Vela CineCast®
API Developer's Guide
Version 2.6

An Application Programming Interface handbook for Vela's CineCast line of SCSI-2 MPEG-2 decoders that run on the Microsoft® Windows® operating system.

Includes support for the following Vela SCSI-2 MPEG-2 audio/video decoders:

- ***CineCast Prime***
- ***CineCast Quad Pro***
- ***CineCast Quad***
- ***CineCast HD***
- ***CineCast 8-Channel***

Document Part Number 9050-2016

Returns must be accompanied by an authorized RMA number obtained from Vela.

Vela OEM Products Division
5733 Myerlake Circle
Clearwater, FL 33760-2804
Phone: (727) 507-5300
Fax: (727) 507-5311

World Wide Web – http://www.vela.com

Shipping / Mailing Address:
5733 Myerlake Circle
Clearwater, FL 33760-2804

Printed in the United States of America 6-02. CCAPI-0204-006

Table of Contents

List of Figures

Chapter 1

Getting Started

Introduction

Welcome to version 2.6 of the Vela Application Programming Interface (API) for the Vela CineCast® line of SCSI-2 audio/video decoders. This guide is applicable to all CineCast SCSI-2 products, except where noted. This document is primarily intended for application developers designing digital video delivery systems.

This chapter discusses system requirements and installation issues. Chapter 2 describes in some detail a number of various functions and routines that the typical application developer uses in his or her development efforts.

Operating System

- Microsoft® Windows® 2000 or Windows NT™ 4.0 (NT Service Pack 6a or later).

Supported Development Systems

- Microsoft Visual C++™ 6.0.
- Microsoft Visual Basic™ 6.0.

System Requirements

Proper operation of CineCast SCSI-2 decoders requires the following minimum system configuration:

IMPORTANT: The PC system must be Windows certified by the system manufacturer in order for Vela to guarantee proper operation of its products. See your dealer for details on Windows certification.

- IBM®-compatible personal computer with PCI bus support and a Pentium® processor, 200 MHz or faster.
- Minimum of 32MB RAM memory.
- Microsoft Windows 2000 or Windows NT 4.0 (SP 6a or later).
- SCSI or IDE hard drive for Windows operating system (an IDE drive is recommended to prevent possible conflicts).

- A separate SCSI-2 hard drive for MPEG file storage and transfer. The drive must support a minimum sustained data rate of 10Mbps and must not contain the operating system. See Setup Note at the end of this bulleted list.
- CD-ROM Drive.
- Floppy disc drive, 3-½ inch.
- Video monitor or television receiver with video input connector.
- Audio amplifier (not required if audio input connector is available on the video monitor or television receiver). See illustrations at the end of this chapter for audio connector pinout information and typical audio cable configurations.

Setup Note: The system should be set up so that the virtual memory file (page-file.sys) is on the drive containing the Windows operating system. Paging files must not be set on the drive(s) containing the MPEG-encoded video files, since the operating system will perform paging when there is not enough memory to start or complete a process or in the process of general housekeeping. Paging will generally "take over" disk I/O activity during this time and will frequently and momentarily halt the transfer of MPEG files to the decoder board. Therefore, MPEG files should be kept on a separate drive for greatest efficiency.

Decoder Installation

If not already installed, refer to the installation and user manual for your Vela CineCast decoder for complete information and instructions on proper installation.

Software Installation

Be sure to review the "readme" files on the system software CD-ROM for the very latest information on installation and performance issues.

Vela CineCast Software Development Kit system software installation is a relatively straightforward process. Follow the instructions in your particular CineCast decoder ver. 2.6 user manual for installation of both CineCast system software and the SDK. If you did not receive a hard copy manual, the document is available in PDF form on the Vela System Software CD-ROM that accompanied either the decoder or this SDK.

Hardware drivers are automatically installed under Windows NT 4.0. Instructions for installing them under Windows 2000 can be found in the above-referenced installation and user manual.

Customer Support

In the event of questions or problems with the Vela Application Programming Interface materials or this manual, do not hesitate to contact Vela Training and Support.

- Phone: (727) 507-5301
- E-mail: support@vela.com
- World Wide Web - http://www.vela.com

Chapter 2

API Development

The following functions used in this section pertain to Vela CineCast SCSI-2 MPEG-2 decoders designed and manufactured by Vela LP.

AudioChannelFade
AudioGain
AVChannelSelect
AVFade
AVStreamID
AVSynch
AVTransportPacketID
BackgroundColor
BackToBackVideo
BlackLevel
Blank
BlankLevel
BufferReset
ClosedCaptionReorder
DataUnderflowErrorCount
DecoderDetect
DecoderShutdown
DefaultResolution
DefaultVideoStandard
EnableFreezeOnLast
EnableGUISignal
EnablePlaybackSignal
EnableWaitForCondition
ErrorMask
FactorySettings
FastMotion
FieldFreeze
FifoBufferStatus
FlushChannelBuffer
FlushInline
FrameAdvance
FrameFreeze
FrameReport
GPIN1Bit
GPIN2Bit
GainU
GainV
Genlock
GenlockTiming
GetMPEGFileInfo
GetMpegPath
GetProductID
GetSettings
GetState
HorizontalPhase
Initialize
IsNonStop
LastDecoderCommand
LastPlayBackError
LtcVitcTimeCode
MpegPlay
Mute
NoBlack
NonStop
PALBufferMode
PauseVideo
Play
PlaybackStarted
PlayBit
ProductInfo
QueueBit
ReadAudioLevel
ReadFrameCount
ReadTimeCode
ReadyForNextVideo
RestoreSettings
Resume
ResumeNormal
ResumePlay
RevisionInfo
SaveSettings
SecondChannel
SessionActive
SetEventLevel
SetMpegPath
SetNextVideo
SetOutputDiscrete
SetSettings
SetVideoPassThrough
SetYuvMode
SlowMotion
SoftReset
StandAlone
Stop
SubcarrierPhase
SwitchAudio
TestPattern
TimeCode
TrickModeFrameCount
UnderFlowBit
UpgradeActive
UpgradeFirmWare
UpgradeResult
UnusedFiFoDataAfterStop
VideoLevel
VolumeControl
WaitForCondition

NOTE: Where indicated, some inline functions are supported only by the CineCast Quad Pro decoder. Inline commands are not executed immediately, but are executed "inline." Inline commands will remain pending until the specific event occurs, as defined in InLine Mode. InLine Mode determines by what event the inlined command is activated. InLine Mode is defined as:

0x0 MPEG Data Mode (default)	0x1 Output SMPTE Time Code Mode
0x2 Input LTC Time Code Mode	0x3 Frame Count Mode
0x4 Playback Start	0x5 Playback End
0x6 GPIN1 Active	0x7 GPIN1 Inactive
0x8 GPIN2 Active	0x9 GPIN2 Inactive

Below is a recommended playback process flow for general use

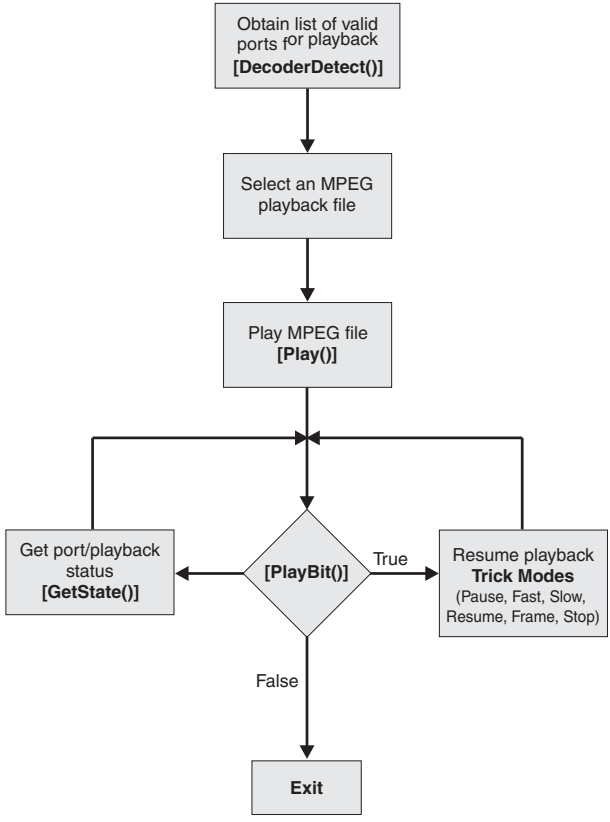

Figure 2-1. Recommended Process Flow

CineCast SDK Q&A

1. What include header files do I use?

A. For C interface implementation:

#include “typedefs.h”
#include “CCmds.h”

(The “typedefs.h” file includes descriptions of all the type definitions required by the CineCast decoders. The “CCmds.h” file includes descriptions for all the CineCast C commands available for the CineCast decoders.)

B. For C++ interface implementation:

#include “typedefs.h”
#include “CDecoder.h”

(The “CDecoder.h” file describes the base interface class which supports all CineCast decoders.)

C. For CineCast Prime and 8-Channel decoders (SCSI-1,SCSI-2, SCSIx8) specific methods:

#include “CCScsi.h”

D. For CineCast Quad and Quad Pro decoders (SCSIx4) specific methods:

#include “CCineCast.h”

E. For CineCast High Definition decoders (SCSI-HD) specific methods:

#include “CCineCastHD.h”

2. What library files do I link with?

The CineCast SDK is implemented in a single Dynamic Link Library, “CineCastU.dll.” The U in the filename refers to the library as Unicode aware. All calls to the library using strings require the strings to be Unicode defined. The project should include the \lib directory in the link tab. CineCastU.lib is the link library used to interface with CineCastU.dll.

3. How do I detect and/or enable the decoders on my system?

For C interface implementation:

```
FUNC_STATUS rc = SUCCESSFUL;
SHORT decoderNum = DecoderDetect();
for (int i = 0; i < decoderNum; i++)
      {
```

```
    rc = Initialize();
    }
```

DecoderDetect() enables the device driver and return the number of MPEG decoders installed in the system. Now any C function may be used as long as a valid Port number (0…n-1) as the first parameter of the function.

For C++ interface implementation:

```
CDecoder ** Channel;
SHORT m_DecoderCount;
FUNC_STATUS rc = SUCCESSFUL;
m_DecoderCount = DecoderCount();
Channel = new CDecoder * [m_DecoderCount];
for ( int i = 0 ; (i < m_DecoderCount) && (rc == SUCCESSFUL) ; i++ )
    {
    Channel[i] = new CDecoder();
    rc = Channel[i]->Open();
    if ( rc == SUCCESSFUL )
    rc = Channel[i]->Initialize();
    }
```

The DecoderCount() returns the number of decoders attached to the local SCSI host adapter. Channel is an array of base class CDecoder objects. After creating each object, Open() method makes the decoder accessible, and Initialize() places the decoder hardware into the last selected mode. For any given entry in Channel, Channel->Open(x) where x is a selected port, enables the programmer to swap the order of the decoders. So instead of the standard Channel[0] pointing to Port 0, Channel[0] could now point to Port 3 instead, allowing for customization of the channel ordering.

4. How do I execute trick modes (Play, Stop, Pause, Fast Forward, Slow Motion, and Frame Advance)?

For C interface implementation:

Play

```
FUNC_STATUS rc = SUCCESSFUL;
SHORT Port = 0;
char *Clip = "D:\\first.mpg";
BOOL ErrorRecovery = TRUE;
BOOL Midstream = FALSE;
BOOL Pause = FALSE;
```

```
BOOL Trigger = FALSE;
STREAM Type = AutoDetect;
UCHAR PreBlack = 1;
ULONG Frames = 0;
WORD PostBlack = 10;
DWORDLONG Offset = 0;
rc = Play( Port , Clip, ErrorRecovery, Midstream, Pause, Trigger, Type,
      PreBlack, Frames, PostBlack, Offset );
```

This call will initialize a playback thread and begin playing the selected clip. Use SessionActive() to determine if the thread is still sending data to the decoder. The thread will terminate before the clip ends, unless WaitForCondition() is enabled.

Stop

```
BOOL Freeze = TRUE;
Stop(Port,Freeze);
```

Pause

```
PauseVideo(Port);
```

Fast Forward

```
USHORT WhichFrame = 5;
FastMotion(Port, WhichFrame);
```

Slow Motion

```
USHORT WhichFrame = 3;
USHORT Repeat = 5;
SlowMotion(Port,WhichFrame,Repeat);
```

Frame Advance

```
FrameAdvance(Port);
```

Resume

```
Resume(Port);
```

For C++ interface implementation:

Same as above, but the calls are issued using the C++ object and no Port number is required, since it was assigned when Open() method was called. For example:

```
CDecoder Channel;
Channel.Open(3);
Channel.Initialize();
Channel.Play( Clip, ErrorRecovery, Midstream, Pause, Trigger, Type, PreBlack,
```

```
        Frames, PostBlack, Offset );
 BOOL Freeze = TRUE;
USHORT WhichFrame = 5;
USHORT Repeat = 5;
Channel.Stop(Freeze);
Channel.PauseVideo();
Channel.FastMotion(WhichFrame);
Channel.SlowMotion(WhichFrame,Repeat);
Channel.FrameAdvance();
Channel.Resume();
```

This will assign Port 3 decoder to Decoder through Open(3), Initialize(), and then Play() the clip.

