TECHNICAL REPORT

ISO/IEC TR 11172-5

First edition
1998-11-01

Information technology — Coding of moving pictures and associated audio for digital storage media at up to about 1,5 Mbit/s —

Part 5:
Software simulation

Technologies de l'information — Codage de l'image animée et du son associé pour les supports de stockage numérique jusqu'à environ 1,5 Mbit/s —

Partie 5: Simulation de logiciel

Reference number
ISO/IEC TR 11172-5:1998(E)

Contents

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work.

In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

The main task of technical committees is to prepare International Standards, but in exceptional circumstances a technical committee may propose the publication of a Technical Report of one of the following types:

— type 1, when the required support cannot be obtained for the publication of an International Standard, despite repeated efforts;

— type 2, when the subject is still under technical development or where for any other reason there is the future but not immediate possibility of an agreement on an International Standard;

— type 3, when a technical committee has collected data of a different kind from that which is normally published as an International Standard ("state of the art", for example).

Technical Reports of types 1 and 2 are subject to review within three years of publication, to decide whether they can be transformed into International Standards. Technical Reports of type 3 do not necessarily have to be reviewed until the data they provide are considered to be no longer valid or useful.

ISO/IEC TR 11172-5, which is a Technical Report of type 3, was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 29, *Coding of audio, picture, multimedia and hypermedia information.*

ISO/IEC 11172 consists of the following parts, under the general title *Information technology — Coding of moving pictures and associated audio for digital storage media at up to about 1,5 Mbit/s*:

— *Part 1: Systems*

— *Part 2: Video*

— *Part 3: Audio*

— *Part 4: Compliance testing*

— *Part 5: Software simulation*

Introduction

Purpose

This Technical Report was developed in response to the growing need for a generic coding method of moving pictures and of associated sound for various applications such as digital storage media, television broadcasting and communication. The use of this specification means that motion video can be manipulated as a form of computer data and can be stored on various storage media, transmitted and received over existing and future networks and distributed on existing and future broadcasting channels.

Information technology — Coding of moving pictures and associated audio for digital storage media at up to about 1,5 Mbit/s —

Part 5:
Software simulation

1 Scope

This Technical Report provides a C language software simulation of an encoder and decoder for Part 1 (Systems), Part 2 (Video), and Part 3 (Audio) of ISO/IEC 11172.

2 References

Recommendations and reports of the CCIR, 1990 XVIIth Plenary Assembly, Dusseldorf, 1990 Volume XI - Part 1 Broadcasting Service (Television) ITU-R Rec. BT.601-3, *Encoding parameters of digital television for studios.*

CCIR Volume X and XI Part 3 ITU-R Rec. BR.648, *Recording of audio signals*.

CCIR Volume X and XI Part 3 Report ITU-R 955-2 Satellite sound broadcasting to vehicular, portable and fixed receivers in the range 500 - 3000Mhz.

ISO/IEC 11172-1:1993, *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 11172-2:1993, *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 11172-3:1993, *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 11172-4:1995, *Information technology - Coding of moving pictures and associated audio for digital storage media at up to about 1,5 Mbit/s – Part 4: Compliance testing*.

IEEE Standard Specifications for the Implementations of 8 by 8 Inverse Discrete Cosine Transform, IEEE Std 1180-1990, December 6, 1990.

IEC 461:1986, *Time and control code for video tape recorders*.

IEC 908:1987, *Compact disc digital audio system.*

ITU-T Rec. H.261 (Formerly CCITT Rec. H.261) Codes for audiovisual services at px64 kbit/s, Geneva 1990.

ITU-T Rec. T.81 | ISO/IEC 10918-1:1994, *Information technology - Digital compression and coding of continuous-tone still images: Requirements and guidelines*.

3 Definitions

For the purposes of this Technical Report, the definitions given in ISO/IEC 11172-1, ISO/IEC 11172-2 and ISO/IEC 11172-3 apply.

4 Symbols and abbreviations

The ISO/IEC 9899:1990, *Programming languages – C* defines the programming language used in the source annexes of this Technical Report.

5 Systems simulation

The diskette included in Annex A contains the implementation of an ISO/IEC 11172-1 codec.

6 Video simulation

The diskette included in Annex A contains the implementation of an ISO/IEC 11172-2 codec.

7 Audio simulation

The diskette included in Annex A contains the implementation of an ISO/IEC 11172-3 codec.

