CONFIDENTIAL

4i2i Communications Ltd

May 2000

H.261 and H.263 Video CODEC Documentation

Revision 1.3

29 May 2000

Software:	Combined H.261 and H.263
Author:	Kathy Kipperman
Reviewed:	Iain Richardson
Updated:	29 May 2000

H.261 and H.263 Video CODEC Documentation

Revision 1.3

TABLE OF CONTENTS

1 INTRODUCTION

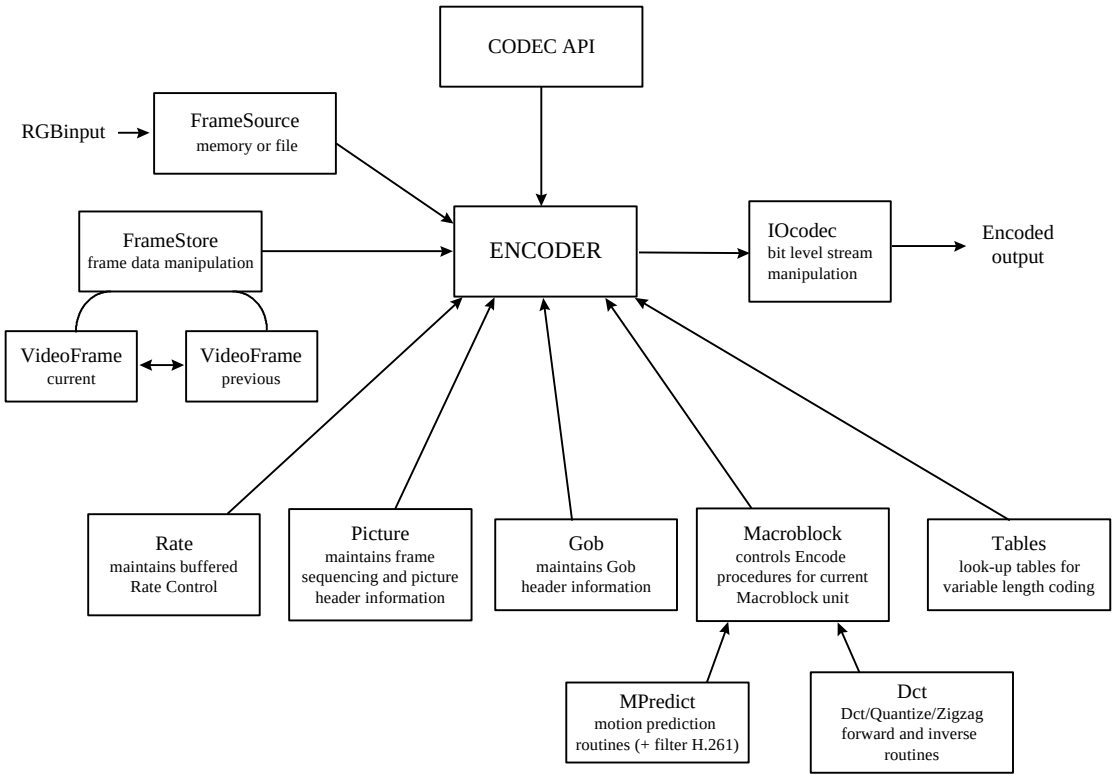
This report documents the combined H.261/H.263 Video CODEC (hereinafter referred to as the CODEC). The CODEC consists of C++ classes which are held in a number of C++ source and header files. The CODEC conforms to the appropriate sections of [1] and [2].

2 FUNCTIONAL DESCRIPTION

2.1 CODEC STRUCTURE

The CODEC consists of a series of classes that are interconnected as shown in the Encoder diagram below.

CODEC configured for Encode



2.2 ENCODER OPERATION

Typical operation is as follows.

- Create an H.261/H.263 Encoder object
- Initialise the encoding parameters
- Call the Encoder to encode frames of video either (a) a frame at a time or (b) a GOB (Group of Blocks) at a time.

The input to the Encoder is a series of frames of video and the output is a bit stream conforming to ITU-T H.261 or H.263. Input/output can be file-based or memory-based.

2.3 DECODER OPERATION

Typical operation is as follows.

- Create an H.261/H.263 Decoder object
- Initialise the decoding parameters
- Call the Decoder to decode a bit stream either (a) a frame at a time or (b) a GOB (Group of Blocks) at a time.

The input to the Decoder is a bit stream conforming to ITU-T H.261 or H.263 and the output is a series of frames of video. Input/output can be file-based or memory-based.

3 API DESCRIPTION

The API consists of a CODEC configuration structure (Section 3.1), plus the API functions listed in Section 3.3.

3.1 CODEC CONFIGURATION

The calling application must include the header file in which the CODEC configuration structure is declared. A separate configuration variable must be generated for each instance of an Encoder or Decoder object. The configuration structure consists of member variables for each CODEC option, plus member variables used to pass information between the CODEC and the calling application.

API CONFIGURATION STRUCTURE – CODECconfigure declared in

CODEC TYPE AND RESOLUTION

```
#
#define H261CIF 1
  define H263QCIF 2
#
```

OPTION SETTINGS

```
#
#define DISABLED 0
  define YES 1
#
```

CONFIGURATION OPTIONS

Resolution (H.261 PTYPE bit 4 H.263 PTYPE bits 6-8)	H.261CODEC – H261QCIF or H261CIF H.263CODEC – H263QCIF or H263CIF
FrameRate	Target frames per second (between 1 – 30 frames per second)
BitRate	Target bit rate (Kbits/second) A value of 0 indicates that Rate Control is to be DISABLED
H261LoopFilter	ENABLED or DISABLED
H263GobSync	ENABLED or DISABLED If ENABLED the H.263 CODEC inserts GOB header information
H263HalfPixelME	ENABLED or DISABLED
FreezePictureRequest	ENABLED or DISABLED
FastUpdateRequest (PTYPE bit 3 in H.261, or bit 5 in H.263, is switched on in response to this signal)	ENABLED or DISABLED
NewSequence (PTYPE bit 9 in H.263) only available in H.263	YES or NO Forces the CODEC to encode the next frame as an INTRA I-picture. Prevents motion artefacts at the start of a new video sequence.

