UDA5 Cod5 API Reference

Contents

1.1. Cod5 API

1.1.1. Functions

This section provides the references to the functions used in the Cod5 API. The following table summarizes the process states used to call each function.

Functions	**Callable State**			**State Change**
	Init	**Setup**	**Run**	
Cod5GetLastErrorCode	√	√	√	
Cod5GetSystemInfo	√	√	√	
Cod5GetBoardInfo	√	√	√	
Cod5QueryInfo	√	√		
Cod5Activate	√			
Cod5Setup	√			Setup
Cod5Endup		√	√	Init
Cod5Run		√		Run
Cod5Stop			√	Setup
Cod5ChannelEnable		√	√	
Cod5ReleaseData			√	
Cod5GetEventHandle		√	√	
Cod5GetEventData			√	
Cod5SetImageSize		√		
Cod5GetImageSize		√	√	
Cod5SetVideoFormat		√		
Cod5GetVideoFormat		√	√	
Cod5SetCodecProperty		√		
Cod5GetCodecProperty		√	√	
Cod5SetCodecAdjust		√	√	
Cod5GetCodecAdjust		√	√	
Cod5GetVideoStatus		√	√	
Cod5SetDO		√	√	
Cod5GetDO		√	√	
Cod5GetDI		√	√	
Cod5SetExtVideoOut		√	√	
Cod5GetExtVideoOut		√	√	
Cod5SetWatchdog			√	

Cod5GetLastErrorCode

This function retrieves the error information if the Cod5Api fails. The API's error codes are saved within the drive in the queue structure and include not only the error code but also the internal error information.

```
BOOL Cod5GetLastErrorCode(
       CMN5_ERROR_CODE_ITEM*pEcode          //
);
```

Parameters

pEcode

[out] Refers to the pointer to variables of CMN5_ERROR_CODE_ITEM type to retrieve the error information.

Return Values

If the function succeeds, the return value is nonzero. If not, it returns a zero value.

Remarks

When the function fails by calling the Cod5 API, call Cod5GetLastErrorCode to obtain information related to the error. CMN5_ERROR_CODE_ITEM structure includes the error code, error time, error string and error occurring session. This function, which is different from all other Cod5 APIs, does not require Cod5GetLastErrorCode to be called if the return is FALSE. It simply shows that there is no more error.

Different from GetLastError of the Win32 API, this function continually saves error information in queue format so that there is no immediate and compulsory need to call this function. Besides, the entire information of the internally occurred error – up to SYS level – can be verified when the calling of any API fails.

Msg can be of great help with debugging. It is a programming option to specify whether original messages will be exposed to users or not, but it is not recommended since there is no currently existing multilingual version driver.

If all API callings fail, call the Cod5GetLastErrorCode and check the possible reasons that may have caused the error.

Process State

Init, Setup, Run

Example Code

```
BOOL rs;
rs=Cod5Setup();
if(!rs){
       ErrorCodeItem errCode;
       Cod5GetLasError(&errCode);
       TRACE(“Error-%s\n”, errCode. AuxMsg);
}
```

See Also

CMN5_ERROR_CODE_ITEM

Cod5GetSystemInfo

Retrieves the Cod5 API's system information.

```
BOOL Cod5GetSystemInfo(
       CMN5_SYSTEM_INFO *pInfo              //
);
```

Parameters

pInfo

[out] Refers to the pointer to a CMN5_SYSTEM_INFO structure to receive the Cod5 API's system information.

Return Values

If the function succeeds, the return value is nonzero. If the function fails, it returns a zero value. To obtain extended error information, call Cod5GetLastErrorCode.

Remarks

Call this function before setting up the board to obtain information on the current driver and HW. The main purpose is to obtain the number of boards. This function can also be used to acquire the information on DLL or SYS versions to verify whether the current version is what APP requires.

Process State

Init, Setup, Run

Example Code

See Also

CMN5_SYSTEM_INFO

Cod5GetBoardInfo

Retrieves the information about the specified board.

```
BOOL Cod5GetBoardInfo(
       ULONG uBdID,           //
       const CMN5_BOARD_INFO_DESC *pDesc,           //
       COD5_BOARD_INFO *pInfo        //
);
```

Parameters

uBdID

[in] Refers to the index from 0 for the board. 0 indicates the 1st board and 1 indicates the 2nd board.

pDesc

[in] Describes *pInfo* information. Users should deliver *pInfo* information such as size and version. In API Version 5, users can transfer the explanation on COD5_BOARD_INFO.

pInfo

[out] Refers to the pointer for a variable to retrieve the board information. Users should use the exact pointer to the variable identical to the described contents in the *pDesc*. Pointer to the COD5_BOARD_INFO structure should be used in API Version 5.

Return Values

If the function succeeds, the return value is nonzero. If the function fails, it returns a zero value. To obtain extended error information, call Cod5GetLastErrorCode.

Remarks

Board information can be found in various types. Basically, COD5_BOARD_INFO Structure-type information is to be provided in Cod5 API. For required information in APP, the description for the corresponding structure is transferred through the *pDesc* and the contents can be received through the *pInfo*. Failure to provide the information in such type as described in *pDesc* will result in a return of FALSE.

The information that can be retrieved through this function includes the number of board channels, DIO figure and ModelID. The board should be activated with this ModelID to make other Cod5 API functions available.

This function can be used in all states (Init, Setup, Run).

Process State

Init, Setup, Run

Example Code

See Also

CMN5_BOARD_INFO_DESC, COD5_BOARD_INFO

Cod5QueryInfo

Retrieves special board information that cannot be retrieved by the Cod5GetSystemInfo or Cod5GetBoardInfo function.

```
BOOL Cod5QueryInfo (
       ULONG uQueryInfoCmd,          //
       ULONG uIn,            //
       void *pOut            //
);
```

Parameters

uQueryInfoCmd
[in] Specifies command to receive special features or attributes that the card supports.

uIn
[in] Specifies additional information.

pOut
[out] Pointer to a variable that a user allocated. When the function returns, this holds the information.

