UNF API

Difference Description

Issue 05

Date 2015-06-15

Trademarks and Permissions

 , HISILICON, and other HiSilicon icons are trademarks of HiSilicon Technologies Co., Ltd.

All other trademarks and trade names mentioned in this document are the property of their respective holders.

Notice

The purchased products, services and features are stipulated by the contract made between HiSilicon and the customer. All or part of the products, services and features described in this document may not be within the purchase scope or the usage scope. Unless otherwise specified in the contract, all statements, information, and recommendations in this document are provided "AS IS" without warranties, guarantees or representations of any kind, either express or implied.

The information in this document is subject to change without notice. Every effort has been made in the preparation of this document to ensure accuracy of the contents, but all statements, information, and recommendations in this document do not constitute a warranty of any kind, express or implied.

HiSilicon Technologies Co., Ltd.

Address: Huawei Industrial Base
Bantian, Longgang
Shenzhen 518129
People's Republic of China

Website: http://www.hisilicon.com

Email: support@hisilicon.com

About This Document

Purpose

This document describes the API differences between different UNF versions for enabling users to familiarize themselves with new UNF APIs rapidly.

Related Versions

The following table lists the product versions related to this document.

Product Name	Version
Hi3716C	V2XX
Hi3719C	V1XX
Hi3719M	V1XX
Hi3718C	V1XX
Hi3718M	V1XX
Hi3716M	V4XX
Hi3798C	V1XX
Hi3796C	V1XX
Hi3798M	V1XX
Hi3796M	V1XX
Hi3798C	V2XX

Intended Audience

This document is intended for:

- Technical support engineers
- Software development engineers

Change History

Changes between document issues are cumulative. Therefore, the latest document issue contains all changes made in previous issues.

Issue 05 (2015-06-15)

This issue is the fifth official release, which incorporates the following changes:

Section 12.1 is delected.

Issue 04 (2015-05-07)

This issue is the fourth official release, which incorporates the following changes:

Hi3798C V200, Hi3716M V420, and Hi3716M V410 are supported, and chapter 12 is added.

Issue 03 (2015-03-30)

This issue is the third official release, which incorporates the following changes:

Chapter 11 is added.

Issue 02 (2015-03-10)

This issue is the second official release, which incorporates the following changes:

Chapters 7, 8, 9, and 10 are added.

Issue 01 (2014-10-30)

This issue is the first official release, which incorporates the following change:

Hi3796M V100 is supported.

Issue 00B02 (2014-08-19)

This issue is the second draft release, which incorporates the following changes:

Chapters 4, 5, and 6 are added.

Issue 00B01 (2014-06-20)

This issue is the first draft release.

Contents

1 Overview

This document describes the API differences between different UNF versions from the following aspects:

- Changes of functions and specifications
- Changes of APIs and data structures
- Changes of API invocation processes

2 Differences Between UNF 3.2.0 and UNF 3.1

2.1 Common

2.1.1 Overview

The UNF 3.2.0 common module has the following changes compared with UNF 3.1:

The function of obtaining chip capabilities is added.

2.1.1.1 Obtaining Chip Capabilities

To support the function of obtaining chip capabilities, the data structure and API are changed as follows:

Data Structure

HI_CHIP_CAP_E is added.

API

HI_SYS_GetChipCapability is added.

2.1.2 Data Structure

2.1.2.1 New Data Structure

HI_CHIP_CAP_E

[Definition]

```
typedef enum hiCHIP_CAP_E
{
    HI_CHIP_CAP_DOLBY,
    HI_CHIP_CAP_DTS,
    HI_CHIP_CAP_ADVCA,
    HI_CHIP_CAP_MACROVISION
```

```
} HI_CHIP_CAP_E;
```

[Reason]

The enumeration is required for the new feature of obtaining chip capabilities.

[Note]

None

[Example]

None

2.1.2.2 Modified Data Structures

HI_CHIP_VERSION_E

[Definition]

Before modification:

```
typedef enum hiCHIP_VERSION_E
{
    HI_CHIP_VERSION_V100 = 0x100,
    HI_CHIP_VERSION_V101 = 0x101,
    HI_CHIP_VERSION_V200 = 0x200,
    HI_CHIP_VERSION_V300 = 0x300,
    HI_CHIP_VERSION_BUTT
}HI_CHIP_VERSION_E;
```

After modification:

```
typedef enum hiCHIP_VERSION_E
{
    HI_CHIP_VERSION_V100 = 0x100,
    HI_CHIP_VERSION_V101 = 0x101,
    HI_CHIP_VERSION_V200 = 0x200,
    HI_CHIP_VERSION_V300 = 0x300,
    HI_CHIP_VERSION_V400 = 0x400,
    HI_CHIP_VERSION_BUTT
}HI_CHIP_VERSION_E;
```

[Description]

The chip version information is added.

[Note]

None

[Example]

None

FLASH_OPT_S

[Definition]

Before modification:

```
typedef struct tagFLASH_OPT_S
{
   int (*raw_read)(int fd, unsigned long long *startaddr, unsigned char *buffer, unsigned long length, unsigned long long openaddr, unsigned long long limit_leng, int read_oob, int skip_badblock);
   int (*raw_write)(int fd, unsigned long long *startaddr, unsigned char *buffer, unsigned long length, unsigned long long openaddr, unsigned long long limit_leng, int write_oob);
   int (*raw_erase)(int fd, unsigned long long startaddr, unsigned long long length, unsigned long long openaddr, unsigned long long limit_leng);
} FLASH_OPT_S;
```

After modification:

```
typedef struct tagFLASH_OPT_S
{
   int (*raw_read)(int fd, unsigned long long *startaddr, unsigned char *buffer, unsigned long length, unsigned long long openaddr, unsigned long long limit_leng, int read_oob, int skip_badblock);
   int (*raw_write)(int fd, unsigned long long *startaddr, unsigned char *buffer, unsigned long length, unsigned long long openaddr, unsigned long long limit_leng, int write_oob);
   long long (*raw_erase)(int fd, unsigned long long startaddr, unsigned long long length, unsigned long long openaddr, unsigned long long limit_leng);
} FLASH_OPT_S;
```

[Description]

The length of data returned by the raw_erase function pointer is greater than that allowed in an int type parameter.

[Note]

None

[Example]

None

2.1.3 API

2.1.3.1 New API

HI_SYS_GetChipCapability

[Definition]

```
HI_S32 HI_SYS_GetChipCapability(HI_CHIP_CAP_E enChipCap, HI_BOOL
*pbSupport);
```

[Reason]

This API is added to obtain chip capabilities.

[Note]

None

[Example]

None

2.2 AVPLAY

2.2.1 Overview

The UNF 3.2.0 AVPLAY module has the following changes:

- The function of reporting the video frame error rate is added.
- The function of setting the trusted video path (TVP) attributes is added.
- The accurate seek function is added.
- The full-band is supported, and programs can be quickly switched when there are multiple tuners.
- The top/bottom field priority flag is added to the CC data.

2.2.1.1 Reporting the Video Frame Error Rate

When errors occur during video frame decoding, the error rate is reported, which ranges from 1 to 100. The data structure is changed as follows:

Data Structure

HI_UNF_AVPLAY_EVENT_E is modified.

2.2.1.2 Setting TVP Attributes

The TVP feature requires that secure memory is used for the entire channel from video stream transfer and decoding to display. Set the TVP attribute **bEnable** to **TRUE** only when the TVP feature is supported. The data structures are changed as follows:

Data Structure

- HI_UNF_AVPLAY_TVP_ATTR_S is added.
- HI_UNF_AVPLAY_ATTR_ID_E is modified.

2.2.1.3 Seeking to a Specific Position

This feature allows you to specify a presentation time stamp (PTS) when the decoder is reset so that the audio/video decoder can seek to the stream around the PTS and continue decoding. The data structure is changed as follows:

Data Structure

HI_UNF_AVPLAY_RESET_OPT_S is modified.

2.2.1.4 Supporting Full-Band

This feature enables fast program switchover. Multiple AVPLAYs can be started at the same time: one is in normal playback state, and others are in pre-play state (only receive TSs but do not decode and output audio/video). When the program is switched, the AVPLAY in normal playback state is switched to pre-play state. If the program to be switched to is in the AVPLAY that is in pre-play state, the AVPLAY is switched to normal playback state. As the AVPLAY has received and demultiplexed ESs before the switch, it decodes and outputs ESs quickly after the switch. In this way, the program is switched quickly. The data structures and APIs are changed as follows:

Data Structure

- HI_UNF_AVPLAY_PRESTART_OPT_S is added.
- HI_UNF_AVPLAY_PRESTOP_OPT_S is added.
- HI_UNF_AVPLAY_STATUS_E is modified.

API

The following APIs are added:

- HI_UNF_AVPLAY_PreStart
- HI_UNF_AVPLAY_PreStop

2.2.1.5 Adding the Top/Bottom Field Priority Flag for CC Data

The top/bottom field priority flag is added to ensure that CC data can be decoded correctly. The data structure is changed as follows:

Data Structure

HI_UNF_VIDEO_USERDATA_S is modified.

