ADV7511 Transmitter API

Revision history

Date	Rev	Author	Description
11.02.2013	1.0		Document creation

Contents

1. Introduction

The ADV7511 is a 225 MHz High-Definition Multimedia Interface (HDMI®) transmitter, which is ideal for home entertainment products including DVD players/recorders, digital set top boxes, A/V receivers, gaming consoles, and PCs.
The digital video interface contains an HDMI 1.4- and a DVI 1.0-compatible transmitter, and supports all HDTV formats (including 1080p with 12-bit Deep Color). The ADV7511 supports the HDMI 1.4-specific features, HEAC (ARC), and 3D video. In addition to these features, the ADV7511 supports x.v.Color™, high bit rate audio, and programmable AVI InfoFrames. With the inclusion of HDCP, the ADV7511 allows the secure transmission of protected content as specified by the HDCP 1.4 protocol.
The ADV7511 supports both S/PDIF and 8-channel I^2S audio. Its high fidelity 8-channel I^2S can transmit either stereo or 7.1 surround audio up to 768 kHz. The S/PDIF can carry compressed audio including Dolby® Digital, DTS®, and THX®. Fabricated in an advanced CMOS process, the ADV7511 is provided in a 100-lead LQFP surface-mount plastic package and is specified over the 0°C to +70°C temperature range.

The transmitter library is a collection of APIs that provide a consistent interface to ADV7511.
The library is a software layer that sits between the application and the TX hardware. The library is intended to serve two purposes:

- Provide the application with a set of APIs that can be used to configure HDMI TX hardware without the need for low-level register access. This makes the application portable across different revisions of the hardware and even across different hardware modules.
- Provide basic services to aid the application in controlling the TX module, such as interrupt service routine, HDCP high-level control and status information.

2. Functions description

The Transmitter library provides a comprehensive set of APIs to control, configure and provide status on all aspects of the HDMI TX module.

Below is the list of all functions implemented in this library:

- **ATV_ERR ADIAPI_TxInit (BOOL FullInit)**

Description

Power-up and initialize HDMI TX hardware and software module. This function will perform a complete TX module reset and brings the chip into a known state. The default behavior of the HDMI TX chip is to automatically power down if HPD is low. To change this default behavior, the API ADIAPI_TxOverrideHpdPD can be used.
The TMDS clock and data lines will be disabled following a call to this API (or whenever the system is initialized) unless the configuration flag TX_ENABLE_TMDS_ON_INIT is set as described in the ADIAPI_TxSetConfig API.

Parameters

FullInit

Select if it is required to perform full initialization (all h/w modules will be reset) or partial initialization (CEC module will not be affected).
Set to TRUE to perform full initialization. This should be used when doing a cold start.
Set to FALSE to perform partial initialization. This should be used for warm start (for example coming out of standby) to preserve the state of the CEC engine.

Return value

ATVERR_OK

Operation completed successfully.

ATVERR_FAILED

The chip is powered down.

- **ATV_ERR ADIAPI_TxShutdown (TX_PD_MODE PdMode)**

Description

Power down HDMI TX hardware. The chip power-down will be set to high and all TMDS lines will be disabled. This API can also be used to enter stand-by mode.

Parameters

PdMode

Select the power-down mode. Three power-down modes are available:

TX_PD_MODE1

Entire chip is powered down except HPD and Rx Sense interrupts and CEC engine.

TX_PD_MODE2

Everything is powered down except CEC engine.

TX_PD_MODE3

Everything is powered down.

Return value

ATVERR_OK

- **ATV_ERR ADIAPI_TxSetConfig (TX_CONFIG UserConfig, BOOL Set)**

Description

Configure how TX ISR responds to various events and set other generals operational parameters.

Parameters

TxConfig

Any of the TX_CONFIG values ORed together. Possible values are:

TX_INIT_ON_HPD_HIGH

This flag causes the TX ISR to perform h/w initialization when the HPD signal changes from low to high. The initialization is done by calling the ADIAPI_TxInit API.

TX_INIT_ON_HPD_LOW

This flag causes the TX ISR to perform h/w initialization when the HPD signal changes from high to low. The initialization is done by calling the ADIAPI_TxInit API.

TX_INIT_ON_EDID_ERROR

This flag causes the TX ISR to perform h/w initialization if an EDID segment is received that was not expected or requested. The initialization is done by calling the ADIAPI_TxInit API.

TX_HDCP_DISABLE_ON_ERROR

This flag causes the TX ISR to disable HDCP engine on any HDCP errors. Note that while this flag instructs the HDCP h/w to disable HDCP, the HDCP engine will not be actually disabled until it reaches authenticated state.

TX_ENABLE_TMDS_ON_INIT

This flag causes the any call to the ADIAPI_TxInit API to power-up TMDS clock and data lines. For more information, please refer to ADIAPI_TxInit API.

TX_ENABLE_DBG

This flag enables the debug messages from the TX module to be sent to the platform‟s console output channel. This flag only directs the debug messages to the output channel. To be able to see the messages on the console, the console must be enabled using the compilation switch “UART_DEBUG” as described in the ATV software architecture document.

Set

Set to TRUE to set the flags supplied in TxConfig Set to FALSE to reset the flags supplied in TxConfig

Return value

ATVERR_OK

- **ATV_ERR ADIAPI_TxIsr (void)**

Description

Process the TX device interrupts. This function should be called by the application as soon as a TX device interrupt is detected. If the application uses polling and the interrupt line from the HDMI TX to the MCU is not connected, this function can be called periodically to poll and process any outstanding interrupts. It should be noted that some of the TX interrupts can take relatively long time to process. For example, an EDID interrupt will consume at least the amount of time needed to read all 256 bytes of EDID via I2C. It is thus advisable for interrupt-driven real-time applications to disable the transmitter interrupt to the MCU before calling this function and re-enable it after this function returns.
The application will be notified on any change of the operating conditions if the notification events are enabled using the ADIAPI_TxSetEnabledEvents API.
The various interrupts, the action taken in the ISR and the associated notification event are listed in the table below. The details of the notification events can be found in the Notification events section.

Interrupt	Default Action	Notification Event
HPD Low	None	TX_EVENT_HPD_CHG
HPD High	Hardware reset is performed (ADIAPI_TxInit will be called)	TX_EVENT_HPD_CHG
Rx Sense Low/High	None	TX_EVENT_MSEN_CHG
EDID Ready	Read the next EDID segment up to TX_SUPPORTED_EDID_SEGMENT	TX_EVENT_EDID_READY when a new segment is fully read
BKSV ready	Read and concatenate downstream BKSV into internal buffer	TX_EVENT_BKSV_READY when ALL downstream BKSVs are received
HDCP Authenticated	None	TX_EVENT_HDCP_AUTHENTICATED
HDCP Error	None	TX_EVENT_HDCP_ERROR
Vsync Edge	None	TX_EVENT_VSYNC_EDGE
Audio FIFO Full	None	TX_EVENT_AUDIO_FIFO_FULL
Embedded Sync Polarity	None	TX_EVENT_EMB_SYNC_ERROR
CEC TX Ready	None	TX_EVENT_CEC_TX_READY
CEC RX Ready	None	TX_EVENT_CEC_RX_READY
CEC TX Retry timeout	None	TX_EVENT_CEC_ERR_TIMEOUT
CEC TX Arbitration Lost	None	TX_EVENT_CEC_ERR_ARB_LOST

Parameters

None

Return value

ATVERR_OK
: The ISR completed execution and no other interrupts are pending. All pending interrupts are cleared.

ATVERR_FAILED
: No pending interrupts detected.

- **ATV_ERR ADIAPI_TxIntPending (void)**

Description

Check if TX interrupt is pending.

Parameters

None

Return value

ATVERR_TRUE
: Pending interrupt detected.

ATVERR_FALSE
: No pending interrupt detected.

