IT9500 Windows SDK Programmer's Guide

Revision Log:

Revision	Date	Author	Remarks
1.0	2012/07/19	Jason	First release
1.01	2012/08/07	Jeff	1. ITE_TxSendTSData() document update
1.02	2012/09/03	Jeff	1. Testkit update 2. API update ITE_TxGetNumOfDevice ITE_TxGetOutputGain ITE_TxGetChipType ITE_TxGetTPS ITE_TxSetTPS ITE_TxGetGainRange ITE_TxSendCustomPacketOnce ITE_TxSetPeridicCustomPacket ITE_TxSetPeridicCustomPacketTimer
1.5	2013/02/01	Jeff	1. Sync to latest v1.5 API
1.6	2013/7/1		1. IQ calibration table update 2. TPS cell id in big-endian 3. ITE_TxSetDCCalibrationValue()

ITE Tech. Inc.

Easy HD Expressway

Table of Contents

1 Introduction

This document describes how to program IT9500 (also known as Eagle) Digital TV modulator USB Dongle under Microsoft Windows platforms.. The intended readers of this document are Windows software programmers. For Linux developers, please refer to IT9500 Linux SDK Programmer's Guide.
IT9500 series include IT9507 and IT9503.

Figure 1: IT9500 USB Dongle, DB-01-01 v03

1.1. Control IT9500

A host CPU can control IT9500 through either IIC or USB bus. This document describes how to control and send video transport streams to IT9500 via USB bus.
For IIC interface programming, please refer to IT9500_Programming Guide.

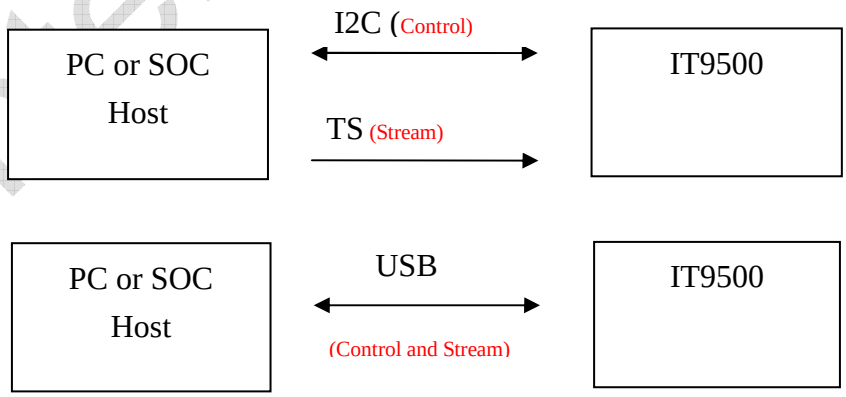

Figure 2: IT9500 controlled by a Host

1.2.IT9500 Windows application software Hierarchy

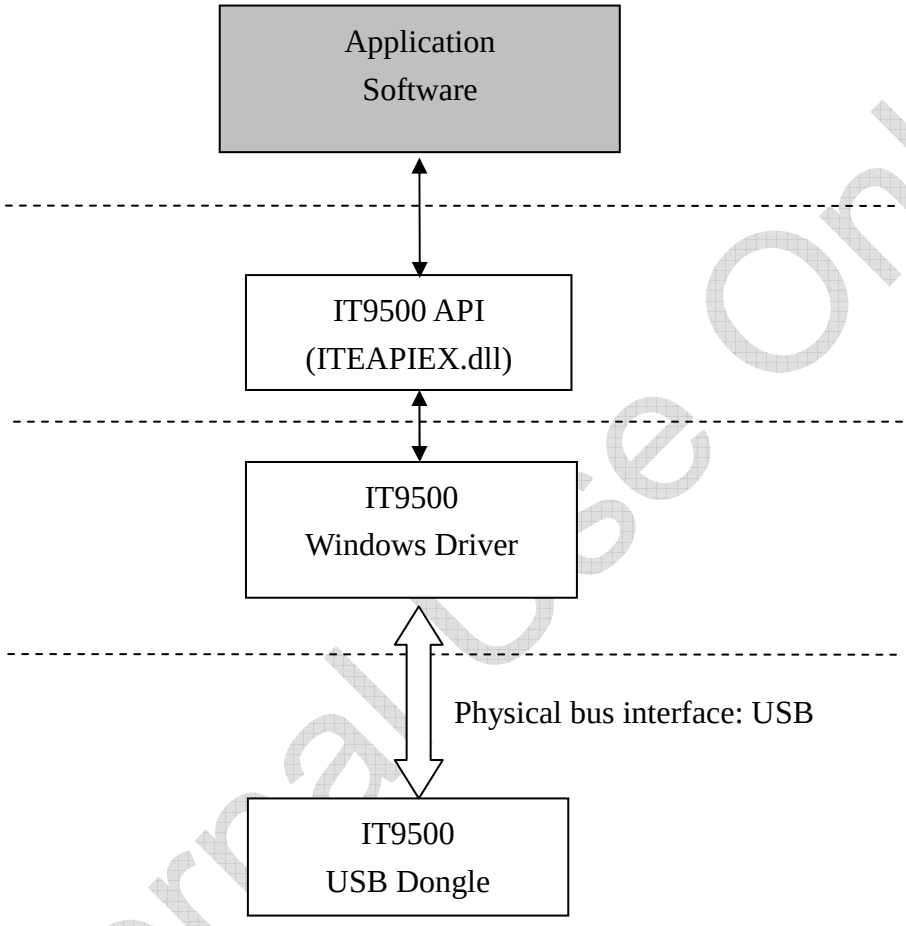

Figure 3: IT9500 Software Hierarchy

2 Package Contents

2.1 IT9500 Testkit and SDK binary file

A sample compiled test kit and IT9500 API DLL file can be found in the folder \Bin. Both 32-bit and 64-bit files are included.

ITEAPIEX.dll: The SDK DLL file for transmission (TX) API's.
ITEAPIEXTest_TX.exe: A sample test tool for Windows SDK

For user's convenience, a sample test transport stream Test.ts is included. The data rate of Test.ts is around 9.952 Mbps.
The sample testkit can only read and handle 188-byte TS files, while 204-byte format TS files are not supported.

