sys/fcntl.h>
#include <unistd.h>
#include <memory.h>
```

```
/**
*** The MPEG system codes
**/
int nullvalue           = 0x00000000;
int end_of_sequence_code    = 0x000001b7;
int pack_start_code         = 0x000001ba;
int system_header_start_code = 0x000001bb;
int packet_start_code_prefix = 0x00000100;
int stream_id_private_low   = 0x000000bc;
int stream_id_private_high  = 0x000000bf;
int stream_id_audio_low     = 0x000000c0;
int stream_id_audio_high    = 0x000000df;
int stream_id_video_low     = 0x000000e0;
int stream_id_video_high    = 0x000000ef;
int stream_id_reserved_low  = 0x000000f0;
int stream_id_reserved_high = 0x000000ff;
int end_of_iso_stream       = 0x000001b9;

int user_data_start_code    = 0x000001b2;
int sequence_header_code    = 0x000001b3;
int sequence_error_code     = 0x000001b4;
int extension_start_code    = 0x000001b5;
int sequence_end_code       = 0x000001b7;
int group_start_code        = 0x000001b8;

#define PPLOW       (packet_start_code_prefix | stream_id_private_low)
#define PPHIGH      (packet_start_code_prefix | stream_id_private_high)

#define VPLOW       (packet_start_code_prefix | stream_id_video_low)
#define VPHIGH      (packet_start_code_prefix | stream_id_video_high)

#define APLOW       (packet_start_code_prefix | stream_id_audio_low)
#define APHIGH      (packet_start_code_prefix | stream_id_audio_high)

#define RPLOW       (packet_start_code_prefix | stream_id_reserved_low)
#define RPHIGH      (packet_start_code_prefix | stream_id_reserved_high)

/**
*** GLOBALS
**/
#define MAX_BUF   8192
#define MAX_POS    512

int  sysfile;                   /* system stream */
long fpos=0;                    /* current file position */

unsigned char *buffer;          /* data buffer */
long eospos[MAX_POS];               /* EOS file positions */
char patchit[MAX_POS];              /* patch the EOS ? */
long sospos[MAX_POS];               /* first SOS after EOS file positions */

unsigned char keep_sos[16];     /* buffer to keep last SOS data */

int  nreos=0;                   /* nr of EOS found */
int  nrsos=0;                   /* nr of SOS after EOS found */
```

```
unsigned int val=0;                /* value FIFO */
unsigned int vidval=0;             /* video packet value FIFO */
int        eosfound=1;     /* got an EOS */
int          wait_for_length=0;  /* get the length field of a packet */
int        packet_length=0;        /* this is the packets length */

int  in_packet=0;                          /* are we in a packet ? */
int  in_video_packet=0;                    /* are we in a video_packet ? */

int  test_mode;                            /* no patch !! */

/***
**** statistics
***/
int nr_of_video_packets=0;       /* number of video packets scanned */
int nr_of_audio_packets=0;       /* number of audio packets scanned */
int nr_of_other_packets=0;       /* number of other packets scanned */

int nr_of_eos_found=0;           /* total number of EOS found */
int nr_of_eos_patched=0;         /* total number of EOS patched */
int nr_of_sos_starts=0;          /* total number of SOS sequence starts */
int nr_of_sos_differ=0;          /* total number of SOS switches */

/**
*** MAIN
**/
main (argc, argv)
int argc;
char *argv[];
{
unsigned int  code;
int  buflength;

fprintf(stderr, "Philips Interactive Media Centre\n");
fprintf(stderr, "DV Tools (c)1994\n");
fprintf(stderr, "  End Of Sequence Patching utility\n");
fprintf(stderr, "  Release 2.1,  mar 1, 1994 \n\n");

if (argc < 2)
  {
  print_usage();
  exit(1);
  }

sysfile = open (argv[1], O_RDWR);
if (sysfile < 0)
  {
  fprintf(stderr, ".ERROR : can't open: %s\n", argv[1]);
  print_usage();
  exit(1);
  }

if ((argc == 3) && (strcmp(argv[2], "-s") == 0))
  test_mode = 1;
else
  test_mode = 0;
```