- **ATV_ERR ADIAPI_TxSetEnabledEvents (TX_EVENT Events, BOOL Enable)**

Description

This API enables or disables user notification on certain events as described in the Notification events section.

Parameters

Events
: The events that needs to be enabled or disabled ORed together. For a list of valid events, please refer to the Notification events section. Only the events supplied in this parameter will be affected. All other events‟ state (Enabled/Disabled) will remain unchanged.

The TX_EVENT enum also offers 3 additional values that are used as an event groups: This values are:

TX_EVENT_ALL_EVENTS

This value defines all supported events.

TX_EVENT_HDMI_EVENTS

This value groups all HDMI events. HDMI events are enabled by default and constitute the following events:

TX_EVENT_HPD_CHG
TX_EVENT_MSEN_CHG
TX_EVENT_EDID_READY
TX_EVENT_BKSV_READY
TX_EVENT_HDCP_AUTHENTICATED
TX_EVENT_HDCP_ERROR

TX_EVENT_CEC_EVENTS

This value defines all CEC events. CEC events are:

TX_EVENT_CEC_RX_MSG
TX_EVENT_CEC_TX_DONE
TX_EVENT_CEC_TX_TIMEOUT
TX_EVENT_CEC_TX_ARB_LOST
TX_EVENT_CEC_TX_LOG_ADDR_ALLOC

Enable

TRUE to enable notification on the supplied events.
FALSE to disable notification on the supplied events.

Return value

ATVERR_OK

- **ATV_ERR ADIAPI_TxGetChipRevision (UINT16 *TxRev)**

Description

Get HDMI TX chip revision.

Parameters

TxRev

Pointer to receive HDMI TX chip revision.

Return value

ATVERR_OK

- **ATV_ERR ADIAPI_TxOverrideHpdPd (BOOL Override)**

Description

Maintain the current power state regardless of the state of the sink HPD signal. By default, HDMI TX chip will automatically power-down if the sink‟s HPD signal changes state from HIGH to LOW. This API can be used to change this default behaviour so that the chip will remain powered-up regardless of the state of the sink HPD.

Parameters

Override

TRUE to disable automatic power-down when sink HPD goes low.
FALSE to enable automatic power-down when sink HPD goes low.

Return value

ATVERR_OK

- **ATV_ERR ADIAPI_TxEnableTmds (BOOL Enable, BOOL SoftOn)**

Description

Enable or disable TMDS output clock and data lines.

Parameters

Enable

TRUE to enable TMDS clock and data lines.
FALSE to disable TMDS clock and data lines.

SoftOn

TRUE to enable soft TMDS clock turn on. This avoids glitches in the TMDS clock when it is turned on.
FALSE to disable the soft turn-on feature

Return value

ATVERR_OK

- **ATV_ERR ADIAPI_TxSetInputPixelFormat (UCHAR BitsPerColor, TX_IN_FORMAT Format, UCHAR Style, TX_CHAN_ALIGN Align, BOOL RisingEdge, BOOL BitSwap)**

Description

Set HDMI TX input pixel data format. The HDMI TX can accept video data from 8 to 36 pins, with various configurations to accommodate 4:4:4 or 4:2:2 format, embedded or separate sync, single or double data rate, repeated pixels and different pin assignments to interface with video data sources. For most applications, the input pixel format needs to be set only once, unless the video source can change its output pixel format on the fly, in which case the HDMI TX input format must also be changed to match. For detailed information regarding HDMI TX input video pin assignments, please refer to the ADV7510 programming guide.

Parameters

BitsPerColor

Specify the number of bits per color component. This can be 8, 10 or 12.

Format

Video input format. This value indicates if the input video is SDR or DDR, with embedded or separate sync and if the input pixel clock is twice the pixel rate. This can be one of the following:

SDR_444_SEP_SYNC
SDR_422_SEP_SYNC
SDR_422_EMP_SYNC
SDR_422_SEP_SYNC_2X_CLK
SDR_422_EMB_SYNC_2X_CLK
DDR_444_SEP_SYNC
DDR_422_SEP_SYNC
DDR_422_EMB_SYNC

For more information about input pin assignment for each case, please refer to the ADV7510 programming guide.

Style

Three input styles are available: 1, 2 or 3. For more information about input pin assignment for each style, please refer to the ADV7510 programming guide.

Alignment

This value specifies the bit alignment of each channel in 4:2:2 modes. Three alignment types are available:

ALIGN_LEFT
ALIGN_RIGHT
ALIGN_EVEN

For more information about input pin assignment for each type, please refer to the ADV7510 programming guide.

RisingEdge

This value specifies the clocking edge for DDR modes.
Set to TRUE to select rising edge.
Set to FALSE to select falling edge.

Return value

ATVERR_OK

Indicate the function completed successfully.

ATVERR_INV_PARM

Indicate that one or more of the input parameters are invalid or the selected combination of the input parameters is invalid.

- **ATV_ERR ADIAPI_TxSetInputVideoClock (UCHAR ClkDivide)**

Description

Divide HDMI TX input video clock to generate the correct pixel clock. The input video clock to HDMI TX must be equal to the pixel clock. In cases where the input clock is NOT equal to the pixel clock, this function should be used to divide the input clock to generate the correct pixel clock. This function is designed to be used for compatibility with some older source devices.

Parameters

ClkDivide

Set to 1, 2 or 4 to divide HDMI TX input clock by 1, 2 or 4 to generate the pixel clock.

Return value

ATVERR_OK
ATVERR_INV_PARM

- **ATV_ERR ADIAPI_TxSetOutputPixelFormat (TX_OUT_ENCODING OutEnc, BOOL Interpolate)**

Description

Sets HDMI TX output pixel format. This API defines the up-conversion from 4:2:2 to 4:4:4 encoding or the down-conversion from 4:4:4 to 4:2:2 encoding along with the method to be used for up-conversion.

Parameters

OutFormat

This defines the required output pixel encoding format. This can be one of the following:

OUT_ENC_RGB_444
OUT_ENC_YUV_444
OUT_ENC_YUV_422

This value should match the Y1Y0 value sent in the AV info-frame.

Interpolate

This parameter is used only when up-converting the input from 4:2:2 to 4:4:4 encoding.
Set to TRUE to up-convert using linear interpolation.
Set to FALSE to up-convert using line duplication.

Return value

ATVERR_OK
ATVERR_INV_PARM

- **ATV_ERR ADIAPI_TxSetManualPixelRepeat (UCHAR Vic, UCHAR Factor, UCHAR PrValue)**

Description

Manually set HDMI TX output pixel repetition rate and parameters.

Parameters

Vic

This value defines the Video Identification Code that should be sent in the AV info-frame. See ADIAPI_TxSendAVInfoframe for more details.

Factor

This value defines the required multiplication factor of the input pixel clock. Possible values are 1, 2, 3 and 4.

PrValue

This value defined the PR (Pixel Repeat) value that should be sent in the AV info-frame. See ADIAPI_TxSendAVInfoframe for more details.

Return value

ATVERR_OK
ATVERR_INV_PARM

- **ATV_ERR ADIAPI_TxSetAutoPixelRepeat (UCHAR Mode, UCHAR Vic, TX_REFR_RATE RefRate, UCHAR AspectRatio)**

Description

Set HDMI TX to automatically calculate and output the correct pixel repetition rate based on detected video format and audio sampling frequency. The audio sampling frequency is either extracted from the stream or defined by the user. See ADIAPI_TxSetAudChStatSampFreq for more details.

Parameters

Mode

PR_NORMAL = Normal automatic pixel repetition. In this mode, HDMI TX will automatically calculate the required pixel repetition based on audio sampling rate and detected VIC. The resulting video identification code (VIC) and Pixel Repeat value (PR) will be automatically inserted in the AV info-frame. See ADIAPI_TxSendAVInfoframe for more details.

PR_MAX = Maximum automatic pixel repetition. This mode is similar to normal automatic mode, except the required pixel repetition will always be set to the highest possible value the HDMI TX is capable of. This makes video timing independent of audio timing.