INFORMATION FROM CALLING APPLICATION TO CODEC

IncludeBMPheaders	YES or NO
AccessToP_TR	ENABLED or DISABLED
TempRef	see Section 3.1.1

INFORMATION FROM CODEC TO CALLING APPLICATION

FramesSkipped	In order to maintain the target frame rate and acceptable picture quality the Encoder may decide to drop frames
EncodedBytes	Number of bytes taken to encode current task
EncodeVoid	Encode frame failure
DecodedBytes	Number of bytes taken to decode current task
DecodeVoid	Decoder has detected an error condition Decoded frame data corrupted

PLEASE NOTE

- The configuration option **H263GobSync** *must* be enabled when coding one GOB at a time (H.263).

- The Encoder **FastUpdateRequest** and Decoder **FreezePictureRequest** options are only functional during complete frame unit operation.
  FreezePictureRequest
  When enabled the CODEC sets an internal flag to freeze output to the current frame held in the Decoder frame store. On each subsequent call to the API EncodeFrame function, the Decoder will continue to output this frozen frame until it either detects the Freeze Picture Release signal or a time-out period of at least six seconds has elapsed.
  FastUpdateRequest
  When enabled the CODEC encodes the next frame in INTRA mode and switches Freeze Picture Release ON in the Picture header PType information field.

- **Frames Skipped**
  The CODEC will only decide to skip frames when operating with complete frame units. The maximum number of frames that can be dropped between transmitted frames are:
  H.261 – 1frame
  H.263 – 1frame

3.1.1 API CONTROL OF PICTURE LAYER VARIABLES

The CODEC has internal functions, which would normally determine the picture layer control variables.

- **TEMPORAL REFERENCE**
  To enable outside control of the Temporal Reference code, the API provides the AccessToP_TR flag. When this flag is ENABLED the CODEC uses the TempRef value provided by the calling application for the TR field of the Picture layer header.
  Please Note:
  The TempRef code for H.261 is a 5-bit number, which can have 32 possible values.
  The TempRef code for H.263 is an 8-bit number, which can have 256 possible values.
- **PTYPE INFORMATION**
  The PTYPE information bits, which are available for control by an external application, are listed in the above configuration options.

3.2 HOW TO CREATE A CODEC OBJECT

The CODEC has separate H.261 and H.263 functionality, it does not support inter-working between H.261 and H.263 processing within the same operation.

A CODEC object is created when that object's declaration statement is executed. The object's constructor functions are called each time one of the following types of declaration statements are executed.

- ***h261Encode h261Encoder:*** creates an H.261 Encoder object
- ***h261Decode h261Decoder:*** creates an H.261 Decoder object
- ***h263Encode h263Encoder:*** creates an H.263 Encoder object
- ***h263Decode h263Decoder*** : creates an H.263 Decoder object

If more than one object of the same type were to be created in the same application, the objects would have to take different variable names. For example, the following two H.261 Encoders could be created:

- ***h261Encode h261EncA***
- ***h261Encode h261EncB***

It is the responsibility of the calling application to create the CODEC object, then keep that object in scope for the required duration of CODEC processing so that information is not "lost" between CODEC processing calls. The object's destructor functions are automatically called when that object goes out of scope. It is at this point that the CODEC object is destroyed.

When the CODEC object is created the constructor functions automatically initialise the CODEC to operate with the following default settings:

CODEC option	H.261 and H.263 Encoder	H.261 and H.263 Decoder
BitRate	384 Kbits/second	No effect
FrameRate	10 frames per second	No effect
Resolution	CIF	CIF
IncludeBMPheaders	NO	NO
H261LoopFilter	DISABLED (H.261 only)	No effect
H263GobSync	DISABLED (H.263 only)	No effect
H263HalfPixelME	DISABLED (H.263 only)	No effect
FreezePictureRequest	No effect	DISABLED
FastUpdateRequest	DISABLED	No effect
NewSequence	YES	No effect
AccessToP_TR	DISABLED	No effect
TempRef	0	No effect

If different option settings are required, it is the responsibility of the calling application to change the CODECconfigure structure then call the API function to update CODEC operation.

3.3 API FUNCTIONS

The API source code for the H.263 and H.261 CODECs are in files h263codec.cpp and h261codec.cpp.
The API functions extract operational information from the CODECconfigure structure. This information is used to select the appropriate internal function calls.

3.3.1 INITIALISATION

1. CODECinitialise(CODECconfigure*)

INPUTS

CODECconfigure*	Configuration structure containing new CODEC options

DESCRIPTION
This function MUST be called to update CODEC internal variables after any change in option settings.

2. ConfigEncoder(CODECconfigure* configE)

INPUTS and OUTPUTS

CODECconfigure* configE	Configuration structure to be initialised for Encoder operation

DESCRIPTION
This function initialises the CODEC options for default Encoder operation (H.261 and H.263).

3. ConfigDecoder(CODECconfigure* configD)

INPUTS and OUTPUTS

CODECconfigure* configD	Configuration structure to be initialised for Decoder operation

DESCRIPTION
This function initialises the CODEC options for default Decoder operation (H.261 and H.263).

3.3.2 MEMORY BASED OPERATION

H.261 Information

In order to exit the memory based DecodeFrame and DecodeGOB H.261 API function calls when macroblocks have not been transmitted at the end of the last GOB unit, the input parameter, "size", must be the actual number of valid data bytes to be decoded from the buffer.

1. M_EncodeFrame(CODECconfigure* configE, BMPobject* RGBin, unsigned char* VideoStream, int size)

INPUTS

BMPobject* RGBin	Memory location of video frame in RGB format (with or without Bitmap header)
unsigned char* VideoStream	Memory location of video stream buffer
int size	Size (bytes) of video stream buffer

OUTPUTS

configE.FramesSkipped	Following number of frames to be skipped
configE.EncodedBytes	Number of bytes taken to encode frame unit
configE.EncodeVoid	Set to YES if video stream buffer is not large enough to store the encoded frame unit

DESCRIPTION
Memory based operation - encodes a single frame unit of video.

