VIVOTEK NETWORK DEVELOPMENT PLATFORM

Data Packet Parser

Version 5.1.0.3

2009/8/31

VIVOTEK may make changes to specifications and product descriptions at any time, without notice.

The following is trademarks of VIVOTEK Inc., and may be used to identify VIVOTEK products only: VIVOTEK. Other product and company names contained herein may be trademarks of their respective owners.

No part of this publication may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm, without permission in writing from VIVOTEK Inc.

TABLE OF CONTENTS

1.Overview

1.1. Introduction 3
File Structure 3

2.SAMPLE CODE

2.1. SaveAndLoadV3Packet 4

3.API Reference

3.1. Data Structure 9
TMediaDataPacketInfo 10
TMediaDataPacketInfoV3 13
TMediaDataPacketInfoEx 14
TMediaPacketFirstReserved 15
TVUExtInfo 16
TMotionTriggerInfo 17
TCaptureWinInfo 18
TMediaPacketTimeZoneInfo 19
TMediaAudioInfo 20
3.2. Enumeration 21
EMediaCodecType 22
TMediaDBFrameType 24
3.3. API Definition 25
DataPacket_Parse 26
DataPacket_ParseV3 27
DataPacket_GetEveryAudioInfo 28

1. Overview

1.1. Introduction

This document describes the properties and methods supported by the VIVOTEK Data Packet parser.

The Data Packet parser provides functions to parse and get the content of a Data Packet.

File Structure

FILE	DESCRIPTION
doc\VNDP_DataPacketParser_API.pdf	This manual
lib\d_parsedatapacket.lib	Dynamic linking
lib\parsedatapacket.dll	Dynamic runtime library
Inc\datapacketdef.h	Data Packet definition header file
Inc\parsedatapacket.h	Data Packet parser header file

2. SAMPLE CODE

2.1. SaveAndLoadV3Packet

DESCRIPTION

User may want to save and load the TMediaDataPacketInfoV3, but the structure of TMediaDataPacketInfoV3 is more complex than TMediaDataPacketInfo. Here we will introduce you how to save the packet into file and than re-load it from file.

SAMPLE CODE

Save Packet

```
DWORD dwDummyData = 0; // NOTE: for buffer is null
// STEP 1: write out pbyTLVExt

if (ptPacket->tIfEx.tRv1.tExt.pbyTLVExt != NULL)
{
    fwrite(ptPacket->tIfEx.tRv1.tExt.pbyTLVExt, 1, ptPacket->tIfEx.tRv1.tExt.dwTLVExtLen, fp);
}
else
{
    fwrite(&dwDummyData, 1, sizeof(dwDummyData), fp);
}
// STEP 2: write out pbyBuff

if (ptPacket->tIfEx.tInfo.pbyBuff != NULL)
{
    fwrite(ptPacket->tIfEx.tInfo.pbyBuff, 1, ptPacket->tIfEx.tInfo.dwOffset + ptPacket->tIfEx.
tInfo.dwBitstreamSize, fp);
}
else
{
    fwrite(&dwDummyData, 1, sizeof(dwDummyData), fp);
}
// STEP 3: write out pbyVExtBuf

if (ptPacket->pbyVExtBuf != NULL)
{
    fwrite(&ptPacket->dwVExtLen, 1, sizeof(DWORD), fp);
    fwrite(ptPacket->pbyVExtBuf, 1, ptPacket->dwVExtLen, fp);
}
else
{
    fwrite(&dwDummyData, 1, sizeof(dwDummyData), fp);
}
```

Load Packet

// STEP 1: initial packet

```
memset(ptPacket, 0, sizeof(TMediaDataPacketInfoV3));
ptPacket->tIfEx.tRv1.tExt.dwStructSize = sizeof(TMediaDataPacketInfoV3);
```

// STEP 2: read into pbyTLVExt

```
fread(&ptPacket->tIfEx.tRv1.tExt.dwTLVExtLen, 1, sizeof(DWORD), fp);
if (ptPacket->tIfEx.tRv1.tExt.dwTLVExtLen != 0)
{
    memcpy(pbyTLVExt, &ptPacket->tIfEx.tRv1.tExt.dwTLVExtLen, sizeof(DWORD));
    ptPacket->tIfEx.tRv1.tExt.dwTLVExtLen = HTONL(ptPacket->tIfEx.tRv1.tExt.dwTLVExtLen);
    fread(pbyTLVExt + sizeof(DWORD), 1, ptPacket->tIfEx.tRv1.tExt.dwTLVExtLen, fp);
    ptPacket->tIfEx.tRv1.tExt.pbyTLVExt = pbyTLVExt;
}
ptPacket->tIfEx.tRv1.tExt.dwTLVExtLen += sizeof(DWORD);
```