2.2 IT9500 Testkit source code

The sample test kit source codes can be found in \Source\ITEAPIEXTest_TX

3 Launch Test kit

3.1 Environment Setup

Step 1: Plug in IT9500 USB dongle and install Windows driver properly

Step 2: Copy Test.ts , ITEAPIEX.dll and ITEAPIEXTest_TX.exe to the same folder

Step 3: Run ITEAPIEXTest_TX.exe

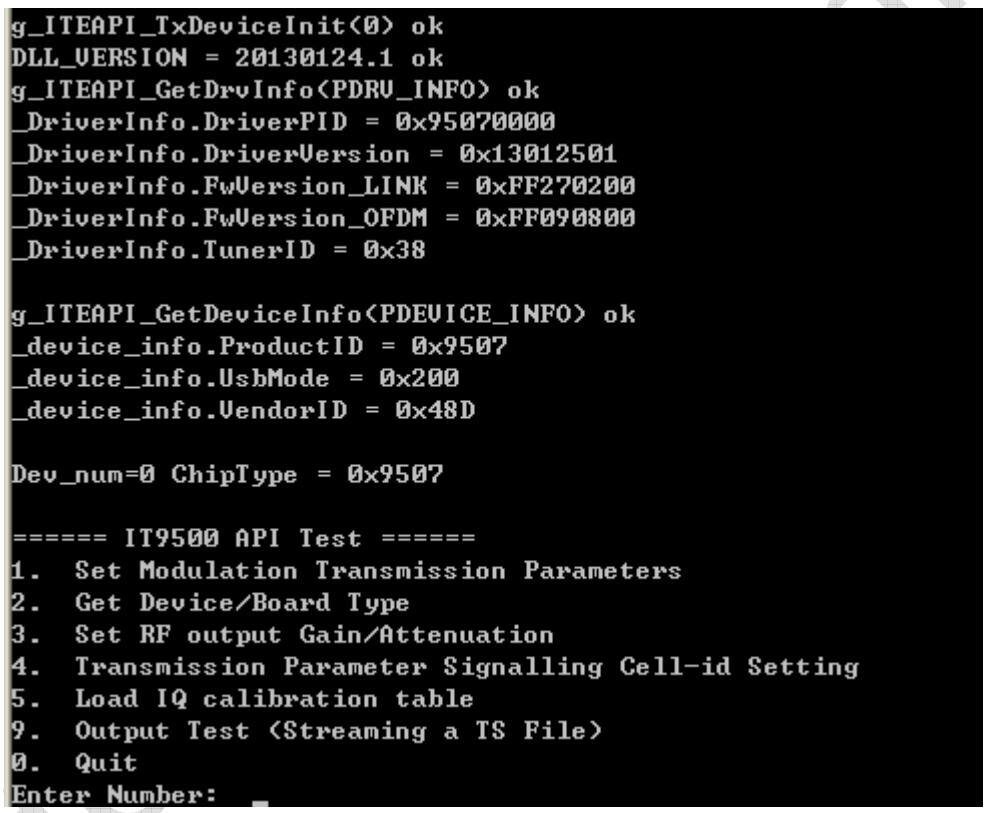

Figure 4: Initialization Messages

The number of IT9500 devices attached, version codes and chip type information will be shown while test kit started.

Multiple IT9500 devices can be supported. By default, the test kit selects the first device (Dev_num=0).

3.2 Test Kit Usage

3.2.1 Set Transmission Parameters

Select 1 "Set Modulation Transmission Parameters"
The following example shows how to set
Channel Frequency 533MHz, 6MHz bandwidth
16QAM, Code Rate 2/3, Guard Interval 1/4, FFT 8K.

```
Enter Number:  1

Enter frequency in KHz (ex. 666000Khz):  533000

Enter bandwidth in KHz (ex. 8000Khz):  6000
g_ITEAPI_TxSetChannel ok

Enter Constellation(0: QPSK, 1: 16QAM, 2: 64QAM):  1

Enter Code Rate(0: 1/2, 1: 2/3, 2: 3/4, 3:5/6, 4: 7/8):  1

Enter Guard Interval(0: 1/32, 1: 1/16, 2: 1/8, 3: 1/4):  3

Enter Transmission Mode(0: 2K, 1: 8K):  1
g_ITEAPI_TxSetChannelModulation ok
The Maximum Channel Capacity is 9952767 bps(9952 Kbps)
```

Figure 5: Transmission Parameter Settings

The transmission parameters also decides the channel capacity, i.e. the maximum channel throughput or output data rate. The configuration above will give a maximum data rate of 9952767 bps (about 9.952 Mbps).

A piece of sample code is included to calculate the maximum channel data rate. Also, refer to Appendix A for data rate calculation formula.

The maximum channel throughput should be equal to or larger than the input stream file's data rate.

3.2.2 Get Device Board Type

Select 2 "Get Device/Board Type"

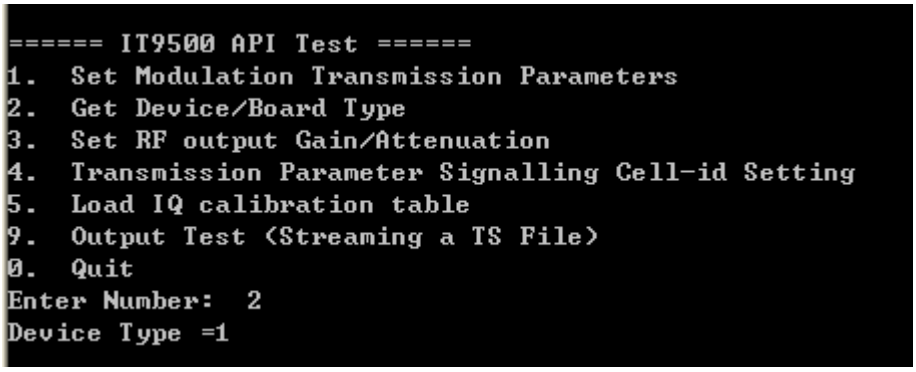

Figure 6: Device type

The device type corresponds to the IT9500 board version.
Some device type enumerations are listed below,
0: EVB
1:DB-01-01 v01
2:DB-01-02 v01
3:DB-01-01 v03

The board device type number is recorded in the EEPROM of the USB device.