5. How do I get back the decoder to its factory state?

For C interface implementation:

```
FactorySettings(Port);
Initialize(Port);
```

For C++ interface implementation:

```
Channel.FactorySettings();
Channel.Initialize();
```

6. What does the sample code show?

Three sample applications are included in the installation. The VC directory contains a Visual Studio 6.0 Unicode playback application using the C interface. The VB directory contains a Visual Basic 6.0 playback application using the C interface. The ConsoleCpp directory includes a console application using the C++ interface.

Listing of Public Functions

AudioChannelFade

C++ Methods

FUNC_STATUS Cinecast::AudioChannelFade(SOURCE PrimarySelect, MODE PrimaryStereo, SOURCE SecondarySelect, MODE Secondary-Stereo, BOOL SwapLRChn, SHORT FadeTime, BYTE VideoSwitch, BOOLEAN bInlined, BYTE Mode)

Description

This function controls output selection for the two outputs (primary and secondary) from four possible sources.

Parameters

SOURCE PrimarySelect: Select Primary Audio source.

MODE PrimaryStereo: Select Primary Stereo mode.

SOURCE SecondarySelect: Select Secondary Audio source.

MODE SecondaryStereo: Select Secondary Stereo mode.

BOOL SwapLRChn: Swap Left/Right channels for each source.

SHORT FadeTime: Transition time for fade between clips.

BYTE VideoSwitch: Switch video source.

BOOLEAN bInline:

BYTE Mode:

Return Value

See page 120 for FUNC_STATUS definition.

Comments

- This command is supported only by the CineCast Quad Pro.
- Video switching requires both channels to be genlocked to the same genlock source. VideoSwitch is defined as:
  - 0x0 - No change.
  - 0x2 - Video is switched to MPEG output.
  - 0x3 - Video is switched to MPEG output of adjacent channel pair.
- See page 122 for MODE definition.
- See page 126 for SOURCE definition.
- See page 128 for for Fade Time Table.

AudioGain

C Calls

FUNC_STATUS AudioGain(SHORT Port, WORD PrimaryAudio, WORD SecondaryAudio, BOOL PrimaryHeadroom, BOOL SecondaryHeadroom)

FUNC_STATUS AudioGainInlined(SHORT Port, WORD PrimaryAudio, WORD SecondaryAudio, BOOL PrimaryHeadroom, BOOL SecondaryHeadroom, BOOL binline, BYTE Mode)

C++ Method

FUNC_STATUS CCinecast::AudioGain(WORD PrimaryAudio, WORD SecondaryAudio, BOOL PrimaryHeadroom, BOOL SecondaryHeadroom, BOOL binline, BYTE Mode)

Description

Adjusts audio gain control.

Parameters

SHORT Port: Port number of decoder to accept the command.

WORD PrimaryAudio: Primary Audio gain level in dB.

WORD Secondary Audio: Secondary Audio gain level in dB.

BOOL PrimaryHeadroom: TRUE: 20dB; FALSE: 18dB.

BOOL SecondaryHeadroom: TRUE: 20dB; FALSE: 18dB.

BOOL binline:

FALSE: Commands are executed immediately.
TRUE: Commands will remain pending until the specific event occurs.

BYTE Mode: Determines in what event the inlined command is activated.

Return Value

See page 120 for FUNC_STATUS definition.

Comments

- This command is supported only by the CineCast Quad Pro.

AVChannelSelect

C Call

FUNC_STATUS AVChannelSelect(SHORT Port, SOURCE PrimarySelect,
MODE PrimaryStereo, SOURCE SecondarySelect,
MODE SecondaryStereo, BOOL SwapLRChn, SHORT FadeTime,
BYTE VideoSwitch)

C++ Method

FUNC_STATUS CCinecast::AVChannelSelect(SOURCE PrimarySelect,
MODE PrimaryStereo, SOURCE SecondarySelect,
MODE SecondaryStereo, BOOL SwapLRChn, SHORT FadeTime,
BYTE VideoSwitch)

Description

Controls output selection for the two audio outputs (primary and secondary) from six possible sources.

Parameters

SHORT Port: Port number of decoder to accept the command.

SOURCE PrimarySelect: Select Primary Audio source.

MODE PrimaryStereo: Select Primary Stereo mode.

SOURCE SecondarySelect: Select Secondary Audio source.

MODE SecondaryStereo: Select Secondary Stereo mode.

BOOL SwapLRChn: Swap Left/Right channels for each source.

SHORT FadeTime: Transition time for fade between clips.

BYTE VideoSwitch: Switch video source.

Return Value

See page 120 for FUNC_STATUS definition.

Comments

- This command is supported only by the CineCast Quad Pro.
- Video switching requires both channels to be genlocked to the same genlock source. VideoSwitch is defined as:
  - 0x0 — No change.
  - 0x2 — Video is switched to MPEG output.
  - 0x3 — Video is switched to MPEG output of adjacent channel pair.
- See page 122 for MODE definition.
- See page 126 for SOURCE definition.
- See page 128 for for Fade Time Table.

AVFade

C++ Methods

BOOLEAN Cinecast::AVFade(SOURCE PrimAudio, SOURCE SecAudio, BOOL EnableVideo, BOOL EnableAudio, BOOL FadeToBlack, SELECT VideoSource, BYTE FadeVideo, BYTE FadeAudio, BOOLEAN bInline, BYTE Mode)

Description

This function provides both audio and video fading capability.

Parameters

SOURCE PrimAudio: Select Primary Audio source.

SOURCE SecAudio: Select Secondary Audio source.

BOOL EnableVideo: Enable Video Switch.

BOOL EnableAudio: Enable Audio Switch.

BOOL FadeToBlack: Fade to black during transition.

SELECT VideoSource: Set video source to OWN, CHN1, or ADJ channel.

BYTE FadeVideo: Fade time is specified in terms of video frames.

BYTE FadeAudio: Fade time is specified in terms of video frames.

BOOLEAN bInline:

BYTE Mode:

Return Value

See page 120 for FUNC_STATUS definition.

Comments

- This command is supported on the Models 2000-0422-1 "Dash-1" only.
- It can be also used in place of AUDIOCHANNELFADE().
- With FadeToBlack enabled, the primary video source will fade to black (or background color) and then fade from black (or background color) to the secondary video source.

AVStreamID

C Call
FUNC_STATUS AVStreamID (SHORT Port, AV_STREAM Cmd, BYTE Id)

C++ Method
FUNC_STATUS CDecoder::AVStreamID (AV_STREAM_CMD AvCmd, BYTE AvId)

Description
Selects which audio and video stream IDs to decode, and determines which audio/video stream ID to use. The actual ID must be determined beforehand.

Parameters
SHORT Port: Port number of decoder to accept the command.
AV_STREAM_CMD cmd: Command for all audio and video IDs.
BYTE Id: Stream ID value.

Return Value
See page 120 for FUNC_STATUS definition.

Comments
- See page 118 for AV_STREAM definition.

AVSynch

C Call

FUNC_STATUS AVSynch(SHORT Port, BYTE Coarse, BYTE Medium, BYTE Fine)

C++ Method

FUNC_STATUS CSCSI::AVSynch(BYTE Coarse, BYTE Medium, BYTE Fine)

Description

This function will set the thresholds for the synchronization of video and audio.

Parameters

SHORT Port: Port number of decoder to accept the command.

BYTE coarse: Set Video Frame threshold.

BYTE med: Set Audio Access Unit threshold.

BYTE fine: Set Audio Sample Rate.

Return Value

See page 120 for FUNC_STATUS definition.

Comments

• See "MPEG Audio/Video Synchronization" in the appropriate decoder user manual for explanation of settings.

AVTransportPacketId

C Call

FUNC_STATUS AVTransportPacketId (SHORT Port, PID_TYPE PidAv, ULONG PidValue)

C++ Method

FUNC_STATUS CDecoder::AVTransportPacketId (PID_TYPE PidAv, ULONG PidValue)

Description

This function allows the user to specify the transport packet IDs for video and audio to be decoded.

Parameters

SHORT Port: Port number of decoder to accept the command.

PID_TYPE PidAv: Type of PID to be set for Video/Audio Primary/Secondary and Sequence.

ULONG PidValue: Value to be set for PID selection.

Return Value

See page 120 for FUNC_STATUS definition.

Comments

- See page 123 for PID_TYPE definition.

BackgroundColor

C Call

FUNC_STATUS BackgroundColor(SHORT Port, COLORMODE Mode, BYTE Y, BYTE Cb, BYTE Cr)

C++ Method

FUNC_STATUS CCinecast::BackgroundColor(COLORMODE Mode, BYTE Y, BYTE Cb, BYTE Cr)

Description

Outputs the specified color during idle.

Parameters

SHORT Port: Port number of decoder to accept the command.

COLORMODE Mode: Background color mode

BYTE Y: Color value for Y component

BYTE Cb: Color value for Cb component

BYTE Cr: Color value for Cr component

Return Value

See page 120 for FUNC_STATUS definition.

Comments

- Whenever the decoder is idle (i.e. playback not active), the decoder outputs black. If "Set Background Color" is sent, the decoder will output the specified color. The default is Black, and can be optionally set to blue, or any color specified by the user.
- See page 119 for COLORMODE definition.

BackToBackVideo

C Call

FUNC_STATUS BackToBackVideo(SHORT Port, CHAR * Clip, BOOL ErrorRecovery, BOOL Midstream, BOOL Pause, BOOL Trigger, STREAM Type, UCHAR PreBlack, ULONG Frames, WORD PostBlack, TRIGGER TriggerSource, BYTE Microcode, DWORDLONG Offset)

C++ Method

FUNC_STATUS CDecoder::BackToBackVideo(CHAR * Clip, BOOL ErrorRecovery, BOOL Midstream, BOOL Pause, BOOL Trigger, STREAM Type, UCHAR PreBlack, ULONG Frames, WORD PostBlack, TRIGGER TriggerSource, BYTE Microcode, DWORDLONG Offset)

Description

This function will playback a selected clip immediately following the video now playing.

Parameters

SHORT Port: Port number of decoder to accept the command.

CHAR * Clip: Name of video clip.

BOOL ErrorRecovery: TRUE: Automatically pause when data underflows; FALSE: Stop playback and go to black.

BOOL Midstream: TRUE: Search for next valid sequence header; FALSE: Immediate playback.

BOOL Pause: TRUE: Pause on non-black video frame; FALSE: Immediate playback.

BOOL Trigger: TRUE: Wait for external trigger; FALSE: Immediate playback.

STREAM Type: Stream type of clip.

SHORT PreBlack: Option for number of black frames to display before clip start.

ULONG Frames: Number of frames to playback.

WORD PostBlack: Option for number of black frames to display after clip end.

TRIGGER TriggerSource: Source for event to start playback.

BYTE Microcode: Chroma Microcode to load.

DWORDLONG Offset: Midstream start offset.

Return Value

See page 120 for FUNC_STATUS definition.

Comments

- NOBLACK() disabled will cause MPEGPLAY() to be issued for each clip.
- Microcode is defined as:

  0x0 — IBM
  0x1 — 4:2:0
  0x2 — 4:2:2

- See page 126 for STREAM definition.
- See page 127 for TRIGGER definition.

BlackLevel

C Call
FUNC_STATUS BlackLevel(SHORT Port, SHORT Data)

C++ Method
FUNC_STATUS CSCSI::BlackLevel(SHORT Data)

Description
Sets the Black Level to the desired point.

Parameters
SHORT Port: Port number of decoder to accept the command.
SHORT Data: Range is 0 to 0x1F.

Return Value
See page 120 for FUNC_STATUS definition.

Comments
• Use of this command may drastically reduce picture quality. Only experienced users should attempt to customize the NTSC / PAL video. Consult with Vela for additional information.

Blank

C Calls

Export FUNC_STATUS Blank(SHORT Port , BOOL On)

Export FUNC_STATUS BlankInlined(SHORT Port, BOOL On, BOOL bInline, BYTE Mode)

C++ Methods

Export FUNC_STATUS CDecoder::Blank(BOOL On)

Export FUNC_STATUS CCinecast::Blank(BOOL BlankOn, BOOL binline, BYTE Mode)

Description

This function will turn the video on or off. Playback of the video clip will continue unaffected, even if the video is not visible.

Parameters

SHORT Port: Port number of decoder to accept the command.

BOOL On: TRUE: Blank the screen; FALSE: Turn the screen back on.

BOOL binline:

FALSE: Commands are executed immediately.

TRUE: Commands will remain pending until the specific event occurs.

BYTE Mode: Determines in what event the inlined command is activated.

Return Value

See page 120 for FUNC_STATUS definition.

Comments

None

BlankLevel

C Call
FUNC_STATUS BlankLevel(SHORT Port, SHORT Data)

C++ Method
FUNC_STATUS CSCSI::BlankLevel(SHORT Data)

Description
Sets the Blank Level to the desired point.

Parameters
SHORT Port: Port number of decoder to accept the command.
SHORT Data: Range is 0 to 0x1F.

Return Value
See page 120 for FUNC_STATUS definition.

Comments
• Use of this command may drastically reduce picture quality. Only experienced users should attempt to customize the NTSC / PAL video. Consult with Vela for additional information.