Return Values
If the function succeeds, the return value is nonzero. If the function fails, it returns a zero value. To obtain extended error information, call Cod5GetLastErrorCode.

Remarks
This function is used to retrieve the card features or related information that cannot be obtained by either Cod5GetSystemInfo or Cod5GetBoardInfo. Please refer to model SDK for detailed information.

Process State
Init, Setup, Run

Example Code

See Also

Cod5Activate

Activates the board to enable all Cod5 APIs including Cod5Setup.

```
BOOL Cod5Activate(
        ULONG uBdID,          //
        const UCHAR ActivationCode[CMN5_ACTIVATION_CODE_SIZE]              //
);
```

Parameters

uBdID

[in] Refers to the index from 0 for the board. 0 indicates the 1st board and 1 indicates the 2nd board.

ActivationCode

[in] Refers to the pointer to a 16 byte activation code matching to the corresponding ModelID. It is an encrypted code matching to the ModelID granted to each HW model.

Return Values

If the function succeeds, the return value is nonzero. If the function fails, it returns a zero value. To obtain extended error information, call Cod5GetLastErrorCode.

Remarks

If the Cod5 is not activated or an activation code entered is not corresponding to the ModelID, the Cod5 API can no longer be used. To use the API, all boards must be activated. All boards must be successfully activated to proceed to the setup step. An activation code will be provided separately by the company.

Process State

Init

Example Code

See Also

Cod5Setup

Changes the process state to setup state to enable the setup of driver properties.

```
BOOL Cod5Setup(void);
```

Parameters

This function has no parameters.

Return Values

If the function succeeds, the return value is nonzero. If the function fails, it returns a zero value. To obtain extended error information, call Cod5GetLastErrorCode.

Remarks

All boards should be activated by using the Cod5Activate before running this function. Then, set up all the related boards. Users can set up parameters for each board when the process state is changed to Setup state. It must be in Run State to enable data generation because no data will be generated in Setup State. Users must call Endup to return init-state

Process State

Init → Setup

Example Code

See Also

Cod5Endup

Ends all Cod5 API related activities, releases resources and turns the process state to Init State.

```
BOOL Cod5Endup(void);
```

Parameters

This function has no parameters.

Return Values

If the function succeeds, the return value is nonzero. If the function fails, it returns a zero value. To obtain extended error information, call Cod5GetLastErrorCode.

Remarks

This function releases all resources assigned by the driver and terminates the currently running code. It also changes the process state to Init State. Although in Run State, the function internally stops by force and carries out the process functionally similar to the Endup function.

However, it is recommended that users call Endup when the program is properly stopped and ready.

Process State

Setup, Run → Init

Example Code

See Also

Cod5Run

Runs the driver to capture compressed data. The function alters the process state to Run State in which the data can be generated and the change of state can be detected.

```
BOOL Cod5Run(void);
```

Parameters

This function has no parameters.

Return Values

If the function succeeds, the return value is nonzero. If the function fails, it returns a zero value. To obtain extended error information, call Cod5GetLastErrorCode.

Remarks

This function runs the driver that actually captures the data. This function must be called in the Setup State only after all the properties are successfully set up. If this function succeeds, the state alters to Run State and users can call available functions. In order to end the process after Run is successfully completed, users must call Stop. All data retrieving processes will only run in Run State.

Process State

Run → Stop

Example Code

See Also

Cod5Stop

Stops the driver to capture compressed data. It also changes the process state to Setup State.

```
BOOL Cod5Stop(void);
```

Parameters

This function has no parameters.

Return Values

If the function succeeds, the return value is nonzero. If the function fails, it returns a zero value. To obtain extended error information, call Cod5GetLastErrorCode.

Remarks

This function stops the driver process. Users can call this when in Run State and the state turns to Setup State if this function succeeds. Users can also call this function to end programs or edit the parameters that can only be changed in Setup State.

Process State

Run → Setup

Example Code

See Also

Cod5ChannelEnable

The Cod5ChannelEanble function changes the states of each channel to capture compressed data.

```
BOOL Cod5ChannelEnable(
       ULONG uBdID,          //
       ULONG uCh,            //
       BOOL bEnable          //
);
```

Parameters

uBdID

[in] Refers to the tndex from 0 for the board. 0 indicates the 1st board and 1 indicates the 2nd board.

uCh

[in] Refers to the tndex from 0 for the channel. 0 indicates the 1st channel and 1 indicates the 2nd channel.

bEnable

[in] If the function returns FALSE in Run State, the system stops generating the compressed data stream. If TRUE is returned, the data will be generated.
In Setup State it is decided whether to enable or disable data transfer in the next Run.

Return Values

If the function succeeds, the return value is nonzero. If the function fails, it returns a zero value. To obtain extended error information, call Cod5GetLastErrorCode.

Remarks

This function can force each channel to stop or start generating data. Users can call the function in both Setup and Run States. After setup, all channels are enabled by default, and after Run, they begin generating data. All changes are immediately applied if the function is called in Run State, while they will be applied in the next Run if the function is called in Setup State.

Process State

Setup, Run

Example Code

See Also

Cod5ReleaseData

Releases the data obtained from the driver.

```
BOOL Cod5ReleaseData(
       const void *pData             //
);
```

Parameters

pData

[in] Refers to the pointer for a variable that contains the data to be released. Transfer the pointer once again to the COD5_DATA_INFO structure among the data received through the Cod5GetEventData function.

Return Values

If the function succeeds, the return value is nonzero. If the function fails, it returns a zero value. To obtain extended error information, call Cod5GetLastErrorCode.