InVic

This value defines the video identification code (VIC) of the input video format. If the VIC of the input video is not known, this value must be set to 0xff and the „RefreshRate" and „Aspect" parameters must be used.

RefreshRate

This value defines the refresh rate for video modes with low refresh rate or with 2x or 4x the normal refresh rate, when the „Vic" parameter is undefined (set to 0xff) Possible values are:

REFRESH_NORMAL
REFRESH_LOW
REFRESH_2X
REFRESH_4X

AspectRatio

This value defines the aspect ratio of the input video, when the „Vic" parameter is undefined (set to 0xff). Possible values are:

4*3 (=12 for 4x3 aspect)
16*9 (=144 for 16x9 aspect)

Return value

ATVERR_OK
ATVERR_INV_PARM

- **ATV_ERR ADIAPI_TxSetOutputColorDepth (UCHAR Depth, TX_DC_METHOD DcMethod)**

Description

Set output colour depth and the method used to handle deep colour down-conversion. When the input colour depth to the HDMI TX is less than the colour depth of the output, the remaining least-significant bits of the output will be filled with 0s. When the input colour depth is larger than the output, truncation or active dithering can be used to reduce the colour depth.

Parameters

Depth

Required colour depth of the output. This value can be 24, 30 or 36 and will correctly set the general control packet colour depth field. Any other value will be written un-modified to the General Control Packet CD (Colour Depth) field.

DcMethod

This value specifies the down-conversion method that will be used if the input colour depth is larger than the output colour depth. Possible values are:

TX_DC_TRUNCATE
TX_DC_ACTIVE_DITHER

Return value

ATVERR_OK
ATVERR_INV_PARM

- **ATV_ERR ADIAPI_TxSetCSC (TX_CS_MODE InColorSpace, TX_CS_MODE OutColorSpace)**

Description

Set colour space conversion for HDMI TX chip.

Parameters

InColorSpace

Colour space input to TX device. This can be one of the following values:

TX_CS_MODE	Meaning	Range
TX_CS_RGB	RGB	0-255
TX_CS_YUV_601	YCrCb 601 (SDTV)	16-235
TX_CS_YUV_709	YCrCb 709 (HDTV)	16-235
TX_CS_YCC_601	xvYCC 601 (Extended gamut SDTV)	0-255
TX_CS_YCC_709	xvYCC 709 (Extended gamut HDTV)	0-255
TX_CS_AUTO	Disable CSC (out CS = In CS)	

The TX_CS_AUTO setting will disable Colour Space Conversion.

OutColorSpace

Colour space output from TX device. This can be one of the following values:

TX_CS_MODE	Meaning	Range
TX_CS_RGB	RGB	0-255
TX_CS_YUV_601	YCrCb 601 (SDTV)	16-235
TX_CS_YUV_709	YCrCb 709 (HDTV)	16-235
TX_CS_YCC_601	xvYCC 601 (Extended gamut SDTV)	0-255
TX_CS_YCC_709	xvYCC 709 (Extended gamut HDTV)	0-255
TX_CS_AUTO	Disable CSC (out CS = In CS)	

The TX_CS_AUTO setting will disable Colour Space Conversion.

Return value

ATVERR_OK
ATVERR_INV_PARM

- **ATV_ERR ADIAPI_TxSetAudioInterface (TX_AUD_FORMAT InputFormat, TX_AUD_PKT_TYPE OutType, UCHAR HbrStrmCount)**

Description

Set the input audio interface and output audio packet type. The TX device has 3 physical audio interfaces:

4 I2S inputs (8 channels)
SPDIF
DSD

The I2S interface can accept data in the following formats:

Standard I2S
Right justified I2S
Left justified I2S
AES3 (IEC 60958-3)

Two different output packet formats can be selected when the input is I2S: Audio sample packet or HBR packet.

When the input interface is I2S, for all formats except AES3, the channel status data to be sent to the receiver (in the ASP/HBR packet) must be explicitly set in the TX registers, since I2S contains pure audio samples. For AES3 format, the TX device can extract the channel status from the data stream or can use user-defined values from registers.

The mapping between I2S input and the data sent in the audio sample packet is configurable (For example, I2S3 input left channel can be sent in the audio sample packet sub-frame 0 instead of the default sub-frame 6) See ADIAPI_TxSetAudChanMapping for details.

The SPDIF interface can accept 2 channel L-PCM audio or AES3 (IEC 60958-3) audio at sampling rates of up to 192 KHz. The sampling frequency extracted from the stream will be sent in the Audio Sample Packet channel status bits. The sampling frequency used for pixel repeat can be either the one extracted from the stream or a user-defined value.
As with I2S, the TX device can output either Audio Sample Packet or Hight Bit Rate packet when the input is SPDIF.
The DSD interface can be used to input DSD or DST audio. The output audio packets will be either one-bit audio for DSD or DST audio packet for DST.

Parameters

InputFormat

This value defines the audio interface and format to be used for inputting audio to the HDMI TX. This can be one of the values defined in the table below. Note that some input formats cannot be used with certain output packet types.

TX_AUD_FORMAT	Input Interface	Input Format	Valid Output Packet Type (TX_AUD_PKT_TYPE)
TX_I2S_STD	I2S	Standard I2S	AUD_SAMP_PKT HBR_STRM_PKT
TX_I2S_RJUST	I2S	Right justified I2S	AUD_SAMP_PKT HBR_STRM_PKT
TX_I2S_LJUST	I2S	Left justified I2S	AUD_SAMP_PKT HBR_STRM_PKT
TX_I2S_AES3	I2S	AES3 direct	AUD_SAMP_PKT HBR_STRM_PKT
TX_I2S_SPDIF	I2S	IEC61937 Bi-phase Mark	HBR_STRM_PKT
TX_SPDIF	SPDIF	IEC61937 Bi-phase Mark	AUD_SAMP_PKT
TX_DSD_NORM	DSD	DSD normal	ONE_BIT_ASP
TX_DSD_SDIF3	DSD	SDIF-3	ONE_BIT_ASP
TX_DSD_DST	DSD	DST normal	DST_AUD_PKT
TX_DSD_DST_SDR	DSD	DST 2X	DST_AUD_PKT
TX_DSD_DST_DDR	DSD	DSD 1X (DDR)	DST_AUD_PKT

OutType

This value defines the audio type (packet type) that will be output by the HDMI TX as illustrated in the table below. Note that some packet types can be used only with certain input formats.

Output Packet Type (TX_AUD_PKT_TYPE)	Definition	Valid Input Formats
AUD_SAMP_PKT	Audio Sample Packet	I2S_STD, I2S_RJUST, I2S_LJUST, I2S_AES3
HBR_STRM_PKT	High Bit Rate Audio Stream Packet	I2S_STD, I2S_RJUST, I2S_LJUST, I2S_AES3, I2S_SPDIF
ONE_BIT_ASP	One Bit Audio Sample Packet	DSD_NORM, DSD_SDIF3
DST_AUD_PKT	DST Audio Packet	DSD_DST, DSD_DST_SDR, DSD_DST_DDR

HbrStrmCount

This parameter is used only when the output audio packet type is HBR_STRM_PKT. It specifies the number of HBR streams encoding. This value can only be 1 or 4.

Return value

ATVERR_OK

Function completed successfully.

ATVERR_INV_PARM

Invalid input parameter value.

- **ATV_ERR ADIAPI_TxSetAudChanMapping (TX_AUD_CHAN InChan, TX_AUD_CHAN OutSample)**

Description

Set the mapping between I2S input channels and the output audio samples. The default setting is one-to-one according to the following table. The default setting can be changed using this API.