// STEP 3: read into pbyBuff

```
DWORD dwFrameSize = 0;
fread(&dwFrameSize, 1, sizeof(DWORD), fp);
if (dwFrameSize != 0)
{
    memcpy(pbyBuff, &dwFrameSize, sizeof(DWORD));
    dwFrameSize = HTONL(dwFrameSize);
    fread(pbyBuff + sizeof(DWORD), 1, dwFrameSize, fp);
    ptPacket->tIfEx.tInfo.pbyBuff = pbyBuff;
}
```

```
// STEP 4: read into pbyVExtBuf
DWORD dwVExtLen = 0;
fread(&dwVExtLen, 1, sizeof(DWORD), fp);
if (dwVExtLen)
{
    dwVExtLen = HTONL(dwVExtLen);
    fread(pbyVExtBuf, 1, dwVExtLen, fp);
    ptPacket->pbyVExtBuf = pbyVExtBuf;
}

// STEP 5: reconstruct packet

DataPacket_ParseV3(ptPacket);
```

TIPS

Remember to set ptPacket->tIfEx.tRv1.tExt.dwStructSize = sizeof(TMediaDataPacketInfoV3), otherwise the DataPacket_ParseV3 function won't parse the V3 information.

3. API Reference

This chapter contains the API function calls for the Data Packet Parser.

3.1. Data Structure

The data structure is depicted here.

TMediaDataPacketInfo

This structure provides the information in the Data Packet.

```
typedef struct
{
BYTE                    *pbyBuff;
DWORD                   dwOffset;
DWORD                   dwBitstreamSize;
DWORD                   dwStreamType;
DWORD                   dwFrameNumber;
TMediaDBFrameType       tFrameType;
DWORD                   dwFirstUnitSecond;
DWORD                   dwFirstUnitMilliSecond;
BOOL                    bFixedFrameSize;
DWORD                   dwAudioSamplingFreq;
BYTE                    byAudioChannelNum;
DWORD                   dwDIAlert;
DWORD                   dwDO;
BOOL                    bMotionDetection[3];
BOOL                    bMotionDetectionAlertFlag[3];
BYTE                    byMotionDetectionPercent[3];
WORD                    wMotionDetectionAxis[3][4];
BOOL                    bTimeModified;
BOOL                    bAudioDI;
BOOL                    bNoVideoSignal;
}
TMediaDataPacketInfo;
```

Members

***pbyBuff**

the pointer of buffer containing this Data Packet

dwOffset

the offset of buffer refer to the media bitstream, 32-bits align

dwBitstreamSize

the size of media bitstream

dwStreamType

the type of stream in the Data Packet, indicated by enumeration EMediaCodecType

dwFrameNumber

the number of frame in the Data Packet

tFrameType

the type of frame

dwFirstUnitSecond

the second of first frame in the Data Packet (time in seconds from midnight, January 1, 1970)

dwFirstUnitMilliSecond

the millisecond of first frame in the Data Packet

bFixedFrameSize

the flag indicates the frame size is fixed

dwAudioSamplingFreq

audio sampling frequency, if the stream has only one kind of sampling frequency, this field is ignored.

byAudioChannelNum

the channel number of audio, if the stream has only one kind of channel number, this field is ignored.

dwDIAlert

the digital input alert, each bit presents a digital input source. Bit 0 is for DI 0, bit 1 is for DI 1, bit 2 is for DI 2, and bit 3 is for ID 3. Other bits are reserved.

dwDO

Each bit is used to indicate the DO (H/L). It supports two digital output in the present. The LSB indicates the first digital output.

bMotionDetection[3]

the flags of motion detection which indicate the motion detection windows are active or not

bMotionDetectionAlertFlag[3]

the flags of motion detection alert

byMotionDetectionPercent[3]

the percentages of motion detection

wMotionDetectionAxis[3][4]

the window of motion detection, 1st element is the left coordinate of rectangular, 2nd element is the top coordinate of rectangular, 3rd element is the width of rectangular and 4th element is the height of rectangular.

bTimeModified

the flag indicates the time is modified according to timezone. If the value is FALSE, the time is dependent the time zone

bAudioDI

the flag indicates audio packets take the DI Alert information

bNoVideoSignal

the flag indicates the loss of video signal

Remarks

Variable “dwDIMask” is removed in version 2.0.0.0. Variable “dwDO” is added in version 3.0.0.0. Variable “bTimeModified” and “bAudioDI” are added in version 4.0.0.0. Variable “bNoVideoSignal” is added in version 5.0.0.0.