2. M_DecodeFrame(CODECconfigure* configD, unsigned char* VideoStream, int size, BMPobject* RGBout)

INPUTS

unsigned char* VideoStream	Memory location of video stream buffer to decode
int size	Size(bytes) of buffer to decode
BMPobject* RGBout	Memory location to direct decoded video frame (in RGB format with or without Bitmap header)

OUTPUTS

configD.DecodedBytes	Number of bytes decoded from video stream buffer
configD.DecodeVoid	Set to YES if an error condition is detected

DESCRIPTION
Memory based operation - decodes a single frame unit.

3. M_EncodePSC(CODECconfigure* configE, BMPobject* RGBin, unsigned char* PSCstream, int size)

INPUTS

BMPobject* RGBin	Pointer to video frame stored in memory in RGB format (with or without Bitmap header)
unsigned char* PSCstream	Memory location of buffer used to output encoded Picture Start Code
int size	Size (bytes) of Picture Start Code buffer

OUTPUTS

configE.EncodedBytes	Number of bytes taken to encode Picture Start Code
configE.EncodeVoid	Set to YES if buffer is not large enough to store the encoded Picture Start Code

DESCRIPTION
Memory based operation - encodes Picture Start Code of current frame unit.

4. M_EncodeGOB(CODECconfigure* configE, unsigned char* GOBstream, int size)

INPUTS

unsigned char* GOBstream	Memory location of buffer to direct current encoded GOB unit
int size	Size (bytes) of GOB buffer

OUTPUTS

configE.EncodedBytes	Number of bytes taken to encode GOB unit
configE.EncodeVoid	Set to YES if buffer is not large enough to store the encoded GOB

DESCRIPTION
Memory based operation - encodes one GOB unit

5. M_DecodePSC(CODECconfigure* configD, unsigned char* PSCstream, int size, BMPobject* RGBout)

INPUTS

unsigned char* PSCstream	Memory location of buffer containing encoded PSC of current frame unit
int size	Size(bytes) of data to decode
BMPobject* RGBout	Memory location to direct decoded frame unit (RGB format with or without Bitmap header)

OUTPUTS

configD.DecodedBytes	Number of bytes decoded from Picture Start Code buffer
configD.DecodeVoid	Set to YES if an error condition is detected

DESCRIPTION
Memory based operation - decodes Picture Start Code of current frame unit.
Supplies CODEC with memory location to store decoded frame unit

6. M_DecodeGOB(CODECconfigure* configD, unsigned char* GOBstream, int size)

INPUTS

unsigned char* GOBstream	Memory location of buffer containing current encoded GOB unit
int size	Size(bytes) of data to decode

OUTPUTS

configD.DecodedBytes	Number of bytes decoded from GOB buffer
configD.DecodeVoid	Set to YES if an error condition is detected

DESCRIPTION
Memory based operation - decodes one GOB unit

3.3.3 FILE BASED OPERATION

1. F_EncodeFrame(CODECconfigure* configE, char* RGBinFile, char* StreamOut)

INPUTS

unsigned char* RGBinFile	File containing video frame stored in RGB format (with or without Bitmap header)
unsigned char* StreamOut	File to direct encoded video data output

OUTPUTS

configE.FramesSkipped	Following number of frames to be skipped
configE.EncodedBytes	Number of bytes taken to encode frame unit

DESCRIPTION
File based operation - encodes a single video frame unit.

2. F_DecodeFrame(CODECconfigure* configD, char* StreamIn, char* RGBoutFile)

INPUTS

unsigned char* StreamIn	File containing encoded video data
unsigned char* RGBoutFile	File to direct decoded frame output (RGB format with or without bitmap header)

OUTPUTS

configD.DecodedBytes	Number of bytes taken to decode current frame unit
configD.DecodeVoid	Set to YES if an error condition is detected

DESCRIPTION
File based operation - decodes a single frame unit.

3. F_EncodePSC(CODECconfigure* configE, char* RGBinFile, char* PSCoutFile)

INPUTS

unsigned char* RGBinFile	File containing video frame to encode (in RGB format with or without Bitmap header)
unsigned char* PSCoutFile	File to output encoded Picture Start Code of current frame unit

OUTPUTS

configE.EncodedBytes	Number of bytes taken to encode Picture Start Code

DESCRIPTION
File based operation - encodes Picture Start Code of current frame unit.

4. F_DecodePSC(CODECconfigure* configD, char* PSCinFile, char* RGBoutFile)

INPUTS

unsigned char* PSCinFile	File containing encoded Picture Start Code to decode
unsigned char* RGBoutFile	File to direct decoded frame unit (in RGB format with or without Bitmap header)

OUTPUTS

configD.DecodedBytes	Number of bytes taken to decode Picture Start Code
configD.DecodeVoid	Set to YES if an error condition is detected

DESCRIPTION
File based operation - decodes Picture Start Code of current frame unit.

Supplies Decoder with the filename to direct output of decoded frame unit

5. F_EncodeGOB(CODECconfigure* configE, char* GOBoutFile)

INPUTS

unsigned char* GOBoutFile	File to output encoded GOB unit

OUTPUTS

configE.EncodedBytes	Number of bytes taken to encode GOB unit

DESCRIPTION
File based operation - encodes a single GOB unit.

6. F_DecodeGOB(CODECconfigure* configD, char* GOBinFile)

INPUTS

unsigned char* GOBinFile	File containing GOB unit to decode

OUTPUTS

configD.DecodedBytes	Number of bytes taken to decode GOB unit
configD.DecodeVoid	Set to YES if an error condition is detected

DESCRIPTION
File based operation - decodes a singe GOB unit.

4 FORMAT OF VIDEO FRAME DATA

Encoder input or Decoder output are frames of video stored in memory or file in RGB format. The CODECconfigure member, IncludeBMPheaders, informs the CODEC whether this data is raw RGB data only or includes Bitmap header information.