```
/** allocate buffer **/
buflength = MAX_BUF;
while ((buffer = (unsigned char *) malloc (buflength)) == 0)
    {
    buflength >> 2;
    if (buflength < 512)
      {
      fprintf (stderr, ".ERROR : Could not allocate buffer.\n");
          exit (1);
      }
    }

if (test_mode)
  fprintf(stderr, "\n . Patcheos SCAN mode !!!...\n");

/** parse process input file **/
fpos  = lseek(sysfile, 0, SEEK_SET);

fprintf(stderr, " . Scanning system stream...\n");
parse_it (buflength);

fprintf(stderr, "\n . Checking for difference in Sequence Header...\n");
diff_seq_header(buflength);

fprintf(stderr, "\n . Patching EOS...\n");
patch_eos();

fprintf(stderr, "\n . Patcheos statistics...\n");
print_statistics();

if (test_mode)
  fprintf(stderr, "\n . Patcheos SCAN mode : no EOS patched !!!...\n");

fprintf(stderr, "\n . patcheos done...\n");
close(sysfile);
}

/**
*** name :        print_usage()
***
*** function :    display usage message
***
*** version :     1.0
**/
int print_usage()
{
fprintf(stderr, "  use : patcheos <filename> [-s]\n");
fprintf(stderr, "        with : filename = name of multiplexed stream.\n");
fprintf(stderr, "               [-s]    = scan mode; do not patch.\n\n");
}
```

```
/**
*** name :        file_length
***
*** function :    Retrieve file length
***
*** version :     1.0
**/
long file_length (fid)
int fid;
{
long length;
long oldp;

oldp = lseek(fid, 0, SEEK_CUR);
lseek(fid, 0, SEEK_END);
length = lseek(fid, 0, SEEK_CUR);
lseek (fid, oldp, SEEK_SET);
return (length);
}

/**
*** name :        set_scan_pct
***
*** function :    print the  % of file still to do
***
*** version :     1.0
**/
int set_scan_pct (done, todo)
long done;
long todo;
{
static long cmt=101;
long  pct;

pct = done/(todo/100);
if (pct != cmt)
  {
  if ((pct%10) == 0)
    {
    fprintf(stderr, "  . %3ld %% to do...\n", pct);
    }
  cmt = pct;
  }
}
```

```
/**
*** name :        parse_it
***
*** function :    block read the system file
***
*** version :     1.0
**/
int parse_it (bl)
int bl;
{
long remain;
long filesize;

filesize = file_length(sysfile);
remain = filesize;
while (remain > bl)
  {
  set_scan_pct (remain, filesize);
  read (sysfile, buffer, bl);
  /** parse data **/
  parse_data(bl);
  fpos += bl;
  remain -= bl;
  }
if (remain > 0)
  {
  read (sysfile, buffer, remain);
  parse_data(remain);
  fpos += remain;
  }
}

/**
*** name :        vpacket
***
*** function :    check the packet & toggle the video_packet switch
***
*** version :     1.0
**/
int vpacket(x)
unsigned int x;
{
/** video packet **/
if ((x >= VPLOW) && (x <= VPHIGH))
  {
  nr_of_video_packets++;
  wait_for_length = 2;
  in_packet=1;
  return(1);
  }
```

```
/** private, reserved & padding packets **/
if ((x >= PPLOW) && (x <= PPHIGH))
  {
  nr_of_other_packets++;
  wait_for_length = 2;
  in_packet=1;
  return(0);
  }

/** audio packet **/
if ((x >= APLOW) && (x <= APHIGH))
  {
  nr_of_audio_packets++;
  wait_for_length = 2;
  in_packet=1;
  return(0);
  }

/** reserved data stream **/
if ((x >= RPLOW) && (x <= RPHIGH))
  {
  nr_of_other_packets++;
  wait_for_length = 2;
  in_packet=1;
  return(0);
  }

return(in_video_packet);
}

/**
*** name :        parse_data
***
*** function :    parse the data buffer
***                       We use a FIFO kind STATIC 4-byte buffer to
***                       match with the codes. This scheme should ensure
***                       the codes are also checked on block-boundaries.
***
*** version :     1.0
**/
int parse_data(length)
int length;
{
unsigned int      newval;
unsigned int      newvidval;
int               i;
```