2.2.2 Data Structure

2.2.2.1 New Data Structures

HI_UNF_AVPLAY_PRESTART_OPT_S

[Definition]

```
typedef struct hiAVPLAY_PRESTART_OPT_S
{
    HI_U32      u32Reserved;
} HI_UNF_AVPLAY_PRESTART_OPT_S;
```

[Reason]

This data structure is added to support the full-band function.

[Note]

This data structure is reserved for future extension.

[Example]

None

HI_UNF_AVPLAY_PRESTOP_OPT_S

[Definition]

```
typedef struct hiAVPLAY_PRESTOP_OPT_S
{
    HI_U32      u32Reserved;
} HI_UNF_AVPLAY_PRESTOP_OPT_S;
```

[Reason]

This data structure is added to support the full-band function.

[Note]

This data structure is reserved for future extension.

[Example]

None

HI_UNF_AVPLAY_TVP_ATTR_S

[Definition]

```
typedef struct hiUNF_AVPLAY_TVP_ATTR_S
{
    HI_BOOL             bEnable;
}HI_UNF_AVPLAY_TVP_ATTR_S;
```

[Reason]

This data structure is added to support the TVP function.

[Note]

Set **bEnable** to **TRUE** only when the TVP feature is supported.

[Example]

None

2.2.2.2 Modified Data Structures

HI_UNF_AVPLAY_EVENT_E

[Definition]

Before modification:

```
typedef enum hiUNF_AVPLAY_EVENT_E
{
    HI_UNF_AVPLAY_EVENT_EOS,
    HI_UNF_AVPLAY_EVENT_STOP,
    HI_UNF_AVPLAY_EVENT_RNG_BUF_STATE,
    HI_UNF_AVPLAY_EVENT_NORM_SWITCH,
    HI_UNF_AVPLAY_EVENT_FRAMEPACKING_CHANGE,
    HI_UNF_AVPLAY_EVENT_NEW_VID_FRAME,
    HI_UNF_AVPLAY_EVENT_NEW_AUD_FRAME,
    HI_UNF_AVPLAY_EVENT_NEW_USER_DATA,
    HI_UNF_AVPLAY_EVENT_GET_AUD_ES,
    HI_UNF_AVPLAY_EVENT_IFRAME_ERR,
    HI_UNF_AVPLAY_EVENT_SYNC_PTS_JUMP,
    HI_UNF_AVPLAY_EVENT_SYNC_STAT_CHANGE,
    HI_UNF_AVPLAY_EVENT_VID_BUF_STATE,
    HI_UNF_AVPLAY_EVENT_AUD_BUF_STATE,
    HI_UNF_AVPLAY_EVENT_VID_UNSUPPORT,
    HI_UNF_AVPLAY_EVENT_BUTT
} HI_UNF_AVPLAY_EVENT_E;
```

After modification:

```
typedef enum hiUNF_AVPLAY_EVENT_E
{
    HI_UNF_AVPLAY_EVENT_EOS,
    HI_UNF_AVPLAY_EVENT_STOP,
    HI_UNF_AVPLAY_EVENT_RNG_BUF_STATE,
    HI_UNF_AVPLAY_EVENT_NORM_SWITCH,
    HI_UNF_AVPLAY_EVENT_FRAMEPACKING_CHANGE,
    HI_UNF_AVPLAY_EVENT_NEW_VID_FRAME,
    HI_UNF_AVPLAY_EVENT_NEW_AUD_FRAME,
    HI_UNF_AVPLAY_EVENT_NEW_USER_DATA,
```

```
    HI_UNF_AVPLAY_EVENT_GET_AUD_ES,
    HI_UNF_AVPLAY_EVENT_IFRAME_ERR,
    HI_UNF_AVPLAY_EVENT_SYNC_PTS_JUMP,
    HI_UNF_AVPLAY_EVENT_SYNC_STAT_CHANGE,
    HI_UNF_AVPLAY_EVENT_VID_BUF_STATE,
    HI_UNF_AVPLAY_EVENT_AUD_BUF_STATE,
    HI_UNF_AVPLAY_EVENT_VID_UNSUPPORT,
    HI_UNF_AVPLAY_EVENT_VID_ERR_RATIO,
    HI_UNF_AVPLAY_EVENT_BUTT
} HI_UNF_AVPLAY_EVENT_E;
```

[Reason]

This data structure is modified to support the function of reporting the video frame error rate.

[Note]

The ERR_RATIO event is not reported when no error occurs during video decoding. The error rate ranges from 1 to 100.

[Example]

None

HI_UNF_AVPLAY_ATTR_ID_E

[Definition]

Before modification:

```
typedef enum hiUNF_AVPLAY_ATTR_ID_E
{
    HI_UNF_AVPLAY_ATTR_ID_STREAM_MODE = 0,
    HI_UNF_AVPLAY_ATTR_ID_ADEC,
    HI_UNF_AVPLAY_ATTR_ID_VDEC,
    HI_UNF_AVPLAY_ATTR_ID_AUD_PID,
    HI_UNF_AVPLAY_ATTR_ID_VID_PID,
    HI_UNF_AVPLAY_ATTR_ID_PCR_PID,
    HI_UNF_AVPLAY_ATTR_ID_SYNC,
    HI_UNF_AVPLAY_ATTR_ID_AFD,
    HI_UNF_AVPLAY_ATTR_ID_OVERFLOW,
    HI_UNF_AVPLAY_ATTR_ID_MULTIAUD,
    HI_UNF_AVPLAY_ATTR_ID_FRMRATE_PARAM,
    HI_UNF_AVPLAY_ATTR_ID_FRMPACK_TYPE,
    HI_UNF_AVPLAY_ATTR_ID_LOW_DELAY,
    HI_UNF_AVPLAY_ATTR_ID_BUTT
} HI_UNF_AVPLAY_ATTR_ID_E;
```

After modification:

```
typedef enum hiUNF_AVPLAY_ATTR_ID_E
```

```
{
    HI_UNF_AVPLAY_ATTR_ID_STREAM_MODE = 0,
    HI_UNF_AVPLAY_ATTR_ID_ADEC,
    HI_UNF_AVPLAY_ATTR_ID_VDEC,
    HI_UNF_AVPLAY_ATTR_ID_AUD_PID,
    HI_UNF_AVPLAY_ATTR_ID_VID_PID,
    HI_UNF_AVPLAY_ATTR_ID_PCR_PID,
    HI_UNF_AVPLAY_ATTR_ID_SYNC,
    HI_UNF_AVPLAY_ATTR_ID_AFD,
    HI_UNF_AVPLAY_ATTR_ID_OVERFLOW,
    HI_UNF_AVPLAY_ATTR_ID_MULTIAUD,
    HI_UNF_AVPLAY_ATTR_ID_FRMRATE_PARAM,
    HI_UNF_AVPLAY_ATTR_ID_FRMPACK_TYPE,
    HI_UNF_AVPLAY_ATTR_ID_LOW_DELAY,
    HI_UNF_AVPLAY_ATTR_ID_TVP,
    HI_UNF_AVPLAY_ATTR_ID_BUTT
} HI_UNF_AVPLAY_ATTR_ID_E;
```

[Reason]

This data structure is modified to support the function of setting/obtaining TVP attributes.

[Note]

Set **bEnable** of HI_UNF_AVPLAY_TVP_ATTR_S to **TRUE** only when the TVP feature is supported.

[Example]

None

HI_UNF_AVPLAY_STATUS_E

[Definition]

Before modification:

```
typedef enum hiUNF_AVPLAY_STATUS_E
{
    HI_UNF_AVPLAY_STATUS_STOP = 0,
    HI_UNF_AVPLAY_STATUS_PLAY,
    HI_UNF_AVPLAY_STATUS_TPLAY,
    HI_UNF_AVPLAY_STATUS_PAUSE,
    HI_UNF_AVPLAY_STATUS_EOS,
    HI_UNF_AVPLAY_STATUS_SEEK ,

    HI_UNF_AVPLAY_STATUS_BUTT
} HI_UNF_AVPLAY_STATUS_E;
```

After modification:

```
typedef enum hiUNF_AVPLAY_STATUS_E
{
    HI_UNF_AVPLAY_STATUS_STOP = 0,
    HI_UNF_AVPLAY_STATUS_PREPLAY,
    HI_UNF_AVPLAY_STATUS_PLAY,
    HI_UNF_AVPLAY_STATUS_TPLAY,
    HI_UNF_AVPLAY_STATUS_PAUSE,
    HI_UNF_AVPLAY_STATUS_EOS,
    HI_UNF_AVPLAY_STATUS_SEEK ,

    HI_UNF_AVPLAY_STATUS_BUTT
} HI_UNF_AVPLAY_STATUS_E;
```

[Reason]

This data structure is modified to support the full-band function.