4.1 VIDEO FRAME STORED IN MEMORY

The video frame stored in memory should be in the format of the CODEC defined Bitmap Object.
This BMPobject is declared in config.hpp.

struct BMPobject {
Bitmap Header Information

unsigned char identifier[2]	'B' followed by 'M' to identify valid bitmap
unsigned long size_in_bytes	size, in bytes, of the bitmap object
unsigned short reserved[2]	reserved bytes (both must be 0)
unsigned long offset	offset, in bytes, to start of image data (54)
unsigned long size	number of bytes required by header (40)
unsigned long width	picture width in pixels
unsigned long height	picture height in pixels
unsigned short planes	biPlanes - this value must be set to 1
unsigned short bit_count	bits per pixel
unsigned long compression	type of compression – zero for no compression
unsigned long image_size	image size in bytes
unsigned long x_pixels	horizontal resolution, in pixels per meter
unsigned long y_pixels	vertical resolution, in pixels per meter
unsigned long number_colors	number of color indexes used – zero for all
unsigned long colors_important	if this value is zero, all colors are required

Raw RGB data

RGBTRIPLE * t	picture data, stored as an array of RGBTRIPLE structures

};

The **RGBTRIPLE** structure describes the colour consisting of relative intensities of red, green, and blue.

```
typedef struct tagRGBTRIPLE {
   BYTE rgbtBlue;
   BYTE rgbtGreen;
   BYTE rgbtRed;
} RGBTRIPLE;
```

It is the responsibility of the calling application to present the video frame in memory in the above format. If the Bitmap header is not included the CODEC will only extract the raw RGB data.

4.2 VIDEO FRAME STORED IN FILE

If the video frame stored in a file is to include the Bitmap header, it should include the same bitmap header information as listed for the bitmap object, followed by the raw picture data. The RGB values for each pixel should be stored in the file in the following order, blue followed by green then red.
When the bitmap header is not included the file should only contain the blue, green and red RGB values for each pixel.

PLEASE NOTE
The CODEC will only accept video frames stored as BottomUp, RGBTRIPLE bitmaps.

5 USE OF SEPARATE GOB BUFFERS

The CODEC provides the following constants in config.hpp header:

```
#define H261QCIFGOBS     3
#define H261CIFGOBS     12
#define H263QCIFGOBS     9
#define H263CIFGOBS     18
```

To encode/decode a frame unit using separate GOB buffers:

- If using H.263, set configuration option H263GobSync to ENABLED.
- Call the appropriate API function to encode/decode the Picture Start Code.
- Determine how many GOBs are present. This is dependent upon the CODEC type and resolution. The calling application can use the constants listed above. The API function to encode/decode a single GOB unit should then be called for each GOB present.

6 SOURCE FILES AND FUNCTIONS

6.1 SOURCE FILES

C++ source files:

Filename	Description
h261codec.cpp	h261Codec Class and derived h261Encoder and h261Decoder classes
h261fsource.cpp	h261FrameSource Class: Frame input and output
h261fstore.cpp	h261FrameStore Class : H.261 frame handling routines
h261vframe.cpp	h261VideoFrame Class : manages video frame YUV data
h261iocodec.cpp	h261IOcodec Class : coded data IO manipulations
h261rlcoding.cpp	h261Rlcoding Class : VLC routines
h261picture.cpp	h261Picture Class: Controls H.261 picture header data
h261gob.cpp	h261Gob Class : Group of Blocks handling

Filename	Description
h261mblock.cpp	h261Mblock Class : macroblock handling
h261mpredict.cpp	h261Mpredict Class: Determines ENCODER Motion Prediction. Performs ENCODER/DECODER motion compensation and picture reconstruction.
h261dct.cpp	h261DCT Class: DCT, Quantization, Zigzag translation routines
h261rate.cpp	h261Rate Class: Handles video codec rate control – determines quantization step
h263codec.cpp	h263Codec Class and derived h263Encoder and h263Decoder classes
h263fsource.cpp	h263FrameSource Class: Frame input and output
h263fstore.cpp	h263FrameStore Class : H.263 frame handling routines
h263vframe.cpp	h263VideoFrame Class : manages video frame YUV data
h263iocodec.cpp	h263IOcodec Class : coded data IO manipulations
h263tables.cpp	h263Tables Class : VLC routines
h263picture.cpp	h263Picture Class: Controls H.263 picture header data
h263gob.cpp	h263Gob Class : Group of Blocks handling
h263mblock.cpp	h263Mblock Class : macroblock handling
h263mpredict.cpp	h263Mpredict Class: Determines ENCODER Motion Prediction. Performs ENCODER/DECODER motion compensation and picture reconstruction.
h263dct.cpp	h263DCT Class: DCT, Quantization, Zigzag translation routines
h263rate.cpp	h263Rate Class: Handles video codec rate control – determines quantization step

Header files:

Filename	Description
config.hpp	Header file required by calling application. Includes CODECconfigure and BMPobject declarations. Defines CODEC operational constants
h261codec.hpp	h261Codec Class and derived h263Encoder and h263Decoder classes
h261dct.hpp	h261DCT Class
h261fsource.hpp	h261FrameSource Class
h261fstore.hpp	h261FrameStore Class
h261gob.hpp	h261Gob (Group of Blocks) Class
h261iocodec.hpp	h261IOCodec Class
h261mblock.hpp	h261Mblock (Macroblock) Class
h261mpredict.hpp	h261Mpredict (Motion Prediction) Class
h261picture.hpp	h261Picture Class
h261rate.hpp	h261Rate (Rate Control) Class
h261rlcoding.hpp	h261RLcoding (Variable Length Codes) Class
h261vframe.hpp	h261VideoFrame Class
h263codec.hpp	h263Codec Class and derived h263Encoder and h263Decoder classes
h263dct.hpp	h263DCT Class
h263fsource.hpp	h263FrameSource Class
h263fstore.hpp	h263FrameStore Class
h263gob.hpp	h263Gob (Group of Blocks) Class
h263iocodec.hpp	h263IOCodec Class
h263mblock.hpp	h263Mblock (Macroblock) Class
h263mpredict.hpp	h263Mpredict (Motion Prediction) Class
h263picture.hpp	h263Picture Class
h263rate.hpp	h263Rate (Rate Control) Class
h263tables.hpp	h263Tables (Variable Length Codes) Class
h263vframe.hpp	h263VideoFrame Class

6.2 CLASSES AND MEMBER FUNCTIONS

Note: where two versions of a function are listed (e.g. InitMBlock (1) and InitMBlock (2)), the version used depends on the parameter list supplied by the calling function (polymorphism).