Remarks

Users must call Cod5ReleaseData and inform them that the data obtained through Cod5GetEventData is no longer used in the APP. If the data is released, the driver can recycle the buffer that has contained the data. Cod5GetEventData retrieves all the board or channel-related data contained in Cod5 API Data in the order in which they are generated. It is not necessary to release such data in the order in which they are retrieved, but the data obtained from each channel must be released in the same order as they are retrieved. It is generally recommended to release the data obtained from each channel in the same order as they are retrieved.

Process State

Run

Example Code

See Also

Cod5GetEventHandle

Retrieves EventHandles of specified data types.

```
BOOL Cod5GetEventHandle(
        ULONG uType,            //
        HANDLE *pHandle                 //
);
```

Parameters

uType,

[in] Refers to the data type that users want to obtain. Use the specified DataType. CMN5_DT_VIDEO, CMN5_DT_SENSOR, and CMN5_DT_VSTATUS are available.

pHandle

[out] Refers to the pointer for a variable used to save Event Handle values.

Return Values

If the function succeeds, the return value is nonzero. If the function fails, it returns zero. To obtain extended error information, call Cod5GetLastErrorCode.

Remarks

This function retrieves the EventHandles. It will import only EventHandle data of Auto Reset type. If related data is generated or its status is changed, it sends signals. The APP should not close those EventHandles.

Process State

Setup, Run

Example Code

See the Cod5GetEventData section.

See Also

Cod5GetEventData

Cod5GetEventData

Retrieves only specified data among those generated by driver.

```
BOOL Cod5GetEventData(
       ULONG uType,          //
       const void *pData            //
);
```

Parameters

uType

[in] Refers to the data type that users want to obtain. Use the specified DataType. Use one value among CMN5_DT_COD, CMN5_DT_SENSOR, and CMN5_DT_VSTATUS.

pData

[out] Refers to the pointer for a variable that will receive the information of the data starting with the CMN5_DATA_INFO_HEADER structure type or the data itself. It is necessary to transfer a structure proper to uType as follows:

uType	pData
CMN5_DT_COD	COD5_DATA_INFO*
CMN5_DT_SENSOR	CMN5_SENSOR_STATUS_INFO*
CMN5_DT_VSTATUS	CMN5_VIDEO_STATUS_INFO*

Return Values

If the function succeeds, the return value is nonzero. If the function fails, it returns a zero value. To obtain extended error information, call Cod5GetLastErrorCode.

Remarks

This function is mainly used to retrieve data of different API Sets. Users must check through the Wait Function of the Win32 if the event obtained by the Cod5GetEventHandle function is actually signalized, and must call Cod5GetEventData if the signal is confirmed.

Although users can call the Cod5GetEventData function without checking the event signal, such calling may be meaningless (however, the system does not enforce users to check the signals.) If any data is retrieved, it must be released through Cod5ReleaseData. If not, users can no longer receive any generated data.

Depending on data type, users can change the pointer: COD5_DATA_INFO for CMN5_DT_COD, CMN5_VIDEO_STATUS_INFO for CMN5_DT_VSTATUS, and CMN5_SENSOR_STATUS_INFO for

DT_VSTATUS.

Process State

Run

Example Code

```
DWORD DataLoopThread ()
{
        BOOL retVal;
        HANDLE events[3];

        events[0] = hStopEvent;
        Cod5GetEventHandle(CMN5_DT_SENSOR,   &events[1]);
        Cod5GetEventHandle(CMN5_DT_COD,      &events[2]);

        while(1) {
                DWORD obj = WaitForMultipleObjects(3, events, FALSE, INFINITE);
                switch(obj){
                case WAIT_OBJECT+1: {
                        do {
                                COD5_DATA_INFO info={0,};
                                if(retVal=Cod5GetEventData(CMN5_DT_COD,&info)) {
                                        if(info.pDataBuffer) {
                        OnCapture((CMN5_DATA_INFO_HEADER*)&info);
                                        }
                                        Cod5ReleaseData((void*)&info);
                                }
                        } while(retVal && info.uHasNextData);
                        break;
                }

                case WAIT_OBJECT+2:{
                        CMN5_SENSOR_STATUS_INFO info={0,};
                        if(Cod5GetEventData(CMN5_DT_SENSOR, &info)) {
                                OnSensor(&info);
                        }
                        break;
                }
                default:
                        break;
                }
        }
}
```

See Also

Cod5GetEventHandle WaitForMultipleObjects, COD5_DATA_INFO, CMN5_SENSOR_STATUS_INFO , CMN5_VIDEO_STATUS_INFO

Cod5SetImageSize

Sets the image size (resolution) of compressed video data.

```
BOOL Cod5SetImageSize(
       ULONG uBdID,           //
       ULONG uCh,             //
       ULONG uImageSize              //
);
```

Parameters

uBdID

[in] Refers to the index from 0 for the board. 0 indicates the 1st board and 1 indicates the 2nd board.

uCh

[in] Refers to the index from 0 for the channel. 0 indicates the 1st channel and 1 indicates the 2nd channel.

uImageSize

[in] Refers to the image value to be set. Bit [0..11] (12 bits) indicates the width, and bit [12..23] (12 bits) indicates the height.

Return Values

The function fails if the image value is set out of the range that the driver can support. If the function succeeds, the return value is nonzero. If the function fails, it returns a zero value. To obtain extended error information, call Cod5GetLastErrorCode.

Remarks

Each board or driver can support limited types of image sizes. To check if any resolution is supported by the driver or not, call Cod5GetBoardInfo for each board, receive the information regarding the COD5_BOARD_INFO structure type, and check the pResInfo variable. pResInfo is a pointer to structures of CMN5_RESOLUTION_INFO type. For details, refer to CMN5_RESOLUTION_INFO descriptions.