3.2.3 Set RF output Gain/Attenuation

Select 3 "Set RF output Gain/Attenuation"

RF output gain/attenuation:
The RF output power gain/attenuation is configurable, step 1 db. The valid value range depends on the transmission parameter frequency/bandwidth in step 1.
0 db gain/attenuation is recommended in normal operation.
The gain/attenuation control is achieved by software digital attenuation algorithm, so the noise figure could be increased and MER might become poorer if it's set to a large (either positive or negative) value.

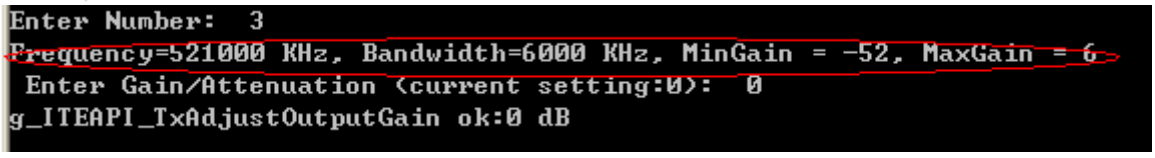

Figure 7: Gain Settings

3.2.4 Transmission Parameter Signalling Cell-id Setting

It's optional to set the cell id in the TPS.

3.2.5 Load IQ calibration table

It's optional to load an IQ calibration table.
If there is an IQ calibration table, the RF signal quality (MER) will be further optimized.
However, IT9500 still works well without any calibration table loaded.
Loading a wrong calibration file may get worse performance result.

3.2.6 Output Test (Streaming a TS File)

Select 9 "Output Test (Streaming a TS File)" to start RF signal transmission.

A sample transport stream file Test.ts is included with this package. You may specify any other valid transport stream file as well.

IT9500 can support custom SI/PSI table insertion. Input "y" to enable SDT insertion.

```
Input the TS file path name:  test.ts
test.ts size = 10729536
Insert Periodical Custom Packets for SI SDT Table (y:yes) ?
PAT TS ID:0x001a and Service ID:0x00c9 found
```

Figure 8: Custom Table Insertion

In the sample test kit, a piece of codes demonstrate how to insert DVB SDT table; the source TS file is analyzed to get TS ID and Service ID which is required to compose a SDT table, the composed SDT table is set to IT9500, and then an repetition interval is assigned to activate the periodical SDT packet transmission.
At maximum, 5 custom packets can be defined. Each packet can be assigned its repetition rate (transmission interval) individually. The interval is specified in ms. Setting 0 ms interval will disable the corresponding custom packet. Refer to the same source codes for details.

When a valid file is input, the file's data rate will be checked and shown for reference.

The data rate should be set correctly, or else the TV receiver will not be able to play the received program smoothly without glitch.

By default, the data rate can be set to the same as shown if the stream file is intact and the bit rate calculated is correct.

Also, the maximum channel throughput (which is determined by the transmission parameters) should be equal to or larger than the input stream file's data rate.

The data rate of Test.ts is 9952941 bps (about 9.952 Mbps).

With a simple data rate control mechanism in the sample code, the test kit will push (stream out) the stream data file in the specified data rate.

The test kit calculates the input file data rate automatically by investigating the PCR time stamps in the stream file. Refer to Appendix B. More complicated algorithm can be implemented to ensure the correctness of the calculated stream data rate, in case of a partially corrupted stream file.

```
Enter Number:  9
Input the TS file path name:  test.ts
test.ts size = 10729536
Insert Periodical Custom Packets for SI SDT Table (y:yes) ?
PAT TS ID:0x001a and Service ID:0x00c9 found
TS Sync byte found in offset:0
1'st PCR Offset:376,PID:2031(0x7ef),PCRB:3752086029(0xdfa44a0d),PCRE:36(0x24),PC
R:(0x1486c760)
The recommended input file data rate for test.ts is = 9952 KBps
Enter Data Bit rate in Kbps(ex. 10000 for 10 Mbps):  9952
Repeat Loop: 1.Repeat, 2.Once:  1
Press 'A' or 'a' to abort...
TS Sync byte found in offset:0
```

Figure 9: Start Transmission

4 SDK Library Interface and API Reference

Interface	Description
ITE_TxDeviceInit	Initialize IT9500 modulator device. This function will create a IT9500 device handle to access device interfaces.
ITE_TxDeviceExit	Exit modulator (IT9500) device. This function will close a IT9500 device handle.
ITE_TxPowerCtl	Control the device power.
ITE_TxSetChannel	Set the frequency and bandwidth to IT9500 TX.
ITE_TxSetChannelModulation	Set the channel modulation parameters of IT9500 TX.
ITE_TxGetDeviceType	Get device/board type
ITE_TxAdjustOutputGain	Set RF output Gain/Attenuation
ITE_TxSetModeEnable	Enable/Disable modulation transmitter front end RF output
ITE_TxGetNumOfDevice	
ITE_TxGetOutputGain	
ITE_TxGetChipType	
ITE_TxGetTPS	
ITE_TxSetTPS	
ITE_TxGetGainRange	
ITE_TxSendCustomPacketOnce	
ITE_TxSetPeridicCustomPacket	
ITE_TxSetPeridicCustomPacketTimer	
ITE_TxSetIQtable	

4.1 System and Version Information

4.1.1 ITE_GetDllVersion

Syntax

char* (WINAPIV *ITE_GetDllVersion)();

Purpose

Retrieve the SDK API Dll version code.

Parameters

None

Return Values

SDK API Dll version code string.

4.1.2 ITE_TxGetDrvInfo

Syntax

int (WINAPIV *ITE_TxGetDrvInfo)(PDRV_INFO pDriverInfo, int DevNo);

Purpose

Retrieve the device driver version code.

Parameters

PDRV_INFO pDriverInfo:

```
{
    DWORD DriverPID;
    DWORD DriverVersion;
    DWORD FwVersion_LINK;
    DWORD FwVersion_OFDM;
    BYTE    TunerID;
};
```

int DevNo:

Specify the device index number to be checked, the first device installed is 0.

Note: In a system, there might be multiple IT9500 devices installed.

Return Values

ERROR_NO_ERROR: successful, non-zero error code otherwise.

4.1.3 ITE_TxGetDeviceInfo

Syntax

bool (WINAPIV *ITE_TxGetDeviceInfo)(PDEVICE_INFO pDeviceInfo, int DevNo);

Purpose

Retrieve the SDK API Dll version code.