6.2.1 H.261 CLASSES AND FUNCTIONS

Class: h261Encoder
Source File: h261codec.cpp

Function	Description
h261Encoder	Constructor
CODECinitialise	API function called to update CODEC internal variables after any change in option settings
ConfigEncoder	API function called to configure the CODECconfigure structure to default Encoder settings
M_EncodeFrame	API Function called to Encode one frame from/to memory
F_EncodeFrame	API Function called to Encode one frame from/to file
EncFrame	Encodes one complete frame unit
M_EncodePSC	API Function called to input the current frame from memory, encode and output the Picture Start Code
F_EncodePSC	API Function called to input the current frame from file, encode the Picture Start Code and output to file
M_EncodeGOB	API Function called to encode a gob unit from/to memory
F_EncodeGOB	API Function called to encode a gob unit from/to file

Class: h261Decoder
Source File: h261codec.cpp

Function	Description
h261Decoder	Constructor
CODECinitialise	API function called to update Decoder internal variables after any change in option settings
ConfigDecoder	API function called to configure the CODECconfigure structure to default Decoder settings
M_DecodeFrame	API Function called to Decode one frame from/to memory
F_DecodeFrame	API Function called to Decode one frame from/to file
M_DecodePSC	API Function called to decode the PSC from memory and set internal parameters to direct Decoder output to memory
F_DecodePSC	API Function called to decode the PSC from file and set internal parameters to direct Decoder output to file
M_DecodeGOB	API Function called to decode a GOB unit from/to memory
F_DecodeGOB	API Function called to decode a GOB unit from/to file
DecFrame	Decodes one complete frame unit
WriteImage	Outputs decoded image

Class: h261Picture
Source File: h261picture.cpp

Function	Description
h261Picture	Constructor.
InitPicture	Initialises Encoder h261Picture object for new configuration
TransmitP	Determines if picture is to be dropped or transmitted
GenerateTempRef	Generates temporal reference timestamp for each new frame
EncodePicture	Outputs picture header information
DecodePHeader	Decodes Picture header information
GetTempRef	Updates Decoder current Temporal Reference information

Class: h261FrameSource
Source File: h261fsource.cpp

Function	Description
MemFrameSource	Updates h261FrameSource for memory operation
FileFrameSource	Updates h261FrameSource for file operation
F_ReadImage	Inputs file-based image in RGB format and stores it in h261VideoFrame
F_WriteImage	Outputs the current h261VideoFrame image to a file in RGB format
M_ReadImage	Gets an image from bitmap object in memory and stores it in h261VideoFrame
M_WriteImage	Outputs current h261VideoFrame image data to bitmap object in memory

Class: h261Dct
Source File: h261dct.cpp

Function	Description
h261Dct	Constructor
ForwardDct	Performs Forward Dct on one block
InverseDct	Performs Chen-Wang Inverse Dct on one block
Quantize	Quantize and clip output coefficients from ForwardDct. Keeps reference block copy for MB reconstruction.
Zigzag	Performs zigzag translation.
InverseZigzag	Performs Inverse Zigzag Translation.
GetRefData	Inputs reference data for block reconstruction – ENCODER
InverseQuantize	Inverse quantize on current block

Class: h261FrameStore
Source File: h261fstore.cpp

Function	Description
h261FrameStore	Constructor
InitFrameStore	Updates h261FrameStore internal variables after any change in option settings
InitConversionConstants	Initializes the arrays used in yuv_bmp conversions
LocateMB	Locates the coordinates of the current Mblock
SwitchFrames	Switches current and previous frames
GetCurrentImage	Returns reference to current video frame
GetPreviousImage	Returns reference to previous video frame
GetNewImage	Gets a new frame from FrameSource
PutCurrentImage	Updates temporal reference of current picture and outputs current frame via h261FrameSource

Class: h261Gob
Source File: h261gob.cpp

Function	Description
h261Gob	Constructor
InitGob	Updates h261Gob object for change in resolution
EncodeGOB	Collects current data and outputs gob header information
ReadGobHeader	Decodes Gob header information
ReadGobHinfo	Decodes Gob Gquant and GEI/GSpare info

Class: h261IOcodec
Source File: h261iocodec.cpp

Function	Description
h261IOcodec	Constructor
MemIOcodec	Updates h261IOcodec object for memory based operation
FileIOcodec	Updates h261IOcodec object for file based operation
CloseFile	Closes current h261IOcodec file
ReadBit	Reads one bit from stream, left shifts current code by 1 and inserts new bit into the right most bit of the 16 bit word
WriteToStream	Packs then writes one char at a time from header/code data to the coded stream
ReadStream	Reads one char at a time the requested number of bits from coded stream
ReadSCode	Read first 16 bits of start code
ReadSCodeTrailer	Reads start code trailer. If trailer is not valid this function stays in the search loop until it finds a valid start code followed by a valid trailer.
PutFlush	Flushes unused bits to allow byte write to stream
RemoveFlushBits	Removes flush bits from internal Buffer when code is forced to be byte aligned during separate GOB operation
CheckEOF	Throws exception if end of stream input file has been reached

Class: h261RLcoding
Source File: h261rlcoding.cpp

Function	Description
h26RLcoding	Constructor
TcoeffTable	Set up TCoeff look-up table
RLEncode	Uses Tcoeff look-up table to output Rlcodes for block data
FindVLC	Finds VLC for given combination of zero-run level
FindRunLevel	Finds run/level combination for Vlcode
DecodeTCoeff	Decodes VLC for block TCoeffs
MakeVLCode	Determines new code value and bit number
ExtractRLvalue	Extracts run and level values from RL combined code