Annex A

Diskette containing software

A.1 General

This Annex contains C source code and information for Systems, Video and Audio codec and is distributed on the attached diskette. The diskette contains the ISO/IEC 11172 source code for Systems, Video, and Audio.

Source code files use MS-DOS line feeds. Files need to be translated to alternate line-feed formats if used on UNIX, for example.

The source code, once compiled, is capable of producing compliant ISO/IEC 11172 bitstreams, except as noted, per ISO/IEC 11172-4. Clauses A.2, A.3, and A.4 and their subclauses contain the README files associated with the source code directories on the diskette.

A.2 Systems

A.2.1 Decoder

```
Every line starts with an ID, followed by a colon. The ID is one of PACK, SYS,
or an ISO stream ID, representing a pack header, system header information, or
a stream packet of the type and number represented by the ID.

Depending upon the object being described, the following information is
available, separated by at least one character of white space. Objects are
separated by linefeeds.

PACK: number of pack in sequence
      SCR value
      MUX rate

SYS:  length
      rate bound
      audio bound
      fixed flag
      CSPS flag
      audio lock
      video lock
      video bound
/* the following may be repeated for each stream referenced */
      streamID
      STD_buffer_bound_scale
      STD_buffer_size_bound

<ID>: length
      # bytes stuffing ( * if omitted )
      STD_buffer_scale
        STD_buffer_size
        PTS or *
        DTS or *
        data size
```

A.2.2 Encoder

```
Introduction:

This program is designed to create ISO/IEC 11172-1 system streams from
ISO/IEC 11172-2 and -3 compressed video and audio files.  It is limited

Compilation/Installation Instructions:
      - Modify the Makefile to match your system.  If you want
      all types of messages define one of the DEBUG's.
```

```
      - Type "make binary"
      - Put wherever you wish.

Usage:
      cdisys paramfile

Parameter file explanation:

      The parameter file is used to control the behavior of this
      encoder.  It is line oriented, and therefore it is very important
      for the right parameters to be on the correct line.

      Here is a sample parameter file that would create a multiplexed
      MPEG file suitable for playing back from a 2x cdrom, using
      mode 2 form 2 sectors.

--------------Start of parameter file, cut here ----------------------
# Sample parameter file for system encoder, your comments may go on this line
audio.abs      # Name of the input audio file
video.vbs      # Name of the input video file
/tmp/sys.mpg   # Name of the output system file
e0                          # hex value for video id, normally e0 (dec 224)
c0                          # hex value for audio id, normally c0 (dec 192)
3                           # MPEG video M value, ie size between P frames
1152000.0             # Video bit rate
224000.0              # Audio bit rate
0                           # bool flag, if set, encoder continues even on EOF of aud/vid
999999                # max number of sectors to create
7056                  # muxrate
21000                 # First Audio pts in 90khz clock units
21000                 # First Video pts in 90khz clock units
0                           # Fixed bitrate flag 1->set, 0->variable bitrate stream
1                           # CSPS flag
1                           # Audio lock flag 1 or 0
1                           # Video lock flag 1 or 0
44100.0               # Audio sample rate
29.97                 # Video frame rate
2324                  # Number of bytes per pack (CDI = 2324)
2352                  # Number of bytes per pack + any DSM related stuff (CDI = 2352)
2312                  # Number of bytes per video packet (CDI=2312)
2294                  # Number of data bytes per video packet (CDI=2294)
2312                  # Number of bytes per audio packet (CDI=2312)
2299                  # Number of data bytes per audio packet (CDI=2279, don't include the 20 bytes of
nulls here)
1                           # Number of packets per pack (not implemented, fixed to 1, CDI=1)
0                           # (0|1) If set then output CDI type headers and audio (20byte) padding
0                           # (0|1) If set output Video CD NULLS with audio packs
----------------------End of parameter file, cut here ----------------------------

      The first line is for any comments you may wish to identify the
      parameter file.

      The next two lines are the audio and video input files that
      are to be multiplexed into a system stream specified in the
      following parameter.

      The stream ID for video is specified next as a hexadecimal value.
      Values from 0xE0 to 0xEF are allowed as specified according to
      ISO 11172-1.  In other words up to 16 video elementary streams
      can be multiplexed into one system stream (but not with this encoder).

      The stream ID for audio allows values from 0xC0 to 0xDF are allowed.
      This allows for as many as 32 separate audio elementary streams
      to be multiplexed together.

      The number of video frames (plus 1) between 'reference' frames is specified
      next (more specifically, the 11172-2 parameter of M).  For example,
      if the number of B frames between reference frames is 2 then this
      number is 2+1

      Video bitrate is the encoded 11172-2 video bitrate of the video stream.

      Audio bitrate is the encoded 11172-3 audio bitrate of the audio stream.
```

The next parameter is used to control when the encoder stops encoding. If this is set the encoder will continue even after the EOF of the elementary streams. This will result in the encoder simply creating padding streams.