[Note]

After HI_UNF_AVPLAY_PreStart is called, the AVPLAY is in pre-play state. It receives and demultiplexes TSs to output audio/video ESs but does not decode and output audio/video data. Calling HI_UNF_AVPLAY_Start switches the AVPLAY from pre-play state to playback state. In this way, the program is quickly switched.

[Example]

None

HI_UNF_AVPLAY_RESET_OPT_S

[Definition]

Before modification:

```
typedef struct hiAVPLAY_RESET_OPT_S
{
    HI_U32      u32Reserved;
} HI_UNF_AVPLAY_RESET_OPT_S;
```

After modification:

```
typedef struct hiAVPLAY_RESET_OPT_S
{
    HI_U32      u32SeekPtsMs;
} HI_UNF_AVPLAY_RESET_OPT_S;
```

[Reason]

This data structure is modified to support the accurate seek function.

[Note]

This data structure is used only by the HiSilicon HiPlayer and can be ignored.

[Example]

None

HI_UNF_VIDEO_USERDATA_S

[Definition]

Before modification:

```
typedef struct hiUNF_VIDEO_USERDATA_S
{
    HI_UNF_VIDEO_BROADCAST_PROFILE_E   enBroadcastProfile;
    HI_UNF_VIDEO_USER_DATA_POSITION_E  enPositionInStream;
    HI_U32                             u32Pts;
    HI_U32                             u32SeqCnt;
    HI_U32                             u32SeqFrameCnt;
    HI_U8                              *pu8Buffer;
    HI_U32                             u32Length;
    HI_BOOL                            bBufferOverflow;
}HI_UNF_VIDEO_USERDATA_S;
```

After modification:

```
typedef struct hiUNF_VIDEO_USERDATA_S
{
    HI_UNF_VIDEO_BROADCAST_PROFILE_E   enBroadcastProfile;
    HI_UNF_VIDEO_USER_DATA_POSITION_E  enPositionInStream;
    HI_U32                             u32Pts;
    HI_U32                             u32SeqCnt;
    HI_U32                             u32SeqFrameCnt;
    HI_U8                              *pu8Buffer;
    HI_U32                             u32Length;
    HI_BOOL                            bBufferOverflow;
    HI_BOOL                            bTopFieldFirst;
}HI_UNF_VIDEO_USERDATA_S;
```

[Reason]

The top/bottom field priority flag is added to ensure the CC data can be correctly decoded.

[Note]

None

[Example]

None

2.2.3 API

2.2.3.1 New APIs

HI_UNF_AVPLAY_PreStart

[Definition]

```
HI_S32 HI_UNF_AVPLAY_PreStart(HI_HANDLE hAvplay,
HI_UNF_AVPLAY_MEDIA_CHAN_E enChn, const HI_UNF_AVPLAY_PRESTART_OPT_S
*pstPreStartOpt);
```

[Reason]

After HI_UNF_AVPLAY_PreStart is called, the AVPLAY is in pre-play state. It receives and demultiplexes TSs to output audio/video ESs but does not decode and output audio/video data. Calling HI_UNF_AVPLAY_Start switches the AVPLAY from pre-play state to playback state. In this way, the program is quickly switched.

[Note]

None

[Example]

See **sample/fullband/ fbc_xswitch.c**.

HI_UNF_AVPLAY_PreStop

[Definition]

```
HI_S32 HI_UNF_AVPLAY_PreStop(HI_HANDLE hAvplay,
HI_UNF_AVPLAY_MEDIA_CHAN_E enChn, const HI_UNF_AVPLAY_PRESTOP_OPT_S
*pstPreStopOpt);
```

[Reason]

This API is used to switch the AVPLAY state to the pre-stop state.

[Note]

This API is reserved for future extension. Currently you can call HI_UNF_AVPLAY_Stop to stop the playback.

[Example]

See **sample/fullband/ fbc_xswitch.c**.

2.3 Cipher

2.3.1 Overview

The UNF 3.2.0 Cipher module has the following change compared with UNF 3.1:

The advanced encryption standard (AES) cipher block chaining (CBC) cipher text stealing (CTS) decryption function is added.

2.3.1.1 AES CBC CTS Mode Decryption

The AES CBC CTS decryption function is added. The data structure is changed as follows:

Data Structure

HI_UNF_CIPHER_WORK_MODE_E is modified.

2.3.2 Data Structure

2.3.2.1 Modified Data Structure

HI_UNF_CIPHER_WORK_MODE_E

[Definition]

Before modification:

```
typedef enum hiHI_UNF_CIPHER_WORK_MODE_E
{
    HI_UNF_CIPHER_WORK_MODE_ECB     = 0x0,
    HI_UNF_CIPHER_WORK_MODE_CBC     = 0x1,
    HI_UNF_CIPHER_WORK_MODE_CFB     = 0x2,
    HI_UNF_CIPHER_WORK_MODE_OFB     = 0x3,
    HI_UNF_CIPHER_WORK_MODE_CTR     = 0x4,
    HI_UNF_CIPHER_WORK_MODE_BUTT    = 0x5
}HI_UNF_CIPHER_WORK_MODE_E;
```

After modification:

```
typedef enum hiHI_UNF_CIPHER_WORK_MODE_E
{
    HI_UNF_CIPHER_WORK_MODE_ECB,
    HI_UNF_CIPHER_WORK_MODE_CBC,
    HI_UNF_CIPHER_WORK_MODE_CFB,
    HI_UNF_CIPHER_WORK_MODE_OFB,
    HI_UNF_CIPHER_WORK_MODE_CTR,
    HI_UNF_CIPHER_WORK_MODE_CBC_CTS,
    HI_UNF_CIPHER_WORK_MODE_BUTT  = 0xffffffff
}HI_UNF_CIPHER_WORK_MODE_E;
```

[Description]

This data structure is modified to support the AES CBC CTS decryption. The widevine digital rights management (DRM) decryption is supported.

[Note]

None

[Example]

None

2.4 Frontend

2.4.1 Overview

The UNF 3.2.0 frontend module has the following changes compared with UNF 3.1:

- The 32-/64-/128-/256-point options are added for the sampling length of the satellite finder.
- The DVB-T/T2 tuner (CXD2861/Si2147) driver is supported.
- The DVB-T/T2 Demod (CXD2837/Hi3137) driver is supported.

2.4.1.1 Adding 32-/64-/128-/256-Point Options for the Sampling Length of the Satellite Finder

Data Structure

HI_UNF_TUNER_SAMPLE_DATALEN_E is modified.

2.4.1.2 Supporting the DVB-T/T2 Tuner (CXD2861/Si2147) Driver

Data Structure

HI_UNF_TUNER_DEV_TYPE_E is modified.

2.4.1.3 Supporting the DVB-T/T2 Demod (CXD2837/Hi3137) Driver

This feature implements the following functions:

- Supports the CXD2837/Hi3137 Demod driver.
- Automatically identifies the multi-carrier debugging mode.
- Receives the DVB-T2 base or lite channel data.
- Automatically identifies the TS sync header length.
- Configures initialization attributes of the terrestrial demodulation chip, including the crystal oscillator frequency, reset pin, and TS output mode.
- Enables/Disables the power supply for the antenna by the Hi3137.
- Searches for only DVB-T signals.
- Searches for only DVB-T2 signals.
- Searches for both DVB-T and DVB-T2 signals.
- Automatically identifies the terrestrial signal pilot mode.
- Automatically identifies whether DVB-T2 signals are mixed.
- Automatically identifies the normal or extended carrier mode.
- Automatically identifies the constellation mode.
- Automatically identifies the normal or short forward error correction (FEC) frames for DVB-T2 signals.

- Configures lock frequency parameters, including the frequency, bandwidth, DVB-T2 channel, and DVB-T stream priority.
- Identifies terrestrial signal information, including the frequency, bandwidth, modulation mode, FEC bit rate, guard interval, fast Fourier transform (FFT) size, TS priority, PLP mode, pilot mode, carrier mode, constellation mode, and FEC frame length
- Reports the program search status of the terrestrial tuner to upper-layer software, including idle, searching, completed, exited, and failed.
- Saves frequency information after program search is successful, including the frequency, bandwidth, DVB mode, PLP index number, PLP ID, common PLP ID, combination flag, and TS priority.
- Reports terrestrial program search information, including the status, progress and searched frequencies.
- Configures parameters for searching terrestrial tuner signals at a single frequency, including the frequency, bandwidth, program search mode, and DVB-T2 channel.
- Configures initialization parameters for terrestrial tuner program searching.
- Stores information about all frequencies for program searching of the terrestrial tuner.
- Counts the number of found frequencies during program searching of the terrestrial tuner.
- Identifies PLP information, including the index number, pipe ID, pipe group ID, and pipe type.
- Identifies PLP group information, including the pipe ID, shared pipe ID, combination flag, and DVB-T2 channel.
- Identifies program searching information at a single frequency for the terrestrial tuner, including the number of programs, DVB mode, DVB-T2 channel, DVB-T2 signal mode, and description of each PLP.