Class: h261Mblock
Source File: h261mblock.cpp

Function	Description
h261Mblock	Constructor
InitMBlock (1)	Updates Encoder h261MBlock object for new configuration
InitMBlock (2)	Updates Decoder h261MBlock object for new configuration
SetMBTypes	Sets to zero all data used to determine the current Mtype
MBATable	Set up MBA look-up table.
MtypeTable	Set up MTYPE look-up table.
MVDTable	Set up MVD look-up table
CBPTable	Set up CBP look-up table
PredictMB	Determines MBpredict type. If Inter – calls Mpredict for motion compensation.
HeaderMB	Builds current MBlock header then calls h261IOcodec object to output header information
Find	Attempts to find a match for the given search item. Returns table index when match is found
DecodeVLC	Attempts to find a match for the given code. Returns the decoded value taken from table
GetMType	Determines current MType number
EncodeMB	Performs forward DCT, quantization and Zigzag Translation on one Mblock. Reconstructs data for picture reference.
EncoderRefMB	Reconstructs Encoder reference MBlock

BlockCBP	Determines whether any blocks hold transform coefficients then calculates coded block pattern.
FindMBdiff	Determines diff between current and previous MBno. If diff is greater than 1 then codec knows that MBs have not been transmitted.
DetermineMVdata	Computes the current horizontal and vertical motion vector data from the decoded values.
SkipMB	Determines if MBs have been skipped in current gob. If so copies skipped data into current stores.
DecodeMB	Decodes current MBlock unit
SetVar	Used in Encoder and Decoder. Sets lastMB variables to their current values.
ResetMB	Before Decoder moves on to the next Gob, checks if any MBs have been skipped at the end of this current gob. If so, copies skipped data into current Frame. Sets MBno and lastMBno.
DecodeMBheader	Decodes Macroblock VLC header information
DecodeTable	Decodes MBA or MVD look up table
DecodeMType	Decodes MType code
DecodeCBP	Decodes CBP VLC table to get current CBPnumber
RLcodeMB	Coordinates RLcoding of blocks with non zero coeff data

Class: h261Rate
Source File: h261rate.cpp

Function	Description
h261Rate	Constructor
InitRate	Updates Rate object for new configuration
GetQuant	Determines new GQuant/MQuant value
UpdateQuant	Updates UseQuant to newly decoded quantization value

Class: h261Mpredict
Source File: h261mpredict.cpp

Function	Description
h261Mpredict	Constructor.
InitMPredict	Updates h261MPredict object for new picture format
Mestimate	Searches for best matching block between current and previous block data and calculates difference MB values. Using Parallel Hierarchical One-Dimensional Search (PHODS).
FindMinD	Computes sum of absolute values of the differences between corresponding row and column projection values
MBreconstruct	Motion compensates for all blocks comprising the current MBlock
GetMBlocation	Gets current MB position for the decoder
CopyPrevMB	FOR UNCODED OR SKIPPED MBLOCKs - finds position of ref area in previous frame and copies filtered/non-filtered values into the current MB position
MBfilter	2D Filter applied to reference area in previous frame

Class: h261VideoFrame
Source File: h261vframe.cpp

Function	Description
h261VideoFrame	Constructor.
InitVideoFrame	Updates VideoFrame object for new configuration
SetStores	Dynamic memory allocation of Y,U,V stores
SetStoreZero	Initialises store elements to zero
~h261VideoFrame	Destructor.
ReadBMP	Reads contents from istream in bmp format to memory in Y,U,V format (File based input with or without BMP header).

WriteBMP	Writes frame contents from memory (Y,U,V format) to ostream in bmp format (File based output with or without BMP header).
GrabBMP	Reads bmp data(RGB) from memory to Y,U,V format. Memory based input - only extracts raw RGB data.
MakeBMP	Outputs current image data to memory as bitmap object(RGBvalues) with or without Bitmap header.

6.2.2 H.263 Classes and Functions

Class: h263Encoder
Source File: h263codec.cpp

Function	Description
h263Encoder	Constructor
CODECinitialise	API function called to update Encoder internal variables after any change in option settings
ConfigEncoder	API function called to configure the CODECconfigure structure to default Encoder settings
M_EncodeFrame	API Function called to Encode one frame from/to memory
F_EncodeFrame	API Function called to Encode one frame from/to file
EncFrame	Encodes one complete frame unit
M_EncodePSC	API Function called to input the current frame from memory, encode and output the Picture Start Code
F_EncodePSC	API Function called to input the current frame from file, encode the Picture Start Code and output to file
M_EncodeGOB	API Function called to encode a gob unit from/to memory
F_EncodeGOB	API Function called to encode a gob unit from/to file

Class: h263Decoder
Source File: h263codec.cpp

Function	Description
h263Decoder	Constructor
CODECinitialise	API function called to update Decoder internal variables after any change in option settings
ConfigDecoder	API function called to configure the CODECconfigure structure to default Decoder settings
M_DecodeFrame	API Function called to Decode one frame from/to memory
F_DecodeFrame	API Function called to Decode one frame from/to file
M_DecodePSC	API Function called to decode the PSC from memory and set internal parameters to direct Decoder output to memory
F_DecodePSC	API Function called to decode the PSC from file and set internal parameters to direct Decoder output to file
M_DecodeGOB	API Function called to decode a GOB unit from/to memory
F_DecodeGOB	API Function called to decode a GOB unit from/to file
DecFrame	Decodes one complete frame unit
WriteImage	Outputs decoded image

Class: h263Picture
Source File: h263picture.cpp

Function	Description
h263Picture	Constructor.
InitPicture	Initialises Encoder h263Picture object for new configuration
TransmitP	Determines if picture is to be dropped or transmitted
GenerateTempRef	Generates temporal reference timestamp for each new frame

GetNoRateTR	Gets temporal reference when rate control is off
EncodePicture	Outputs picture header information
PictureEOS	Outputs byte aligned End Of Sequence code
DecodePHeader	Decodes Picture header information
GetTempRef	Updates Decoder current Temporal Reference information
ComparePType	Returns true if PType in current Picture header differs from PType of the previous Picture header