Max number of sectors is used to control how many packs/sectors that the encoder can generate. This can be used to terminate the encoder before the EOF of one or more of the elementary streams.

Muxrate is the number (divided by 50) of bytes that are delivered to the system decoder every second. For example 7056 is equal to 352800 bytes/second, or looking up the specs on cdrom's this is what 2x cdroms can actually deliver.

First audio and video PTS are used by the encoder to determine when each stream should start playback. If the streams are initially in sync, then make sure that these values are set to the same value. If there is a startup difference between audio to video then set the audio video PTS value to reflect the difference.

Fixed bitrate, constrained system parameters, audio lock and video lock are parameters that come right out of the 11172-1 specification.

Audio sample rate and video sample rate are the next parameters. Each parameter is set according to the value actually set within the elementary stream.

Bytes per pack is the number of bytes that each pack may contain.

Bytes per pack + DSM, is the total number of bytes that are contained within a "sector". This value is necessary to account for any 'time' that takes place for sector headers and other non-MPEG data. For example, with CD's a sector is actually 2352 bytes, of which only 2324 bytes are used for ISO/IEC 11172-1 systems data. The remaining bytes still take time to pass under the read head, so the system encoder uses this size to calculate the amount of time this will take.

The next parameter controls the size of a video packet. The parameter after this one controls how many video bytes are copied from the video elementary stream into the video packet.

The next parameter controls the size of a audio packet. The parameter after this one controls how many audio bytes are copied from the audio elementary stream into the audio packet.

Number of packets per pack controls the max number of packets that can be put within a ISO/IEC 11172-1 system pack. At present this is fixed to 1.

The next parameter is used to put the encoder into a CDI or non-CDI generating mode. If you are creating CDI/VCD type MPEG files, this should be set to 1. It will turn on the notion of each elementary stream being multiplexed into its own system stream (albeit still producing 1 muxed file).

The last parameter controls the generation of Video CD NULLS. If set the encoder will output 20 NULLS after each audio packet. This is a requirement of Video CD's. Many CD burning software packages put the NULLS on the file when burning to the disk. If using these types of software, set this to 0. The Philips verifier assumes that the NULLS are left off for verification, and only placed in the stream when actually burned to disc.

```
Bugs:
      The buffering model is not completely accurate and can produce
      minor overflow/underflow errors when multiplexing long runs.

      Access Unit (AU) detection is not 100% reliable on audio AU's
      due to the small size of a syncword.  A heuristic is used
      to reduce the odds of mismatching an audio AU, however, the
      chances are not zero.

      Upon termination of an elementary stream this code will
      just tack on a ISO end code.  There are requirements (non-ISO) such
      as video CD or CDI that really want the EOF to happen on
      an access unit boundary for all elementary streams being multiplexed.
```