Parameters

PDEVICE_INFO pDeviceInfo**:**

```
{
    WORD    UsbMode;    //0x0200, 0x0110
    WORD    VendorID;
    WORD    ProductID;
};
```

int DevNo:

Specify the device index number to be checked, the first IT9500 device installed is 0.

Note: In a system, there might be multiple IT9500 devices installed.

Return Values

true: successful, false: fail.

4.1.4 ITE_TxGetNumOfDevice

Syntax

int (WINAPIV *ITE_TxGetNumOfDevice)(int *Tx_NumOfDev);

Purpose

Check how many IT9500 devices are attached in the system

Parameters

int * Tx_NumOfDev:

Number of IT9500 devices

Return Values

true: successful, false: fail.

4.1.5 ITE_TxGetChipType

Syntax

int (WINAPIV *ITE_TxGetChipType)(int *pChipType, int Tx_DevNo);

Purpose

Check IT9500 chip model number.

Parameters

int *pChipType:

IT9500 chip model number. Currently, there are two types of chips,

0x9507: IT9507, full featured

0x9503: IT9503, support only QPSK, CR:1/2, GI:1/4 mode.

int DevNo:

Specify the device index number to be checked, the first IT9500 device installed is 0.

Note: In a system, there might be multiple IT9500 devices installed.

Return Values

true: successful, false: fail.

4.2 Device Control

4.2.1 ITE_TxDeviceInit

Syntax

int (WINAPIV *ITE_TxDeviceInit)(int Tx_DevNo);

Purpose

Initialize the software access interface of IT9500 device. An internal device handle will be created to access device interfaces.

The internal device handle is invisible to the SDK developers.

Parameters

int DevNo:

Specify the device index number to be accessed, the first device installed is 0.

Note: In a system, there might be multiple IT9500 devices installed.

Return Values

true: open device handle successful, false: failure

4.2.2 ITE_TxDeviceExit

Syntax

int (WINAPIV *ITE_TxDeviceExit)(int Tx_DevNo);

Purpose

Finalize the software access interface of IT9500 device. The internal opened device handle will be closed.

Parameters

int DevNo:

Specify the device index number to be accessed, the first device installed is 0.

Note: In a system, there might be multiple IT9500 devices installed.

Return Values

true: close device handle successful, false: failure

4.2.3 ITE_TxPowerCtl

Syntax

int (WINAPIV * ITE_TxPowerCtl)(bool bOn, int Tx_DevNo);

Purpose

Control the device power state.

An IT9500 device should be powered on before setting channel configurations.

Parameters

bool bOn:

true: on, false: off

int DevNo:

Specify the device index number to be accessed, the first device installed is 0.

Note: In a system, there might be multiple IT9500 devices installed.

Return Values

Error_NO_ERROR: successful, non-zero error code otherwise.

4.2.4 ITE_TxSetChannel

Syntax

int (WINAPIV *ITE_TxSetChannel)(DWORD Frequency, WORD Bandwidth, int DevNo);

Purpose

Set the transmission channel frequency and bandwidth.

Parameters

DWORD Frequency:

The channel RF frequency, in KHz, range 70000~950000KHz (70MHz~950MHz).

WORD Bandwidth:

The channel RF bandwidth, in KHz, range 1000KHz~8000KHz (1MHz~8MHz.).

int DevNo:

Specify the device index number to be accessed, the first device installed is 0.

Note: In a system, there might be multiple IT9500 devices installed.

Return Values

Error_NO_ERROR: successful, non-zero error code otherwise.

4.2.5 ITE _TxSetChannelModulation

Syntax

int (WINAPIV *ITE_TxSetChannelModulation)(PMODULATION_PARAM pModulationParam, int DevNo);

Purpose

Set the transmission parameters.

Parameters

PMODULATION_PARAM pModulationParam:

```
{
    DWORD IOCTLCode;
    BYTE highCodeRate;
    BYTE transmissionMode;
    BYTE constellation;
    BYTE interval;
};
```

constellation:

0: QPSK, 1: 16QAM, 2: 64QAM

highCodeRate:

The code rate, only one high priority stream is supported by IT9500, i.e. no hierachical mode support.

0: 1/2, 1: 2/3, 2: 3/4, 3:5/6, 4: 7/8

interval:

Guard Interval

0: 1/32, 1: 1/16, 2: 1/8, 3: 1/4

transmissionMode:

0: 2K, 1: 8K

int DevNo:

Specify the device index number to be accessed, the first device installed is 0.

Note: In a system, there might be multiple IT9500 devices installed.

Return Values

Error_NO_ERROR: successful, non-zero error code otherwise.

4.2.6 ITE_TxGetDeviceType

Syntax

int (WINAPIV *ITE_TxGetDeviceType)(BYTE *pDeviceType, int Tx_DevNo);

Purpose

Get the current device board type setting on EEPROM.

Parameters

BYTE *pDeviceType:

The board type of the device

0: EVB, 1:DB-01-01 v01. 2:DB-01-02 v01, 3:DB-01-01 v03

int DevNo:

Specify the device index number to be accessed, the first device installed is 0.

Note: In a system, there might be multiple IT9500 devices installed.

Return Values

Error_NO_ERROR: successful, non-zero error code otherwise.

4.2.7 ITE_TxGetGainRange

Syntax

int (WINAPIV *ITE_TxGetGainRange)(DWORD Frequency, WORD Bandwidth, int *pMaxGain, int *pMinGain, int DevNo);

Purpose

Get the valid range of gain/attenuation settings. The valid range is not fixed, but depends on the channel frequency and bandwidth.

Parameters

DWORD Frequency:

The channel RF frequency, in KHz, range 70000~950000KHz (70MHz~950MHz).

WORD Bandwidth:

The channel RF bandwidth, in KHz, range 1000KHz~8000KHz (1MHz~8MHz.).

int *pMaxGain: in dB

int *pMinGain: in dB

int DevNo:

Specify the device index number to be accessed, the first device installed is 0.

Note: In a system, there might be multiple IT9500 devices installed.

Return Values

Error_NO_ERROR: successful, non-zero error code otherwise.