Class: h263FrameSource
Source File: h263fsource.cpp

Function	Description
MemFrameSource	Updates h263FrameSource for memory operation
FileFrameSource	Updates h263FrameSource for file operation
F_ReadImage	Inputs file-based image in RGB format and stores it in h263VideoFrame
F_WriteImage	Outputs the current h263VideoFrame image to a file in RGB format
M_ReadImage	Gets an image from bitmap object in memory and stores it in VideoFrame
M_WriteImage	Outputs current h263VideoFrame image data to bitmap object in memory

Class: h263Dct
Source File: h263dct.cpp

Function	Description
h263Dct	Constructor
ForwardDct	Performs Forward Dct on one block
InverseDct	Performs Chen-Wang Inverse Dct on one block
Quantize	Quantize and clip output coefficients from ForwardDct. Keeps reference block copy for MB reconstruction.
Zigzag	Performs zigzag translation.
InverseZigzag	Performs Inverse Zigzag Translation.
GetRefData	Inputs reference data for block reconstruction – ENCODER
InverseQuantize	Inverse quantize on current block

Class: h263FrameStore
Source File: h263fstore.cpp

Function	Description
h263FrameStore	Constructor
InitFrameStore	Updates h263FrameStore internal variables after any change in option settings
InitConversionConstants	Initializes the arrays used in yuv_bmp conversions
LocateMB	Locates the coordinates of the current Mblock
SwitchFrames	Switches current and previous frames
GetCurrentImage	Returns reference to current video frame
GetPreviousImage	Returns reference to previous video frame
GetNewImage	Gets a new frame from FrameSource
PutCurrentImage	Updates temporal reference of current picture and outputs current frame via h263FrameSource

Class: h263Gob
Source File: h263gob.cpp

Function	Description
h263Gob	Constructor
EncodeGOB	Collects current data and outputs gob header information
ReadGobHeader	Decodes Gob header information

Class: h263IOcodec
Source File: h263iocodec.cpp

Function	Description
h263IOcodec	Constructor
~h263IOcodec	Destructor
MemIOcodec	Updates h263IOcodec object for memory based operation
FileIOcodec	Updates h263IOcodec object for file based operation
CloseFile	Closes current h263IOcodec file
FindNextSCode	"Scan" through bitstream to find next valid 17-bit start code
WriteToStream	Packs then writes one char at a time from header/code data to the coded stream
ReadStream	Reads one char at a time the requested number of bits from coded stream
ReadSCode	Read valid 17-bit start code (if not valid, call FindNextSCode to search for next valid code)
ReadSCodeTrailer	Reads 5 bit start code trailer
PutFlush	Flushes unused bits to allow byte write to stream
RemoveFlushBits	Removes flush bits from internal Buffer when code is forced to be byte aligned during separate GOB operation
CheckEOF	Throws exception if end of stream input file has been reached

Class: h263Tables
Source File: h263tables.cpp

Function	Description
h263Tables	Constructor
InitMCBPC_I	Initialises VLC look-up table for MCBPC (I-pictures)
InitMCBPC_P	Initialises VLC look-up table for MCBPC (P-pictures)
InitCBPY	Initialises VLC look-up table for CBPY
InitMVD	Initialises VLC look-up table for MVD
InitTCOEF	Initialises VLC look-up table for TCOEF
EncodeMCBPC_I	Finds the VLC for MCBPC (I-pictures)
EncodeMCBPC_P	Finds the VLC for MCBPC (P-pictures)
EncodeCBPY	Finds the VLC for CBPY
EncodeMVD	Finds the VLC for motion vector data
EncodeTCOEF	Finds the VLC for transform coefficients
FindTCOEF_VLC	Searches TCOEF look_up table for given combination of parameters. Returns VLC value or zero if no match is found.
DecodeMCBPC_I	Decodes MCBPC VLC (I-pictures)
DecodeMCBPC_P	Decodes MCBPC VLC (P-pictures)
DecodeCBPY	Decodes CBPY VLC
DecodeMVD	Decodes motion vector data VLC
DecodeMVD_VLC	Searches MVD table
DecodeTCOEF	Decodes transform coefficient VLCs for one block unit
WriteTCOEF	Reads in sign bit from bitstream, looks up TCOEF table to find values for RUN and LEVEL, then enters the transform coefficients into the block store

Class: h263Mblock
Source File: h263mblock.cpp

Function	Description
h263Mblock	Constructor
InitMBlock (1)	Updates Encoder h263MBlock object for new configuration
InitMBlock (2)	Updates Decoder h263MBlock object for new configuration
~h263Mblock	Destructor
ResetMB_type	Sets to zero all data used to determine the current Mtype
PredictMB	Determines MBpredict type. If INTRA - copies data into macroblock unit store.

HeaderMB	Builds current MBlock header then calls h263IOcodec object to output header information
GetMType	Determines current MType number
EncodeMB	Performs forward DCT, quantization and Zigzag Translation on one Mblock. Reconstructs data for picture reference.
BlockCBP	Determines whether any blocks hold transform coefficients then calculates CBPC and CBPY
GetMVdiff	Encoder function - called by INTER coded MB to determine MVD vector difference values used to obtain MVD VLC
GetPredictors	Determines motion vector predictors in both the Encoder and Decoder
GetMVvector	Decoder function - determines motion vectors for current INTER coded macroblock
DecodeMB	Decodes and reconstructs coded MBlock unit
SetPredictor	Sets PredictorFlag for next macroblock
ResetMVdata	Called before starting each new row of macroblocks. Resets MVdata pointers for current and previous MVrows.
DecodeMBheader	Decodes Macroblock VLC header information
MakeCBParray	Determines which blocks hold transform coefficients then stores the information in an array
EncodeBlockData	Encodes macroblock coefficients
FindMB	Determines location of current macroblock
Get_picture_type	Sets MB variable to current Picture Coding Type (required to access correct VLC tables).