In the video format settings, image size turns to default so that users must change it later according to their needs. It must be remembered that if users change the video format after setting the image size, the new video format will have, by default, the image size set for the previous format.

Users can set image size more easily if using macros such as CMN5_MAKEIMGSIZE, CMN5_GETIMGWIDTH, or CMN5_GETIMGHEIGHT.

Process State

Setup

Example Code

See Also

Cod5GetImageSize

Retrieves the values set for image size (resolution).

```
BOOL Cod5GetImageSize(
       ULONG uBdID,          //
       ULONG uCh,            //
       ULONG *puImageSize            //
);
```

Parameters

uBdID

[in] Refers to the index from 0 for the board. 0 indicates the 1st board and 1 indicates the 2nd board.

uCh

[in] Refers to the index from 0 for the channel. 0 indicates the 1st channel and 1 indicates the 2nd channel.

puImageSize

[in] Refers to the pointer for ULONG-typed variables that will receive image size values. Bit [0..11] (12 bits) indicates the width, and bit [12..23] (12 bits) indicates the height

Return Values

If the function succeeds, the return value is nonzero. If the function fails, it returns a zero value. To obtain extended error information, call Cod5GetLastErrorCode.

Remarks

Process State

Setup, Run

Example Code

See Also

Cod5SetVideoFormat

Sets the video format of the Cod5.

```
BOOL Cod5SetVideoFormat(
       ULONG uBdID,          //
       ULONG uFormat         //
);
```

Parameters

uBdID

[in] Refers to the index from 0 for the board. 0 indicates the 1st board and 1 indicates the 2nd board.

uFormat

[in] Use the defined value.

CMN5_VIDEO_FORMAT_NTSC_M, and CMN5_VIDEO_FORMAT_PAL_B are currently available.

Return Values

If the function succeeds, the return value is nonzero. If the function fails, it returns a zero value. To obtain extended error information, call Cod5GetLastErrorCode.

Remarks

This function can be called only in Setup State. One format must be specified before proceeding to Run State. It is possible to specify different video formats to each board, but not to each channel. The format must be set before specifying the image size.

Process State

Setup

Example Code

See Also

Cod5GetVideoFormat

Retrieves the video format currently set in the board.

```
BOOL Cod5GetVideoFormat(
        ULONG uBdID,            //
        ULONG *puFormat                 //
);
```

Parameters

uBdID

[in] Refers to the index from 0 for the board. 0 indicates the 1st board and 1 indicates the 2nd board.

puFormat

[out] Refers to the pointer for a variable to receive the value of the current video format.

Return Values

If the function succeeds, the return value is nonzero. If the function fails, it returns a zero value. To obtain extended error information, call Cod5GetLastErrorCode.

Remarks

Process State

Setup, Run

Example Code

See Also

Cod5SetVideoFormat

Cod5SetCodecProperty

Sets the Cod5's variables available in Setup State.

```
BOOL Cod5SetCodecProperty(
       ULONG uBdID,          //
       ULONG uCh,            //
       ULONG uPropertyCmd,          //
       ULONG uParam1,        //
       ULONG uParam2,        //
       ULONG uParam3,        //
       ULONG uParam4         //
);
```

Parameters

uBdID

[in] Refers to the index from 0 for the board. 0 indicates the 1st board and 1 indicates the 2nd board.

uCh

[in] Refers to the index from 0 for the channel. 0 indicates the 1st channel and 1 indicates the 2nd channel.

uPropertyCmd

[in] Uses the value of enum CodecPropertyCmd. This parameter decides the actual role of the Cod5SetProperty. The meanings of uParam1 to uParam4 vary according to this value. The following values are used:

Command	Description and Parameters
COD5_CPC_STREAM_TYPE	**Defines the stream type to be generated by the codec. Combines and uses one or more values of enum CodecStreamType for *uParam1*. For example, COD5_CST_VIDEO|COD5_CST_AUDIO means the codec generates video/audio streams. COD5_CST_VIDEO means the codec generates only video stream. Remember that some drivers do not support several enum CodecStreamType values.**
COD5_CPC_VIDEO_FORMAT	**Sets the video format and the codec. Uses *uParam1* and enters the video format be used.**
COD5_CPC_VIDEO_IMAGE_SIZE	**Defines the video image size. Uses *uParam1* and enters the value in the form of**

	CMN5_MAKEIMGSIZE(w, h).
COD5_CPC_BITRATE	**Determines the bit rate mode to be set by *uParam1*. Uses the value of num BitrateMode. Other parameters are determined depending on the value of *uParam1*. For details, see the following table.**
COD5_CPC_GOP_SIZE	**Sets the GOP (Group of Pictures) size of the Mpeg video. If the total number of the pictures that makes up *uParam1* GOP is 1, the size will be set to I Picture. If the number is 2, the size will be set to IP, and if 3, IPP. Defines the number of the B Pictures between I and P Pictures in *uParam2* GOP. The number zero means there is no B Picture, and the number 1 means there is one B Picture between I and P Pictures.**
COD5_CPC_AUDIO_ATTR	**Sets the values related to audio compression. Sets the *uParam1* Sampling rate. Also able to set the value of enum AudioCodecSamplFreq. Sets the *uParam2 Bit* rate. All the values are indicated in kbps unit.**
COD5_CPC_SKIP_FRAME	**Defines the skip frames of the compressed video stream. The actual frame rate will be the total frames divided by (skipframe+1). If the frame rate decreases, the real bit rate decreases in proportion to the defined bit rate. *uParam1* is the skip frame value to be set. For the NTSC, if the parameter is 0, the skip frame will be 30; if 1, 25, and if 2, 30.**
COD5_CPC_CODEC_TYPE	**Sets the codec. uParam1 defines the video codec and uParam2 defines the audio codec.**