Input Channel	Output Sample
I2S0 Left Channel	Audio Sample 0 Left Channel
I2S0 Right Channel	Audio Sample 0 Right Channel
I2S1 Left Channel	Audio Sample 1 Left Channel
I2S1 Right Channel	Audio Sample 1 Right Channel
I2S2 Left Channel	Audio Sample 2 Left Channel
I2S2 Right Channel	Audio Sample 2 Right Channel
I2S3 Left Channel	Audio Sample 3 Left Channel
I2S3 Right Channel	Audio Sample 3 Right Channel

Parameters

InChan

Input channel ID. This can be one of the following:

CH0_LEFT
CH0_RIGHT
CH1_LEFT
CH1_RIGHT
CH2_LEFT
CH2_RIGHT
CH3_LEFT
CH3_RIGHT

OutSample

Output sample position. This can be one of the following:

CH0_LEFT
CH0_RIGHT
CH1_LEFT
CH1_RIGHT
CH2_LEFT
CH2_RIGHT
CH3_LEFT
CH3_RIGHT

Return value

ATVERR_OK

Function completed successfully.

ATVERR_INV_PARM

Invalid input parameter value.

- **ATV_ERR ADIAPI_TxSetAudNValue (UINT32 NValue)**

Description

Set „N" value that will be used by the HDMI TX to calculate the audio sampling frequency. CTS (Cycle Time Stamp) can be set using the API ADIAPI_TxSetAudCTS.
HDMI TX uses both N and CTS to calculate the audio sampling frequency according to the formula:

$$128\ fS = fTMDS\_CLK\ N / CTS$$

Parameters

NValue

Specify the 20-bit „N" value that HDMI TX will use to calculate the audio sampling frequency.
If this value is set to 0, the N value will be calculated using the sampling frequency obtained from the current audio info-frame/audio channel status. If new audio info-frame or channel status is received, this function must be called to update the N value.

Return value

ATVERR_OK

Function completed successfully.

ATVERR_INV_PARM

Invalid N value.

- **ATV_ERR ADIAPI_TxSetAudCTS (UINT32 CTS)**

Description

Set CTS (Cycle Time Stamp) value that will be used by the HDMI TX to calculate the audio sampling frequency. The „N" value can be set using the API ADIAPI_TxSetAudNValue.
HDMI TX uses both N and CTS to calculate the audio sampling frequency according to the formula:

128 fS = fTMDS_CLK N / CTS

Parameters

CTS

Specify the 20-bit „CTS" value that HDMI TX will use to calculate the audio sampling frequency. If this value is set to 0, the CTS value will be automatically calculated by the chip using the SCLK.

Return value

ATVERR_OK

Function completed successfully.

ATVERR_INV_PARM

Invalid CTS value.

- **ATV_ERR ADIAPI_TxSetAudMCLK (TX_MCLK_FREQ MClk)**

Description

Set HDMI TX input audio master clock (MCLK) frequency. The MCLK can be externally supplied or internally generated using SCLK.

Parameters

MClk

Define the HDMI TX audio master clock frequency. This can be one of the following values:

TX_MCLK_128FS

Set MCLK = 128 * Sampling frequency.

TX_MCLK_256FS

Set MCLK = 256 * Sampling frequency.

TX_MCLK_384FS

Set MCLK = 384 * Sampling frequency.

TX_MCLK_512FS

Set MCLK = 512 * Sampling frequency.

TX_MCLK_HBR
Set MCLK for High Bit Rate audio.
TX_MCLK_AUTO
Generate MCLK internally using SCLK.

Return value

ATVERR_OK
Function completed successfully.
ATVERR_INV_PARM
Invalid MCLK value.

- **ATV_ERR ADIAPI_TxSetAudClkPolarity (BOOL RisingEdge)**

Description

Set the input clock polarity for MCLK, SCLK and DSD clock.

Parameters

RisingEdge
Clock polarity for MCLK, (if externally supplied) SCLK and DSD clock.
Set to TRUE to latch input data on rising edge.
Set to FALSE to latch input data on falling edge.

Return value

ATVERR_OK

- **ATV_ERR ADIAPI_TxSetAudChStatSampFreq (TX_AUD_FS SampFreq)**

Description

Set the sampling frequency to be sent in the Audio Sample Packet‟s channel status bits. The source of the sampling frequency (Extracted from input stream or defined by user) can be set independently from the rest of the channel status fields defined in the ADIAPI_Tx SetAudChanStatus API.
The TX device can use the sampling frequency extracted from the input stream or the sampling frequency defined by user (in the channel status bits) for automatic pixel repeat calculation. See ADIAPI_TxSetAutoPixelRepeat for more details.

The source of channel status bits and/or sampling frequency sent in the sample packet can be user-defined, extracted from input stream, or both, depending on the input audio format as described in the following table:

Input Audio	Channel Status Source	Sampling Frequency Source	Sampling frequency source for Pixel Repeat
I2S	User defined	User defined	User defined
I2S – AES3	User defined / From stream	User defined / From stream	User defined / From Stream
SPDIF	From stream	From Stream	User defined / From Stream

This API only defines the source and value of the channel status sampling frequency. Incorrect settings of the sampling frequency source will be ignored by the TX device (e.g., if input audio is I2S and the user selects the sampling frequency source to be from stream, the TX device will ignore the setting and use the latest programmed sampling frequency. The default sampling frequency is 44.1 KHz).

Parameters

SampFreq

Specify the channel status sampling frequency that will be used for pixel repeat calculation and will be sent in the audio sample packet. This can be one of the following:

TX_FS_32KHZ
32 KHz.

TX_FS_44KHZ
44.1 KHz.

TX_FS_48KHZ
48 KHz.

TX_FS_88KHZ
88.2 KHz.

TX_FS_96KHZ
96 KHz.

TX_FS_176KHZ
176.4 KHz.

TX_FS_192KHZ
192 KHz.

TX_FS_HBR
Setting for HBR audio (768 KHz).

TX_FS_FROM_STRM
Use sampling freq extracted from audio stream.

Note that for HBR (High Bit Rate) audio, the sampling frequency must be set to TX_FS_HBR. This is done implicitly by the ADIAPI_TxSetAudioInterface API. Setting the sampling frequency to TX_FS_FROM_STRM will only change the source of the sampling frequency; the sampling frequency value programmed into the chip will not change.

Return value

ATVERR_OK
Function completed successfully.
ATVERR_INV_PARM
Invalid SampFreq value.

- **ATV_ERR ADIAPI_TxSetAudChanStatus (BOOL FromStream, TX_CHAN_STATUS *ChanStat)**

Description

Set the sampling frequency to be sent in the Audio Sample Packet‟s channel status bits. The source of the sampling frequency (Extracted from input stream or defined by user) can be set independently from the rest of the channel status fields defined in the ADIAPI_Tx SetAudChanStatus API.
The TX device can use the sampling frequency extracted from the input stream or the sampling frequency defined by user (in the channel status bits) for automatic pixel repeat calculation. See ADIAPI_TxSetAutoPixelRepeat for more details.
The source of channel status bits and/or sampling frequency sent in the sample packet can be user-defined, extracted from input stream, or both, depending on the input audio format as described in the following table:

Input Audio	Channel Status Source	Sampling Frequency Source	Sampling frequency source for Pixel Repeat
I2S	User defined	User defined	User defined
I2S – AES3	User defined / From stream	User defined / From stream	User defined / From Stream
SPDIF	From stream	From Stream	User defined / From Stream

This API only defines the source and value of the channel status sampling frequency. Incorrect settings of the sampling frequency source will be ignored by the TX device (e.g., if input audio is I2S and the user selects the sampling frequency source to be from stream, the TX device will ignore the setting and use the latest programmed sampling frequency. The default sampling frequency is 44.1 KHz).

Parameters

SampFreq
Specify the channel status sampling frequency that will be used for pixel repeat calculation and will be sent in the audio sample packet. This can be one of the following:
TX_FS_32KHZ
32 KHz.
TX_FS_44KHZ
44.1 KHz.
TX_FS_48KHZ
48 KHz.