A.3 Video

A.3.1 Decoder

```
Usage

The parameter file, decode.ini, contains user selectable settings
which determine the tracelevel (how detail the statistics are
printed to stdout or file), and reconstruction file format.
The .SIF format is also known as "Abekus."  YCbCr components are
interleaved (multiplexed) as a repeating sequence of {Cb,Y,Cr,Y} samples
with 4:2:2 sampling ratio.  The .YUV format is compatible with Berkeley's
encoder.  Components are non-interleaved:  {Y picture buffer, Cb buffer, Cr
buffer} are concatenated into a single file.

Bug fixes:
- wrong motion vector reconstruction fixed: MV range was positive biased
  instead of negative (e.g.: for f_code=1 [-15..16] instead of [-16..15])
- fixed bug which caused pictures to be "displayed" (=stored) in wrong
  order (I/P frames were delayed for 2 I/P intervals instead of 1)
- error in predictor resetting fixed (DC predictors were not reset after
  an MB sequence of 'intra, skipped, intra' in P pictures)
- fixed interpolation expression (forward and backward prediction have
  to be rounded individually before averaging, these operations aren't
  commutative)
- Little Endian support: no longer casts a byte stream to an unsigned int array

Functional enhancements:
- arbitrary picture dimensions implemented
- allow repeat sequence headers
- arbitrary slice sizes implemented
- full pel motion vectors implemented
- additional output picture file formats: TGA, PPM
- optional trace output

General:
- separated decoder from encoder, moved all .c files into one directory
- changed most 'short' variables to 'int' (C converts them to int in
  expressions and function parameters / return values anyway and gcc's
  optimizer seems to have problems with shorts)
- SNR statistics output removed
- changed several variable names to match the 11172 terminology
- renamed mpeg1.h to types.h and BitCount.h to vlc.h

procMB.c:
- new procedure procBlock() introduced: block layer (2.4.2.8) and DC prediction
- procMB adapted to call procBlock()

procPicture.c:
- split into procPicture.c (picture layer) and procSlice.c (slice layer)
- decoding (i.e. skipping) of extension and user data fields
- simplified structure of procSlice
```

```
GlobalVar.h:
- redundant kMCType removed (also from consts.h)
- redundant entries in MB type tables removed

iquant.c:
- ambiguous reconstruction expressions parenthesized: q1*q2*v/8 -> (q1*q2*v)/8

FConstr.c:
FIConstr.c:
- simplified break-down of MVs into pel and half-pel offsets

GetBits.c:
- structure simplified: buffer is now refreshed before extracting the next bit,
  not afterwards.
- NextBits() introduced: returns next n bits without advancing current position

GetCode.c:
- changed error criterion in findCode() and findACCode from numBits > maxBits
  to numBits >= maxBits
- added mismatch error message to findCode()
- added checking for invalid level codes after escape
- merged findCode() and findACCode()
- AC VLC table short -> BYTE

changes from 1.3 to 1.4

  Implementation of Video Buffering Verification (VBV) Operation on Sep.2.93

  Changed file          Modifications
  ========================================================
  include/consts.h  - add macros "kMaxPictureDelay" and "kMaxVbvBufferSize"

  include/mpeg1.h   - add an integer member "vbvBufferSize" to a type
                      defined structure "tParameter".
                    - add a float member "vbvOccupy" and an integer
                    - member "vbvDelay" to a type defined structure
                      "tRateControl".

src/initial.c       - add the definition of "aParameter->vbvBufferSize"
                      and the initialization of "aRateControl->vbvOccupy"
                      in the function "Initial()"
                    - change the unit of "aParameter->bitRate" to in bit/s
                    - add a float value "vbvBufferStart"

  encoder/BitCount.c- modify the coding of "vbv_buffer_size" in the
                      function "BC_SequenceHeader()"
                    - modify the coding of "vbv_delay" in the
                        function "BC_Picture()"

  encoder/PreFrame.c- include a header file "flc.h"
                    - call the function "calcVbvDelay()"
                    - add the function "calcVbvDelay()"
                    - add the function "checkUpdateVbv()"

  encoder/encoder.proto
                    - add the definition of the function "calcVbvDelay()"
                    - add the definition of the function "checkUpdateVbv()"

  encoder/procPicture.c
                      - call the function "checkUpdateVbv()"
```