```
/** val is a FIFO **/
for (i=0; i<length; i++)
  {
  newval = (val << 8) + ((*(buffer+i)) & 0xff);
  in_video_packet = vpacket(newval);
  if (in_packet)
    {
    if (wait_for_length)
      {
      wait_for_length--;
      if (wait_for_length == 0)
        {
        packet_length = newval & 0x0000ffff;
        }
      }
    else
      {
      if (in_video_packet)
        {
            newvidval = (vidval << 8) + ((*(buffer+i)) & 0xff);
            if (newvidval == end_of_sequence_code)
          {
          eospos[nreos] = (long) (fpos+i-3);
          patchit[nreos++] = 1;
          eosfound = 1;
          }
        if (eosfound && (newvidval == sequence_header_code))
          {
          sospos[nrsos++] = (long) (fpos+i-3);
          eosfound = 0;
          }
        vidval = newvidval;
        }
      /* wait for packet to finish */
      packet_length--;
      if (packet_length == 0)
        {
        in_packet = 0;
        in_video_packet = 0;
        }
      }
    }
  val = newval;
  }
}
```

```
/**
*** name :        diff_seq_header
***
*** function :    compare SOS before EOS & after EOS
***          If new info in SOS, flag off the previous EOS patch
***
*** version :     1.0
**/
int diff_seq_header(bl)
int bl;
{
int i;
int differ=0;

for (i=0; i<nrsos; i++)
  {
  /** get the SOS data **/
  lseek (sysfile, sospos[i]+4, SEEK_SET); /** seek after sos **/
  read (sysfile, buffer, 16);
  /** check it with the previous **/
  /* last two bits are for quant matrix, may change in between
    SOS headers of same sequence (ISO11172-2 p62-63) */
  *(buffer+7) &= 0xfc;
  if (memcmp (buffer, keep_sos, 8) != 0)
    {
    /* do not patch the previous EOS !!!, (the first SOS is always new) */
    if (i != 0)
      {
      fprintf(stderr, "   . SOS found at [%ld] & differ\n", sospos[i]);
      patchit[i-1] = 0;
      nr_of_sos_differ++;
      }
    /* keep this as last SOS */
    memcpy(keep_sos, buffer, 16);
    }
  }
nr_of_sos_starts = nrsos;
}

/**
*** name :        patch_eos
***
*** function :    if the EOS has to be patched, overwrite it with zeroes
***
*** version :     1.0
**/
```

```
int patch_eos()
{
int i;

patchit[nreos-1] = 0; /* do not patch the last EOS */
for (i=0; i<nreos; i++)
  {
  if (patchit[i])
    {
    fprintf(stderr, "   . EOS found at [%ld] & patched \n", eospos[i]);
    lseek (sysfile, eospos[i], SEEK_SET);
    if (! test_mode) write (sysfile, &nullvalue, 4);
    nr_of_eos_patched++;
    }
  else
    {
    fprintf(stderr, "   . EOS found at [%ld]\n", eospos[i]);
    }
  }
nr_of_eos_found = nreos;
}

/**
*** name :        print_statistics
***
*** function :    print the parsing statistics
***
*** version :     1.0
**/
int print_statistics()
{
fprintf(stderr, "   . Video packets scanned    [%d]\n", nr_of_video_packets);
fprintf(stderr, "   . Audio packets scanned    [%d]\n", nr_of_audio_packets);
fprintf(stderr, "   . Other packets scanned    [%d]\n\n", nr_of_other_packets);
fprintf(stderr, "   . Total EOS found          [%d]\n", nr_of_eos_found);
fprintf(stderr, "   . Total SOS sequence starts [%d]\n", nr_of_sos_starts);
fprintf(stderr, "   . Total SOS changes        [%d]\n", nr_of_sos_differ);
fprintf(stderr, "   . Total EOS patched        [%d]\n\n", nr_of_eos_patched);
}
```