The data structures and APIs are changed as follows:

Data Structure

The following data structures are added:

- HI_UNF_TUNER_TER_MODE_E
- HI_UNF_TUNER_TS_SYNC_HEAD_E
- HI_UNF_TUNER_TER_ATTR_S
- HI_UNF_TUNER_TER_ANTENNA_POWER_E
- HI_UNF_TUNER_TER_SCAN_MODE_E
- HI_UNF_TUNER_TER_PILOT_PATTERN_E
- HI_UNF_TUNER_TER_CHANNEL_MODE_E
- HI_UNF_TUNER_TER_CARRIER_MODE_E
- HI_UNF_TUNER_CONSTELLATION_MODE_E
- HI_UNF_TUNER_TER_FEC_FRAME_MODE_E
- HI_UNF_TER_CONNECT_PARA_S
- HI_UNF_TUNER_TER_SCAN_STATUS_E
- HI_UNF_TUNER_TER_CHANNEL_ATTR_S
- HI_UNF_TUNER_TER_SCAN_NOTIFY_U
- HI_UNF_TUNER_TER_SCAN_ATTR_S
- HI_UNF_TUNER_TER_SCAN_PARA_S

- HI_UNF_TUNER_TER_PLP_ATTR_S
- HI_UNF_TUNER_TER_ACC_S
- HI_UNF_TUNER_TER_TPINFO_S

The following data structures are modified:

- HI_UNF_DEMOD_DEV_TYPE_E
- HI_UNF_TUNER_FE_FFT_E
- HI_UNF_TUNER_TER_SIGNALINFO_S

API

The following APIs are added:

- HI_UNF_TUNER_SetTerAttr
- HI_UNF_TUNER_SetAntennaPower
- HI_UNF_TUNER_SetCommonPLPID
- HI_UNF_TUNER_SetCommonPLPCombination
- HI_UNF_TUNER_TerScanStart
- HI_UNF_TUNER_TerScanStop
- HI_UNF_TUNER_SetPLPMode
- HI_UNF_TUNER_GetPLPId
- HI_UNF_TUNER_GetPLPGrpId

2.4.2 Data Structure

2.4.2.1 New Data Structures

HI_UNF_TUNER_TER_MODE_E

[Definition]

```
typedef enum hiUNF_TUNER_TER_MODE_E
{
   HI_UNF_TUNER_TER_MODE_BASE = 0,
   HI_UNF_TUNER_TER_MODE_LITE,
   HI_UNF_TUNER_TER_MODE_BUTT
} HI_UNF_TUNER_TER_MODE_E;
```

[Reason]

This data structure is added for receiving DVB-T/DVB-T2 signals.

[Note]

None

[Example]

See **sample/tuner/tuner_demo.c**.

HI_UNF_TUNER_TS_SYNC_HEAD_E

[Definition]

```
typedef enum hiUNF_TUNER_TS_SYNC_HEAD_E
{
    HI_UNF_TUNER_TS_SYNC_HEAD_AUTO,
    HI_UNF_TUNER_TS_SYNC_HEAD_8BIT,
    HI_UNF_TUNER_TS_SYNC_HEAD_BUTT
} HI_UNF_TUNER_TS_SYNC_HEAD_E;
```

[Reason]

This data structure is added for receiving DVB-T/DVB-T2 signals.

[Note]

None

[Example]

See **sample/tuner/tuner_demo.c**.

HI_UNF_TUNER_TER_ATTR_S

[Definition]

```
typedef struct  hiUNF_TUNER_TER_ATTR_S
{
    HI_U32                      u32DemodClk;
    HI_U32                      u32ResetGpioNo;
    HI_U16                      u16TunerMaxLPF;
    HI_U16                      u16TunerI2CClk;
    HI_UNF_TUNER_RFAGC_MODE_E     enRFAGC;
    HI_UNF_TUNER_IQSPECTRUM_MODE_E enIQSpectrum;
    HI_UNF_TUNER_TSCLK_POLAR_E    enTSClkPolar;
    HI_UNF_TUNER_TS_FORMAT_E      enTSFormat;
    HI_UNF_TUNER_TS_SERIAL_PIN_E   enTSSerialPIN;
    HI_UNF_TUNER_TS_SYNC_HEAD_E    enTSSyncHead;
} HI_UNF_TUNER_TER_ATTR_S;
```

[Reason]

This data structure is added for receiving DVB-T/DVB-T2 signals.

[Note]

None

[Example]

See **sample/tuner/tuner_demo.c**.

HI_UNF_TUNER_TER_ANTENNA_POWER_E

[Definition]

```
typedef enum hiUNF_TUNER_TER_ANTENNA_POWER_E
{
    HI_UNF_TUNER_TER_ANTENNA_POWER_OFF,
    HI_UNF_TUNER_TER_ANTENNA_POWER_ON,
    HI_UNF_TUNER_TER_ANTENNA_POWER_BUTT
} HI_UNF_TUNER_TER_ANTENNA_POWER_E;
```

[Reason]

This data structure is added for receiving DVB-T/DVB-T2 signals.

[Note]

None

[Example]

See **sample/tuner/tuner_demo.c**.

HI_UNF_TUNER_TER_SCAN_MODE_E

[Definition]

```
typedef enum hiUNF_TUNER_TER_SCAN_MODE_E
{
    HI_UNF_TUNER_TER_SCAN_DVB_T2 = 0,
    HI_UNF_TUNER_TER_SCAN_DVB_T,
    HI_UNF_TUNER_TER_SCAN_DVB_T_T2_ALL,
    HI_UNF_TUNER_TER_SCAN_DVB_T_T2_BUTT
} HI_UNF_TUNER_TER_SCAN_MODE_E;
```

[Reason]

This data structure is added for receiving DVB-T/DVB-T2 signals.

[Note]

None

[Example]

See **sample/tuner/tuner_demo.c**.

HI_UNF_TUNER_TER_PILOT_PATTERN_E

[Definition]

```
typedef enum hiUNF_TUNER_TER_PILOT_PATTERN_E
{
    HI_UNF_TUNER_T2_PILOT_PATTERN_PP1=0,
    HI_UNF_TUNER_T2_PILOT_PATTERN_PP2,
```

```
    HI_UNF_TUNER_T2_PILOT_PATTERN_PP3,
    HI_UNF_TUNER_T2_PILOT_PATTERN_PP4,
    HI_UNF_TUNER_T2_PILOT_PATTERN_PP5,
    HI_UNF_TUNER_T2_PILOT_PATTERN_PP6,
    HI_UNF_TUNER_T2_PILOT_PATTERN_PP7,
    HI_UNF_TUNER_T2_PILOT_PATTERN_PP8,
    HI_UNF_TUNER_T2_PILOT_PATTERN_BUTT
} HI_UNF_TUNER_TER_PILOT_PATTERN_E;
```

[Reason]

This data structure is added for receiving DVB-T/DVB-T2 signals.

[Note]

None

[Example]

See **sample/tuner/tuner_demo.c**.

HI_UNF_TUNER_TER_CHANNEL_MODE_E

[Definition]

```
typedef enum hiUNF_TUNER_TER_CHANNEL_MODE_E
{
    HI_UNF_TUNER_TER_PURE_CHANNEL = 0,
    HI_UNF_TUNER_TER_MIXED_CHANNEL,
    HI_UNF_TUNER_TER_CHANNEL_MODE_BUTT
} HI_UNF_TUNER_TER_CHANNEL_MODE_E;
```

[Reason]

This data structure is added for receiving DVB-T/DVB-T2 signals.

[Note]

None

[Example]

See **sample/tuner/tuner_demo.c**.

HI_UNF_TUNER_TER_CARRIER_MODE_E

[Definition]

```
typedef enum hiUNF_TUNER_TER_CARRIER_MODE_E
{
    HI_UNF_TUNER_TER_EXTEND_CARRIER = 0,
    HI_UNF_TUNER_TER_NORMAL_CARRIER,
    HI_UNF_TUNER_TER_CARRIER_MODE_BUTT
} HI_UNF_TUNER_TER_CARRIER_MODE_E;
```

[Reason]

This data structure is added for receiving DVB-T/DVB-T2 signals.

[Note]

None

[Example]

See **sample/tuner/tuner_demo.c**.

HI_UNF_TUNER_CONSTELLATION_MODE_E

[Definition]

```
typedef enum hiUNF_TUNER_CONSTELLATION_MODE_E
{
    HI_UNF_TUNER_CONSTELLATION_STANDARD = 0,
    HI_UNF_TUNER_CONSTELLATION_ROTATION,
    HI_UNF_TUNER_CONSTELLATION_MODE_BUTT
} HI_UNF_TUNER_CONSTELLATION_MODE_E;
```

[Reason]

This data structure is added for receiving DVB-T/DVB-T2 signals.

[Note]

None

[Example]

See **sample/tuner/tuner_demo.c**.