In case of CAC_BITRATE, the values of *uParam1* and the meanings of other parameters are as follows:

Bit Rate Mode	Description and Parameters
COD5_BM_CBR	Sets bitrate as Constant Bitrate. uParam2 is the average bitrate indicated in bps.
COD5_BM_VBRR	Sets the average and peak values of Variable Bitrate. Variable Bitrate per second is usually maintained at the average level and the maximum bitrate

	must be set smaller than the peak value. uParam2 is the average bitrate indicated in bps. uParam3 is the peak bitrate indicated in bps
COD5_BM_VBRQ	Sets the value of Variable Bitrate Quantization. uParam2 is the Quantization value between 1 to 63
COD5_BM_VBRH	Sets the minimum and maximum values of Variable Bitrate Hybrid. uParam2 is the minimum bitrate indicated in bps. uParam3 is the maximum bitrate indicated in bps

uParam1
[in] Refers to the additional parameter, whose meaning varies according to *uPropertyCmd.*

uParam2
[in] Refers to the additional parameter, whose meaning varies according to *uPropertyCmd.*

uParam3
[in] Refers to the additional parameter, whose meaning varies according to *uPropertyCmd.*

uParam4
[in] Refers to the additional parameter, whose meaning varies according to *uPropertyCmd.*

Return Values
If the function succeeds, the return value is nonzero. If the function fails, it returns a zero value. To obtain extended error information, call Cod5GetLastErrorCode.

Remarks

Process State
Setup

Example Code

See Also

Cod5GetCodectProperty

Retrieves the values set by the Cod5SetProperty function.

```
BOOL Cod5GetCodecProperty(
       ULONG uBdID,          //
       ULONG uCh,            //
       ULONG uPropertyCmd,          //
       ULONG* puParam1,             //
       ULONG* puParam2,             //
       ULONG* puParam3,             //
       ULONG* puParam4              //
);
```

Parameters

uBdID

[in] Refers to the index from 0 for the board. 0 indicates the 1st board and 1 indicates the 2nd board.

uCh

[in] Refers to the index from 0 for the channel. 0 indicates the 1st channel and 1 indicates the 2nd channel.

uPropertyCmd

[in] Uses the value of enum CodecPropertyCmd. This parameter decides the actual role of Cod5GetProperty. The meanings of puParam1 to puParam4 vary according to this value. The following values are used:

Command	Description and Parameters
COD5_CPC_STREAM_TYPE	**Retrieve which stream is set. Combines and transfers one or more values of enum CodecStreamType to p*uParam1*. For example, COD5_CST_VIDEO|COD5_CST_AUDIO means the codec generates video/audio streams. COD5_CST_VIDEO means only the video stream is generated.**
COD5_CPC_VIDEO_FORMAT	**Retrieve the video format and the codec set in the system. The video format is transferred to the lower 16 bit of *puParam1* and the codec definition value to the upper 16 bit. (uParam1 = uVideoFormat | uCodec << 16)**

	If the upper 16 bit is zero, it means that the default codec will be used.
COD5_CPC_VIDEO_IMAGE_SIZE	**Retrieve the defined video image size.**
COD5_CPC_BITRATE	**Retrieve the defined bitrate mode.**
COD5_CPC_GOP_SIZE	**Retrieve the defined GOP (Group of Pictures) size of the Mpeg video.**
COD5_CPC_AUDIO_ATTR	**Retrieve the value set for the audio compression.**
COD5_CPC_SKIP_FRAME	**Retrieve the number of defined Skip Frames.**
COD5_CPC_CODEC_TYPE	**Retrieve the defined codec type.**

puParam1
[out] Refers to the additional parameter, whose meaning varies according to *uPropertyCmd.*

puParam2
[out] Refers to the additional parameter, whose meaning varies according to *uPropertyCmd.*

puParam3
[out] Refers to the additional parameter, whose meaning varies according to *uPropertyCmd.*

puParam4
[out] Refers to the additional parameter, whose meaning varies according to *uPropertyCmd.*

Return Values
If the function succeeds, the return value is nonzero. If the function fails, it returns a zero value. To obtain extended error information, call Cod5GetLastErrorCode.

Remarks

Process State
Setup

Example Code

See Also

Cod5SetProperty

Cod5SetCodecAdjust

Sets the Cod5's variables available in Setup/Run States.

```
BOOL Cod5SetCodecAdjust(
        ULONG uBdID,          //
        ULONG uCh,            //
        ULONG uColorAdjustCmd,        //
        ULONG uParam1,        //
        ULONG uParam2,        //
        ULONG uParam3,        //
        ULONG uParam4         //
);
```

Parameters

uBdID

[in] index from 0 for the board. 0 indicates the 1st board and 1 indicates the 2nd board.

uCh

[in] Iindex from 0 for the channel. 0 indicates the 1st channel and 1 indicates the 2nd channel.

uAdjustCmd

[in] Uses the value of enum CodecAdjustCmd. This parameter decides the actual role of Cod5SetAjust. The meanings of uParam1 to uParam4 vary according to this value. The following values are used:

Command	Description and Parameters
COD5_CAC_AUDIO_GAIN	Sets the value of audio volume. *uparam1* sets the value from 0 to 255. Default value is 128.
COD5_CAC_BRIGHTNESS	Sets the value of video brightness. *uparam1* sets the value from 0 to 255. Default value is 128.
COD5_CAC_CONTRAST	Sets the value of video contrast. *uparam1* sets the value from 0 to 255. Default value is 128.
COD5_CAC_SATURATION_U	Sets the value of video saturation U. *uparam1* sets the value from 0 to 255. Default value is 128.
COD5_CAC_SATURATION_V	Sets the value of video saturation V. *uparam1* sets the value from 0 to 255. Default value is 128.
COD5_CAC_HUE	Sets the value of video hue. *uparam1* sets the value from 0 to 255. Default value is 128.