Class: h263Rate
Source File: h263rate.cpp

Function	Description
h263Rate	Constructor
InitRate	Updates Rate object for new configuration
GetQuant	Determines current Quantization step (PQuant and GQuant).
GetDQuant	Determines Dquant code (difference between previous and current Quant values).
SetQuant	Sets cur_Quant to value of newly decoded PQuant/GQuant (decoder only).
GetMB_Quant	Takes newly decoded DQuant code, determines the new current Quant value (decoder only).
UpdateRateControl	Updates Virtual Buffer when Rate Control is enabled

Class: h263Mpredict
Source File: h263mpredict.cpp

Function	Description
h263Mpredict	Constructor.
InitMPredict (1)	Updates Encoder h263MPredict object for new configuration
InitMPredict (2)	Updates Decoder h263MPredict object for new configuration
~h263Mpredict	Destructor
Mestimate	Searches for best matching block in the reference picture. If prediction is INTER - determines MV components and calculates difference MB. Uses Logarithmic Search.
HalfPixelEstimate	Determines half-pixel accurate motion vector
FindMinSAD	Computes sum of absolute difference between Mblocks
INTRA_MBreconstruct	Reconstructs INTRA macroblock unit
INTER_MBreconstruct	Reconstructs INTER macroblock unit
GetChrominanceMV	Determines chrominance motion vector components
GetUVdiffMB	Computes difference block values for chrominance components
GetMBlocation	Gets current MB position for the decoder
CopyPrevMB	For non-coded macroblocks, finds corresponding macroblock in previous frame and copies values into the current MB position.

Class: h263VideoFrame
Source File: h263vframe.cpp

Function	Description
h263VideoFrame	Constructor.
InitVideoFrame	Updates VideoFrame object for new configuration
SetStores	Dynamic memory allocation of Y,U,V stores
SetStoreZero	Initialises store elements to zero
~h263VideoFrame	Destructor.
ReadBMP	Reads contents from istream in bmp format to memory in Y,U,V format (File based input with or without BMP header).
WriteBMP	Writes frame contents from memory (Y,U,V format) to ostream in bmp format (File based output with or without BMP header).
GrabBMP (1)	Reads bmp data(RGB) from memory to Y,U,V format. Memory based input - only extracts raw RGB data.
MakeBMP	Outputs current image data to memory as bitmap object(RGBvalues) with or without Bitmap header.
GrabBMP (2)	Reads bmp data(RGB) from memory to Y,U,V format. Access RGB values directly from the bitmap object.

6.3 TESTBENCH

Testing is carried out using the source file **testbench.cpp**

The following operations are carried out by the testbench:

- create **either** h263Encoder and h263Decoder objects **or** h261Encoder and h261Decoder objects
- initialise the configuration structures of each object for default operation
- make required modifications to the configuration options then update CODEC by calling CODECinitialise
- you require a video sequence saved as bitmap files
- carry out **one** of the following tests (the code for the other 7 tests should be commented out):
  - H.263:
    - Test 1 (file-based encoding/decoding of complete frames): encode each bitmap file, decode the resulting bit stream file
    - Test 2 (file-based encoding/decoding of separate GOBs): encode each bitmap files (one GOB at a time), decode the resulting bit stream file
    - Test 3 (memory-based encoding/decoding of complete frames): create bitmap objects (from bitmap files), encode and decode from/to memory, convert the resulting bitmap objects to bitmap files.
    - Test 4 (memory-based encoding/decoding of separate GOBs): create bitmap objects (from bitmap files), encode and decode (one GOB at a time) from/to memory, convert the resulting bitmap objects to bitmap files.
  - H.261:
    - Test 5 (file-based encoding/decoding of complete frames): encode each bitmap file, decode the resulting bit stream file
    - Test 6 (file-based encoding/decoding of separate GOBs): encode each bitmap files (one GOB at a time), decode the resulting bit stream file
    - Test 7 (memory-based encoding/decoding of complete frames): create bitmap objects (from bitmap files), encode and decode from/to memory, convert the resulting bitmap objects to bitmap files.
    - Test 8 (memory-based encoding/decoding of separate GOBs): create bitmap objects (from bitmap files), encode and decode (one GOB at a time) from/to memory, convert the resulting bitmap objects to bitmap files.

- to test option changes during a video sequence, make the required modification to the configuration options within the test loop, then update CODEC by calling CODECinitialise

The following utility functions are declared within testbench.cpp:

InitBMPobject	Creates BMP object from BITMAP file (with/without header)
ReconstructImage	Reconstructs BITMAP file from BMP object (with or without header)

6.4 DEBUG DIAGNOSTIC COMMENTS

In order to aid development and confirm correct operation during testing, diagnostic comments can be directed to the file "codecDEBUG.txt". This option should only be used during debugging. It is switched off in the release version.
To switch ON diagnostic comments:

BORLAND VERSION

- right click node on project.ide
- select edit local options
- select Topics: Compiler, Defines
- type DIAG
- select OK

MICROSOFT VISUAL C++ VERSION

- create debug version
- select Project from Main Menu then click on Settings. This produces the Project Settings dialog box.
- select the C/C++ folder, category General and add DIAG into the Preprocessor definitions edit box
- click on OK to confirm and exit
- rebuild project

6.5 ERROR DETECTION

If the CODEC detects an error condition it will flag the calling application by switching EncodeVoid or DecodeVoid ON. It is the responsibility of the calling application to decide on an appropriate course of action for error recovery.

6.5.1 ERROR CONDITION IN H.261

The H.261 CODEC does not transmit macroblocks that contain no information for that part of the picture. This may result in macroblocks not being transmitted at the end of the last GOB unit in the current frame. The Decoder has no way of knowing this and may exit the DecodeFrame or DecodeGOB API function call with the error condition flag, DecodeVoid, switched ON. This is not necessarily an error but is a feature of the H.261 decoding decision process.

7 REFERENCES

[1] ITU-T Recommendation H.261, "Video CODEC For Audiovisual Services At p x 64 kbits", March 1993

[2] ITU-T Recommendation H.263, "Video Coding For Low Bitrate Communication", Version 1, May 1996