HI_UNF_TUNER_TER_FEC_FRAME_MODE_E

[Definition]

```
typedef enum hiUNF_TUNER_TER_FEC_FRAME_MODE_E
{
    HI_UNF_TUNER_TER_FEC_FRAME_NORMAL = 0,
    HI_UNF_TUNER_TER_FEC_FRAME_SHORT,
    HI_UNF_TUNER_TER_FEC_FRAME_MODE_BUTT
} HI_UNF_TUNER_TER_FEC_FRAME_MODE_E;
```

[Reason]

This data structure is added for receiving DVB-T/DVB-T2 signals.

[Note]

None

[Example]

See **sample/tuner/tuner_demo.c**.

HI_UNF_TER_CONNECT_PARA_S

[Definition]

```
typedef struct  hiUNF_TER_CONNECT_PARA_S
{
    HI_U32                  u32Freq;
    HI_U32                  u32BandWidth;
    HI_UNF_MODULATION_TYPE_E    enModType;
    HI_BOOL                 bReverse;
    HI_UNF_TUNER_TER_MODE_E     enChannelMode;
    HI_UNF_TUNER_TS_PRIORITY_E  enDVBTPrio;
} HI_UNF_TER_CONNECT_PARA_S;
```

[Reason]

This data structure is added for receiving DVB-T/DVB-T2 signals.

[Note]

None

[Example]

See **sample/tuner/tuner_demo.c**.

HI_UNF_TUNER_TER_SCAN_STATUS_E

[Definition]

```
typedef enum hiUNF_TUNER_TER_SCAN_STATUS_E
{
    HI_UNF_TUNER_TER_SCAN_STATUS_IDLE,
    HI_UNF_TUNER_TER_SCAN_STATUS_SCANNING,
    HI_UNF_TUNER_TER_SCAN_STATUS_FINISH,
    HI_UNF_TUNER_TER_SCAN_STATUS_QUIT,
    HI_UNF_TUNER_TER_SCAN_STATUS_FAIL,
    HI_UNF_TUNER_TER_SCAN_STATUS_BUTT
} HI_UNF_TUNER_TER_SCAN_STATUS_E;
```

[Reason]

This data structure is added for receiving DVB-T/DVB-T2 signals.

[Note]

None

[Example]

See **sample/tuner/tuner_demo.c**.

HI_UNF_TUNER_TER_CHANNEL_ATTR_S

[Definition]

```
typedef struct hiUNF_TUNER_TER_CHANNEL_ATTR_S
{
    HI_U32 u32Frequency;
    HI_U32 u32BandWidth;
    HI_U8  u8DVBTMode;
    HI_U8  u8PlpIndex;
    HI_U8  u8PlpId;
    HI_U8  u8CommId;
    HI_U8  u8Combination;
    HI_UNF_TUNER_TER_MODE_E    enChannelMode;
    HI_UNF_TUNER_TS_PRIORITY_E  enTSPri;
} HI_UNF_TUNER_TER_CHANNEL_ATTR_S;
```

[Reason]

This data structure is added for receiving DVB-T/DVB-T2 signals.

[Note]

None

[Example]

See **sample/tuner/tuner_demo.c**.

HI_UNF_TUNER_TER_SCAN_NOTIFY_U

[Definition]

```
typedef union hiUNF_TUNER_TER_SCAN_NOTIFY_U
{
    HI_UNF_TUNER_TER_SCAN_STATUS_E* penStatus;
    HI_U16*                      pu16ProgressPercent;
    HI_UNF_TUNER_TER_CHANNEL_ATTR_S *pstResult;
} HI_UNF_TUNER_TER_SCAN_NOTIFY_U;
```

[Reason]

This data structure is added for receiving DVB-T/DVB-T2 signals.

[Note]

None

[Example]

See **sample/tuner/tuner_demo.c**.

HI_UNF_TUNER_TER_SCAN_ATTR_S

[Definition]

```
typedef struct hiUNF_TUNER_TER_SCAN_ATTR_S
{
```

```
    HI_U32 u32Frequency;
    HI_U32 u32BandWidth;
    HI_UNF_TUNER_TER_SCAN_MODE_E enScanMode;
    HI_BOOL  bScanLite;
    HI_VOID (*pfnEVTNotify)(HI_U32 u32TunerId,
HI_UNF_TUNER_TER_SCAN_STATUS_E enEVT, HI_UNF_TUNER_TER_SCAN_NOTIFY_U *
punNotify);
}HI_UNF_TUNER_TER_SCAN_ATTR_S;
```

[Reason]

This data structure is added for receiving DVB-T/DVB-T2 signals.

[Note]

None

[Example]

See **sample/tuner/tuner_demo.c**.

HI_UNF_TUNER_TER_SCAN_PARA_S

[Definition]

```
typedef struct hiUNF_TUNER_TER_SCAN_PARA_S
{
    HI_UNF_TUNER_TER_SCAN_ATTR_S stTer;
    HI_UNF_TUNER_TER_CHANNEL_ATTR_S enChanArray[TER_MAX_TP];
    HI_U32 u32ChanNum;
}HI_UNF_TUNER_TER_SCAN_PARA_S;
```

[Reason]

This data structure is added for receiving DVB-T/DVB-T2 signals.

[Note]

None

[Example]

See **sample/tuner/tuner_demo.c**.

HI_UNF_TUNER_TER_PLP_ATTR_S

[Definition]

```
typedef struct hiUNF_TUNER_TER_PLP_ATTR_S
{
    HI_U8                    u8PlpIndex;
    HI_U8                    u8PlpId;
    HI_U8                    u8PlpGrpId;
    HI_UNF_TUNER_T2_PLP_TYPE_E  enPlpType;
```

```
} HI_UNF_TUNER_TER_PLP_ATTR_S;
```

[Reason]

This data structure is added for receiving DVB-T/DVB-T2 signals.

[Note]

None

[Example]

See **sample/tuner/tuner_demo.c**.

HI_UNF_TUNER_TER_ACC_S

[Definition]

```
typedef struct hiUNF_TUNER_TER_ACC_S
{
    HI_U8                   u8PlpId;
    HI_U8                   u8CommPlpId;
    HI_U8                   u8Combination;
    HI_UNF_TUNER_TER_MODE_E    enChannelAttr;
} HI_UNF_TUNER_TER_ACC_S;
```

[Reason]

This data structure is added for receiving DVB-T/DVB-T2 signals.

[Note]

None

[Example]

See **sample/tuner/tuner_demo.c**.

HI_UNF_TUNER_TER_TPINFO_S

[Definition]

```
typedef struct hiUNF_TUNER_TER_TPINFO_S
{
    HI_U8                       u8ProgNum;
    HI_U8                       u8DVBTMode;
    HI_U8                       u8DVBTHier;
    HI_UNF_TUNER_TER_MODE_E        enChannelAttr;
    HI_UNF_TUNER_TER_CHANNEL_MODE_E enChannelMode;
    HI_UNF_TUNER_TER_PLP_ATTR_S    enPlpAttr[16];
} HI_UNF_TUNER_TER_TPINFO_S;
```

[Reason]

This data structure is added for receiving DVB-T/DVB-T2 signals.

[Note]

None

[Example]

See **sample/tuner/tuner_demo.c**.

2.4.2.2 Modified Data Structures

HI_UNF_TUNER_SAMPLE_DATALEN_E

[Definition]

Before modification:

```
typedef enum hiUNF_TUNER_SAMPLE_DATALEN_E
{
    HI_UNF_TUNER_SAMPLE_DATALEN_512,
    HI_UNF_TUNER_SAMPLE_DATALEN_1024,
    HI_UNF_TUNER_SAMPLE_DATALEN_2048,
    HI_UNF_TUNER_SAMPLE_DATALEN_BUTT
} HI_UNF_TUNER_SAMPLE_DATALEN_E;
```

After modification:

```
typedef enum hiUNF_TUNER_SAMPLE_DATALEN_E
{
    HI_UNF_TUNER_SAMPLE_DATALEN_32,
    HI_UNF_TUNER_SAMPLE_DATALEN_64,
    HI_UNF_TUNER_SAMPLE_DATALEN_128,
    HI_UNF_TUNER_SAMPLE_DATALEN_256,
    HI_UNF_TUNER_SAMPLE_DATALEN_512,
    HI_UNF_TUNER_SAMPLE_DATALEN_1024,
    HI_UNF_TUNER_SAMPLE_DATALEN_2048,
    HI_UNF_TUNER_SAMPLE_DATALEN_BUTT
} HI_UNF_TUNER_SAMPLE_DATALEN_E;
```

[Reason]

This data structure is modified to add the 32-/64-/128-/256-point options for the sampling length of the satellite finder.

[Note]

None

[Example]

See **sample/tuner/tuner_demo.c**.