BufferReset

C Calls

FUNC_STATUS BufferReset(SHORT Port)

FUNC_STATUS BufferResetCC(SHORT Port, BOOL FreezeFrame)

C++ Methods

FUNC_STATUS CDecoder::BufferReset()

FUNC_STATUS CCinecast::BufferReset()

FUNC_STATUS CCinecast::BufferReset(BOOL FreezeFrame)

Description

Resets the MPEG buffer pointers before starting the next download of MPEG data to the decoder.

Parameters

SHORT Port: Port number of decoder to accept the command.

BOOL FreezeFrame: Freeze frame on the last decoded picture of the MPEG stream.

Return Value

See page 120 for FUNC_STATUS definition.

Comments

None

ClosedCaptionReorder

C Call
FUNC_STATUS ClosedCaptionReorder(SHORT Port, BOOL Temporal)

C++ Method
FUNC_STATUS CCinecast::ClosedCaptionReorder(BOOL Temporal)

Description
Displays Closed Caption based on bit stream or display order.

Parameters
SHORT Port: Port number of decoder to accept the command.
BOOL Temporal: TRUE: Temporal reference order; FALSE: Bit Order.

Return Value
See page 120 for FUNC_STATUS definition.

Comments
- This command is supported only by the CineCast Quad Pro.

DataUnderflowErrorCount

C Call
WORD DataUnderflowErrorCount(SHORT Port)

C++ Method
WORD CDecoder::DataUnderflowErrorCount()

Description
Returns MPEG Decoder Status Page (30h) value of Data Underflow Error Count.

Parameters
SHORT Port: Port number of decoder to accept the command.

Return Value
Data Underflow Error Count

Comments
None

DecoderDetect

C Call
SHORT DecoderDetect(VOID)

Description
Opens the device driver and returns the number of MPEG decoders installed in the system.

Return Value
Number of decoder boards detected. A value of -1 indicates the device driver and DLL are incompatible.

Comments
• If no decoder is detected, make sure the device driver has been properly installed. Follow the instructions provided in the appropriate user manual for your decoder.

DecoderShutdown

C Call
void DecoderShutdown(SHORT ChannelCount)

Description
Closes each MPEG decoder device opened.

Parameters
SHORT ChannelCount: Total number of decoders returned by DECODERDETECT()

Return Value
None

Comments
None

DefaultResolution

C Calls

FUNC_STATUS DefaultResolution (SHORT Port, VIDEO_RESOLUTION Resolution)

FUNC_STATUS DefaultResolutionCC(SHORT Port, BOOL Pal)

C++ Methods

FUNC_STATUS CDecoder::DefaultResolution (VIDEO_RESOLUTION Resolution)

FUNC_STATUS CCinecast::DefaultResolution(BOOL Pal)

Description

Sets the video standard output to NTSC or PAL. The definition can also be set to C601, SIF, and Half D1. The display resolution may be set to 704×480, 352×240, and 720×576.

Parameters

SHORT Port: Port number of decoder to accept the command.

VIDEO_RESOLUTION Resolution: Selected resolution to display.

BOOL Pal: Flag TRUE for PAL; FALSE for NTSC.

Return Value

See page 120 for FUNC_STATUS definition.

Comments

- See page 127 for VIDEO_RESOLUTION definition.

DefaultVideoStandard

C Call

FUNC_STATUS DefaultVideoStandard(SHORT Port, VIDEO_STANDARD VideoStd)

C++ Method

FUNC_STATUS CSCSI::DefaultVideoStandard (VIDEO_STANDARD VideoStd)

Description

Sets NTSC or PAL standards.

Parameters

SHORT Port: Port number of decoder to accept the command.

VIDEO_STANDARD video_std: NTSC60hz, NTSC50hz, NTSC_M60hz, PAL_BG50hz, PAL_M60hz, PAL_N50hz.

Return Value

See page 120 for FUNC_STATUS definition.

Comments

- See page 128 for VIDEO_STANDARD definition.

EnableFreezeOnLast

C Call
FUNC_STATUS EnableFreezeOnLast(SHORT Port , BOOL Flag)

Description
Enables BufferReset function to freeze frame on the last decoded picture of the MPEG stream.

Parameters
SHORT Port: Port number of decoder to accept the command.
BOOL Flag: TRUE: Freeze Last Frame; FALSE: Fade to black.

Return Value
See page 120 for FUNC_STATUS definition.

Comments
None

EnableGUISignal

C Call

FUNC_STATUS EnableGUISignal(SHORT Port, BOOL Enable,
void (* Callback) (SHORT Port))

C++ Method

FUNC_STATUS CDecoder::EnableGUISignal(BOOL Enable,
void (*Callback))

Description

During playlist supplied (*.vpl) playback, this feature will report at the transition between the present clip and next clip via a supplied callback function.

Parameters

SHORT Port: Port number of decoder to accept the command.

BOOL Enable: Flag TRUE: Enabled; FALSE: Disabled.

void (* Callback) (SHORT nPort)): Pointer to callback function when status return is encountered.

Return Value

See page 120 for FUNC_STATUS definition.

Comments

• Note that C++ Callback does not require a port since the CDecoder object has a port number assigned to it.

EnablePlaybackSignal

C Call

FUNC_STATUS EnablePlaybackSignal(SHORT Port, BOOL Enable, void (* Callback) (SHORT Port))

C++ Method

FUNC_STATUS CDecoder::EnablePlaybackSignal(BOOL Enable, void (*Callback))

Description

During back-to-back playback, this feature will detect a SCSI Status Immediate Condition Met (14h) when the playback transition between the present and next video occurs and report it via a supplied callback function.

Parameters

SHORT Port: Port number of decoder to accept the command.

BOOL Enable: Flag TRUE: Enabled; FALSE: Disabled.

void (* Callback) (SHORT nPort)): Pointer to callback function when status return is encountered.

Return Value

See page 120 for FUNC_STATUS definition.

Comments

• Note that C++ Callback does not require a port since the CDecoder object has a port number assigned to it.

EnableWaitForCondition

C Call

FUNC_STATUS EnableWaitForCondition(SHORT Port , WORD TimeOut, CHAR Complete , BOOL Enable)

C++ Method

FUNC_STATUS CDecoder::EnableWaitForCondition(WORD TimeOut, CHAR Complete, BOOL Enable)

Description

Activates WAITFORCONDITION() function at the end of playback. This will cause the playback thread to suspend until the conditions are met.

Parameters

SHORT Port: Port number of decoder to accept the command.

WORD TimeOut: Expiration time 1/60 second for NTSC; 1/50 second for PAL

CHAR Complete: Option to signal on two or zero frames left.

BOOL Enable: TRUE: Enable; FALSE: Disable (WaitForCondition).

Return Value

See page 120 for FUNC_STATUS definition.

Comments

- Use WAITFORCONDITION() for synchronous wait.

ErrorMask

C Calls

FUNC_STATUS ErrorMask(SHORT Port, MPEG_ERROR_MASK Error)

FUNC_STATUS ErrorMaskInlined(SHORT Port, MPEG_ERROR_MASK Error, BOOL bInline, BYTE Mode)

C++ Methods

FUNC_STATUS CDecoder::ErrorMask(MPEG_ERROR_MASK Error)

FUNC_STATUS CCinecast::ErrorMask(MPEG_ERROR_MASK Error, BOOL bInline, BYTE Mode)

Description

This function will return MPEG_ERROR bit mask which indicates the latest MPEG Stream or Hardware errors encountered.

Parameters

SHORT Port: Port number of decoder to accept the command.

MPEG_ERROR_MASK Error: Structure storing the error mask.

BOOL binline:

FALSE: Commands are executed immediately.

TRUE: Commands will remain pending until the specific event occurs.

BYTE Mode: Determines in what event the inlined command is activated.

Return Value

See page 120 for FUNC_STATUS definition.

Comments

- The Error Mask will be cleared every time this function is called. This allows the user to poll and check for changes in the state. Use this function to monitor Recovered Data, Genlock Sync, Lip Sync, SCSI Not Ready, SCSI Time Out, End Of Data signal, SCSI Write Protect, SCSI Unit Attention, Microcode Changed signal, and Wait For 2 Frames signal.
- See page 122 for MPEG_ERROR_MASK definition.

FactorySettings

C Call
BOOL FactorySettings(SHORT Port)

C++ Methods
BOOL CDecoder::FactorySettings()

BOOL CSCSI::FactorySettings()

BOOL CCinecast::FactorySettings()

Description
This function will restore the decoder settings in the Registry to the original factory settings.

Parameters
SHORT Port: Port number of decoder to accept the command.

Return Value
TRUE if save succeeds; FALSE if save fails.

Comments
• Exercise caution as all previous settings will be lost. INITIALIZE() must be called to effect the settings onto the decoder.

FastMotion

C Calls

FUNC_STATUS FastMotion(SHORT Port , USHORT Which)

FUNC_STATUS FastMotionCC(SHORT Port)

FUNC_STATUS FastMotionCCInlined(SHORT Port, BOOL bInline, BYTE Mode)

C++ Methods

FUNC_STATUS CDecoder::FastMotion(USHORT Which)

FUNC_STATUS CCinecast::FastMotion(BOOL binline, BYTE Mode)

Description

Plays the current video clip in fast motion by skipping frames.

Parameters

SHORT Port: Port number of decoder to accept the command.

USHORT Which: Number of frames to skip.

BOOL binline:

FALSE: Commands are executed immediately.

TRUE: Commands will remain pending until the specific event occurs.

BYTE Mode: Determines in what event the inlined command is activated.

Return Value

See page 120 for FUNC_STATUS definition.

Comments

- CineCast Quad Pro version of Fast Motion only has one speed.

FieldFreeze

C Call
FUNC_STATUS FieldFreeze(SHORT Port)

C++ Method
FUNC_STATUS CDecoder::FieldFreeze()

Description
Enables field freeze when using the "Pause" or "Pause on Underrun."

Parameters
SHORT Port: Port number of decoder to accept the command.

Return Value
See page 120 for FUNC_STATUS definition.

Comments
- Use FRAMEFREEZE() to pause on frame instead of field.

FifoBufferStatus

C Call
BYTE FifoBufferStatus(SHORT Port)

C++ Method
BYTE CDecoder::FifoBufferStatus()

Description
Returns MPEG Decoder Status Page (30h) value of Fifo Buffer Status.

Parameters
SHORT Port: Port number of decoder to accept the command.

Return Value
Fifo Buffer Status

Comments
None

FlushChannelBuffer

C Call
FUNC_STATUS FlushChannelBuffer(SHORT Port)

Description
Stops playback after all data in the decoder's buffers is used.

Parameters
SHORT Port: Port number of decoder to accept the command.

Return Value
See page 120 for FUNC_STATUS definition.

Comments
- Equivalent to RESET BUFFER (0x21).

FlushInline

C Call
FUNC_STATUS FlushInline(SHORT Port)

C++ Method
FUNC_STATUS CDecoder::FlushInline()

Description
Flushes all “inline” commands previously sent.

Parameters
SHORT Port: Port number of decoder to accept the command.

Return Value
See page 120 for FUNC_STATUS definition.

Comments
None

FrameAdvance

C Calls

FUNC_STATUS FrameAdvance(SHORT Port)

FUNC_STATUS FrameAdvanceCC(SHORT Port, DWORD Frames)

FUNC_STATUS FrameAdvanceCCInlined(SHORT Port, DWORD Frames, BOOL bInline, BYTE Mode)

C++ Methods

FUNC_STATUS CDecoder::FrameAdvance()

FUNC_STATUS CCinecast::FrameAdvance(DWORD Frames, BOOL binline, BYTE Mode)

Description

Advances the video one frame at a time.

Parameters

SHORT Port: Port number of decoder to accept the command.

DWORD Frames: Number of frames to advance.

BOOL binline:

FALSE: Commands are executed immediately.

TRUE: Commands will remain pending until the specific event occurs.

BYTE Mode: Determines in what event the inlined command is activated.

Return Value

See page 120 for FUNC_STATUS definition.

Comments

- CineCast Quad Pro decoders advance the given number frames.

FrameFreeze

C Call
FUNC_STATUS FrameFreeze(SHORT Port)

C++ Method
FUNC_STATUS CDecoder::FrameFreeze()

Description
Enables frame freeze when using the “Pause” or “Pause on Underrun.”

Parameters
SHORT Port: Port number of decoder to accept the command.

Return Value
See page 120 for FUNC_STATUS definition.

Comments
- Use FIELDFREEZE() to pause on field instead of frame.

FrameReport

C Call
DWORD FrameReport(SHORT Port)

C++ Method
DWORD CDecoder::FrameReport()

Description
Returns MPEG Decoder Status Page (30h) value of Frame Reporting.

Parameters
SHORT Port: Port number of decoder to accept the command.

Return Value
Frame Reporting

Comments
None

GPIN1Bit

C Call
BOOL PreLoadBit(SHORT Port)

C++ Method
BOOL CCinecast::PreLoadBit()

Description
Indicates that the decoder is loading data into the MPEG hardware but playback has not begun.

Parameters
SHORT Port: Port number of decoder to accept the command.

Return Value
State of PreLoad Bit.

Comments
- Preload is cleared when playback starts,

GPIN2Bit

C Call
BOOL GPIN2Bit(SHORT Port)

C++ Method
BOOL CCinecast::GPIN2Bit()

Description
Returns MPEG Decoder Status Page (30h) the state of the general purpose input discrete 2.