Caution!

If your item is one of the IPC series or NVE100 with PAL video format, Hue value adjustment doesn’t work at all. This is because of the characteristic of the decoder chip built in IPC series and NVE100. If your video format is NTSC, it has no problem in hue value adjustment.

uParam1
[in] Refers to the additional parameter, whose meaning varies according to *uAdjustCmd.*

uParam2
[in] Refers to the additional parameter, whose meaning varies according to *uAdjustCmd.*

uParam3
[in] Refers to the additional parameter, whose meaning varies according to *uAdjustCmd.*

uParam4
[in] Refers to the additional parameter, whose meaning varies according to *uAdjustCmd.*

Return Values
If the function succeeds, the return value is nonzero. If the function fails, it returns a zero value. To obtain extended error information, call Cod5GetLastErrorCode.

Remarks
Available in Setup/Run States.

Process State
Setup, Run

Example Code

See Also

Cod5GetCodecAdjust

Retrieves the values set by the Cod5SetAdjust function.

```
BOOL Cod5GetCodecAdjust(
       ULONG uBdID,          //
       ULONG uCh,            //
       ULONG  uColorAdjustCmd,              //
       ULONG* puParam1,             //
       ULONG* puParam2,             //
       ULONG* puParam3,             //
       ULONG* puParam4              //
);
```

Parameters

uBdID

[in] Refers to the index from 0 for the board. 0 indicates the 1st board and 1 indicates the 2nd board.

uCh

[in] Refers to the index from 0 for the channel. 0 indicates the 1st channel and 1 indicates the 2nd channel.

uAdjustCmd

[in] Uses the value of enum CodecAdjustCmd. This parameter decides the actual role of Cod5SetAjust. The meanings of puParam1 to puParam4 vary according to this value. The following values are used:

Command	Description and Parameters
COD5_CAC_AUDIO_GAIN	The value of audio volume is to be transferred to puParam1.
COD5_CAC_BRIGHTNESS	The value of video brightness is to be transferred to puParam1
COD5_CAC_CONTRAST	The value of video contrast is to be transferred to puParam1
COD5_CAC_SATURATION_U	The value of video saturation U is to be transferred to puParam1.
COD5_CAC_SATURATION_V	The value of video saturation V is to be transferred to puParam1.
COD5_CAC_HUE	The value of video hue is to be transferred to puParam1.

puParam1

[out] Refers to the additional parameter, whose meaning varies according to *uAdjustCmd.*

puParam2

[out] Refers to the additional parameter, whose meaning varies according to *uAdjustCmd.*

puParam3

[out] Refers to the additional parameter, whose meaning varies according to *uAdjustCmd.*

puParam4

[out] Refers to the additional parameter, whose meaning varies according to *uAdjustCmd.*

Return Values

If the function succeeds, the return value is nonzero. If the function fails, it returns a zero value. To obtain extended error information, call Cod5GetLastErrorCode.

Remarks

Available in Setup/Run States.

Process State

Setup, Run

Example Code

See Also

Cod5SetAdjust

Cod5GetVideoStatus

Finds the video input states of all the channels of the specified Cod5 boards.

```
BOOL Cod5GetVideoStatus(
       ULONG uBdID,          //
       ULONG uMaxCh,         //
       ULONG *pbwData        //
);
```

Parameters

uBdID

[in] Refers to the index from 0 for the board. 0 indicates the 1st board and 1 indicates the 2nd board.

uMaxCh

[in] Refers to the number of channels to be obtained. This parameter must have the same number of total bits as the real variables of *pbwData* or less.

pbwData

[out] The pointer for a variable to receive the video status information of each channel in a bitwise format. The variable name of *pbwData* is the abbreviation of “pointer to bitwised Data.” It is the pointer to ULONG-typed variables or arrays. If the number of channels exceed 32 (bits of ULONG), an array pointer must be transferred. Send the variables like ULONG Data [2] for 64 channels and ULONG Data [8] for 256 channels.

Return Values

If the function succeeds, the return value is nonzero. If the function fails, it returns a zero value. To obtain extended error information, call Cod5GetLastErrorCode.

Remarks

Each board can have up to 256 channels. If one 32-bit ULONG is used, it is impossible to retrieve the status information of all the channels of the board. If the APP is designed to support less than 32 channels per board, pbwData just sends the pointer to a ULONG-typed variable. In this case, enter the number of the board channels into the uMaxCh parameter.

Process State

Setup, Run

Example Code

See Also

Cod5SetDO

Changes the states of all the DOs of the specified board.

```
BOOL Cod5SetDO(
       ULONG uBdID,           //
       ULONG uMaxDO,          //
       const ULONG *pbwData          //
);
```

Parameters

uBdID

[in] Refers to the index from 0 for the board. 0 indicates the 1st board and 1 indicates the 2nd board.

uMaxDO

[in] Refers to the number of channels to be set. This parameter must have the same number of total bits as the real variables of *pbwData* or less.

pbwData

[in] The pointer for a variable that will set the values for each DO. The variable name of *pbwData* is the abbreviation of "pointer to bitwised Data." It is the pointer to ULONG-typed variables or arrays. An array pointer must be transferred if the number of DOs exceeds 32 (bits of ULONG). For example, transfer the variables like ULONG Data [2] to 64 DOs, and ULONG Data [8] to 256 DOs.

Return Values

If the function succeeds, the return value is nonzero. If the function fails, it returns a zero value. To obtain extended error information, call Cod5GetLastErrorCode.