HI_UNF_TUNER_DEV_TYPE_E

[Definition]

Before modification:

```
typedef enum    hiUNF_TUNER_DEV_TYPE_E
{
    HI_UNF_TUNER_DEV_TYPE_XG_3BL,
    HI_UNF_TUNER_DEV_TYPE_CD1616,
    HI_UNF_TUNER_DEV_TYPE_ALPS_TDAE,
    HI_UNF_TUNER_DEV_TYPE_TDCC,
    HI_UNF_TUNER_DEV_TYPE_TDA18250,
    HI_UNF_TUNER_DEV_TYPE_CD1616_DOUBLE,
    HI_UNF_TUNER_DEV_TYPE_MT2081,
    HI_UNF_TUNER_DEV_TYPE_TMX7070X,
    HI_UNF_TUNER_DEV_TYPE_R820C,
    HI_UNF_TUNER_DEV_TYPE_MXL203,
    HI_UNF_TUNER_DEV_TYPE_AV2011,
    HI_UNF_TUNER_DEV_TYPE_SHARP7903,
    HI_UNF_TUNER_DEV_TYPE_MXL101,
    HI_UNF_TUNER_DEV_TYPE_MXL603,
    HI_UNF_TUNER_DEV_TYPE_IT9170,
    HI_UNF_TUNER_DEV_TYPE_IT9133,
    HI_UNF_TUNER_DEV_TYPE_TDA6651,
    HI_UNF_TUNER_DEV_TYPE_TDA18250B,
    HI_UNF_TUNER_DEV_TYPE_M88TS2022,
    HI_UNF_TUNER_DEV_TYPE_RDA5815,
    HI_UNF_TUNER_DEV_TYPE_MXL254,
    HI_UNF_TUNER_DEV_TYPE_BUTT,
}HI_UNF_TUNER_DEV_TYPE_E ;
```

After modification:

```
typedef enum    hiUNF_TUNER_DEV_TYPE_E
{
    HI_UNF_TUNER_DEV_TYPE_XG_3BL,
    HI_UNF_TUNER_DEV_TYPE_CD1616,
    HI_UNF_TUNER_DEV_TYPE_ALPS_TDAE,
    HI_UNF_TUNER_DEV_TYPE_TDCC,
    HI_UNF_TUNER_DEV_TYPE_TDA18250,
    HI_UNF_TUNER_DEV_TYPE_CD1616_DOUBLE,
    HI_UNF_TUNER_DEV_TYPE_MT2081,
    HI_UNF_TUNER_DEV_TYPE_TMX7070X,
    HI_UNF_TUNER_DEV_TYPE_R820C,
    HI_UNF_TUNER_DEV_TYPE_MXL203,
    HI_UNF_TUNER_DEV_TYPE_AV2011,
    HI_UNF_TUNER_DEV_TYPE_SHARP7903,
    HI_UNF_TUNER_DEV_TYPE_MXL101,
    HI_UNF_TUNER_DEV_TYPE_MXL603,
```

```
    HI_UNF_TUNER_DEV_TYPE_IT9170,
    HI_UNF_TUNER_DEV_TYPE_IT9133,
    HI_UNF_TUNER_DEV_TYPE_TDA6651,
    HI_UNF_TUNER_DEV_TYPE_TDA18250B,
    HI_UNF_TUNER_DEV_TYPE_M88TS2022,
    HI_UNF_TUNER_DEV_TYPE_RDA5815,
    HI_UNF_TUNER_DEV_TYPE_MXL254,
    HI_UNF_TUNER_DEV_TYPE_CXD2861,
    HI_UNF_TUNER_DEV_TYPE_SI2147,
    HI_UNF_TUNER_DEV_TYPE_BUTT
} HI_UNF_TUNER_DEV_TYPE_E ;
```

[Reason]

This data structure is modified to support the CXD2861/Si2147 tuner driver.

[Note]

CXD2861 works with CXD2837 to receive DVB-T/-T2 signals, and Si2147 works with Hi3137 to receive DVB-T/-T2 signals.

[Example]

See **sample/tuner/tuner_demo.c**.

HI_UNF_DEMOD_DEV_TYPE_E

[Definition]

Before modification:

```
typedef enum   hiUNF_DEMOD_DEV_TYPE_E
{
    HI_UNF_DEMOD_DEV_TYPE_NONE,
    HI_UNF_DEMOD_DEV_TYPE_3130I= 0x100,
    HI_UNF_DEMOD_DEV_TYPE_3130E,
    HI_UNF_DEMOD_DEV_TYPE_J83B,
    HI_UNF_DEMOD_DEV_TYPE_AVL6211,
    HI_UNF_DEMOD_DEV_TYPE_MXL101,
    HI_UNF_DEMOD_DEV_TYPE_MN88472,
    HI_UNF_DEMOD_DEV_TYPE_IT9170,
    HI_UNF_DEMOD_DEV_TYPE_IT9133,
    HI_UNF_DEMOD_DEV_TYPE_3136,
    HI_UNF_DEMOD_DEV_TYPE_3136I,
    HI_UNF_DEMOD_DEV_TYPE_MXL254,
    HI_UNF_DEMOD_DEV_TYPE_BUTT,
}HI_UNF_DEMOD_DEV_TYPE_E ;
```

After modification:

```
typedef enum   hiUNF_DEMOD_DEV_TYPE_E
```

```
{
    HI_UNF_DEMOD_DEV_TYPE_NONE,
    HI_UNF_DEMOD_DEV_TYPE_3130I = 0x100,
    HI_UNF_DEMOD_DEV_TYPE_3130E,
    HI_UNF_DEMOD_DEV_TYPE_J83B,
    HI_UNF_DEMOD_DEV_TYPE_AVL6211,
    HI_UNF_DEMOD_DEV_TYPE_MXL101,
    HI_UNF_DEMOD_DEV_TYPE_MN88472,
    HI_UNF_DEMOD_DEV_TYPE_IT9170,
    HI_UNF_DEMOD_DEV_TYPE_IT9133,
    HI_UNF_DEMOD_DEV_TYPE_3136,
    HI_UNF_DEMOD_DEV_TYPE_3136I,
    HI_UNF_DEMOD_DEV_TYPE_MXL254,
    HI_UNF_DEMOD_DEV_TYPE_CXD2837,
    HI_UNF_DEMOD_DEV_TYPE_3137,
    HI_UNF_DEMOD_DEV_TYPE_BUTT
} HI_UNF_DEMOD_DEV_TYPE_E;
```

[Reason]

This data structure is modified to support the CXD2837/Hi3137 Demod driver.

[Note]

CXD2861 works with CXD2837 to receive DVB-T/-T2 signals, and Si2147 works with Hi3137 to receive DVB-T/-T2 signals.

[Example]

See **sample/tuner/tuner_demo.c**.

HI_UNF_TUNER_FE_FFT_E

[Definition]

Before modification:

```
typedef enum hiUNF_TUNER_FE_FFT_E
{
    HI_UNF_TUNER_FE_FFT_DEFAULT = 0,
    HI_UNF_TUNER_FE_FFT_1K ,
    HI_UNF_TUNER_FE_FFT_2K ,
    HI_UNF_TUNER_FE_FFT_4K ,
    HI_UNF_TUNER_FE_FFT_8K ,
    HI_UNF_TUNER_FE_FFT_16K ,
    HI_UNF_TUNER_FE_FFT_32K ,
    HI_UNF_TUNER_FE_FFT_BUTT ,
}HI_UNF_TUNER_FE_FFT_E;
```

After modification:

```
typedef enum hiUNF_TUNER_FE_FFT_E
{
    HI_UNF_TUNER_FE_FFT_DEFAULT = 0,
    HI_UNF_TUNER_FE_FFT_1K ,
    HI_UNF_TUNER_FE_FFT_2K ,
    HI_UNF_TUNER_FE_FFT_4K ,
    HI_UNF_TUNER_FE_FFT_8K ,
    HI_UNF_TUNER_FE_FFT_16K ,
    HI_UNF_TUNER_FE_FFT_32K ,
    HI_UNF_TUNER_FE_FFT_64K ,
    HI_UNF_TUNER_FE_FFT_BUTT
} HI_UNF_TUNER_FE_FFT_E;
```

[Reason]

This data structure is modified to add the 64 KB FFT size.