A.3.2 Encoder

```
ISO/IEC 11172-1 Video encoder   version 1.5g
```

```
Version 1.5g of the ISO/IEC 11172-1 Video Encoder has been printed as
document CD 11172-5 in April 1994.  This program has been successfully
ported and tested on the IBM PC and various Unix workstations using
the GNU GCC compiler.

Changes

1.4f to 1.4g:
- added .YUV, .TGA, and .PPM file formats

since 1.4

 - added big/little endian I/O to writebit.c
 - corrected temporal reference
 - corrected half-pel interpolation in B construction
 - added .TGA, .PPM, .YUV file formats
 - clarified a lot of code with better organization and nomenclature
 - fixed bug in motion.c that confused current_MB with current_picture
 - fixed bug in Write_MB() which did not reset forward MV to zero

since 1.3
- user programmamble picture dimensions

1.2 Remaining task items
(as of April 14, 1994 version 1.5a) are:

1.  Proper filtering (co-siting of chroma samples) in the
     4:2:0 to 4:2:2 upsampling and downsampling process

2.  Temporal reference needs to be corrected.  This is generated
     in Write_GOP_Header() module.

3.  VBV delay needs extra numerical precision.   It drifts 4 counts
     each picture from the correct value.

 4.  Implement drop frame in encoder.ini and t_Parameter
     [this is part of the Write_GOP_Header overhaul]

 5.  Implement tracefile

Usage.

 Variables and coding style are temporary.

 Legal picture rates are:

        No.     Picture rate (Hz)
        ----    -----------------
        1       23.976
        2       24
        3       25
        4       29.97
        5       30
        6       50
        7       59.94
        8       60

Drop frame may be only used in Case No. 5.

List of files

encoder.pro
        File containing all module prototypes for all .c files.  This file
        is then called in the header (#include) of each .c file.
```

```
encoder.ini
        Initialization script file for encoder.  Sets frame rate,
        picture formats, picture dimensions, file names, etc.
        (Formerly known as "paramFile").

reconstr.c
        macroblock reconstruction and prediction modules for
        intra, forward or backward, and interpolated (bi-directional).

decision.c
        Files related to macroblock decision making: intra/non-intra
        decision [ IntraCheck() ], motion compensated/no motion compensated
        [ CalSE() ], interpolated [ ICalSE() ],

ratectrl.c
        Rate control functions.

dct.c
        Forward and Inverse quantization and DCT modules.

quantize.c
        Forward and Inverse quantization routines.

motion.c
        Block matching algorithm modules

writepic.c
        Reconstruction picture to file modules

readpic.c
        Source input picture modules

initial.c
        Initialization routines

encoder.c
        Main() module

convert.c
        Picture format conversion modules involving any filtering,
        interpolation, decimation or sample rate conversion.
```

A.4 Audio

A.4.1 Decoder/Encoder

```
The software is functioning properly if the following equations hold:

      a. decoded(orig.mpg) == deco.dec
         byte-swapping of deco.dec will be necessary for this
         equation to hold for little-endian computers
      b. encoded(deco.dec) == renc.mpg

           (encode with the default options except for the following:
           48 kHz sampling rate and 256 kbits/sec coded bit rate)

      If the bitstream tests fail, make sure that the following
variable types have at least the precision listed below:

      integer     -     16 or 32bits
      float       -     32 bits
      double      -     64 bits.

Organization of the Code:
```

The Audio software package consists of:

21 data files tables
8 source files (*.c)
3 definitions files (*.h)
3 test bitstreams
* makefiles

Table 1 illustrates how the encoder and decoder is formed from the component files. In this table the definition files are enclosed in parenthesis and listed immediately below the primary source file which uses them. The data file names are listed within braces and also placed immediately below the source file which uses them.

Table 1

encoder files	common files	decoder files
musicin.c	common.c	musicout.c
encode.c	(common.h)	decode.c
(encoder.h)	{alloc_0}	(decoder.h)
{enwindow}	{alloc_1}	{dewindow}
psy.c, subs.c	{alloc_2}	
{absthr_0}	{alloc_3}	
{absthr_1}		
{absthr_2}		
tonal.c		
{1cb0}, {1cb1}, {1cb2}		
{2cb0}, {2cb1}, {2cb2}		
{1th0}, {1th1}, {1th2}		
{2th0}, {2th1}, {2th2}		

Running the Software

To run this software, compile the programs to form an encoder executable file, musicin, and a decoder executable file, musicout.

To run the code type the name of the file followed by a carriage return. The programs will prompt you to input the appropriate parameters. The sound input file for the encoder should be sound data, monophonic or stereophonic, sampled at 32, 44.1, or 48 kHz with 16 bits per sample. For stereophonic data the left channel sample should precede the right channel sample. The sound output file of the decoder will be the same format as the sound input file used by the decoder, except for possible byte order differences if the encoder and decoder programs are run on different computer systems which have different byte ordering conventions.

Special notes for MSDOS users:

1. The default bitrate option does not work.
2. The input/output filename defaults not compatible with MSDOS.
3. Use the large memory model for compilation.

Notes on the Software

The encoder and decoder software are configured to output the coded audio bitstreams as a string of hexadecimal ascii characters. For greater compression efficiency, compile flag, BS_FORMAT, in common.h can be switched to configure the bitstream reading and writing routines to process raw binary bitstreams.

The decoder program has a very crude implementation of bitstream synchword detection. It may not be able to correctly decode valid bitstreams which have false synchword patterns in the ancillary data portion of the bitstream.