Requirements

datapacketdef.h

TMediaDataPacketInfoV3

This structure provides the information in the new Data Packet (version 3). The most important part of the V3 is the capturing and cropping information, you could refer to the sample code “GetV3Information” for the detail.

```
typedef struct
{
TMediaDataPacketInfoEx          tIfEX;
TMediaPacketTimeZoneInfo        tTZ;
DWORD                           dwUTCTime;
TVUExtInfo                      tVUExt;
BYTE                            *pbyVExtBuf;
DWORD                           dwVExtLen

} TMediaDataPacketInfoV3;
```

Members

tIfEx

TMediaDataPacketInfoEx

tTZ

TMediaPacktTimeZoneInfo

dwUTCTime;

The UTC time of this packet.This value could be got from dwFristUnitSecond and time zone information, too.

tVUExt

TVUExtInfo

pbyVExtBuf

Video further extension. If the value is null, it means this packet don’t have such extension.

dwVExtLen

The size of the pbyVExtBuf.

Requirements

datapacketdef.h

TMediaDataPacketInfoEx

This structure provides the information in the Data Packet.

```
typedef struct
{
TMediaDataPacketInfo        tInfo;
UMediaPktReserved1          tRv1;
DWORD                       dwWidth;
DWORD                       dwHeight;
DWORD                       dwWidthPadLeft
DWORD                       dwWidthPadRight
DWORD                       dwHeightPadTop
DWORD                       dwHeightPadBottom
}  TMediaDataPacketInfoEx;
```

Members

tInfo

TMediaDataPacketInfo

tRv1

UMediaPktReserved1. It is a union of TMediaPacketFirstReserved(tExt) and void*(apvReserved[4]). In most case, we use the tExt field.

For example:PacketV3.tIfEx.tRv1.tExt.pbyTLVExt

dwWidth

The width of the video frame if it's video (includes padding if any)

dwHeight

The height of the video frame if it's video (includes padding if any)

dwWidthPadLeft

Padded width of left, the value is usually zero.

dwWidthPadRight

Padded width of right, the value is usually zero.

dwHeightPadTop

Padded height of top, the value is usually zero.

dwHeightPadBottom

Padded height of bottom, the value is usually zero.

Requirements

datapacketdef.h

TMediaPacketFirstReserved

This structure provides the information of extension data of a frame

```
typedef struct
{
BYTE                    *pbyTLVExt;
DWORD                   dwTLVExtLen;
DWORD                   dwStructureSize;
}  TMediaPacketFirstReserved;
```

Members

pbyTLVExt

The pointer to the tag/length/data extension of a frame. The pointer would point to the location after media data in pbyBuff

dwTLVExtLen

the length of the pbyTLVExt

dwStructureSize

the size of the structure. If this is 0, it means the packet is Ex only, else this maps the size of the overall packet structure

Requirements

datapacketdef.h

TVUExtInfo

This structure provides the information in the Video Extension.

```
typedef struct
{
DWORD                   dwPIR;
DWORD                   dwWLLed;
TCaptureWinInfo         tCapWinInfo;
DWORD                   dwTamperingAlert;
TMotionTriggerInfo      *ptMTI;

}  TMediaDataPacketInfoEx;
```

Members

dwPIR

the PIR status, HIGH word means the PIR enabled flag, LOW word contains the values.

dwWLLed

the status for white light LED.

tCapWinInfo

the capture window info

dwTamperingAlert

tamperingalert info

***ptMTI**

point to the data in TMediaDataPacketInfo

Requirements

datapacketdef.h

TMotionTriggerInfo

This structure provides the information of motion window. All the value is the same as the value in TMediaDataPacketInfo.

```
typedef struct
{
BOOL                    bMotionDetection[3];
BOOL                    bMotionDetectionAlertFlag[3];
BYTE                    byMotionDetectionPercent[3];
WORD                    wMotionDetectionAxis[3][4];

}  TMotionTriggerInfo;
```

Members

bMotionDetection[3]

the flags of motion detection which indicate the motion detection windows are active or not

bMotionDetectionAlertFlag[3]

the flags of motion detection alert

byMotionDetectionPercent[3]

the percentages of motion detection

wMotionDetectionAxis[3][4]

the window of motion detection, 1st element is the left coordinate of rectangular, 2nd element is the top coordinate of rectangular, 3rd element is the width of rectangular and 4th element is the height of rectangular.