Parameters
SHORT Port: Port number of decoder to accept the command.

Return Value
State of GPIN2.

Comments
None

GainU

C Call
FUNC_STATUS GainU(SHORT Port , SHORT Data)

C++ Method
FUNC_STATUS CSCSI::GainU(SHORT Data)

Description
Sets the GainU Level to the desired point.

Parameters
SHORT nPort: Port number of decoder to accept the command.
SHORT Data: Range is 0x00 to 0xFF.

Return Value
See page 120 for FUNC_STATUS definition.

Comments
• Use of this command may drastically reduce picture quality. Only experienced users should attempt to customize the NTSC / PAL video. Consult with Vela for additional information.

GainV

C Call
FUNC_STATUS GainV(SHORT Port , SHORT Data)

C++ Method
FUNC_STATUS CSCSI::GainV(SHORT Data)

Description
Sets the GainV Level to the desired point.

Parameters
SHORT nPort: Port number of decoder to accept the command.
SHORT Data: Range is 0x00 to 0xFF.

Return Value
See page 120 for FUNC_STATUS definition.

Comments
• Use of this command may drastically reduce picture quality. Only experienced users should attempt to customize the NTSC/PAL video. Consult with Vela for additional information.

Genlock

C Calls

FUNC_STATUS Genlock (SHORT Port , BOOL AutoSwitch)

FUNC_STATUS GenlockCC (SHORT Port, BOOL AutoSwitch, SELECT Source)

C++ Methods

FUNC_STATUS CCinecast::Genlock(BOOL AutoSwitch, SELECT Source)

FUNC_STATUS CDecoder::Genlock (BOOL AutoSwitch)

Description

Enables the decoder to synchronize to a genlock input source.

Parameters

SHORT Port: Port number of decoder to accept the command.

BOOL AutoSwitch: Flag TRUE: Switch to Standalone when Genlock Loss; FALSE: No Switching.

SELECT Source: Source of genlock input.

Return Value

See page 120 for FUNC_STATUS definition.

Comments

- Use STANDALONE() to remove synchronizing to a genlock input source.
- CineCast Quad Pro version of GENLOCK() allows source selection.
- See page 124 for SELECT definition.

GenlockTiming

C++ Methods
FUNC_STATUS CCinecastEx::GenlockTiming (WORD Horiz, BYTE Vert)

Description
This function adjusts the relative position of the SDI output to that of the composite genlock input.

Parameters
WORD Horiz: Adjustment is in terms of 27MHz clock delays.
BYTE Vert: Adjustment is -8 lines to +7 lines.

Return Value
See page 120 for FUNC_STATUS definition.

Comments
- Horizontal Adjustment is in the following ranges:
  NTSC: 0 to 1715 (0 to 63.5 µS)
  PAL: 0 to 1727 (0 to 64.0 µS)
- Vertical Adjustment with a positive time indicates that the SDI output will precede genlock, negative time indicates that SDI follows genlock.

GetMPEGFileInfo

C Call
BOOL GetMPEGFileInfo(PMPEG_FILE_DETAILS pFile)

Description
This function will return information on MPEG files such as size, total frame count, stream type, bit rate, and frame rate.

Parameters
PMPEG_FILE_DETAILS pfile: Pointer to structure.

Return Value
TRUE if valid MPEG data is retrieved; FALSE if no file or data is found.

Comments
• Calculation of PAL file size is based on the Frame Rate Code. If the Code is 2,3,4, or 5, the fps will be set to 24.00, 25.00, 29.97, and 30.00. If not, it will default to 29.97. BitRate is described in kbps. Support for any file size, including those over 4-GB range.
• **pFile->szFileName** must be initialized to address of string holding a valid path/filename.
• See page 122 for PMPEG_FILE_DETAILS definition.

GetMpegPath

C Call
FUNC_STATUS GetMpegPath(SHORT Port , LPCSTR Path)

C++ Method
FUNC_STATUS CDecoder::GetMpegPath(LPCSTR Path)

Description
This function will return the default path of the clip location.

Parameters
SHORT Port: Port number of decoder to accept the command.
LPCSTR Path: Path of last decoder clip.

Return Value
See page 120 for FUNC_STATUS definition.

Comments
None

GetProductID

C Call

FUNC_STATUS GetProductID (SHORT Port, PDECODER_TYPE this_decoder)

C++ Method

FUNC_STATUS CDecoder::GetProductID (PDECODER_TYPE this_decoder)

Description

This function will return the decoder type.

Parameters

SHORT Port: Port number of decoder to accept the command.

PDECODER_TYPE this_decoder: Decoder model.

Return Value

See page 120 for FUNC_STATUS definition.

Comments

- See page 119 for PDECODER_TYPE definition.

GetSettings

C Calls
BOOL GetSettings(SHORT Port , SETTINGS * pSettings)

C++ Methods
BOOL CSCSI::GetSettings(SETTINGS * pSettings)

BOOL CCinecast::GetSettings(SETTINGS * pSettings)

Description
This function will return the decoder settings stored in the Registry.

Parameters
SHORT Port: Port number of decoder to accept the command.
SETTINGS * pSettings: Decoder Settings.

Return Value
Not yet documented.

Comments
- Use this function along with RESTORESETTINGS(), SETSETTINGS() and SAVESETTINGS() to make the decoder settings persistent between executions.
- See page 124 for SETTINGS definition.

GetState

C Call
FUNC_STATUS GetState(SHORT Port, DECODER_STATE_MASK *state)

C++ Method
FUNC_STATUS CDecoder::GetState(DECODER_STATE_MASK *state)

Description
This function will return the decoder state.

Parameters
SHORT Port: Port number of decoder to accept the command.
DECODER_STATE_MASK *state: State of the decoder.

Return Value
See page 120 for FUNC_STATUS definition.

Comments
- Use this function to determine the software state of the selected decoder.
- See page 120 for DECODER_STATE_MASK definition.

HorizontalPhase

C++ Method
FUNC_STATUS SCSI::HorizontalPhase(SHORT Data)

Description
This function will set the Horizontal Phase to the desired point.

Parameters
SHORT Data: Range for SCSI is 0x00 to 0xFF;
Range for SCSI-2 and SCSI-4 is 0x0000 to 0x07FF.

Return Value
See page 120 for FUNC_STATUS definition.

Comments
- Function works for SCSI-2 and SCSI-4 only.
- Use of this command may drastically reduce picture quality. Only experienced users should attempt to customize the NTSC / PAL video. Consult with Vela for additional information.

Initialize

C Calls
FUNC_STATUS Initialize(SHORT Port)

C++ Methods
FUNC_STATUS CSCSI::Initialize()

FUNC_STATUS CCinecast::Initialize()

Description
This function will initialize the decoder based on the saved settings stored in the Registry.

Parameters
SHORT Port: Port number of decoder to accept the command.

Return Value
See page 120 for FUNC_STATUS definition.

Comments
• CDecoder::Initialize() calls the following functions: DefaultResolution(), ErrorMask(), FrameFreeze(), FieldFreeze(), NoBlack(), Genlock(), StandAlone(), SetEventLevel().
• CSCSI::Initialize() calls the following functions: CDecoder::Initialize(), DefaultVideoStandard(), AVSynch(), VolumeControl(), BlackLevel(), BlankLevel(), GainU(), GainV().
• CCinecast::Initialize() calls the following functions: CDecoder::Initialize(), BackgroundColor(), ClosedCaptionReorder(), DefaultResolution(), LtcVitcTimeCode(), SecondChannel(), SetOutputDiscrete (), AudioGain().

IsNonStop

C Call
BOOLEAN IsNonStop(SHORT Port)

C++ Method
FUNC_STATUS CDecoder::IsNonStop()

Description
Returns last NonStop value.

Parameters
SHORT Port: Port number of decoder to accept the command.

Return Value
BOOLEAN TRUE | FALSE

Comments
None

LastDecoderCommand

C Call
DWORD LastDecoderCommand(SHORT Port)

C++ Method
DWORD CDecoder::LastDecoderCommand()

Description
Returns MPEG Decoder Status Page (30h) value of last decoder command.

Parameters
SHORT Port: Port number of decoder to accept the command.

Return Value
Last decoder command.

Comments
None

LastPlayBackError

C Call
FUNC_STATUS LastPlayBackError(SHORT Port)

Description
Returns the FUNC_STATUS return code from the last playback thread.

Parameters
SHORT Port: Port number of decoder to accept the command.

Return Value
See page 120 for FUNC_STATUS definition.

Comments
None

LtcVitcTimeCode

C Call

FUNC_STATUS LtcVitcTimeCode(SHORT Port, BOOL LTC, BOOL VITC, BYTE OutputSource, WORD LineField1, WORD LineField2)

C++ Method

FUNC_STATUS CCinecast::LtcVitcTimeCode(BOOL LTC, BOOL VITC, BYTE OutputSource, WORD LineField1, WORD LineField2)

Description

Configures LTC Output and VITC insertion/extraction.

Parameters

SHORT Port: Port number of decoder to accept the command.

BOOL LTC: TRUE: LTC disabled; FALSE: LTC enabled

BOOL VITC: TRUE: VITC disabled; FALSE: VITC enabled

BYTE OutputSource: Source of time code.

WORD LineField1: Line/Field for extraction/insertion of time code.

WORD LineField2: Line/Field for extraction/insertion of time code.

Return Value

See page 120 for FUNC_STATUS definition.

Comments

- This command is supported only by the CineCast Quad Pro.
- OutputSource is defined as:

  0x0 — Use GOP header SMPTE time code (default)

  0x1 — Use user-defined starting time code, set using TIMECODE().

MpegPlay

C Calls

FUNC_STATUS MpegPlay(SHORT Port, BOOL Trigger, BOOL Pause, BOOL MidStrm, BOOL ErrRcvy, STREAM Stream, UCHAR PreBlack, DWORD FrameCount, WORD PostBlack)

FUNC_STATUS MpegPlayCC(SHORT Port, BOOL Trigger, BOOL Pause, BOOL MidStrm, BOOL ErrRcvy, STREAM Stream, UCHAR PreBlack, DWORD FrameCount, WORD PostBlack, TRIGGER TrgEvent, BYTE Chroma, UCHAR Control)

C++ Methods

FUNC_STATUS CDecoder::MpegPlay(BOOL Trigger, BOOL Pause, BOOL MidStrm, BOOL ErrRcvy, STREAM Stream, UCHAR PreBlack, DWORD FrameCount, WORD PostBlack)

FUNC_STATUS CCinecast::MpegPlay(BOOL Trigger, BOOL Pause, BOOL MidStrm, BOOL ErrRcvy, STREAM Stream, UCHAR PreBlack, DWORD FrameCount, WORD PostBlack)

FUNC_STATUS CCinecast::MpegPlay(BOOL Trigger, BOOL Pause, BOOL MidStrm, BOOL ErrRcvy, STREAM Stream, UCHAR PreBlack, DWORD FrameCount, WORD PostBlack, TRIGGER TrgEvent, BYTE Chroma, UCHAR Control)

Description

Starts decoder playback.

Parameters

SHORT Port: Port number of decoder to accept the command.

BOOL Trigger: TRUE: External trigger start; FALSE: Immediate playback**.**

BOOL Pause: TRUE: Pause on non-black video frame; FALSE: Immediate playback.

BOOL MidStrm: TRUE: Search for next valid sequence header; FALSE: Immediate playback.

BOOL ErrRcvy: TRUE: Automatically pause when data underflow; FALSE: Stop playback go to black when data underflows.

STREAM Stream: Stream type of clip.

UCHAR PreBlack: Number of black frames to display before clip start.

DWORD FrameCount: Number of frames to play back.

WORD PostBlack: Number of black frames to display after clip end.
TRIGGER TrgEvent: Source for event to start playback.
BYTE Chroma: 4:2:0 or 4:2:2 chroma format.
UCHAR Control: Default 0.

Return Value

See page 120 for FUNC_STATUS definition.

Comments

- A BUFFERRESET() followed by at least 500-k of data written to the decoder must occur before MPEGPLAY() may be issued.
- Do not use in conjunction with PLAY().
- Chroma is defined as:
  0x0 — IBM
  0x1 — 4:2:0
  0x2 — 4:2:2
- See page 126 for STREAM definition.
- See page 127 for TRIGGER definition.

Mute

C Calls

FUNC_STATUS Mute(SHORT Port , BOOL Enabled)

FUNC_STATUS MuteInlined(SHORT Port, BOOL Enabled, BOOL bInline, BYTE Mode)

C++ Methods

FUNC_STATUS CDecoder::Mute(BOOL Enabled)

FUNC_STATUS CCinecast::Mute(BOOL Enabled, BOOL binline, BYTE Mode)

Description

Mutes audio output.

Parameters

SHORT Port: Port number of decoder to accept the command.

BOOL Enabled: TRUE: Enable; FALSE: Disable Mute.

BOOL binline:

FALSE: Commands are executed immediately.
TRUE: Commands will remain pending until the specific event occurs.

BYTE Mode: Determines in what event the inlined command is activated.

Return Value

See page 120 for FUNC_STATUS definition.

Comments

None

NoBlack

C Call
FUNC_STATUS NoBlack(SHORT Port, UCHAR Seamless)

C++ Method
FUNC_STATUS CDecoder::NoBlack(UCHAR Seamless)

Description
Enables seamless transition between clips.