4.2.8 ITE_TxAdjustOutputGain

Syntax

int (WINAPIV *ITE_TxAdjustOutputGain)(int *pGain, int Tx_DevNo);

Purpose

Set the RF output Gain/Attenuation

Parameters

int *pGain:

It's specified in dB. The valid range can be retrieved by calling the API ITE_TxGetGainRange.

After the function call returns successfully, check the value in *pGain to find the real gain/attenuation set.

int DevNo:

Specify the device index number to be accessed, the first device installed is 0.

Note: In a system, there might be multiple IT9500 devices installed.

Return Values

Error_NO_ERROR: successful, non-zero error code otherwise.

4.2.9 ITE_TxGetOutputGain

Syntax

int (WINAPIV *ITE_ TxGetOutputGain)(int *pGain, int Tx_DevNo);

Purpose

Get the current RF output Gain/Attenuation setting

Parameters

int *pGain:

It's specified in dB.

int DevNo:

Specify the device index number to be accessed, the first device installed is 0.

Note: In a system, there might be multiple IT9500 devices installed.

Return Values

Error_NO_ERROR: successful, non-zero error code otherwise.

4.2.10 ITE_TxSetTPS

Syntax

int (WINAPIV *ITE_TxSetTPS)(TPS Tps, int Tx_DevNo);;

Purpose

Set TPS (Transmission Parameter Signaling) .

Currently, only cell-id field is configurable.

Parameters

TPS Tps:

```
struct _TPS{
    BYTE highCodeRate;
    BYTE lowCodeRate;
    BYTE transmissionMode;
    BYTE constellation;
    BYTE interval;
    WORD cellid;
};
```

int DevNo:

Specify the device index number to be accessed, the first device installed is 0.

The cellid should be input in big-endian

Note: In a system, there might be multiple IT9500 devices installed.

Return Values

Error_NO_ERROR: successful, non-zero error code otherwise.

4.2.11 ITE_TxGetTPS

Syntax

int (WINAPIV *ITE_TxGetTPS)(TPS *pTps, int Tx_DevNo);;

Purpose

Retrieve TPS (Transmission Parameter Signaling) .

Parameters

TPS *pTps:

```
struct _TPS{
    BYTE highCodeRate;
    BYTE lowCodeRate;
    BYTE transmissionMode;
    BYTE constellation;
    BYTE interval;
    WORD cellid;
};
```

int DevNo:

Specify the device index number to be accessed, the first device installed is 0.

The cellid should be input in big-endian

Note: In a system, there might be multiple IT9500 devices installed.

Return Values

Error_NO_ERROR: successful, non-zero error code otherwise.

4.2.12 ITE_TxSetModeEnable

Syntax

```
int (WINAPIV *ITE_TxSetModeEnable)(bool bEnable, int DevNo);
```

Purpose

Enable/Disable the transmitter output, i.e. start/stop the stream transmission.

Parameters

bool bEnable:

true: on, false: off

int DevNo:

Specify the device index number to be accessed, the first device installed is 0.

Note: In a system, there might be multiple IT9500 devices installed.

Return Values

Error_NO_ERROR: successful, non-zero error code otherwise.

4.2.13 ITE_TxSendTSData

Syntax

int (WINAPIV *ITE_TxSendTSData)(DWORD* BufferLength, BYTE* Buffer, int DevNo);

Purpose

Send the data stream to be transmitted to the modulator.

Parameters

DWORD* BufferLength:

Buffer size in bytes, the maximum value is 188*348 bytes.

BYTE* Buffer:

1. **The TS packets must be in 188-byte format. 204-byte TS format is not supported.**
2. **The first byte of the data buffer block must be sync byte, i.e. 0x47.**

int DevNo:

Specify the device index number to be accessed, the first device installed is 0.

Note: In a system, there might be multiple IT9500 devices installed.

Return Values

Error_NO_ERROR: successful, non-zero error code otherwise.

4.3 Custom table/packet insertion

4.3.1 ITE_TxSendCustomPacketOnce

Syntax

int (WINAPIV *ITE_TxSendCustomPacketOnce)(DWORD Buffer_Length, BYTE* Buffer, int Tx_DevNo);

Purpose

Send a custom packet for SI/PSI table insertion.

The custom packet will be sent once (one shot only) when null packets are required to stuff the channel.

Parameters

DWORD* BufferLength:

Buffer size in bytes, the maximum value is 188 bytes.

BYTE* Buffer:

TS packet buffer

int DevNo:

Specify the device index number to be accessed, the first device installed is 0.

Note: In a system, there might be multiple IT9500 devices installed.

Return Values

Error_NO_ERROR: successful, non-zero error code otherwise.

4.3.2 ITE_TxSetPeridicCustomPacket

Syntax

int (WINAPIV *ITE_TxSetPeridicCustomPacket)(DWORD Buffer_Length, BYTE* Buffer, BYTE Index, int Tx_DevNo);

Purpose

Set a custom packet to the internal custom table/packet buffer.

There are 5 188-byte packet buffers within IT9500. Each buffer holds a single 188-byte custom packet, and is indexed as 1~5.

Each packet buffer is also associated with a periodic timer. Whenever the timer expires, the packet will be sent once and the timer restarts. Thus, the packet will be sent periodically in fixed interval.

Parameters

DWORD* BufferLength:

Buffer size in bytes, the maximum value is 188 bytes.

BYTE* Buffer:

TS packet buffer

BYTE Index:

Packet buffer index, 1~5

int DevNo:

Specify the device index number to be accessed, the first device installed is 0.

Note: In a system, there might be multiple IT9500 devices installed.

Return Values

Error_NO_ERROR: successful, non-zero error code otherwise.

4.3.3 ITE_TxSetPeridicCustomPacketTimer

Syntax

int (WINAPIV *ITE_TxSetPeridicCustomPacketTimer)(BYTE Index, BYTE Timer, int Tx_DevNo);

Purpose

Set custom packet buffer timer interval in milliseconds.

Parameters

BYTE Index:

Packet buffer index, 1~5

BYTE Timer:

Timer interval in milliseconds.

int DevNo:

Specify the device index number to be accessed, the first device installed is 0.

Note: In a system, there might be multiple IT9500 devices installed.

Return Values

Error_NO_ERROR: successful, non-zero error code otherwise.

4.4 Performance optimization with IQ calibration

Because of different characteristics with different designs, it's necessary to compensate IT9500 IQ output for optimized signal quality.