Requirements

datapacketdef.h

TCaptureWinInfo

This structure provides the information of Capture Window

```
typedef struct
{
BOOL                    bWithInfo;
WORD                    wCapW;
WORD                    wCapH;
WORD                    wOffX;
WORD                    wOffY;
WORD                    wCropW;
WORD                    wCropH;

}  TCaptureWinInfo;
```

Members

bWithInfo
If the information is valid.

wCapW
Capture Width

wCapH
Capture Height

wOffX
Offset X

wOffY
Offset Y

wCropW
Cropping Width

wCropH
Cropping Height

Requirements

datapacketdef.h

TMediaPacketTimeZoneInfo

This structure provides the information of time zone and daylight saving.

```
typedef struct
{
BOOL                    bTimeZone;
long                    lDLSaving;
long                    lOffsetSeconds;

}  TMediaPacketTimeZoneInfo;
```

Members

bTimeZone

if the packet contains time zone information, if no, the following fields are from client machine

lDLSaving

the daylight saving time in seconds (if 0 it means no day-light saving), -3600 for most case

lOffsetSeconds

offset of seconds from GMT time; for example Taipei will be 8 * 3600 = 28800

Requirements

datapacketdef.h

TMediaAudioInfo

This structure is used to get the size of audio units in a Data Packet through the DataPacket_GetEveryAudioInfo function.

```
typedef struct
{
DWORD                   dwSize;
} TMediaAudioInfo;
```

Members

dwSize

the size of an audio unit

Remarks

Requirements

parsedatapacket.h

3.2. Enumeration

The enumeration used is depicted here.

EMediaCodecType

This enumeration indicates the media codec type.

```
typedef enum
{
mctJPEG                 = 0x0001,
mctH263                 = 0x0002,
mctMP4V                 = 0x0004,
mctH264                 = 0x0008,
mctG7221                = 0x0100,
mctG729A                = 0x0200,
mctAAC                  = 0x0400,
mctGAMR                 = 0x0800,
mctSAMR                 = 0x1000
} EMediaCodecType;
```

Values

mctJPEG
the codec type is JPEG (image, video)

mctH263
the codec type is H.263 (video)

mctMP4V
the codec type is MPEG-4 video (video)

mctH264
the codec type is H.264 video (video)

mctG7221
the codec type is G.722.1 (audio)

mctG729A
the codec type is G.729A (audio)

mctAAC
the codec type is AAC (audio)

mctGAMR
the codec type is AMR (audio)

mctSAMR
the codec type is SAMR (audio)

Remarks

Requirements

mediatypedef.h

TMediaDBFrameType

This enumeration indicates the frame type of media.

```
typedef enum
{
MEDIADB_FRAME_INTRA          = 0,
MEDIADB_FRAME_PRED           = 1,
MEDIADB_FRAME_BIPRED         = 2
} TMediaDBFrameType;
```

Values

MEDIADB_FRAME_INTRA
the intra frame

MEDIADB_FRAME_PRED
the prediction frame

MEDIADB_FRAME_BIDIR
the bi-direction prediction frame

Remarks

Requirements

mediatypedef.h

3.3. API Definition

The API definition is depicted here.

DataPacket_Parse

Parse the Data Packet in the input buffer.

Syntax

```
SCODE DataPacket_Parse (    TMediaDataPacketInfo *ptDataPacketInfo      );
```

Parameters

***ptDataPacketInfo**

[in/out] A pointer of data structure storing information of Data Packet

Return Values

S_OK

Parse Data Packet successfully.

S_FAIL

Parse Data Packet failed.

Remarks

Before calling this function, the field of pbyBuff in the Data Packet structure must be valid.

Requirements

parsedatapacket.h

See Also

DataPacket_ParseV3

Parse the Data Packet in the input buffer into TMediaDataPacketInfoV3.

Syntax

```
SCODE DataPacket_ParseV3 (    TMediaDataPacketInfoV3 *ptDataPacketInfo      );
```

Parameters

***ptDataPacketInfoV3**

[in/out] A pointer of data structure storing information of Data Packet

Return Values

S_OK

Parse Data Packet successfully.

S_FAIL

Parse Data Packet failed.

Remarks

The usage of DataPacket_ParseV3 is more complex than DataPacket_Parser, please refer to the sample code “SaveAndLoadV3Packet” for the usage.

Requirements

parsedatapacket.h

See Also

DataPacket_GetEveryAudioInfo

This function gets the size and buffer address of every audio unit in a Data Packet.

Syntax

```
SCODE DataPacket_GetEveryAudioInfo (   BYTE *pbyInBuf,
                                       TMediaAudioInfo *ptAudioInfo );
```

Parameters

pbyInBuf

[in] the pointer of buffer containing an audio Data Packet

ptAudioInfo

[out] the pointer of structure to store the list of audio information, the size must be larger than audionumber * sizeof(TMediaAudioInfo)

Return Values

S_OK

Get audio info successfully.

S_FAIL

Get audio info failed.

Remarks

Requirements

parsedatapacket.h

See Also