Remarks

Each board can have up to 256 DOs. If one 32-bit ULONG is used, it is impossible to set the status information of all the channels of the board. If the APP is designed to support less than 32 DOs per board, pbwData just sends the pointer to a ULONG-typed variable. In this case, enter the number of the board DOs into the uMaxDO parameter.

Process State

Setup, Run

Example Code

Process State

See Also

Cod5GetDO

Retrieves the DO settings of the specified Cod5 board.

```
BOOL Cod5GetDO(
       ULONG uBdID,           //
       ULONG uMaxDO,          //
       ULONG *pbwData         //
);
```

Parameters

uBdID

[in] Refers to the index from 0 for the board. 0 indicates the 1st board and 1 indicates the 2nd board.

uMaxDO

[in] Refers to the number of channels to be obtained. This parameter must have the same number of total bits as the real variables of *pbwData* or less.

pbwData

[in] Refers to the pointer for a variable that will set the values for each DO. The variable name of *pbwData* is the abbreviation of "pointer to bitwised Data." It is the pointer to ULONG-typed variables or arrays. An array pointer must be transferred if the number of DOs exceeds 32 (bits of ULONG). For example, transfer the variables like ULONG Data [2] to 64 DOs, and ULONG Data [8] to 256 DOs.

Return Values

If the function succeeds, the return value is nonzero. If the function fails, it returns a zero value. To obtain extended error information, call Cod5GetLastErrorCode.

Remarks

Each board can have up to 256 DOs. If one 32-bit ULONG is used, it is impossible to retrieve the status information of all the DOs of the board. If the APP is designed to support less than 32 DOs per board, pbwData just sends the pointer to a ULONG-typed variable. In this case, enter the number of the board DOs into the uMaxDO parameter.

Process State

Setup, Run

Example Code

Process State

See Also

Cod5GetDI

Reads the current states of DIs.

```
BOOL Cod5GetDI(
       ULONG uBdID,           //
       ULONG uMaxDI,          //
       ULONG *pbwData         //
);
```

Parameters

uBdID

[in] Refers to the index from 0 for the board. 0 indicates the 1st board and 1 indicates the 2nd board.

uMaxDI,

[in] Refers to the number of DIs to be obtained. This parameter must have the same number of total bits as the real variables of *pbwData* or less.

pbwData

[out] The pointer for a variable to receive the state information of each DI in a bitwise format. The variable name of *pbwData* is the abbreviation of "pointer to bitwised Data." It is the pointer to ULONG-typed variables or arrays. An array pointer must be transferred if the number of DIs exceeds 32 (bits of ULONG). For example, transfer the variables like ULONG Data [2] to 64 DIs, and ULONG Data [8] to 256 DIs.

Return Values

If the function succeeds, the return value is nonzero. If the function fails, it returns a zero value. To obtain extended error information, call Cod5GetLastErrorCode.

Remarks

Users can obtain event notifications about DIs only in Run State, but can retrieve the state information in both the Setup and Run states. Each board can have up to 256 DIs. If one 32-bit ULONG is used, it is impossible to retrieve the status information of all the DIs of the board. If the APP is designed to support less than 32 DIs per board, pbwData just sends the pointer to a ULONG-typed variable. In this case, enter the number of the board DIs into the uMaxDI parameter.

Process State

Setup, Run

Example Code

See Also

Cod5SetExtVideoOut

Selects a channel that will send out video data to an external video output.

```
BOOL Cod5SetExtVideoOut(
        ULONG uBdID,           //
        ULONG uCh              //
);
```

Parameters

uBdID

[in] Refers to the index from 0 for the board. 0 indicates the 1st board and 1 indicates the 2nd board.

uCh

[in] Refers to the index from 0 for the channel. 0 indicates the 1st channel and 1 indicates the 2nd channel.

Return Values

If the function succeeds, the return value is nonzero. If the function fails, it returns a zero value. To obtain extended error information, call Cod5GetLastErrorCode.

Remarks

Process State

Setup, Run

Example Code

Process State

See Also

Cod5GetExtVideoOut

Gets the channel number of the external video to which the Cod5 is sending data.

```
BOOL Cod5GetExtVideoOut(
       ULONG uBdID,          //
       ULONG *puCh           //
);
```

Parameters

uBdID

[in] Refers to the index from 0 for the board. 0 indicates the 1st board and 1 indicates the 2nd board.

puCh

[out] Refers to the ULONG pointer to receive the channel index. Channel index 0 indicates the 1st channel and 1 indicates the 2nd channel.

Return Values

If the function succeeds, the return value is nonzero. If the function fails, it returns a zero value. To obtain extended error information, call Cod5GetLastErrorCode.

Remarks

Process State

Setup, Run

Example Code

Process State

See Also

Cod5SetWatchdog

Sets and triggers various parameters of the WatchDog.

```
BOOL Cod5SetWatchdog(
       ULONG uBdID,           //
       ULONG uWatchdogCmd,           //
       ULONG uParam1,         //
       ULONG uParam2          //
);
```

Parameters

uBdID

[in] Refers to the index from 0 for the board. 0 indicates the 1st board and 1 indicates the 2nd board.

uWatchdogCmd

[in] Decides the actual role of the function. The meanings of additional parameters are determined according to this value.

Values	**Description and Parameters**
CMN5_WC_ENABLE	Activates WatchDog. No parameter is used
CMN5_WC_DISABLE	Deactivates WatchDog. No parameter is used.
CMN5_WC_SET_TIMEOUT_VALUE	Sets the Timeout value of the WatchDog. *uParam1*[in] sets Timeout in seconds and to values over zero. If the value is zero, the function will fail.
CMN5_WC_TRIGGER	Updates the values according to those set by WC_SET_TIMEOUT_VALUE. No parameter is used.
CMN5_WC_SET_BUZZER_TIMEOUT_VALUE	Sets the period during which the Buzzer makes beep sounds. *uParam1*[in] sets Buzzer Timeout in seconds

uParam1

[in] Refers to the additional parameter, whose meaning varies according to *uWatchdogCmd*.

uParam2

[in] Refers to the additional parameter, whose meaning varies according to *uWatchdogCmd*.