The calibration table is stored in an IQ calibration bin file released by ITE. Users may call ITE_TxSetIQtable() to inform the underlying API that a new IQ calibration table should be referenced instead of the built-in table.

Refer to the following table for the bin file format. A file dump sample is also shown in the following figure.

Table 1 File format of IQ calibration bin file

Offset	Size (bytes)	Descriptions
0~0x9	10	Mnemonic descriptions for this file
0xa~0xd	4	Version code
0xe~0x0f	2	The number of calibration item entries (in Big-endian)
0x10~0x13	4	#1 Frequency in KHz
0x14~0x15	2	dAmp
0x16~0x17	2	dPhi

0x18~0x1b	4	#2 Frequency in KHz
0x1c~0x1d	2	dAmp
0x1e~0x1f	2	dPhi
…		

Note: all entry values are in little-endian except the "number of calibration item entries".

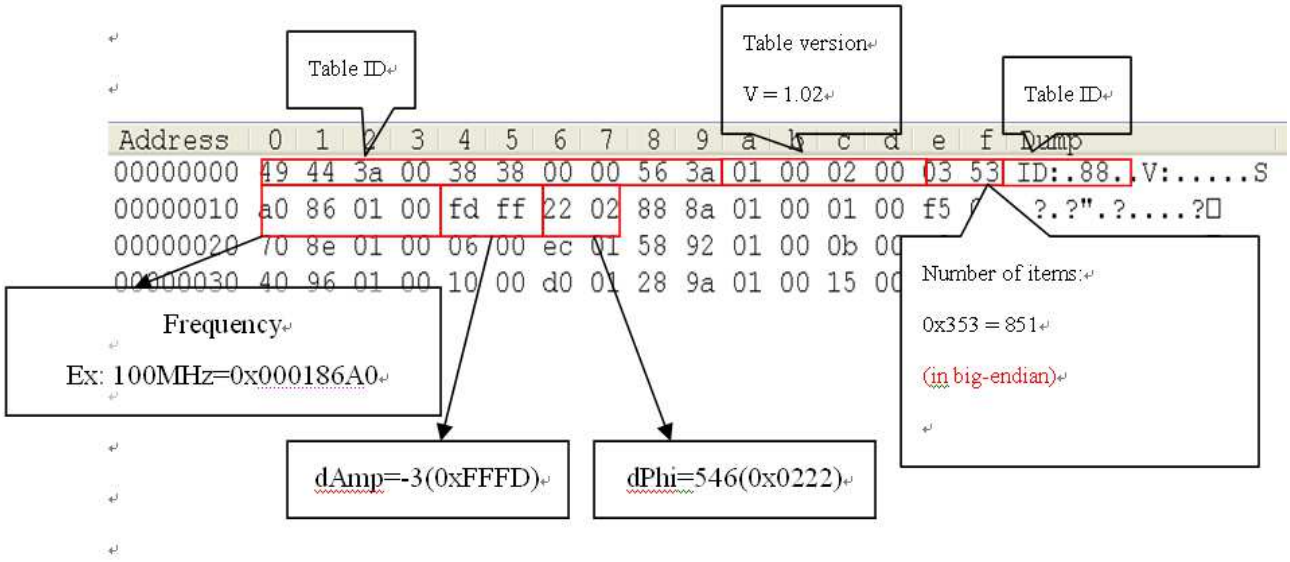

Figure 10 IQ calibration bin file format.

4.4.1 ITE_TxSetIQtable

Syntax

ITE_TxSetIQtable(IQtable *IQ_table, int IQtable_element_number, int Tx_DevNo);

Purpose

Set the IQ calibration table.

Parameters

IQtable:

Pointer to the IQ calibration table entries

IQtable_element_number:

Number of entries.

int DevNo:

Specify the device index number to be accessed, the first device installed is 0.

Note: In a system, there might be multiple IT9500 devices installed.

Return Values

Error_NO_ERROR: successful, non-zero error code otherwise.

4.4.2 ITE_TxSetDCCalibrationValue

Syntax

ITE_TxSetDCCalibrationValue)(int dc_i, int dc_q, int Tx_DevNo);

Purpose

Set the DC compensation value

Parameters

int dc_i:

int dc_q:

Specify the DC offset compensation for I & Q

int DevNo:

Specify the device index number to be accessed, the first device installed is 0.

Note: In a system, there might be multiple IT9500 devices installed.

Return Values

Error_NO_ERROR: successful, non-zero error code otherwise.

5 Appendix A: Bit Rate Calculation for DVB-T Modulation

DVB-T modulator maximum play rate depends on code rate, constellation, guard interval, bandwidth. The maximum bit rate can be calculated as below.

Tbandwidth = {6,000,000, 7,000,000, 8,000,000} in Hz for 6MHz, 7MHz, 8MHz
Tcode_rate = {1/2, 2/3, 3/4, 5/6, 7/8}
TConstellation = {2, 4, 6} <- QPSK = 2, 16QAM=4, 64QAM = 6
TGuardInterval = {4/5, 8/9, 16/17, 32/33}, 1/4 = 4/5, 1/8 = 8/9, 1/16 => 16/17, 1/32 => 32/33}
2K/8K mode does not matter

Maximum bit rate = 1512 / 2048 * 188 / 204 * 64 / 56 * TBandwidth * Tcode_rate * TConstellation * TGuardInterval (bps)
= 423 / 544 * TBandwidth * Tcode_rate * TConstellation * TGuardInterval (bps)

Refer to the following tables for calculated results for various configurations.