TX_FS_88KHZ
88.2 KHz.
TX_FS_96KHZ
96 KHz.
TX_FS_176KHZ
176.4 KHz.
TX_FS_192KHZ
192 KHz.
TX_FS_HBR
Setting for HBR audio (768 KHz).
TX_FS_FROM_STRM
Use sampling freq extracted from audio stream.

Note that for HBR (High Bit Rate) audio, the sampling frequency must be set to TX_FS_HBR. This is done implicitly by the ADIAPI_TxSetAudioInterface API. Setting the sampling frequency to TX_FS_FROM_STRM will only change the source of the sampling frequency; the sampling frequency value programmed into the chip will not change.

Return value

ATVERR_OK
Function completed successfully.
ATVERR_INV_PARM
Invalid SampFreq value.

- **ATV_ERR ADIAPI_TxAudInputEnable (TX_AUD_INTERFACE Interface, BOOL Enable)**

Description

Enable/Disable audio input signal to HDMI TX.

Parameters

Interface
Specify the audio interface to enable/disable. This can be one of the following values:
TX_AUD_IN_I2S0
I2S channel 0.
TX_AUD_IN_I2S1
I2S channel 1.
TX_AUD_IN_I2S2
I2S channel 2.
TX_AUD_IN_I2S3
I2S channel 3.

TX_AUD_IN_I2S
All I2S channels (0-3).
TX_AUD_IN_SPDIF
SPDIF.
TX_AUD_IN_DSD0
DSD channel 0.
TX_AUD_IN_DSD1
DSD channel 1.
TX_AUD_IN_DSD2
DSD channel 2.
TX_AUD_IN_DSD3
DSD channel 3.
TX_AUD_IN_DSD4
DSD channel 4.
TX_AUD_IN_DSD5
DSD channel 5.
TX_AUD_IN_DSD6
DSD channel 6.
TX_AUD_IN_DSD7
DSD channel 7.
TX_AUD_IN_DSD
All DSD channels (0-7).
TX_AUD_IN_ALL
All audio inputs.

Enable
Set to TRUE to enable the audio interface specified in the “Interface” parameter.
Set to FALSE to disable the audio interface specified in the “Interface” parameter.

Return value

ATVERR_OK
Function completed successfully.
ATVERR_INV_PARM
Invalid Interface value.

- **ATV_ERR ADIAPI_TxSetI2sInput(UCHAR ChanCount, UCHAR ChanAlloc, TX_AUD_PKT_TYPE AudType)**

Description

Enable I2S audio input 0-3 based on audio info-frame channel allocation and if the input stream is HBR.

Parameters

ChanCount

Number of audio channels (0-7).

ChanAlloc

Channel allocation field from Audio InfoFrame.

AudType

Input audio packet type as defined in the table below.

Audio Packet Type Packet Type (TX_AUD_PKT_TYPE)	Definition
AUD_SAMP_PKT	Audio Sample Packet
HBR_STRM_PKT	High Bit Rate Audio Stream Packet
ONE_BIT_ASP	One Bit Audio Sample Packet
DST_AUD_PKT	DST Audio Packet

Return value

ATVERR_OK

Function completed successfully.

- **ATV_ERR ADIAPI_TxSetOutputMode (TX_OUTPUT_MODE OutMode)**

Description

Set output video mode of HDMI TX to HDMI or DVI.

Parameters

OutMode

Required HDMI TX output mode.

Can be TX_OUT_MODE_HDMI or TX_OUT_MODE_DVI

Return value

ATVERR_OK

Function completed successfully.

ATVERR_INV_PARM

Invalid OutMode value.

- **ATV_ERR ADIAPI_TxHdcpEnable (BOOL EncEnable, BOOL FrameEncEnable)**

Description

Enable or disable HDCP on HDMI TX output.

Parameters

EncEnable

TRUE to enable HDCP.

FALSE to disable HDCP.

FrameEncEnable

TRUE to enable encryption of the current frame.

FALSE to disable encryption of the current frame while maintaining HDCP synchronization.

Return value

ATVERR_OK

- **ATV_ERR ADIAPI_TxGetBksvList (UCHAR *BksvList, UCHAR *NumOfBksvs)**

Description

Read BKSVs list from HDMI TX once all BKSVs are read.

Parameters

BksvList

Pointer to a buffer to receive the downstream BKSV list as read by the HDMI TX. This list is available only after HDMI TX successfully read all downstream BKSVs. This list will not be available if HDCP is disabled or if any HDCP errors are detected. The size of the buffer must be large enough to accommodate the number of BKSVs specified in the TX_SUPPORTED_DS_DEVICE_COUNT configuration parameter (i.e., Minimum buffer size will be TX_SUPPORTED_DS_DEVICE_COUNT * 5).

This parameter can be set to NULL to only return the number of available BKSVs in the BksvCount parameter.

BksvCount

This is a pointer to receive the number of BKSVs reported by the downstream device. This number will also include the downstream repeater BKSV if the downstream device is a repeater. This value normally specify the number of BKSVs returned in the BksvList buffer, unless the BKSV count reported by the downstream device exceeds

TX_SUPPORTED_DS_DEVICE_COUNT, in which case the BksvList buffer will hold the first TX_SUPPORTED_DS_DEVICE_COUNT BKSVs.

Return value

ATVERR_OK
Function completed successfully.
ATVERR_FAILED
HDCP is disabled or authentication is not complete or HDCP errors encountered.

- **ATV_ERR ADIAPI_TxGetBstatus (UINT16 *Bstatus, UCHAR *Bcaps)**

Description

Read downstream device Bstatus and Bcaps registers. Bstatus and Bcaps are available only if the state of the HDCP engine is HDCP_BSTATUS_READY, HDCP_BKSV_LIST_READY or HDCP_AUTHENTICATED. The HDCP state can be obtained by calling the ADIAPI_TxGetHdcpState API.

Parameters

Bstatus Bcaps
Pointer to receive the downstream device HDCP Bstatus register. Pointer to receive the downstream device HDCP BCAPS register.

Return value

ATVERR_OK
Function completed successfully.
ATVERR_FAILED
HDCP is disabled or downstream device is not available.

- **ATV_ERR ADIAPI_TxGetHdcpState (TX_HDCP_STATE *HdcpState)**

Description

Read the current status of HDCP engine.

Parameters

HdcpState
This is a pointer to receive the current status of HDCP engine. Possible states are:
TX_HDCP_NO_DS_DEVICE
TX_HDCP_DISABLED

TX_HDCP_BSTATUS_READY
TX_HDCP_BKSV_LIST_READY
TX_HDCP_AUTHENTICATED

Return value

ATVERR_OK
Function completed successfully.

- **ATV_ERR ADIAPI_TxGetLastHdcpError (TX_HDCP_ERR *Error)**

Description

Return the last error status of HDCP engine. This API returns the last encountered HDCP error(s) since the previous read using this API. All returned error bits will be cleared following a call to this API.

Parameters

Status

This is a pointer to receive HDCP errors that occurred since the last call to this API. Returned errors are OR-ed together. Possible error bits are:
TX_HDCP_ERR_BAD_RECV_BKSV
TX_HDCP_ERR_RI_MISMATCH
TX_HDCP_ERR_PJ_MISMATCH
TX_HDCP_ERR_I2C_ERROR
TX_HDCP_ERR_REP_DONE_TIMEOUT
TX_HDCP_ERR_MAX_CASCADE_EXCEEDED
TX_HDCP_ERR_V_DASH_CHECK_FAILED
TX_HDCP_ERR_MAX_DEVICE_EXCEEDED
All error bits will be cleared upon calling this function. HDMI TX automatically re-starts the authentication process on any HDCP error.

Return value

ATVERR_OK

- **ATV_ERR ADIAPI_TxGetEdidSegment (UCHAR SegNum, UCHAR *SegBuf)**

Description

Read a 256-byte EDID segment received by HDMI TX.

Parameters

SegNum

EDID segment number to read (Starting from 0).