Parameters
SHORT Port: Port number of decoder to accept the command.
UCHAR Seamless: 1: No black between videos; 2: Pre and Post black values will be used.

Return Value
See page 120 for FUNC_STATUS definition.

Comments
• This option eliminates any black between playbacks. NOBLACK() disabled will cause MPEGPLAY() to be issued for each clip.

NonStop

C Call
FUNC_STATUS NonStop(SHORT Port , BOOL Loop)

C++ Method
FUNC_STATUS CDecoder::NonStop(BOOL Loop)

Description
Enables the current video clip to repeat from the beginning when the end of the current file is reached or when the maximum frame count has played.

Parameters
SHORT Port: Port number of decoder to accept the command.
BOOL Loop: TRUE: Enable looping; FALSE: Disable looping.

Return Value
See page 120 for FUNC_STATUS definition.

Comments
None

PALBufferMode

C Call
FUNC_STATUS PALBufferMode(SHORT Port , BOOL flag)

C++ Method
FUNC_STATUS CSCSI::PALBufferMode(BOOL PalBufferFlag)

Description
Changes the decoder's buffer MPEG mode for PAL.

Parameters
SHORT Port: Port number of decoder to accept the command.
BOOL flag: TRUE for PAL; FALSE for NTSC.

Return Value
See page 120 for FUNC_STATUS definition.

Comments
- Not supported by CineCast Quad Pro.

PauseVideo

C Calls
FUNC_STATUS PauseVideo(SHORT Port)

C++ Methods
FUNC_STATUS CDecoder::PauseVideo()

FUNC_STATUS CCinecast::PauseVideo(BOOL bInline, BYTE Mode)

Description
Pauses the video clip that is currently being played.

Parameters
SHORT Port: Port number of decoder to accept the command.
BOOL binline:
FALSE: Commands are executed immediately.
TRUE: Commands will remain pending until the specific event occurs.
BYTE Mode: Determines in what event the inlined command is activated.

Return Value
See page 120 for FUNC_STATUS definition.

Comments
None

Play

C Calls

FUNC_STATUS Play(SHORT Port, CHAR * Clip, BOOL ErrorRecovery, BOOL Midstream, BOOL Pause, BOOL Trigger, STREAM Type, UCHAR PreBlack, ULONG Frames, WORD PostBlack, DWORDLONG Offset)

FUNC_STATUS PlayCC(SHORT Port, CHAR * Clip, BOOL ErrorRecovery, BOOL Midstream, BOOL Pause, BOOL Trigger, STREAM Type, UCHAR PreBlack, ULONG Frames, WORD PostBlack, TRIGGER TriggerSource, BYTE Microcode, DWORDLONG Offset)

C++ Methods

FUNC_STATUS CDecoder::Play(CHAR * Clip, BOOL ErrorRecovery, BOOL Midstream, BOOL Pause, BOOL Trigger, STREAM Type, UCHAR PreBlack, ULONG Frames, WORD PostBlack, TRIGGER TriggerSource, BYTE Microcode, DWORDLONG Offset)

FUNC_STATUS CCinecast::Play(CHAR * Clip, BOOL ErrorRecovery, BOOL Midstream, BOOL Pause, BOOL Trigger, STREAM Type, UCHAR PreBlack, ULONG Frames, WORD PostBlack, DWORDLONG Offset)

FUNC_STATUS CCinecast::Play(CHAR * Clip, BOOL ErrorRecovery, BOOL Midstream, BOOL Pause, BOOL Trigger, STREAM Type, UCHAR PreBlack, ULONG Frames, WORD PostBlack, TRIGGER TriggerSource, BYTE Microcode, DWORDLONG Offset)

Description

This function creates the playback thread which pre-loads data to the decoder, issues MPEGPLAY(), loads the rest of the data, and monitors the state of the clip and decoder.

Parameters

SHORT Port: Port number of decoder to accept the command.

CHAR * Clip: Name of video clip.

BOOL ErrorRecovery: TRUE: Automatically pause when data underflow; FALSE: Stop playback and go to black.

BOOL Midstream: TRUE: Search for next valid sequence header; FALSE: Immediate playback.

BOOL Pause: TRUE: Pause on non-black video frame; FALSE: Immediate playback.

BOOL Trigger: TRUE: Wait for external trigger; FALSE: Immediate playback.

STREAM Type: Stream type of clip.

SHORT PreBlack: Option for number of black frames to display before start of clip.

ULONG Frames: Number of frames to play back.

WORD PostBlack: Option for number of black frames to display after clip end.

TRIGGER TriggerSource: Source for event to start playback.

BYTE Microcode: Chroma microcode to load.

DWORD Offset: Byte offset into file.

Return Value

See page 120 for FUNC_STATUS definition.

Comments

- If Offset is non-zero, MidStream boolean must be set to TRUE.
- Microcode is defined as:
  0x0 — IBM
  0x1 — 4:2:0
  0x2 — 4:2:2
- See page 126 for STREAM definition.
- See page 127 for TRIGGER definition.

PlaybackStarted

C Call
BOOL PlaybackStarted(SHORT Port)

C++ Method
BOOL CDecoder::PlaybackStarted()

Description
Returns BOOL to indicate data has been pre-loaded and MPEGPLAY() issued.

Parameters
SHORT Port: Port number of decoder to accept the command.

Return Value
Not yet documented.

Comments
- Use this to monitor the playback thread during the implementation of a playlist.
- Use READYFORNEXTVIDEO() to determine when to issue SETNEXTVIDEO().

PlayBit

C Call
BOOL PlayBit(SHORT Port)

C++ Method
BOOL CDecoder::PlayBit()

Description
Returns MPEG Decoder Status Page (30h) value of Play Bit.

Parameters
SHORT Port: Port number of decoder to accept the command.

Return Value
Play Bit

Comments
• Use this function to determine if decoder is still playing a clip. Note that if Error Recovery is enabled, the PLAYBIT() will return TRUE even when playback has terminated.

ProductInfo

C Call

FUNC_STATUS ProductInfo(SHORT Port, PPRODUCT_TYPE product, PBOARD_INFO_TYPE board)

C++ Method

FUNC_STATUS CDecoder::ProductInfo(PPRODUCT_TYPE product, PBOARD_INFO_TYPE board)

Description

Returns product and SCSI board information on the selected decoder.

Parameters

S**HORT Port:** Port number of decoder to accept the command.

PPRODUCT_TYPE product: Product information.

PBOARD_INFO_TYPE board: SCSI board information.

Return Value

See page 120 for FUNC_STATUS definition.

Comments

- See page 123 for PPRODUCT_TYPE definition.
- See page 118 for PBOARD_INFO_TYPE definition.

QueueBit

C Call
BOOL QueueBit(SHORT Port)

C++ Method
BOOL CDecoder::QueueBit()

Description
Returns MPEG Decoder Status Page (30h) value of Queue Bit.

Parameters
SHORT Port: Port number of decoder to accept the command.

Return Value
Queue Bit

Comments
- Set after a buffer reset is received; cleared only after playback has started.

ReadAudioLevel

C Call

FUNC_STATUS ReadAudioLevel(SHORT Port, BYTE * PrimRightLevel, BYTE * PrimLeftLevel, BYTE * SecRightLevel, BYTE * SecLeftLevel)

C++ Method

FUNC_STATUS CCinecast::ReadAudioLevel(BYTE * PrimRightLevel, BYTE * PrimLeftLevel, BYTE * SecRightLevel, BYTE * SecLeftLevel)

Description

Returns Audio Level Mode Page (35h) with the latest averaged audio levels for each output audio channel.

Parameters

SHORT Port: Port number of decoder to accept the command.

BYTE * PrimRightLevel: Primary Right Audio Level.

BYTE * PrimLeftLevel: Primary Left Audio Level.

BYTE * SecRightLevel: Secondary Right Audio Level.

BYTE * SecLeftLevel: Secondary Left Audio Level.

Return Value

See page 120 for FUNC_STATUS definition.

Comments

- This command is supported only by the CineCast Quad Pro.

ReadFrameCount

C Call
DWORD ReadFrameCount(SHORT Port)

C++ Method
DWORD CDecoder::ReadFrameCount()

Description
Reads back number of frames displayed.

Parameters
SHORT Port: Port number of decoder to accept the command.

Return Value
Number of frames decoded so far.

Comments
• The frame count is valid only during playback. Reading the value while the decoder is idle or during transition between two videos will result in indeterminate results.

ReadTimeCode

C Call

FUNC_STATUS ReadTimeCode(SHORT Port, TCSRC Source, SHORT *Hours, SHORT *Minutes, SHORT *Seconds, SHORT *Frames)

C++ Method

FUNC_STATUS CCinecast::ReadTimeCode(TCSRC Source, SHORT *Hours, SHORT *Minutes, SHORT *Seconds, SHORT *Frames)

Description

Reads SMPTE time code for next command.

Parameters

SHORT Port: Port number of decoder to accept the command.

TCSRC Source: LTCOut, GOP, LTCIn.

SHORT Hour: SMPTE Hour.

SHORT Minute: SMPTE Minute.

SHORT Seconds: SMPTE Second.

SHORT Frames: SMPTE Frame.

Return Value

TRUE if playback thread ready to accept another clip; FALSE if playback has not transitioned to the next clip.

Comments

- SMPTE time codes can be read from three different sources:
  1. LTC Output time code
  2. MPEG GOP Header SMPTE time code
  3. LTC Input time code
- See page 127 for TCSRC definition.

ReadyForNextVideo

C Call
BOOL ReadyForNextVideo(SHORT Port)

C++ Method
BOOL CDecoder::ReadyForNextVideo()

Description
Returns BOOL to indicate playback thread can accept the next video in a playlist.

Parameters
SHORT Port: Port number of decoder to accept the command.

Return Value
Not yet documented.

Comments
- Use this function to monitor the playback thread during the implementation of a playlist.
- Use SETNEXTVIDEO() to cue next clip.

RestoreSettings

C Call
BOOL RestoreSettings(SHORT Port)

C++ Methods
BOOL CDecoder::RestoreSettings()

BOOL CSCSI::RestoreSettings()

BOOL CCinecast::RestoreSettings()

Description

This function will retrieve the previously saved decoder settings stored in the Registry.

Parameters

SHORT Port: Port number of decoder to accept the command.

Return Value

TRUE if restore succeeds; FALSE if restore fails.

Comments

• Use this function along with GETSETTINGS(), SETSETTINGS() and SAVESETTINGS() to make the decoder settings persistent between executions. INITIALIZE() must be call to effect the settings onto the decoder.

Resume

C Call
FUNC_STATUS Resume(SHORT Port)

C++ Method
FUNC_STATUS CDecoder::Resume()

Description
Resumes the video clip that currently is paused, in slow motion, or in fast forward mode.

Parameters
SHORT Port: Port number of decoder to accept the command.

Return Value
See page 120 for FUNC_STATUS definition.

Comments
None

ResumeNormal

C Calls

FUNC_STATUS ResumeNormal(SHORT Port)

FUNC_STATUS ResumeNormalInlined(SHORT Port, BOOL bInline, BYTE Mode)

***C++ Method*s**

FUNC_STATUS CDecoder::ResumeNormal()

FUNC_STATUS CCinecast::ResumeNormal(BOOL bInline, BYTE Mode)

Description

Resumes the video clip that currently is paused.

Parameters

SHORT Port: Port number of decoder to accept the command.

BOOL binline:

FALSE: Commands are executed immediately.

TRUE: Commands will remain pending until the specific event occurs.

BYTE Mode: Determines in what event the inlined command is activated.

Return Value

See page 120 for FUNC_STATUS definition.

Comments

None

ResumePlay

C Call
FUNC_STATUS ResumePlay(SHORT Port)

C++ Method
FUNC_STATUS CDecoder::ResumePlay()

Description
Resumes the video clip that currently is in slow motion or fast forward mode.

Parameters
SHORT Port: Port number of decoder to accept the command.

Return Value
See page 120 for FUNC_STATUS definition.

Comments
None

RevisionInfo

C Call
FUNC_STATUS RevisionInfo(SHORT Port, PREVISION_TYPE Revision)

C++ Method
FUNC_STATUS CDecoder::RevisionInfo(PREVISION_TYPE Revision)

Description
This function retrieves the Revision Page from the decoder. This includes information on the various revision codes for hardware/firmware and mach versions.

Parameters
SHORT Port: Port number of decoder to accept the command.
PREVISION_TYPE Revision: Hardware/Firmware/Mach Version information.

Return Value
See page 120 for FUNC_STATUS definition.

Comments
• A Test Unit Ready will be issued if the Inquiry fails. This will ensure that the information to be received from the decoder is valid. Function retrieves the firmware, machware, SCSI, and MPEG versions of the decoder board.
• See page 123 for PREVISION_TYPE definition.

SaveSettings

C Call
BOOL SaveSettings(SHORT Port)

C++ Methods
BOOL CDecoder::SaveSettings()

BOOL CSCSI::SaveSettings()

BOOL CCinecast::SaveSettings()

Description
This function will store the decoder settings in the Registry.

Parameters
SHORT Port: Port number of decoder to accept the command.

Return Value
TRUE if save succeeds; FALSE if save fails.