[Note]

None

[Example]

See **sample/tuner/tuner_demo.c**

HI_UNF_TUNER_TER_SIGNALINFO_S

[Definition]

Before modification:

```
typedef struct  hiUNF_TUNER_TER_SIGNALINFO_S
{
    HI_U32                    u32Freq;
    HI_U32                    u32BandWidth;
    HI_UNF_MODULATION_TYPE_E      enModType;
    HI_UNF_TUNER_FE_FECRATE_E     enFECRate;
    HI_UNF_TUNER_FE_GUARD_INTV_E enGuardIntv;
    HI_UNF_TUNER_FE_FFT_E enFFTMode;
    HI_UNF_TUNER_FE_HIERARCHY_E enHierMod;
    HI_UNF_TUNER_TS_PRIORITY_E enTsPriority;
} HI_UNF_TUNER_TER_SIGNALINFO_S;
```

After modification:

```
typedef struct  hiUNF_TUNER_TER_SIGNALINFO_S
{
    HI_U32                          u32Freq;
    HI_U32                          u32BandWidth;
    HI_UNF_MODULATION_TYPE_E           enModType;
    HI_UNF_TUNER_FE_FECRATE_E          enFECRate;
```

```
    HI_UNF_TUNER_FE_FECRATE_E          enLowPriFECRate;
    HI_UNF_TUNER_FE_GUARD_INTV_E       enGuardIntv;
    HI_UNF_TUNER_FE_FFT_E              enFFTMode;
    HI_UNF_TUNER_FE_HIERARCHY_E        enHierMod;
    HI_UNF_TUNER_TS_PRIORITY_E         enTsPriority;
    HI_UNF_TUNER_T2_PLP_TYPE_E         enPLPType;
    HI_UNF_TUNER_TER_PILOT_PATTERN_E   enPilotPattern;
    HI_UNF_TUNER_TER_CARRIER_MODE_E    enCarrierMode;
    HI_UNF_TUNER_CONSTELLATION_MODE_E  enConstellationMode;
    HI_UNF_TUNER_TER_FEC_FRAME_MODE_E  enFECFrameMode;
} HI_UNF_TUNER_TER_SIGNALINFO_S;
```

[Reason]

This data structure is modified to support the low-priority stream FEC rate, PLP type, pilot mode, carrier mode, constellation mode, and frame length mode.

[Note]

None

[Example]

See **sample/tuner/tuner_demo.c**.

2.4.3 API

2.4.3.1 New APIs

HI_UNF_TUNER_SetTerAttr

[Definition]

```
HI_S32 HI_UNF_TUNER_SetTerAttr(HI_U32  u32tunerId , const
HI_UNF_TUNER_TER_ATTR_S *pstTerTunerAttr);
```

[Reason]

A new feature is added.

[Note]

This API is called when the terrestrial tuner is initialized.

[Example]

None

HI_UNF_TUNER_SetPLPMode

[Definition]

```
HI_S32 HI_UNF_TUNER_SetPLPMode(HI_U32 u32TunerId, HI_U8 u8Mode);
```

[Reason]

A new feature is added.

[Note]

This API is called when the terrestrial tuner receives DVB-T2 signals.

[Example]

None

HI_UNF_TUNER_SetCommonPLPID

[Definition]

```
HI_S32 HI_UNF_TUNER_SetCommonPLPID(HI_U32 u32TunerId, HI_U8 u8PLPID);
```

[Reason]

A new feature is added.

[Note]

This API is called when the terrestrial tuner receives DVB-T2 signals.

[Example]

None

HI_UNF_TUNER_SetCommonPLPCombination

[Definition]

```
HI_S32 HI_UNF_TUNER_SetCommonPLPCombination(HI_U32 u32TunerId, HI_U8
u8PLPID);
```

[Reason]

A new feature is added.

[Note]

This API is called when the terrestrial tuner receives DVB-T2 signals.

[Example]

None

HI_UNF_TUNER_GetPLPId

[Definition]

```
HI_S32 HI_UNF_TUNER_GetPLPId(HI_U32 u32TunerId, HI_U8 *pu8PLPId);
```

[Reason]

A new feature is added.

[Note]

This API is called when the terrestrial tuner receives DVB-T2 signals.

[Example]

None

HI_UNF_TUNER_GetPLPGrpId

[Definition]

```
HI_S32 HI_UNF_TUNER_GetPLPGrpId(HI_U32 u32TunerId, HI_U8 *pu8PLPGrpId);
```

[Reason]

A new feature is added.

[Note]

This API is called when the terrestrial tuner receives DVB-T2 signals.

[Example]

None

HI_UNF_TUNER_SetAntennaPower

[Definition]

```
HI_S32 HI_UNF_TUNER_SetAntennaPower(HI_U32 u32TunerId,
HI_UNF_TUNER_TER_ANTENNA_POWER_E enPower);
```

[Reason]

A new feature is added.

[Note]

This API is called when the terrestrial tuner receives signals.

[Example]

None

HI_UNF_TUNER_TerScanStart

[Definition]

```
HI_S32 HI_UNF_TUNER_SetTerAttr(HI_U32  u32tunerId , const
HI_UNF_TUNER_TER_ATTR_S *pstTerTunerAttr);
```

[Reason]

A new feature is added.

[Note]

This API is called when the terrestrial tuner receives signals.

[Example]

None

HI_UNF_TUNER_TerScanStop

[Definition]

```
HI_S32 HI_UNF_TUNER_TerScanStop( HI_U32 u32TunerId);
```

[Reason]

A new feature is added.

[Note]

This API is called when the terrestrial tuner receives signals.

[Example]

None

2.5 HDMI

2.5.1 Overview

The UNF 3.2.0 HDMI module has the following changes compared with UNF 3.1:

- The function of obtaining the high-bandwidth digital content protection (HDCP) handshake status is added.
- The function of registering CEC callback functions is added.

2.5.1.1 Obtaining the HDCP Handshake Status

The HDCP handshake status and BKSV can be obtained for Miracast+HDCP 2.2. The data structure is changed as follows:

Data Structure

HI_UNF_HDMI_STATUS_S is modified.

2.5.1.2 Registering the CEC Callback Function

CEC information can be processed by using the callback function, which simplifies the development process. The data structure and APIs are changed as follows:

Data Structure

HI_UNF_HDMI_CECCALLBACK is added.

API

The following APIs are added:

- HI_UNF_HDMI_RegCECCallBackFunc
- HI_UNF_HDMI_UnRegCECCallBackFunc

2.5.2 Data Structure

2.5.2.1 New Data Structure

HI_UNF_HDMI_CECCALLBACK

[Definition]

```
typedef HI_VOID (*HI_UNF_HDMI_CECCALLBACK)(HI_UNF_HDMI_ID_E enHdmi,
HI_UNF_HDMI_CEC_CMD_S *pstCECCmd, HI_VOID *pData);
```

[Reason]

Unified register event types need to be defined for the CEC callback function.

[Note]

None

[Example]

None

2.5.2.2 Modified Data Structure

HI_UNF_HDMI_STATUS_S

[Definition]

Before modification:

```
typedef struct hiUNF_HDMI_STATUS_S
{
    HI_BOOL           bConnected;
    HI_BOOL           bSinkPowerOn;
}HI_UNF_HDMI_STATUS_S;
```

After modification:

```
typedef struct hiUNF_HDMI_STATUS_S
{
    HI_BOOL           bConnected;
    HI_BOOL           bSinkPowerOn;
    HI_BOOL           bAuthed;
    HI_U8             u8Bksv[5];
}HI_UNF_HDMI_STATUS_S;
```

[Reason]

BKSV must be obtained for the Miracast+HDCP 2.2 function.

[Note]

Before using u8BKSV, check whether authentication (bAuthed) is complete. BKSV is valid only when HDCP authentication is complete.

[Example]

None

2.5.3 API

2.5.3.1 New APIs

HI_UNF_HDMI_RegCECCallBackFunc

[Definition]

```
HI_S32 HI_UNF_HDMI_RegCECCallBackFunc(HI_UNF_HDMI_ID_E enHdmi,
HI_UNF_HDMI_CECCALLBACK pCECCallback);
```

[Reason]

This API is added to register the CEC event.

[Note]

The CEC event does not support multiple processes at the same time currently. If it is registered for multiple times, only the last registration is valid. In addition, HI_UNF_HDMI_GetCECCommand and the CEC event registration function cannot be used at the same time. Otherwise, information may be lost.

[Example]

None

HI_UNF_HDMI_UnRegCECCallBackFunc

[Definition]

```
HI_S32 HI_UNF_HDMI_UnRegCECCallBackFunc(HI_UNF_HDMI_ID_E enHdmi,
HI_UNF_HDMI_CECCALLBACK pCECCallback);
```

[Reason]

This API is added to deregister the CEC event.

[Note]

None

[Example]

None

2.6 VO

2.6.1 Overview

The UNF 3.2.0 VO module has the following changes compared with UNF 3.1:

- The function of creating a window that uses the physical coordinate system is added.

- The function of obtaining window freeze status is added.
- The function of obtaining the window quick output status is added.
- The function of obtaining the window field playback mode is added.
- The function of obtaining the 3D output depth of field of a window is added.

2.6.1.1 Creating a Window That Uses the Physical Coordinate System

The physical coordinate system can be specified as the coordinate system of a window. The API is changed as follows:

API

HI_UNF_VO_CreateWindowExt is added.

2.6.1.2 Obtaining the Window Freeze Status

The window freeze status can be obtained. This function is related to HI_UNF_VO_FreezeWindow. The API is changed as follows:

API

HI_UNF_VO_GetWindowFreezeStatus is added.

2.6.1.3 Obtaining the Window Quick Output Status

The window quick output status can be obtained. This function is related to HI_UNF_VO_SetQuickOutputEnable. The API is changed as follows:

API

HI_UNF_VO_GetQuickOutputStatus is added.