Return Values

If the function succeeds, the return value is nonzero. If the function fails, it returns a zero value. To obtain extended error information, call Cod5GetLastErrorCode.

Remarks

When the system does not respond, the Watchdog resets it by force. The APP continually triggers the driver to inform that the system is working. If there is no triggering during the period set as TimeOut, the buzzer makes alarm sounds for the period set as Buzzer Timeout and the system will be reset by force.
Some models do not include buzzers. If a model does not have any buzzer, the function returns FALSE. It is highly recommended to set within default Timeout 20 minutes after changing to ENABLE. If the Timeout value is set in DISABLE state, the buzzer will not be triggered. Timeout value must be set in DISABLE state and then the buzzer starts triggering in ENABLE state. It is also possible to start triggering at the moment when the Timeout value is being specified. If a command such as TRIGGERING, DISABLE, or SET_TIMEOUT is executed while the buzzer is alarming, the buzzer stops making sounds and is not to be reset.
To reset the buzzer, it takes time as long as the periods are set for Timeout after the first triggering and for additional Buzzer Timeout. In short, the actual RESETTIMEOUT will be TIMEOUT plus BUZZERTIMEOUT.

Process State

Run

Example Code

Process State

See Also

1.1.2. Structures

COD5_BOARD_INFO

Displays the information concerning the Media Data (Compressed Video/Audio/Etc) generated by the Cod5 API.

```
typedef struct {
       ULONG          uBoardID;
       ULONG          uModelID;
       ULONG          uSlotNumber;
       ULONG          uMaxVP;
       ULONG          uMaxChannel;
       ULONG          uMaxDO;
       ULONG          uMaxDI;
       ULONG          uCodecType;
       ULONG          uVideoCodecType;
       ULONG          uAudioCodecType;
       const CMN5_RESOLUTION_INFO* pResInfo;        //132*sizeof(ULONG)
       ULONG          reserved[CMN5_MAX_RESERVED_BOARD_INFO - 11];
} COD5_BOARD_INFO;
```

Members

uBoardID

Refers to the index for the board. 0 indicates the 1st board and 1 indicates the 2nd board.

uModelID

Refers to the model ID of the board.

uSlotNumber

Refers to the number of the physical slots of the board. The slot after the AGP is 0, and the next slot is 1.

uMaxVP

Refers to the number of the VPs of the board.

uMaxChannel

Refers to the number of the channels of the board.

uMaxDO

Refers to the number of the DOs of the board. This number will be zero when there is no DO (Digital Out), and can be set up to 256.

uMaxDI

Refers to the number of the DIs of the board. This number will be zero when there is no DI

(Digital In), and can be set up to 256.

uCodecType

Refers to the generated stream type. Set by enum CodecStreamType. If there are more than two options, this member will be default codec type.

uVideoCodecType

Types of video stream and codec. Set by enum VideoCodecType. If there are more than two options, this member will be default codec type.

uAudioCodecType

Refers to the types of audio stream and codec. Set by enum AudioCodecType. If there are more than two options, this member will be default codec type.

pResInfo

Refers to the resolution information supported by the board.

reserved

Refers to the reserved field. Not used.

Remarks

COD5_DATA_INFO

Displays the information concerning the Media Data (Compressed Video/Audio) generated by the Cod5 API.

```
typedef struct {
       ULONG                 uDataType;
       ULONG                 uStructSize;
       ULONG                 uErrCode;
       ULONG                 uBoardNum;     // Board ID[0..3]
       ULONG                 uChannelNum;   // Channel ID of the Board
       ULONG                 uHasNextData;
       UCHAR                 *pDataBuffer;  // image address
       LARGE_INTEGER  TimeTag;              // KeQuerySystemTime()
       ULONG                 uBufferIdx;    // internal use! do not touch!
       ULONG                 uDataSize;
       ULONG                 uFrameType;
} COD5_DATA_INFO;
```

Members

uDataType

Retrieved data types.

DATA Type	Description
CMN5_DT_VIDEO	Uncompressed video data.
CMN5_DT_AUDIO	Audio data.
CMN5_DT_COD	Codec data, compressed video data.
CMN5_DT_SENSOR	Data entered by the sensor data DI.
CMN5_DT_VSTATUS	Video input state data.
CMN5_DT_NET	Network data.

uStructSize

Refers to the structure size.

uErrCode

Refers to the error code.

Error code	Description
CMN5_EC_NO_ERROR	Normal state without errors.
CMN5_EC_NO_DATA	No data.
CMN5_EC_OVERFLOW	Data buffer overflow. If this error occurs, restarting is generally required.
CMN5_EC_ERROR	PCI error.
CMN5_EC_FIRMWARE_ERROR	Firmware error. If this error occurs, restarting is generally required.
CMN5_EC_FIRMWARE_DEAD	Firmware causes error and the system does not work even after restart. If this error occurs, the system must generally be restarted.

uBoardNum

Refers to the board number of the retrieved data.

uChannelNum

Refers to the channel number of the retrieved data.

uHasNextData

Refers to the availability of the data to be retrieved. If the data exists, it returns TRUE, and if not, it returns FALSE.

pDataBuffer

Refers to the memory pointer where the data is saved.

TimeTag

Refers to the time when the data is generated.

uBufferIdx

Common with CMN5_MEDIA_DATA_INFO.

uDataSize

Refers to the size of the generated data (equal to the size of compressed stream). Indicated in Bytes.

uFrameType

Refers to the types of the generated data. The value of enum FrameType is used.

Remarks