Comments
• Use this function along with GETSETTINGS(), SETSETTINGS() and RESTORESETTINGS() to make the decoder settings persistent between executions. INITIALIZE() must be called to effect the settings onto the decoder.

SecondChannel

C Call

FUNC_STATUS SecondChannelCC(SHORT Port, BOOL Enable, BYTE DecodeOrder, BOOL AC3)

C++ Methods

FUNC_STATUS CDecoder::SecondChannel(BOOL Enable)

FUNC_STATUS CSCSI::SecondChannel(BOOL Enable)

FUNC_STATUS CCinecast::SecondChannel(BOOL Enable, BYTE DecodeOrder, BOOL AC3)

Description

Enables the audio on the second channel.

Parameters

SHORT Port: Port number of decoder to accept the command.

BOOL Enable: TRUE: Enable second audio channel.

BYTE DecodeOrder: Audio decode mode.

BOOL AC3: TRUE for as-is downmix; FALSE for downmix to two channels.

Return Value

See page 120 for FUNC_STATUS definition.

Comments

- Note that second channel audio decode is not currently supported by firmware.
- CineCast Quad Pro version of second channel audio: The IBM MPEG decoder can only decode 2-channel MPEG Audio. The second channel audio decoder can decode either two channel MPEG or Dolby* AC-3 (Dolby Digital*).
- DecodeOrder is defined as:
  0x0 — Auto
  0x1 — MPEG
  0x2 — AC-3

* Dolby and Dolby Digital are trademarks of Dolby Laboratories.

SessionActive

C Call
BOOL SessionActive(SHORT Port)

C++ Method
BOOL CDecoder::SessionActive()

Description
Returns Active status of the current playback thread.

Parameters
SHORT Port: Port number of decoder to accept the command.

Return Value
TRUE: Playback thread active; FALSE: Playback thread inactive.

Comments
None

SetEventLevel

C Call
VOID SetEventLevel(SHORT Port, DEBUG_LEVEL Level)

C++ Method
VOID CDecoder::SetEventLevel(DEBUG_LEVEL Level)

Description
Sets the Event Log to the desired level of messaging to aid in debugging of applications.

Parameters
SHORT Port: Port number of decoder to accept the command.
DEBUG_LEVEL Level: Desired level of debug messages.

Return Value
None

Comments
- Function will accept 0 as a DISABLE option.
- See page 119 for DEBUG_LEVEL definition.

SetMpegPath

C Call
FUNC_STATUS SetMpegPath(SHORT Port, LPCSTR NewPath)

C++ Method
FUNC_STATUS CDecoder::SetMpegPath(LPCSTR NewPath)

Description
This function will set the default path of the clip location.

Parameters
SHORT Port: Port number of decoder to accept the command.
LPCSTR Path: Path of last decoder clip.

Return Value
See page 120 for FUNC_STATUS definition.

Comments
None

SetNextVideo

C Call
BOOL SetNextVideo(SHORT Port , LPCSTR Movie)

Description
This function will playback a selected video immediately following the video now playing.

Parameters
SHORT Port: Port number of decoder to accept the command.
LPCSTR Movie: Name of video clip.

Return Value
TRUE: Playback thread accepted next clip; FALSE: Playback thread rejected next clip.

Comments
- Functionally equivalent to BACKTOBACKVIDEO(). NOBLACK() disabled will cause MPEGPLAY() to be issued for each clip.
- Use PLAYBACKSTARTED() to determine when to check for READYFORNEXTVIDEO().

SetOutputDiscrete

C Calls

FUNC_STATUS SetOutputDiscrete (SHORT Port, BOOL GPOut1, BOOL Sync1, BOOL GPOut2, BOOL Sync2)

FUNC_STATUS SetOutputDiscreteInlined(SHORT Port, BOOL GPOut1, BOOL Sync1, BOOL GPOut2, BOOL Sync2, BOOL bInline, BYTE Mode)

C++ Method

FUNC_STATUS CCinecast::SetOutputDiscrete (BOOL GPOut1, BOOL Sync1, BOOL GPOut2, BOOL Sync2, BOOL binline, BYTE Mode)

Description

Selects OutputDiscrete and when to change state.

Parameters

SHORT Port: Port number of decoder to accept the command.

BOOL GPOut1: TRUE: Discrete Out 1 on.

BOOL Sync1: TRUE: Change on Vsync; FALSE: Change immediately.

BOOL GPOut2: TRUE: Discrete Out 2 on.

BOOL Sync2: TRUE: Change on Vsync; FALSE: Change immediately.

BOOL binline:

FALSE: Commands are executed immediately.
TRUE: Commands will remain pending until the specific event occurs.

BYTE Mode: Determines in what event the inlined command is activated.

Return Value

See page 120 for FUNC_STATUS definition.

Comments

- This command is supported only by the CineCast Quad Pro.

SetSettings

C Call
BOOL SetSettings(SHORT Port, SETTINGS * pSettings)

C++ Methods
BOOL CDecoder::SetSettings(SETTINGS * pSettings)

BOOL CSCSI::SetSettings(SETTINGS * pSettings)

BOOL CCinecast::SetSettings(SETTINGS * pSettings)

Description
This function will store the decoder settings in Memory.

Parameters
SHORT Port: Port number of decoder to accept the command.

Return Value
TRUE if set succeeds; FALSE if set fails.

Comments
• Use this function along with GETSETTINGS(), SAVESETTINGS() and RESTORESETTINGS() to make the decoder settings persistent between executions. INITIALIZE() must be called to effect the settings onto the decoder.

SetVideoPassThrough

C Call

FUNC_STATUS SetVideoPassThrough(SHORT Port, VIDEO_PASS_MODE mode)

C++ Method

FUNC_STATUS CSCSI::SetVideoPassThrough (BOOL AdjChn)

Description

Switches the decoder output from MPEG digital video to analog (genlock) video.

Parameters

BOOL: AdjChn

SHORT Port: Port number of decoder to accept the command.

VIDEO_PASS_MODE mode: Next Vertical Sync, External Trigger, End of Playback, Video only.

Return Value

See page 120 for FUNC_STATUS definition.

Comments

- Not supported by CineCast Quad Pro.
- See page 127 for VIDEO_PASS_MODE definition.

SetYuvMode

C Call
FUNC_STATUS SetYuvMode(SHORT Port, YUV_TYPE Yuv)

C++ Method
FUNC_STATUS CSCSI::SetYuvMode(YUV_TYPE Yuv)

Description
Changes the output mode to Betacam YUV, SVHS, or RGB. PAL/NTSC will be detected by the decoder when it parses the MPEG stream.

Parameters
SHORT Port: Port number of decoder to accept the command.
YUV_TYPE Yuv: Set mode to Betacam YUV, SVHS, or RGB.

Return Value
See page 120 for FUNC_STATUS definition.

Comments
- Only valid for SCSI-2 YUV-enabled decoders.
- See page 128 for YUV_TYPE definition.

SlowMotion

C Calls

FUNC_STATUS SlowMotion(SHORT Port, USHORT which, USHORT repeat)

FUNC_STATUS SlowMotionCC (SHORT Port, BYTE Speed)

FUNC_STATUS SlowMotionCCInlined (SHORT Port, BYTE Speed, BOOL bInline, BYTE Mode)

C++ Methods

FUNC_STATUS CDecoder::SlowMotion(USHORT SlowWhich, USHORT SlowRepeat)

FUNC_STATUS CCinecast::SlowMotion(BYTE Speed, BOOL binline, BYTE Mode)

Description

Plays the current video clip in slow motion.

Parameters

SHORT Port: Port number of decoder to accept the command.

USHORT which: How often a frame should be skipped (0…15).

USHORT repeat: A single frame is repeated (0…15).

BYTE Speed: Rate of display (0…7).

BOOL binline:

FALSE: Commands are executed immediately.

TRUE: Commands will remain pending until the specific event occurs.

BYTE Mode: Determines in what event the inlined command is activated.

Return Value

See page 120 for FUNC_STATUS definition.

Comments

- CineCast Quad Pro version of slow motion only uses a speed rate of 0…7.

SoftReset

C Calls

FUNC_STATUS SoftReset(SHORT Port)

FUNC_STATUS SoftResetInlined(SHORT Port, BOOL bInline, BYTE Mode)

C++ Methods

FUNC_STATUS CDecoder::SoftReset()

FUNC_STATUS CCinecast::SoftReset(BOOL binline, BYTE Mode)

Description

Resets the decoder's audio and video circuits.

Parameters

SHORT Port: Port number of decoder to accept the command.

BOOL binline:

FALSE: Commands are executed immediately.

TRUE: Commands will remain pending until the specific event occurs.

BYTE Mode: Determines in what event the inlined command is activated.

Return Value

See page 120 for FUNC_STATUS definition.

Comments

- Forces the screen to black.

StandAlone

C Call
FUNC_STATUS StandAlone(SHORT Port)

C++ Method
FUNC_STATUS CDecoder::StandAlone()

Description
Enables the decoder to standalone mode without synchronizing to a genlock input source.

Parameters
SHORT Port: Port number of decoder to accept the command.

Return Value
See page 120 for FUNC_STATUS definition.

Comments
- Use GENLOCK() to enable synchronizing to a genlock input source.

Stop

C Calls

FUNC_STATUS Stop(SHORT Port , BOOL Freeze)

FUNC_STATUS StopInlined(SHORT Port, BOOL Freeze, BOOL bInline, BYTE Mode)

C++ Methods

FUNC_STATUS CDecoder::Stop(BOOL Freeze)

FUNC_STATUS CCinecast::Stop(BOOL Freeze, BOOL bInline, BYTE Mode)

Description

Stops the video clip that is currently playing.

Parameters

SHORT Port: Port number of decoder to accept the command.

BOOL Freeze: TRUE: Stop the video on the last frame.

BOOL binline:

FALSE: Commands are executed immediately.
TRUE: Commands will remain pending until the specific event occurs.

BYTE Mode: Determines in what event the inlined command is activated.

Return Value

See page 120 for FUNC_STATUS definition.

Comments

None

SubcarrierPhase

C++ Methods
BOOLEAN SCSI::SetSubcarrierPhase(SHORT Data)

Description
This function will set the Subcarrier Phase to the desired point.

Parameters
SHORT Data: Range is 0x00 to 0xFF.

Return Value
See page 120 for FUNC_STATUS definition.

Comments
- Function works for SCSI-2 and SCSI-4 only.
- Use of this command may drastically reduce picture quality. Only experienced users should attempt to customize the NTSC / PAL video. Consult with Vela for additional information.

SwitchAudio

C Call

FUNC_STATUS SwitchAudio(SHORT Port, AUDIO_FUNC func, AUDIO_DATA data)

C++ Method

FUNC_STATUS CSCSI::SwitchAudio(AUDIO_FUNC func, AUDIO_DATA data)

Description

Selects and sets up audio output.

Parameters

SHORT Port: Port number of decoder to accept the command.

AUDIO_FUNC func: Select audio function to modify.

AUDIO_DATA data: Type of modification.

Return Value

See page 120 for FUNC_STATUS definition.

Comments

- Not supported by CineCast Quad Pro.
- See page 118 for AUDIO_DATA definition.
- See page 118 for AUDIO_FUNC definition.

TestPattern

C Call
FUNC_STATUS TestPattern(SHORT Port)

C++ Method
FUNC_STATUS CDecoder::TestPattern()

Description
Generates a color bar test pattern along with an audio tone.

Parameters
SHORT Port: Port number of decoder to accept the command.

Return Value
See page 120 for FUNC_STATUS definition.

Comments
- Issue SOFTRESET() command to terminate the command.

TimeCode

C Call

FUNC_STATUS TimeCode(SHORT Port, SHORT Hour, SHORT Minute, SHORT Seconds, SHORT Frames)

C++ Method

FUNC_STATUS CCinecast::TimeCode(SHORT Hour, SHORT Minute, SHORT Seconds, SHORT Frames)

Description

Sets SMPTE time code for next command.

Parameters

SHORT Port: Port number of decoder to accept the command.

SHORT Hour: Hour to set.

SHORT Minute: Minute to set.

SHORT Seconds: Second to set.

SHORT Frames: Frame to set.

Return Value

See page 120 for FUNC_STATUS definition.

Comments

- This command is supported only by the CineCast Quad Pro.
- Use in conjunction with LTCVITCTIMECODE()

TrickModeFrameCount

C Call
FUNC_STATUS TrickModeFrameCount (SHORT Port , DWORD Frames)

C++ Method
FUNC_STATUS CCinecast::TrickModeFrameCount (DWORD Frames)

Description
Sets frame count for next trick mode command.

Parameters
SHORT Port: Port number of decoder to accept the command.
DWORD Frames: Desired frame count to activate trick mode.

Return Value
See page 120 for FUNC_STATUS definition.

Comments
- This command is supported only by the CineCast Quad Pro.

UnderFlowBit

C Call
BOOL UnderFlowBit(SHORT Port)

C++ Method
BOOL CDecoder::UnderFlowBit()

Description
Returns MPEG Decoder Status Page (30h) value of UnderFlow Bit.

Parameters
SHORT Port: Port number of decoder to accept the command.

Return Value
UnderFlow Bit

Comments
None

UpgradeActive

C Call
BOOL UpgradeActive(SHORT Port)

C++ Method
BOOL CDecoder::UpgradeActive()

Description
Returns Active State of Firmware Upgrade.