5 M Bandwidth Maximum bit rate(bps):

GI	CR	QPSK	16-QAM	64-QAM
1 / 4	1 / 2	3110294	6220588	9330882
	2 / 3	4147059	8294118	12441176
	3 / 4	4665441	9330882	13996324
	5 / 6	5183824	10367647	15551471
	7 / 8	5443015	10886029	16329044
1 / 8	1 / 2	3455882	6911765	10367647
	2 / 3	4607843	9215686	13823529
	3 / 4	5183824	10367647	15551471
	5 / 6	5759804	11519608	17279412
	7 / 8	6047794	12095588	18143382
1 / 16	1 / 2	3659170	7318339	10977509
	2 / 3	4878893	9757785	14636678
	3 / 4	5488754	10977509	16466263
	5 / 6	6098616	12197232	18295848
	7 / 8	6403547	12807093	19210640

1 / 32	1 / 2	3770053	7540107	11310160
	2 / 3	5026738	10053476	15080214
	3 / 4	5655080	11310160	16965241
	5 / 6	6283422	12566845	18850267
	7 / 8	6597594	13195187	19792781

6 M Bandwidth Maximum bit rate(bps):

GI	CR	QPSK	16-QAM	64-QAM
1 / 4	1 / 2	3732353	7464706	11197059
	2 / 3	4976471	9952941	14929412
	3 / 4	5598529	11197059	16795588
	5 / 6	6220588	12441176	18661765
	7 / 8	6531618	13063235	19593853
1 / 8	1 / 2	4147059	8294118	12441176
	2 / 3	5529412	11058824	16588235
	3 / 4	6220588	12441176	18661765
	5 / 6	6911765	13823529	20735294
	7 / 8	7257353	14514706	21772059
1 / 16	1 / 2	4391003	8782007	13173010
	2 / 3	5854671	11709343	17564014
	3 / 4	6586505	13173010	19759516
	5 / 6	7318339	14636678	21955017
	7 / 8	7684256	15368512	23052768
1 / 32	1 / 2	4524064	9048128	13572193
	2 / 3	6032086	12064171	18096257
	3 / 4	6786096	13572193	20358289
	5 / 6	7540107	15080214	22620321
	7 / 8	7917112	15834225	23751337

7 M Bandwidth Maximum bit rate (bps):

GI	CR	QPSK	16-QAM	64-QAM
1 / 4	1 / 2	4354412	8708824	13063235
	2 / 3	5805882	11611765	17417647
	3 / 4	6531618	13063235	19593853
	5 / 6	7257353	14514706	21772059
	7 / 8	7620221	15240441	22860662
1 / 8	1 / 2	4838235	9676471	14514706

	2 / 3	6450980	12901961	19352941
	3 / 4	7257353	14514706	21772059
	5 / 6	8063725	16127451	24191176
	7 / 8	8466912	16933824	25400735
1 / 16	1 / 2	5122837	10245675	13568512
	2 / 3	6830450	13660900	20491349
	3 / 4	7684256	15368512	23052768
	5 / 6	8538062	17076125	25614187
	7 / 8	8964965	17929931	26894896
1 / 32	1 / 2	5278075	10556150	15834225
	2 / 3	7037433	14074866	21112299
	3 / 4	7917112	15834225	23751337
	5 / 6	8796791	17593583	26390374
	7 / 8	9236631	18473262	27709893

8 M Bandwidth Maximum bit rate (bps):

GI	CR	QPSK	16-QAM	64-QAM
1 / 4	1 / 2	4976471	9952941	14929412
	2 / 3	6635294	13270588	19905882
	3 / 4	7464706	14929412	22394118
	5 / 6	8294118	16588235	24882353
	7 / 8	8708824	17417647	26126471
1 / 8	1 / 2	5529412	11058824	16588235
	2 / 3	7372549	14745098	22117647
	3 / 4	8294118	16588235	24882353
	5 / 6	9215686	18431373	27647059
	7 / 8	9676471	19352941	29029412
1 / 16	1 / 2	5854671	11709343	17564014
	2 / 3	7806228	15612457	23418685
	3 / 4	8782007	17564014	26346021
	5 / 6	9757785	19515571	29273356
	7 / 8	10245675	20491349	30737024
1 / 32	1 / 2	6032086	12064171	18096257
	2 / 3	8042781	16085561	24128342
	3 / 4	9048128	18096257	27144385
	5 / 6	10053476	20106952	30160428

	7 / 8	10556150	21112299	31668449

6 Appendix B: Calculation of TS Bitrate

TS bitrate - depends on the PCR for that particular stream. They're sent out at a somewhat standard time gap (mine's at 90ms). All that's involved is getting the PCR base and ext, calculating the PCR based on the formula in the ISO 13818-1 doc, and then using this formula to get the *byte* rate:

27000000 * (packets between PCR final byte) / (PCR2 - PCR1)

multiply that by 8 to get the *bit* rate. basically the PCRs are measurements on the "system clock" and that system clock for TS is 27Mhz. so the units look like this: ticks/s * bytes / (ticks) === bytes / s

Specifically:

$PCR(i) = PCR\_base(i) \cdot 300 + PCR\_ext(i)$

where:

$PCR\_base(i) = ((system\_clock\_frequency \cdot t(i)) \; DIV \; 300) \; \% \; 2^{33}$

$PCR\_ext(i) = ((system\_clock\_frequency \cdot t(i)) \; DIV \; 1) \; \% \; 300$

$system\_clock\_frequency$ =27 000 000 Hz

Refer to ISO 13818-1 2.4.3 Specification of the Transport Stream syntax and semantics and 2.4.4.8 Program Map Table

Table 2-28 – Transport Stream program map section

Syntax	No. of bits	Mnemonic
TS_program_map_section() {		
table_id	8	uimsbf
section_syntax_indicator	1	bslbf
'0'	1	bslbf
reserved	2	bslbf
section_length	12	uimsbf
program_number	16	uimsbf
reserved	2	bslbf
version_number	5	uimsbf
current_next_indicator	1	bslbf
section_number	8	uimsbf
last_section_number	8	uimsbf
reserved	3	bslbf
PCR_PID	13	uimsbf
reserved	4	bslbf
program_info_length	12	uimsbf
for (i = 0; i < N; i++) {		
descriptor()		
}		
for (i = 0; i < N1; i++) {		
stream_type	8	uimsbf
reserved	3	bslbf
elementary_PID	13	uimsbf
reserved	4	bslbf
ES_info_length	12	uimsbf
for (i = 0; i < N2; i++) {		
descriptor()		
}		
}		
CRC_32	32	rpchof
}		

Table 2-2 – Transport packet of this Recommendation | International Standard

Syntax	No. of bits	Mnemonic
transport_packet(){		
sync_byte	8	bslbf
transport_error_indicator	1	bslbf
payload_unit_start_indicator	1	bslbf
transport_priority	1	bslbf
PID	13	uimsbf
transport_scrambling_control	2	bslbf
adaptation_field_control	2	bslbf
continuity_counter	4	uimsbf
if(adaptation_field_control == '10' || adaptation_field_control == '11'){		
adaptation_field()		
}		
if(adaptation_field_control == '01' || adaptation_field_control == '11') {		
for (i = 0; i < N; i++){		
data_byte	8	bslbf
}		
}		
}		