SegBuf

This is a pointer to a 256-byte buffer to receive the requested EDID segment. This buffer will contain valid data only if the return value is ATVERR_OK.

Return value

ATVERR_OK

EdidBuf will contain the requested EDID segment.

ATVERR_FAILED

Requested EDID segment is not available.

- **ATV_ERR ADIAPI_TxGetHpdMsenState (BOOL *Hpd, BOOL *Msen)**

Description

Return the Hot Plug Detect and Monitor Sense state of HDMI TX.

Parameters

Hpd

This is a pointer to receive the sink device Hot Plug Detect state. This parameter can be set to NULL if the HPD state is not required. If not NULL, on return, it will be set to TRUE if HPD is high and FALSE if HPD is low.

Msen

This is a pointer to receive the sink device monitor sense state. This parameter can be set to NULL if the monitor sense state is not required. If not NULL, on return, it will be TRUE if monitor sense is high and FALSE if monitor sense is low.

Return value

ATVERR_OK

- **ATV_ERR ADIAPI_TxGetEdidControllerState (TX_EDID_CTRL_STATE *State)**

Description

Get EDID/HDCP controller state.

Parameters

State

This is a pointer to the current status of EDID/HDCP engine. Possible states are:

TX_HDCP_NO_DS_DEVICE
TX_HDCP_DISABLED
TX_HDCP_BSTATUS_READY
TX_HDCP_BKSV_LIST_READY
TX_HDCP_AUTHENTICATED

Return value

ATVERR_OK

Function completed successfully.

- **ATV_ERR ADIAPI_TxOutputModeHdmi (BOOL *IsHdmi)**

Description

Get output mode: HDMI or DVI.

Parameters

IsHdmi

This is a pointer to the output mode. TRUE if the output mode is HDMI FALSE if the output mode is DVI.

Return value

ATVERR_TRUE

Output mode is HDMI.

ATVERR_FALSE

Output mode is DVI.

- **ATV_ERR ADIAPI_TxHdcpEnabled (BOOL *HdcpOn)**

Description

Determine if HDCP is currently enabled.

Parameters

HdcpOn

Pointer to the current state of HDCP.
Function will set this to TRUE when called if HDCP is enabled.
Function will set this to FALSE when called if HDCP is disabled.

Return value

ATVERR_TRUE

If HDCP is enabled.

ATVERR_FALSE

If HDCP is disabled.

- **ATV_ERR ADIAPI_TxOutputEncrypted (BOOL *Encrypted)**

Description

Check if the output is encrypted.

Parameters

Encrypted

This is a pointer to encryption state. TRUE if the output is encrypted FALSE if the output is not encrypted.

Return value

ATVERR_TRUE

Output is encrypted.

ATVERR_FALSE

Output is not encrypted.

- **ATV_ERR ADIAPI_TxPllLocked (BOOL *Locked)**

Description

Check if the PLL is locked.

Parameters

Locked

This is a pointer to PLL lock state.
TRUE if the PLL is locked.
FALSE if the PLL is not locked.

Return value

ATVERR_TRUE
PLL is locked.
ATVERR_FALSE
PLL is not locked.

- **ATV_ERR ADIAPI_TxGetStatus (TX_STATUS *TxStat)**

Description

This API provides the status of HDMI TX module. It can be used by the application to get some information regarding HDMI TX current state.

Parameters

TxStat
This is a pointer to TX_STATUS structure to receive HDMI TX status information. The members of this structure are described below:
ChipPd
Will be set to TRUE if HDMI TX chip is powered down.
Will be set to FALSE if HDMI TX chip is in normal operation.
TmdsPd
Will be set to TRUE if any of the TMDS lines (Ch0, Ch1, Ch2 or Clk) are powered down.
Will be set to FALSE if all of the TMDS lines (Ch0, Ch1, Ch2 or Clk) are powered up.
Hpd
Will be set to TRUE if sink Hot Plug Detect is high.
Will be set to FALSE if sink Hot Plug Detect is low.
MonSen
Will be set to TRUE if sink monitor sense is high.
Will be set to FALSE if sink monitor sense is low.

OutputHdmi
Will be set to TRUE if the output is HDMI.
Will be set to FALSE if the output is DVI.
PllLocked
Will be set to TRUE if video PLL is locked.
Will be set to FALSE if video PLL is not locked.
VideoMuted
Will be set to TRUE if video output is blacked out.
Will be set to FALSE if video output is not muted.
ClearAVMute
Will be set to TRUE if Clear AV mute is being sent to sink.
Will be set to FALSE if Clear AV mute is not being sent.
SetAVMute
Will be set to TRUE if set AV mute is being sent to sink.
Will be set to FALSE if set AV mute is not being sent.
AudioRep
Will be set to TRUE if audio output is enabled.
Will be set to FALSE if audio output is disabled (muted).
SpdifEnable
Will be set to TRUE if SPDIF interface is enabled.
Will be set to FALSE if SPDIF interface is disabled.
I2SEnable
Bits 0-3 will be set according to I2S channel 0-3 enable state.
If a channel is enabled, the corresponding bit will be set to 1, otherwise it will be 0.
DetectedVic
Video Identification code detected by HDMI TX
HdcpErr
Will be set to the last received HDCP error.

Return value

ATVERR_OK

- **ATV_ERR ADIAPI_TxMuteAudio (BOOL Mute)**

Description

This API can be used to Mute or un-mute HDMI TX audio output. The audio will be muted by disabling audio sample packets output from HDMI TX.

Parameters

Mute

TRUE to mute HDMI TX audio output.
FALSE to un-mute HDMI TX audio output.

Return value

ATVERR_OK

- **ATV_ERR ADIAPI_TxMuteVideo (BOOL Mute)**

Description

This API can be used to Mute or un-mute HDMI TX video output. The video will be muted by sending black level on all TMDS lines.

Parameters

Mute

TRUE to mute HDMI TX video output.
FALSE to un-mute HDMI TX video output.

Return value

ATVERR_OK

- **ATV_ERR ADIAPI_TxSetAvmute (TX_AVMUTE State)**

Description

Sets or clear general control packet AVMUTE signal.

Parameters

State

The required state of Set AVMUTE and Clear AVMUTE signals in the general control packet. This can be one of the following values:

TX_AVMUTE_ON

Set SET_AVMUTE and clear CLEAR_AVMUTE.

TX_AVMUTE_OFF

Clear SET_AVMUTE and set CLEAR_AVMUTE.

TX_AVMUTE_NONE
Clear both SET_AVMUTE and CLEAR_AVMUTE.
TX_AVMUTE_BOTH
Set both SET_AVMUTE and CLEAR_AVMUTE. Please note that this setting is not allowed by HDMI.

Return value

ATVERR_OK

- **ATV_ERR ADIAPI_TxGetAvmute (TX_AVMUTE *State)**

Description

Get Set/Clear AVMUTE state.

Parameters

State
Pointer to AVMUTE state. This can be one of the following values:
TX_AVMUTE_ON
Set SET_AVMUTE and clear CLEAR_AVMUTE.
TX_AVMUTE_OFF
Clear SET_AVMUTE and set CLEAR_AVMUTE.
TX_AVMUTE_NONE
Clear both SET_AVMUTE and CLEAR_AVMUTE.
TX_AVMUTE_BOTH
Set both SET_AVMUTE and CLEAR_AVMUTE. Please note that this setting is not allowed by HDMI.

Return value

ATVERR_OK

- **ATV_ERR ADIAPI_TxEnablePackets (UINT16 Packets, BOOL Enable)**

Description

Enable or disable HDMI TX sending of selected packets and info frames. Once a packet send is enabled, HDMI TX will continue to send the packet periodically on intervals as specified in HDMI specification.