Parameters
SHORT Port: Port number of decoder to accept the command.

Return Value
TRUE: Active; FALSE: Inactive.

Comments
• Use this function along with UPGRADEFIRMWARE() and UPGRADERESULT() to monitor the asynchronous download of firmware data.

UpgradeFirmWare

C Call
BOOL UpgradeFirmWare(SHORT Port, PCHAR sFileName, DL_TYPE Mode)

C++ Method
BOOL CDecoder::UpgradeFirmWare(PCHAR sFileName, DL_TYPE Mode)

Description
This function downloads new decoder firmware (microcode) over the SCSI bus to the decoder asynchronously.

Parameters
SHORT Port: Port number of decoder to accept the command.
PCHAR sFileName: Path and filename to the firmware binary file.
DL_TYPE Mode: CHECK_FILE (0X4) | BURN_FLASH (0X5).

Return Value
Not yet documented.

Comments
• Use this function along with UPGRADEACTIVE() and UPGRADERESULT() to monitor the asynchronous download of firmware data. See page 121 for DL_TYPE definition.

UpgradeResult

C Call
BOOL UpgradeResult(SHORT Port)

C++ Method
BOOL CDecoder::UpgradeResult()

Description
Returns result of firmware upgrade.

Parameters
SHORT Port: Port number of decoder to accept the command.

Return Value
TRUE: Passed; FALSE: Failed.

Comments
• Use this function along with UPGRADEACTIVE() and UPGRADEFIRMWARE() to monitor the asynchronous download of firmware data.

UnusedFiFoDataAfterStop

C Call
BYTE UnusedFiFoDataAfterStop(SHORT Port)

C++ Method
BYTE CDecoder::UnusedFiFoDataAfterStop()

Description
Returns MPEG Decoder Status Page (30h) value of unused FiFo data after Stop.

Parameters
SHORT Port: Port number of decoder to accept the command.

Return Value
Unused FiFo data after Stop.

Comments
None

VideoLevel

C Call
FUNC_STATUS VideoLevel(SHORT Port , WORD Level , BOOL Up)

C++ Method
FUNC_STATUS CSCSI::VideoLevel(WORD Level, BOOL Up)

Description
Modifies the video reference level.

Parameters
SHORT Port: Port number of decoder to accept the command.
WORD Level: Increment/Decrement Level.
BOOL Up: Direction of change. TRUE: Up; FALSE: Down.

Return Value
See page 120 for FUNC_STATUS definition.

Comments
- Not supported by CineCast Quad Pro.
- Use of this command may drastically reduce picture quality. Only experienced users should attempt to customize the NTSC / PAL video. Consult with Vela for additional information.

VolumeControl

C Call
FUNC_STATUS VolumeControl(SHORT Port , BYTE Level , BYTE Channel)

C++ Method
FUNC_STATUS CSCSI::VolumeControl(BYTE Level, BYTE Channel)

Description
Modifies attenuation of the audio output.

Parameters
SHORT Port: Port number of decoder to accept the command.
BYTE Level: Attenuation level 0…127.
BYTE Channel: Channel selection for Stereo (0x0), Left (0x1), or Right (0x2).

Return Value
See page 120 for FUNC_STATUS definition.

Comments
- Channel is defined as:
  0x0 — Stereo
  0x1 — Left
  0x2 — AC-3
- Not supported by CineCast Quad Pro.

WaitForCondition

C Call

FUNC_STATUS WaitForCondition(SHORT Port , WORD TimeOut, CHAR Complete)

C++ Method

FUNC_STATUS CDecoder::WaitForCondition(WORD TimeOut, CHAR Complete)

Description

Notifies the host of playback decoder events.

Parameters

SHORT Port: Port number of decoder to accept the command.

WORD TimeOut: Expiration time 1/60 second for NTSC; 1/50 second for PAL.

CHAR Complete: Option to signal on two or zero frames left.

Return Value

See page 120 for FUNC_STATUS definition.

Comments

- Use ENABLEWAITFORCONDITION() for asynchronous use of WAITFORCONDITION().

CLASS: CDecoder

Overview

The *CDecoder* class is a base class used to create *CScsi* and *CCinecast* classes. This class provides the generic and common methods for all decoders. The class by itself is will allow playback on any decoder. In order to be able to use functionality of the other decoders, it is suggested that you use CScsi for SCSI1, SCSI2, SCSIx4, and SCSIx8 decoders. While *CCinecast* class is to be used for only with the CineCast Quad Pro SDI decoders.

PUBLIC Interface: *CDecoder* (v = virtual)

CDecoder	Class Constructor
~CDecoder (v)	Class Destructor
Open (v)	Open decoder channel device
Close (v)	Close decoder channel device
Initialize (v)	Initialize decoder based on the saved settings
GetSettings (v)	Return decoder settings stored in the Registry
SetSettings (v)	Store decoder settings in Memory
RestoreSettings (v)	Retrieve decoder settings stored in the Registry
SaveSettings (v)	Store decoder settings in the Registry
//Setup	
ProgramDecoder	Download new decoder firmware (microcode) (synch)
SetEventLevel	Sets the Event Log to the desired level of messaging
UpgradeActive	Returns Active State of Firmware Upgrade
UpgradeFirmWare	Download new decoder firmware (microcode) (asynch)
UpgradeResult	Returns Result of Firmware Upgrade
//Setup Static	
DefaultResolution	Set the video standard output to NTSC or PAL
SecondChannel	Enable the audio on the second channel
TestPattern	Generate color bar test pattern and audio tone
//Setup Dynamic	
Blank	Turn video on or off
BufferReset (v)	Reset the MPEG buffer pointers
EnablePlaybackSignal	Detects transition between clips
EnableWaitForCondition	Enable WaitForCondition() at the end of playback
FieldFreeze	Enable field freeze for “Pause” or “Pause on Underrun”
FlushInline	Flush all “inline” commands previously sent
FrameFreeze	Enable frame freeze for “Pause” or “Pause on Underrun”
Genlock (v)	Enable decoder to synchronize to a genlock input source
Mute	Mute audio output
SoftReset	Reset the decoder's audio and video circuits
StandAlone	Set decoder to standalone mode with no genlock input
WaitForCondition	Notify of playback decoder events

//VCR Play	
BackToBackVideo	Playback selected (clip following the current video)
MpegPlay (v)	Start decoder playback
NoBlack	Enable seamless transition between clips
NonStop	Enable the current video clip to repeat
Play (v)	Playback selected clip
Stop	Stop the current video clip playing
//VCR Trick Mode	
FastMotion	Play the current video clip in fast motion
FrameAdvance	Advance the video one frame at a time
PauseVideo	Pause the current video clip playing
Resume	Resume the video clip paused, in slow or fast motion
ResumeNormal	Resume the video clip paused
ResumePlay	Resume the video clip in slow or fast motion
SlowMotion	Play the current video clip in slow motion
//Status	
DataUnderflowErrorCount	Returns Data Underflow Error Count
FifoBufferStatus	Returns Fifo Buffer Status
FrameReport	Returns Frame Reporting
GetProductID	Returns decoder type
GetState	Returns decoder state
LastDecoderCommand	Returns Last Decoder Command
PlayBit	Returns Play Bit
ProductInfo	Returns Product and SCSI Board information
QueueBit	Returns Queue Bit
ReadFrameCount	Read back number of frames displayed
RevisionInfo	Returns the Revision Page
SessionActive	Returns Active status of the current playback thread
UnderFlowBit	Returns UnderFlow bit
UnusedFiFoDataAfterStop	Returns Unused FiFo data after Stop
//Clip	
AVStreamId	Selects which audio and video stream IDs to decode
AVTransportPacketId	Specify the Transport Packet IDs
ErrorMask	Returns latest MPEG or Hardware errors encountered
GetMpegPath	Returns default path clip location
LastPlayBackError	Returns return code from the last playback thread
PlaybackStarted	Returns Flag indicating MpegPlay() issued
ReadyForNextVideo	Returns Flag if playback thread can accept next video
SetMpegPath	Sets default path clip location

CDecoder

C++ Method
CDecoder:: CDecoder ()

Description
Initializes private members

Parameters
N/A

Return Value
N/A

Comments
None

~CDecoder

C++ Method
CDecoder::~CDecoder()

Description
Issues a CLOSE() and releases attached resources.

Parameters
N/A

Return Value
N/A

Comments
None

Open

C++ Method
FUNC_STATUS CDecoder::Open(SHORT Port)

Description
Function opens device name attached to the selected port. The default if no port is provided will be to assign the next available port number.

Parameters
SHORT Port: Port number of decoder to accept the command.

Return Value
See page 120 for FUNC_STATUS definition.

Comments
To create a decoder port order different than the SCSI ID and LUN sequence, issue **CDecoder::Open(Port)** with port numbers in the sequence desired.

Close

C++ Method
FUNC_STATUS CDecoder::Close(BOOL Save)

Description
Method saves persistent properties, closes the channel, and makes the port available.

Parameters
BOOL Save: Save persistent values to the Registry via SAVEREGISTRY()

Return Value
See page 120 for FUNC_STATUS definition.

Comments
None

CLASS: CSCSI

Overview

The *CSCSI* class is a derived class from *CDecoder*. This class provides the common valid methods for all for SCSI1, SCSI2, SCSIx4, and SCSIx8 decoders.

PUBLIC Interface: *CSCSI* (v = virtual)

CSCSI	Class Constructor
~CSCSI	Class Destructor
Open (v)	Open decoder channel device
Close (v)	Close decoder channel device
Initialize (v)	Initialize decoder based on the saved settings
GetSettings (v)	Return decoder settings stored in the Registry
SetSettings (v)	Store decoder settings in Memory
RestoreSettings (v)	Retrieve decoder settings stored in the Registry
SaveSettings (v)	Store decoder settings in the Registry
// Setup Static	
DefaultVideoStandard	Set NTSC or PAL standards
SetYuvMode	Set output mode to Betacam YUV, SVHS, RGB
// Setup Dynamic	
AVSynch	Set thresholds for the Audio/Video synchronization
PALBufferMode	Set decoder's buffer MPEG mode for PAL
SetVideoPassThrough	Switch output from digital to analog (genlock) video
SwitchAudio	Select and setup audio output
VideoLevel	Modify the video reference level
VolumeControl	Modify attenuation of the audio output
BlackLevel	Set the Black Level to the desired point
BlankLevel	Set the Blank Level to the desired point
GainU	Set the GainU Level to the desired point
GainV	Set the GainV Level to the desired point

CLASS: CCinecast

Overview

The *CCinecast* class is a derived class from *CDecoder*. This class provides the common valid methods for CineCast Quad Pro SDI decoders.

PUBLIC Interface: *CCinecast* (v = virtual)

CCinecast	Class Constructor
~CCinecast	Class Destructor
Open (v)	Open decoder channel device
Close (v)	Close decoder channel device
Initialize (v)	Initialize decoder based on the saved settings
GetSettings (v)	Return decoder settings stored in the Registry
SetSettings (v)	Store decoder settings in Memory
RestoreSettings (v)	Retrieve decoder settings stored in the Registry
SaveSettings (v)	Store decoder settings in the Registry
// Setup Static	
BackgroundColor	Output the specified color during idle
ClosedCaptionReorder	Closed Caption based on bit stream or display order
DefaultResolution	Set the video standard output to NTSC or PAL
// Setup Dynamic	
AVChannelSelect	Control output selection for the two audio outputs
AudioGain	Adjust audio gain control
Blank	Turn video on or off
BufferReset	Reset the MPEG buffer pointers
Genlock	Enable decoder to synchronize to a genlock input source
LtcVitcTimeCode	Configure LTC Output and VITC Insertion/extraction
Mute	Mute audio output
SecondChannel	Enable the audio on the second channel
SetOutputDiscrete	Select Output Discrete and when to change state
SoftReset	Reset the decoder's audio and video circuits
TimeCode	Set SMPTE time code for next command
TrickModeFrameCount	Set frame count for next trick mode command
// VCR Play	
MpegPlay (v)	Start decoder playback
Play (v)	Playback selected clip
Stop	Stop the current video clip playing
// VCR Trick Mode	
FastMotion	Play the current video clip in fast motion
FrameAdvance	Advance the video one frame at a time
PauseVideo	Pause the current video clip playing
ResumeNormal	Resume the video clip paused
SlowMotion	Play the current video clip in slow motion

// Status	
GPIN1Bit	Returns state of general purpose input discrete 1
GPIN2Bit	Returns state of general purpose input discrete 2
ReadTimeCode	Read SMPTE time code for next command
ReadAudioLevel	Returns average Audio Levels
// Clip	
ErrorMask	Returns latest MPEG or Hardware errors encountered

Description

N/A

Parameters

SHORT Port: Port number of decoder to accept the command.

Return Value

Not yet documented.