Table 2-6 – Transport Stream adaptation field

Syntax	No. of bits	Mnemonic
adaptation_field() {		
adaptation_field_length	8	uimsbf
if (adaptation_field_length > 0) {		
discontinuity_indicator	1	bslbf
random_access_indicator	1	bslbf
elementary_stream_priority_indicator	1	bslbf
PCR_flag	1	bslbf
OPCR_flag	1	bslbf
splicing_point_flag	1	bslbf
transport_private_data_flag	1	bslbf
adaptation_field_extension_flag	1	bslbf
if (PCR_flag == '1') {		
program_clock_reference_base	33	uimsbf
reserved	6	bslbf
program_clock_reference_extension	9	uimsbf
}		
if (OPCR_flag == '1') {		
original_program_clock_reference_base	33	uimsbf
reserved	6	bslbf
original_program_clock_reference_extension	9	uimsbf
}		
if (splicing_point_flag == '1') {		

7 Appendix C: PAT, SDT, NIT

Reference:

ISO_IEC_13818-1 2.4.4 Program specific information and Annex C

ETSI EN_300468

Repetition rates and random access:

The minimum time interval between the arrival of the last byte of a section to the first byte of the next transmitted section with the same PID, table_id and table_id_extension and with the same or different section_number shall be 25 ms.

ETSI TR 101 211 defines the maximum timer interval as,

Table	**Maximum Repetition Rate**
PAT	<100ms
CAT	<1s
TSDT	
reserved	
NIT, ST	<10s
SDT, BAT, ST	<2s
EIT, ST	<2s
RST, ST	
TDT, TOT, ST	<30s

Table 1: PID allocation for SI

Table	PID value
PAT	0x0000
CAT	0x0001
TSDT	0x0002
reserved	0x0003 to 0x000F
NIT, ST	0x0010
SDT, BAT, ST	0x0011
EIT, ST CIT (TS 102 323 [15])	0x0012
RST, ST	0x0013
TDT, TOT, ST	0x0014
network synchronization	0x0015
RNT (TS 102 323 [15])	0x0016
reserved for future use	0x0017 to 0x001B
inband signalling	0x001C
measurement	0x001D
DIT	0x001E
SIT	0x001F

Table 2: Allocation of table_id values

Value	Description
0x00	program_association_section
0x01	conditional_access_section
0x02	program_map_section
0x03	transport_stream_description_section
0x04 to 0x3F	reserved
0x40	network_information_section - actual_network
0x41	network_information_section - other_network
0x42	service_description_section - actual_transport_stream
0x43 to 0x45	reserved for future use
0x46	service_description_section - other_transport_stream
0x47 to 0x49	reserved for future use
0x4A	bouquet_association_section
0x4B to 0x4D	reserved for future use
0x4E	event_information_section - actual_transport_stream, present/following
0x4F	event_information_section - other_transport_stream, present/following
0x50 to 0x5F	event_information_section - actual_transport_stream, schedule
0x60 to 0x6F	event_information_section - other_transport_stream, schedule
0x70	time_date_section
0x71	running_status_section
0x72	stuffing_section
0x73	time_offset_section
0x74	application information section (TS 102 812 [17])
0x75	container section (TS 102 323 [15])
0x76	related content section (TS 102 323 [15])
0x77	content identifier section (TS 102 323 [15])
0x78	MPE-FEC section (EN 301 192 [4])
0x79	resolution notification section (TS 102 323 [15])

PAT: **Program Association Table**

Table 2-25 – Program association section

Syntax	No. of bits	Mnemonic
program_association_section() {		
table_id	8	uimsbf
section_syntax_indicator	1	bslbf
'0'	1	bslbf
reserved	2	bslbf
section_length	12	uimsbf
transport_stream_id	16	uimsbf
reserved	2	bslbf
version_number	5	uimsbf
current_next_indicator	1	bslbf
section_number	8	uimsbf
last_section_number	8	uimsbf
for (i = 0; i < N; i++) {		
program_number	16	uimsbf
reserved	3	bslbf
if (program_number == '0') {		
network_PID	13	uimsbf
}		
else {		
program_map_PID	13	uimsbf
}		
}		
CRC_32	32	rpchof
}		

NIT: Network Information Table

Table 3: Network information section

Syntax	Number of bits	Identifier
network_information_section(){		
table_id	8	uimsbf
section_syntax_indicator	1	bslbf
reserved_future_use	1	bslbf
reserved	2	bslbf
section_length	12	uimsbf
network_id	16	uimsbf
reserved	2	bslbf
version_number	5	uimsbf
current_next_indicator	1	bslbf
section_number	8	uimsbf
last_section_number	8	uimsbf
reserved_future_use	4	bslbf
network_descriptors_length	12	uimsbf
for(i=0;i<N;i++){		
descriptor()		
}		
reserved_future_use	4	bslbf
transport_stream_loop_length	12	uimsbf
for(i=0;i<N;i++){		
transport_stream_id	16	uimsbf
original_network_id	16	uimsbf
reserved_future_use	4	bslbf
transport_descriptors_length	12	uimsbf
for(j=0;j<N;j++){		
descriptor()		
}		
}		
CRC_32	32	rpchof
}		

SDT: Service Description Table

Table 5: Service description section

Syntax	Number of bits	Identifier
service_description_section(){		
table_id	8	uimsbf
section_syntax_indicator	1	bslbf
reserved_future_use	1	bslbf
reserved	2	bslbf
section_length	12	uimsbf
transport_stream_id	16	uimsbf
reserved	2	bslbf
version_number	5	uimsbf
current_next_indicator	1	bslbf
section_number	8	uimsbf
last_section_number	8	uimsbf
original_network_id	16	uimsbf
reserved_future_use	8	bslbf
for (i=0;i<N;i++){		
service_id	16	uimsbf
reserved_future_use	6	bslbf
EIT_schedule_flag	1	bslbf
EIT_present_following_flag	1	bslbf
running_status	3	uimsbf
free_CA_mode	1	bslbf
descriptors_loop_length	12	uimsbf
for (j=0;j<N;j++){		
descriptor()		
}		
}		
CRC_32	32	rpchof
}		