Parameters

Packets

Packets that needs to enabled or disables ORed together. Possible values are:

PKT_AV_INFO_FRAME
PKT_AUDIO_INFO_FRAME
PKT_GC_PACKET
PKT_ACP_PACKET
PKT_SPD_PACKET
PKT_GMD_PACKET
PKT_ISRC1_PACKET
PKT_ISRC2_PACKET
PKT_MPEG_PACKET
PKT_VS_PACKET
PKT_ACR_PACKET
PKT_AUDIO_CHANNEL_STATUS
PKT_AUDIO_SAMPLE_PACKET
PKT_ALL_PACKETS

Enable

Set to TRUE to enable sending of packets specified in the “Packets” parameter.
Set to FALSE to disable sending of packets specified in the “Packets” parameter.

Return value

ATVERR_OK

- **ATV_ERR ADIAPI_TxGetEnabledPackets (UINT16 *Packets)**

Description

Get information about which packets are currently enabled.

Parameters

Packets

Pointer to receive information about which packets are currently enabled. Enabled packets are returned as bit values ORed together. Possible values are:

PKT_AV_INFO_FRAME
PKT_AUDIO_INFO_FRAME
PKT_GC_PACKET
PKT_ACP_PACKET
PKT_SPD_PACKET
PKT_GMD_PACKET

PKT_ISRC_PACKET
PKT_ACR_PACKET
PKT_AUDIO_CHANNEL_STATUS
PKT_AUDIO_SAMPLE_PACKET

Return value

ATVERR_OK

- **ATV_ERR ADIAPI_TxSendAVInfoFrame (UCHAR *Packet, UCHAR Size)**

Description

Send AV info-frame to the sink device. The AV info-frame packet repeat must be enabled using ADIAPI_TxEnablePackets to be able to send this packet.

Parameters

Packet

Pointer to the AV info frame HB0 (Header Byte 0) This AV info-frame will be sent as-is to the sink device, except the VIC and PR (pixel repeat) fields. The VIC and PR fields sent to the sink will depend on the pixel repeat mode setting using the APIs ADIAPI_TxSetManualPixelRepeat and ADIAPI_TxSetAutoPixelRepeat.

Size

Byte size of the AV info frame (Must be 16).

Return value

ATVERR_OK
ATVERR_INV_PARM

- **ATV_ERR ADIAPI_TxSendAudioInfoFrame (UCHAR *Packet, UCHAR Size)**

Description

Send Audio info-frame to the sink device. The Audio info-frame packet repeat must be enabled using ADIAPI_TxEnablePackets to be able to send this packet.
This function will automatically set the size of the right-justified I2S word size if the supplied audio info-frame sample size is not 0.
For HBR audio, the CA and CC fields of the audio info-frame must be set to 0x1F and 0x07 respectively, unless the ADIAPI_TxSetAudioInterface API is called afterward to adjust those two fields.

Parameters

Packet Size

Pointer to the audio info-frame packet HB0 (Header Byte 0) Byte size of the packet (Must be 13).

Return value

ATVERR_OK
ATVERR_INV_PARM

- **ATV_ERR ADIAPI_TxSendACPPacket (UCHAR *Packet, UCHAR Size)**

Description

Send ACP packet to the sink device. The ACP packet repeat must be enabled using ADIAPI_TxEnablePackets to be able to send this packet.

Parameters

Packet

Pointer to the packet HB0 (Header Byte 0).

Size

Byte size of the packet.

Return value

ATVERR_OK
ATVERR_INV_PARM

- **ATV_ERR ADIAPI_TxSendSPDPacket (UCHAR *Packet, UCHAR Size)**

Description

Send SPD packet to the sink device. The SPD packet repeat must be enabled using ADIAPI_TxEnablePackets to be able to send this packet.

Parameters

Packet

Pointer to the packet HB0 (Header Byte 0).

Size

Byte size of the packet.

Return value

ATVERR_OK
ATVERR_INV_PARM

- **ATV_ERR ADIAPI_TxSendISRC1Packet (UCHAR *Packet, UCHAR Size)**

Description

Send ISRC1 packet to the sink device. The ISRC1 packet repeat must be enabled using ADIAPI_TxEnablePackets to be able to send this packet.

Parameters

Packet
Pointer to the 31-byte ISRC1 packet HB0 (Header Byte 0).
Size
Byte size of the packet.

Return value

ATVERR_OK
ATVERR_INV_PARM

- **ATV_ERR ADIAPI_TxSendISRC2Packet (UCHAR *Packet, UCHAR Size)**

Description

Send ISRC2 packet to the sink device. The ISRC2 packet repeat must be enabled using ADIAPI_TxEnablePackets to be able to send this packet.

Parameters

Packet
Pointer to the 31-byte ISRC2 packet HB0 (Header Byte 0).
Size
Byte size of the packet.

Return value

ATVERR_OK
ATVERR_INV_PARM

- **ATV_ERR ADIAPI_TxSendGMDPacket (UCHAR *Packet, UCHAR Size)**

Description

Send Gamut Metadata packet to the sink device. The GMD packet repeat must be enabled using ADIAPI_TxEnablePackets to be able to send this packet.

Parameters

Packet
: Pointer to the packet HB0 (Header Byte 0).

Size
: Byte size of the packet.

Return value

ATVERR_OK
ATVERR_INV_PARM

- **ATV_ERR ADIAPI_TxSendMpegPacket (UCHAR *Packet, UCHAR Size)**

Description

Send MPEG packet to the sink device. The MPEG packet repeat must be enabled using ADIAPI_TxEnablePackets to be able to send this packet.

Parameters

Packet
: Pointer to the packet HB0 (Header Byte 0).

Size
: Byte size of the packet.

Return value

ATVERR_OK
ATVERR_INV_PARM

- **ATV_ERR ADIAPI_TxSendSpare1Packet (UCHAR *Packet, UCHAR Size)**

Description

Send any user-defined packet to the sink device. The TX Device has a general-purpose packet memory that can be filled with any data to be sent to the sink. One use of such packets is to send vendor-specific info-frame. The VS (Vendor-specific) packet repeat must be enabled using ADIAPI_TxEnablePackets to be able to send this packet.

Parameters

Packet
: Pointer to the packet HB0 (Header Byte 0).

Size
: Byte size of the packet.

Return value

ATVERR_OK
ATVERR_INV_PARM

- **ATV_ERR ADIAPI_TxSendSpare2Packet (UCHAR *Packet, UCHAR Size)**

Description

Send any user-defined packet to the sink device. The TX Device has a general-purpose packet memory that can be filled with any data to be sent to the sink. One use of such packets is to send vendor-specific info-frame. The VS (Vendor-specific) packet repeat must be enabled using ADIAPI_TxEnablePackets to be able to send this packet.

Parameters

Packet
: Pointer to the packet HB0 (Header Byte 0).

Size
: Byte size of the packet.

Return value

ATVERR_OK
ATVERR_INV_PARM

- **ATV_ERR ADIAPI_TxCecEnable (BOOL Enable)**

Description

This API enables or disables the CEC controller engine. This API is available only if CEC support is included by setting the Configuration switch „TX_INCLUDE_CEC".

Parameters

Enable
TRUE to enable the CEC controller.
FALSE to disable the CEC controller.

Return value

ATVERR_OK
Operation completed successfully.

- **ATV_ERR ADIAPI_TxCecReset (void)**

Description

This API resets the CEC controller engine. It is called as part of the ADIAPI_TxCecEnable API. This API is available only if CEC support is included by setting the Configuration switch „TX_INCLUDE_CEC".

Parameters

None

Return value

ATVERR_OK
Operation completed successfully.

- **ATV_ERR ADIAPI_TxCecSendMessage (UCHAR *MsgPtr, UCHAR MsgLen)**

Description

Send a CEC message. This API is available only if CEC support is included by setting the Configuration switch „TX_INCLUDE_CEC".

Parameters

MsgPtr
: Pointer to the CEC message to be sent.

MsgLen
: CEC message length.

Return value

ATVERR_OK
: Message is queued to be sent. The application must poll ADIAPI_TxCecGetStatus API to determine when send is completed before calling any further APIs.