Comments

None

Definitions

```
typedef enum _AUDIO_DATA
{
    AUDIO_LSI                   = 0x00,
    AUDIO_OVELAY                = 0x01,
    AUDIO_PASS                  = 0x02,
    AUDIO_MUTE                  = 0x03,
    AUDIO_LEFTMONO              = 0x04,
    AUDIO_STEREO                = 0x05,
    AUDIO_RIGHTMONO             = 0x06,
    AUDIO_LOUDOFF               = 0x00,
    AUDIO_LOUDON                = 0x01
}  AUDIO_DATA;

typedef enum _AUDIO_FUNC
{
    INPUT                       = 0x0,
    LOUDNESS                    = 0x1,
    BASS                        = 0x2,
    TREBLE                      = 0x3,
    LEFT_VOL                    = 0x4,
    RIGHT_VOL                   = 0x5,
    MODE_SEL                    = 0x6
}  AUDIO_FUNC;

typedef enum _AV_STREAM_CMD
{
    AV_DEFAULT = 0x00,
    AUDIO_ALL = 0x02,
    AUDIO_SET = 0x03,
    VIDEO_ALL = 0x04,
    VIDEO_SET = 0x05
}  AV_STREAM_CMD;
```

```
typedef struct _BOARD_INFO
{
   UCHAR PortNumber;          // Scsi port number
   UCHAR PathId;              // SCSI path id
   UCHAR TargetId;            // SCSI bus target id
   UCHAR Lun ;                // SCSI bus logical unit number
} BOARD_INFO_TYPE, *PBOARD_INFO_TYPE;

typedef enum _COLORMODE
{
   Black ,
   Blue ,
   UserDefined
} COLORMODE;

typedef enum _DEBUG_LEVEL
{
   DISABLE ,
   CLEAR ,
   COARSE ,
   MEDIUM ,
   FINE
} DEBUG_LEVEL;

typedef enum _DECODER_TYPE
{
   SCSI_ISA,
   SCSI2_ISA,
   SCSI_4X,
   SCSI_8X,
   DEC_SCSI,
   SCSI_CC
} DECODER_TYPE, *PDECODER_TYPE;
```

```
typedef union _DECODER_STATE_MASK
{
    volatile ULONG Mask ;
        struct {
        unsigned Idle:1;                // = 0;
        unsigned Playing:1;             // = 1;
        unsigned Pause:1;               // = 2;
        unsigned Resume:1;              // = 4;
        unsigned Stop:1;                // = 8;
        unsigned Preload:1;             // =0X10;
        unsigned Reset:1;               // = 0x20;
        unsigned Mute:1;                // = 0x40;
        unsigned UnMute:1;              // = 0x40;
        unsigned Test:1;                // = 0x100;
        unsigned Slow:1;                // = 0x200;
        unsigned Fast:1;                // = 0x400;
        unsigned Frame_Advance:1;       // = 0x800;
        unsigned Abort_Video:1;
        unsigned dummy :19 ;
    };
} DECODER_STATE_MASK,*PDECODER_STATE_MASK;

typedef enum _FUNC_STATUS
{
    SUCCESSFUL ,
    FAILURE ,
    CREATE_FILE_ERROR,
    CLOSE_HANDLE_ERROR,
    INVALID_PORT,
    INCORRECT_PORT,
    IOCTL_FAILURE,
    INVALID_BUFFER_LENGTH,
    INVALID_FILE_TYPE,
    GET_FILE_SIZE_ERROR,
    CREATE_FILE_MAPPING_ERROR,
    MAP_VIEW_OF_FILE_ERROR,
```

```
    END_OF_FILE,
    READ_FILE_ERROR,
    NO_DECODERS_FOUND,
    NULL_DECODER_BUFFER,
    NO_MOVIE_FILE,
    NO_PLAY_THREAD,
    VIDEO_STOP,
    UNABLE_TO_ALLOC_MEM,
    SET_FILE_POINTER_ERROR,
    END_OF_PLAY,
    INVALID_VENDOR,
    INVALID_ID,
    UNKNOWN_DECODER,
    INVALID_MICRO_FILE,
    NO_PLAY_LIST,
    NO_MPEG_INFO,
    INVALID_PLAY_LIST,
    INVALID_DRIVER,
    VIDEO_ABORT,
    INVALID_STATE,
    INVALID_DECODER_COMMAND,
    PLAYBACK_ACTIVE,
    INVALID_PARAMETER,
    PLAYBACK_FAILED,
    WRITE_FILE_ERROR,
    LAST_FUNC_STATUS
} FUNC_STATUS, *PFUNC_STATUS;

typedef enum _DL_TYPE
{
    CHECK_FILE              = 0x4,
    BURN_FLASH              = 0x5
} DL_TYPE, *PDL_TYPE;
```

```
typedef union _ERROR_MASK
{
    DWORD Mask ;
    struct {
        BYTE Res0;
        BYTE dummy1: 2;
        BYTE Genlock_Loss: 1;
        BYTE Collision : 1 ;
        BYTE Audio_Channel_Buffer: 1;
        BYTE Audio_Video_Mpeg_Decode: 1;
        BYTE Audio_Video_Channel_Buffer: 1;
        BYTE Lip_Sync: 1;
        BYTE dummy[3];
    };
} MPEG_ERROR_MASK;

typedef enum _SMODE
{
    Stereo,
    Left,
    Right
} MODE;

typedef struct
{
    PCHAR szFileName;           // pointer to file name
    ULONG StreamType;           // SYSTEM,PROGRAM,TRANSPORT etc.
    ULONG BitRate;              // Mbps
    ULONG FrameCount;           // Frame count in file
    SHORT HSize;                // horizontal resolution 704 etc.
    SHORT VSize;                // Vertical resolution 480 etc.
    BYTE FrameRateCode;         // frame code is 4=2997
    ULONG FileKBSize;           // File Kilobyte Size
    DWORDLONG FileSize;         // size of file
} MPEG_FILE_DETAILS,*PMPEG_FILE_DETAILS;
```

```
typedef enum _PID_TYPE
{
    VIDEO_PID      = 0x05,
    AUDIO1_PID     = 0x06,
    AUDIO2_PID     = 0x07,
    AUDIO1_PID_SEQ = 0x86,
    AUDIO2_PID_SEQ = 0x87
} PID_TYPE;

typedef struct _PRODUCT_DATA
{
    UCHAR Vendor[8];
    UCHAR ID[16];
    UCHAR Revision[4];
    UCHAR Serial[16];
} PRODUCT_TYPE, *PPRODUCT_TYPE;

typedef struct _REVISION_DATA
{
    BYTE Decoder_Firmware_Revision[4];
    BYTE SCSI_Programmable_Logic_Revision[4];
    BYTE MPEG_Programmable_Hardware_Revision[4];
    BYTE SCSI_Controller_Hardware_Revision[4];
    BYTE SCSI_Controller_Firmware_Revision[4];
    BYTE MPEG_Decoder_Hardware_Revision[4];
    BYTE Hardware_Revision[4];
    BYTE Res2_0: 5;
    BYTE Model: 1;
    BYTE Second_Audio: 1;
    BYTE Genlock: 1;
    BYTE Res3_0[3];
    BYTE Firmware_Build_Data[8];
} REVISION_TYPE, *PREVISION_TYPE;
```

```
typedef enum _SELECT
{
    OWN,
    CHN1,
    ADJ
}  SELECT;

typedef struct _SETTINGS
{
// Generic
    VIDEO_RESOLUTION Resolution;       // DefaultResolution()
    BOOLEAN FrameFreeze;          // FrameFreeze() | FieldFreeze()
    DWORD Mpegdecode;             // ErrorMask()
    BOOLEAN Seamless;             // NoBlack()
    UCHAR PreBlack;               // MpegPlay()
    WORD PostBlack;
    STREAM StreamType;
    BOOLEAN Trigger;
    TRIGGER TriggerEvent;
    BOOLEAN AutoSwitch;           // Genlock()
    BOOLEAN Genlock;

// SCSI 1,2,4,8
    VIDEO_STANDARD VideoStd;      // DefaultVideoStandard()
    BYTE Course;                  // AVSynch()
    BYTE Medium;
    BYTE Fine;
    BOOLEAN ErrorRecovery;        // MpegPlay() | Play() | BackToBackVideo()
    BOOLEAN Freeze;               // Stop()
    BOOLEAN EnablePlayCallback;   // EnablePlaybackSignal()
    BOOLEAN SoftTrigger;          // MpegPlay() | Play() | BackToBackVideo()
    BYTE LogLevel;                // SetEventLevel()
    BYTE PassMode;                // SetVideoPassThrough()
    BYTE Volume;                  // VolumeControl()
    BYTE Channel;
    SHORT BlackLevel;             // BlackLevel()
```

```
    SHORT BlankLevel;           // BlankLevel()
    SHORT GainU;                // GainU()
    SHORT GainV;                // GainV()
    AUDIO_FUNC AFunc;           // SwitchAudio()
    AUDIO_DATA AData;

// CineCast Quad Pro
    COLORMODE ColorMode;        // BackgroundColor()
    BYTE Y;
    BYTE Cb;
    BYTE Cr;
    DWORD BGColor;
    BOOLEAN Temporal;           // ClosedCaptionReorder()
    BOOLEAN Pal;                // DefaultResolution()

// Setup Dynamic
    BOOLEAN AudioChannelFade;       // AudioChannelFade()
    SOURCE PrimarySelect;
    MODE PrimaryStereo;
    SOURCE SecondarySelect;
    MODE SecondaryStereo;
    BOOLEAN SwapLRChn;
    SHORT FadeTime;
    BYTE VideoSwitch ;
    WORD PrimaryAudio;          // AudioGain()
    WORD SecondaryAudio;
    BOOLEAN PrimaryHeadroom;
    BOOLEAN SecondaryHeadroom;
    BOOLEAN FreezeFrame;        // BufferReset()
    SELECT GenlockSource;       // Genlock()
    BOOLEAN LTC;                // LtcVitcTimeCode()
    BOOLEAN VITC;
    BYTE OutputSource;
    WORD LineField1;
    WORD LineField2;
    BOOLEAN SecChnEnable;       // SecondChannel()
```

```
    BYTE DecodeOrder;
    BOOLEAN AC3;
    BOOLEAN GPOut1;          // SetOutputDiscrete()
    BOOLEAN Sync1;
    BOOLEAN GPOut2;
    BOOLEAN Sync2;
    BYTE SlowSpeed;          // SlowMotion()

// VCR Trick Mode
    DWORD SkipFrames;        // FrameAdvance()
    WORD FastWhich;          // FastMotion()
    WORD SlowWhich;          // SlowMotion()
    WORD SlowRepeat;
    TCSRC TCSource;          // ReadTimeCode()
    BYTE Chroma;             // MpegPlay()
} SETTINGS, * PSETTINGS;

typedef enum _SOURCE
{
    IBM,
    Crystal1,
    Crystal2,
    AdjIBM,
    AdjCrystal1,
    AdjCrystal2
} SOURCE;

typedef enum _STREAM
{
    AutoDetect               = 0x00,
    System                   = 0x10,
    VideoElm                 = 0x11,
    AudioElm                 = 0x12,
    Program                  = 0x20,
    Transport                = 0x28,
    VideoPES                 = 0x21,
```

```
    AudioPES = 0x22
}   STREAM;

typedef enum _TCSRC
{
    LTCOut,
    GOP,
    LTCIn,
    TCFrames
}   TCSRC;

typedef enum _TRIGGER
{
    GPIN1 = 0x0 ,
    GPIN2 = 0x1 ,
    LTCIN = 0x2,
    EOPLAY = 0x4
}   TRIGGER ;

typedef enum _VIDEO_PASS_MODE
{
    DISABLED ,
    VERTICAL_SYNC,
    TRIGGER_LOW,
    PLAYBACK,
    VIDEO_ONLY
}   VIDEO_PASS_MODE;

typedef enum _VIDEO_RESOLUTION
{
    NTSC_601 = 0,
    NTSC_SIF,
    PAL_601,
    NTSC_HALF_D1,
    PAL_HALF_D1,
    PAL_SIF
}   VIDEO_RESOLUTION;
```

```
typedef enum _VIDEO_STANDARD
{
   NTSC60hz = 0x3,              // NTSC 4.43 60HZ CCIR 13.5MHZ
   NTSC50hz = 0x4,              // NTSC 4.43 50HZ CCIR 13.5MHZ
   NTSC_M60hz = 0xD,            // NTSC-M 60HZ CCIR 13.5MHZ
   PAL_BG50hz = 0x85,           // PAL B/G 4.43 50HZ CCIR 13.5MHZ
   PAL_M60hz = 0x09,            // PAL-M 60HZ CCIR 13.5MHZ
   PAL_N50hz = 0x8A,            // PAL-N 50HZ CCIR 13.5MHZ
   PAL_N50hzCmb = 0x86
} VIDEO_STANDARD;

typedef enum _YUV_TYPE
{
   SMPTE_MODE                   = 0x00,
   BETA_MODE                    = 0x01,
   BETA700_MODE                 = 0x02,
   SVHS_MODE                    = 0x04,
   RGB_MODE                     = 0x08
} YUV_TYPE;
```

Fade Time Table

Fade Time	Value Total Fade Time (in msec.)	Fade Time	Value Total Fade Time (in msec.)
(1)	21.4 msec.	(9)	192.0 msec.
(2)	42.6 msec.	(10)	213.4 msec.
(3)	64.0 msec.	(11)	234.6 msec.
(4)	85.4 msec.	(12)	256.0 msec.
(5)	106.6 msec.	(13)	277.4 msec.
(6)	128.0 msec.	(14)	298.6 msec.
(7)	149.4 msec.	(15)	320.0 msec.
(8)	170.6 msec.	(16)	341.4 msec.

Note: Nominal fade time will be 128 msec. fade time.

Index