ATVERR_FAILED
: If CEC controller is busy. The message will not be sent.

ATVERR_INV_PARM
: If MsgLen is larger than maximum message size (16 bytes).

- **ATV_ERR ADIAPI_TxCecSendMessageOut(void)**

Description

Send out CEC message in buffer.

Parameters

None

Return value

ATVERR_OK
: If message is sent successfully.

ATVERR_FAILED
: If CEC controller is busy.

- **ATV_ERR ADIAPI_TxCecResendLastMessage (void)**

Description

This API sends the last CEC massage again. This API is available only if CEC support is included by setting the Configuration switch „TX_INCLUDE_CEC".

Parameters

None

Return value

ATVERR_OK
: Message is queued to be sent. The application must poll ADIAPI_TxCecGetStatus API to determine when send is completed before calling any further APIs.

ATVERR_FAILED
: If CEC controller is busy. The message will not be sent.

- **ATV_ERR ADIAPI_TxCecReadMessage (UCHAR *MsgPtr, UCHAR *MsgLen)**

Description

Read a CEC message if available. This API is available only if CEC support is included by setting the Configuration switch „TX_INCLUDE_CEC".

Parameters

MsgPtr
: Pointer to a buffer to receive CEC message (maximum 16 bytes).

MsgLen
: Pointer to receive CEC message length.

Return value

ATVERR_OK
: Message read into MsgPtr parameter.

ATVERR_FAILED
: If no message is available.

- **ATV_ERR ADIAPI_TxCecSetLogicalAddr (UCHAR LogAddr, UCHAR DevId, BOOL Enable)**

Description

This API sets the device logical address. Up to 3 different logical addresses can be set for the device. To inquire about logical addresses available for use (not allocated) the application can use the ADIAPI_TxCecAllocateLogAddr API. This API is available only if CEC support is included by setting the Configuration switch „TX_INCLUDE_CEC".

Parameters

LogAddr

Logical address to be set for the device.

DevID

The device to set the logical address to. Up to 3 different devices can be used. This value can be 0, 1 or 2.

Enable

Enable or disable the logical address.

Return value

ATVERR_OK

Logical address set.

ATVERR_INV_PARM

If DevID is larger than 2.

- **ATV_ERR ADIAPI_TxCecAllocateLogAddr (UCHAR *LogAddrList)**

Description

This API checks the availability of logical addresses. This API is available only if CEC support is included by setting the Configuration switch „TX_INCLUDE_CEC".

Parameters

LogAddrList

Pointer to a prioritized list of logical addresses that the device will try to obtain, terminated by 0xff.

Return value

ATVERR_OK

Operation is queued to be processed. The application must poll ADIAPI_TxCecGetStatus API to determine the logical address that can be used.

ATVERR_FAILED

If CEC controller is busy. The operation will not be completed.

- **ATV_ERR ADIAPI_TxCecGetStatus (UCHAR *Status)**

Description

This API returns the status of the last performed CEC operation. Some CEC APIs return immediately and the application is required to poll CEC state to determine if the operation was successful.

Parameters

Status

Pointer to receive status on CEC engine last operation. The value returned in the Status parameter depends on the return value of this API.
If the return value is ATVERR_FAILED, the Status parameter will be set to the error code indicating the cause of failure.
If the return value is ATVERR_OK, the Status parameter will be set according to the last requested operation as follows:

API	Status value on success
ADIAPI_TxCecEnable	0
ADIAPI_TxCecReset	0
ADIAPI_TxCecSendMessage	0
ADIAPI_TxCecResendLastMessage	0
ADIAPI_TxCecAllocateLogAddr	First available (not allocated) logical address in the logical address list supplied to the API. If no logical address is available, this value will be 0xFF.

Return value

ATVERR_OK

Last CEC operation was completed successfully. Result is returned in the “Status” parameter.

ATVERR_NOT_AVAILABLE

Last CEC operation is still in progress.

ATVERR_FAILED

Last CEC operation failed. Error code is returned in the “Status” parameter as:

CEC_ERR_TX_TIMEOUT

CEC_ERR_ARB_LOST

- **ATV_ERR ADIAPI_TxArcSetMode (TX_ARC_MODE Mode)**

Description

Enable/Disable Audio Return Channel (ARC) operation. This API is valid only on TX or transceiver-type devices that support ARC feature.

Parameters

Mode

Select required ARC mode. This can be one of the following:

TX_ARC_SINGLE

For single mode.

TX_ARC_COMMON

For common mode.

TX_ARC_OFF

To turn ARC feature off.

Return value

ATVERR_OK

ATVERR_NOT_AVAILABLE

- **ATV_ERR ADIAPI_TxSetVideoClkDelay (TX_VIDEO_CLK_DELAY Delay)**

Description

Set delay for input video clock.

Parameters

Delay

Delay format. This can be one of the followings:

TX_VIDEO_CLK_DELAY_M_1200PS

Delay by -1200 psec.

TX_VIDEO_CLK_DELAY_M_800PS

Delay by -800 psec.

TX_VIDEO_CLK_DELAY_M_400PS

Delay by -400 psec.

TX_VIDEO_CLK_DELAY_NULL

No delay.

TX_VIDEO_CLK_DELAY_P_400PS

Delay by 400 psec.

TX_VIDEO_CLK_DELAY_P_800PS

Delay by 800 psec.

TX_VIDEO_CLK_DELAY_P_1200PS

Delay by 1200 psec.

TX_VIDEO_CLK_DELAY_P_1600PS

Delay by 1600 psec.

Return value

ATVERR_OK

3. Notification events

The Transmitter library provide the option to notify the application of any changes in operating condition, thus relieving the application from polling the library for status changes. The application must define a function that will be called from the library (from inside ADIAPI_TxIsr) upon any change in operating conditions. The prototype for this function is as follows:

UINT16 TRANSMITTER_Notification (TX_EVENT EventID, UINT16 EventSize, void *EventData)

Where:

EventID

This is a unique value that identifies the operating condition that has changed.

EventSize

This parameter either specifies the size of the data pointed to by the EventData parameter or carries some other information related to the event. This value is event dependant as listed in the table below.

EventData

This is a pointer to a buffer that contains event-specific information. If the event does not have an associated data, this value will be undefined. The table below lists all events and their associated data.

Depending on the enabled events (See ADIAPI_TxSetEnabledEvents,) the application will be notified via the above function of the events listed in the following table:

Interrupt	EventID	EventSize	EventData
HPD Low	TX_EVENT_HPD_CHG	0	Pointer to BOOL value = FALSE
HPD High	TX_EVENT_HPD_CHG	0	Pointer to BOOL value = TRUE
Rx Sense Low/High	TX_EVENT_MSEN_CHG	0	Pointer to BOOL value: = FALSE if monitor sense is LOW TRUE if monitor sense is HIGH
EDID Ready	TX_EVENT_EDID_READY when a new EDID segment is fully read	Index of the segment returned in EventData (0, 1, 2, etc)	Pointer to buffer containing the EDID segment read from sink
BKSV ready	TX_EVENT_BKSV_READY when all BKSVs are received	Number of BKSVs returned in EventData	Pointer to a concatenated list of all downstream BKSVs
HDCP Authenticated	TX_EVENT_HDCP_AUTHENTICATED	0	NULL
HDCP Error	TX_EVENT_HDCP_ERROR	0	Pointer to TX_HDCP_ERR value containing the error code
Vsync Edge	TX_EVENT_VSYNC_EDGE	0	NULL
Audio FIFO Full	TX_EVENT_AUDIO_FIFO_FULL	0	NULL
Embedded Sync Polarity Error	TX_EVENT_EMB_SYNC_ERROR	0	NULL
CEC TX Ready	TX_EVENT_CEC_TX_READY	0	NULL
CEC RX Ready	TX_EVENT_CEC_RX_READY	CEC message size	CEC message

The return value of the notification event function is not currently used and should always be set to 0.