2.6.1.4 Obtaining the Window Field Playback Mode

The window field playback mode can be obtained. This function is related to HI_UNF_VO_SetWindowFieldMode. The API is changed as follows:

API

HI_UNF_VO_GetWindowFieldMode is added.

2.6.1.5 Obtaining the 3D Output Depth of Field of a Window

The 3D output depth of field of a window can be obtained. This function is related to HI_UNF_VO_SetStereoDetpth. The API is changed as follows:

API

HI_UNF_VO_GetStereoDetpth is added.

2.6.2 API

2.6.2.1 New APIs

HI_UNF_VO_CreateWindowExt

[Definition]

```
HI_S32 HI_UNF_VO_CreateWindowExt(const HI_UNF_WINDOW_ATTR_S* pWinAttr,
                                 HI_HANDLE *phWindow,
                                 HI_BOOL bVirtScreen);
```

[Reason]

This API is added to create a window that uses the physical coordinate system.

[Note]

If a window is created by using this API, it uses the physical coordinate system during its life cycle.

[Example]

None

HI_UNF_VO_GetWindowFreezeStatus

[Definition]

```
HI_S32 HI_UNF_VO_GetWindowFreezeStatus(HI_HANDLE hWindow, HI_BOOL
*pbEnable, HI_UNF_WINDOW_FREEZE_MODE_E *penWinFreezeMode);
```

[Reason]

This API is added to obtain the window freeze status.

[Note]

None

[Example]

None

HI_UNF_VO_GetQuickOutputStatus

[Definition]

```
HI_S32 HI_UNF_VO_GetQuickOutputStatus(HI_HANDLE hWindow, HI_BOOL
*pbQuickOutputEnable);
```

[Reason]

This API is added to obtain the window quick output status.

[Note]

None

[Example]

None

HI_UNF_VO_GetWindowFieldMode

[Definition]

```
HI_S32 HI_UNF_VO_GetWindowFieldMode(HI_HANDLE hWindow, HI_BOOL *pbEnable);
```

[Reason]

This API is added to obtain the window field playback mode.

[Note]

None

[Example]

None

HI_UNF_VO_GetStereoDetpth

[Definition]

```
HI_S32 HI_UNF_VO_GetStereoDetpth(HI_HANDLE hWindow, HI_S32 *ps32Depth);
```

[Reason]

This API is added to obtain the 3D output depth of field of a window.

[Note]

None

[Example]

None

2.7 PMOC

2.7.1 Overview

The UNF 3.2.0 PMOC module has the following changes compared with UNF 3.1:

The function of obtaining the standby wakeup attributes is added.

2.7.1.1 Obtaining the Standby Wakeup Attributes

The API for obtaining the standby wakeup attributes is added so that the previous attributes are not overwritten when a standby wakeup attribute is configured.

API

HI_UNF_PMOC_GetWakeUpAttr is added.

2.7.2 API

2.7.2.1 New API

HI_UNF_PMOC_GetWakeUpAttr

[Definition]

```
HI_S32 HI_UNF_PMOC_GetWakeUpAttr(HI_UNF_PMOC_WKUP_S_PTR pstAttr);
```

[Reason]

This API is added to obtain the standby wakeup attributes that have been configured in the driver.

[Note]

None

[Example]

None

2.8 PQ

2.8.1 Overview

The UNF 3.2.0 PQ module has the following changes compared with UNF 3.1:

- The function of setting default parameters for access specifications tests is added.
- The function of saving PQ parameters is added.

2.8.1.1 Setting Default Parameters for Access Specifications Tests

The default values for some PQ parameters can be configured as required by the access specifications tests. The API is changed as follows:

API

HI_UNF_PQ_SetDefaultParam is added.

2.8.1.2 Saving PQ Parameters

PQ parameters need to be saved when the power is off. The API is changed as follows:

API

HI_UNF_PQ_UpdatePqParam is added.

2.8.2 API

2.8.2.1 New APIs

HI_UNF_PQ_SetDefaultParam

[Definition]

```
HI_S32 HI_UNF_PQ_SetDefaultParam(HI_VOID);
```

[Reason]

This API is added to set default PQ parameters for access specifications tests.

[Note]

- This API can be used properly only after the PQ parameter partition is burnt.
- You are advised not to call this API on boards released after mass production.

[Example]

None

HI_UNF_PQ_UpdatePqParam

[Definition]

```
HI_S32 HI_UNF_PQ_UpdatePqParam(HI_VOID);
```

[Reason]

This API is added to update the PQ configuration information and save PQ parameters to the flash memory.

[Note]

None

[Example]

None

2.9 PVR

2.9.1 Overview

The UNF 3.2.0 PVR module has the following changes compared with UNF 3.1:

The offset address of the frame being played in the file is added.

2.9.1.1 Adding the Offset Address of the Frame Being Played in the File

The offset address of the frame being played in the file is added in the HI_UNF_PVR_DATA_ATTR_S structure as a member. This function is required by the advanced CA. The data structure is changed as follows:

Data Structure

HI_UNF_PVR_DATA_ATTR_S is modified.

2.9.2 Data Structure

2.9.2.1 Modified Data Structure

HI_UNF_PVR_DATA_ATTR_S

[Definition]

Before modification:

```
typedef struct
{
    HI_U32      u32ChnID;
    HI_CHAR     CurFileName[PVR_MAX_FILENAME_LEN];
    HI_CHAR     IdxFileName[PVR_MAX_FILENAME_LEN+5];
    HI_U64      u64FileStartPos;
    HI_U64      u64FileEndPos;
    HI_U64      u64GlobalOffset;
} HI_UNF_PVR_DATA_ATTR_S;
```

After modification:

```
typedef struct
{
    HI_U32      u32ChnID;
    HI_CHAR     CurFileName[PVR_MAX_FILENAME_LEN];
    HI_CHAR     IdxFileName[PVR_MAX_FILENAME_LEN+5];
    HI_U64      u64FileStartPos;
    HI_U64      u64FileEndPos;
    HI_U64      u64GlobalOffset;
    HI_U64      u64FileReadOffset;
} HI_UNF_PVR_DATA_ATTR_S;
```

[Description]

The offset address of the frame being played in the file is added to the data structure.

[Note]

None

[Example]

None

2.10 Sound

2.10.1 Overview

The UNF 3.2.0 sound module has the following changes compared with UNF 3.1:

- The function of setting and obtaining the SPDIF serial copy management system (SCMS) mode is added.
- The function of setting and obtaining the mute status of all tracks is added.
- The function of setting and obtaining the track channel mode is added.
- The function of obtaining the track delay is added.
- The function of setting and obtaining the cast volume is added.
- The function of setting and obtaining the cast mute status is added.

2.10.1.1 Setting and Obtaining the SPDIF SCMS Mode

Data Structure

HI_UNF_SND_SPDIF_SCMSMODE_E is added.

API

The following APIs are added:

- HI_UNF_SND_SetSpdifSCMSMode
- HI_UNF_SND_GetSpdifSCMSMode

2.10.1.2 Setting and Obtaining the Mute Status of All Tracks

API

The following APIs are added:

- HI_UNF_SND_SetAllTrackMute
- HI_UNF_SND_GetAllTrackMute

2.10.1.3 Setting and Obtaining the Track Channel Mode

API

The following APIs are added:

- HI_UNF_SND_SetTrackChannelMode
- HI_UNF_SND_GetTrackChannelMode

2.10.1.4 Obtaining the Track Delay

API

HI_UNF_SND_GetTrackDelayMs is added.

2.10.1.5 Setting and Obtaining the Cast Volume

API

The following APIs are added:

- HI_UNF_SND_SetCastAbsWeight
- HI_UNF_SND_GetCastAbsWeight

2.10.1.6 Setting and Obtaining the Cast Mute Status

API

The following APIs are added:

- HI_UNF_SND_SetCastMute
- HI_UNF_SND_GetCastMute

2.10.2 Data Structure

2.10.2.1 New Data Structure

HI_UNF_SND_SPDIF_SCMSMODE_E

[Definition]

```
typedef enum hiHI_UNF_SND_SPDIF_SCMSMODE_E
{
    HI_UNF_SND_SPDIF_SCMSMODE_COPYALLOW,
    HI_UNF_SND_SPDIF_SCMSMODE_COPYONCE,
    HI_UNF_SND_SPDIF_SCMSMODE_COPYNOMORE,
    HI_UNF_SND_SPDIF_SCMSMODE_COPYPROHIBITED,
    HI_UNF_SND_SPDIF_SCMSMODE_BUTT
} HI_UNF_SND_OUTPUTPORT_E;
```

[Reason]

This data structure is added to specify the SPDIF SCMS mode.

[Note]

None

[Example]

None

2.10.3 API

2.10.3.1 New APIs

HI_UNF_SND_SetSpdifSCMSMode

[Definition]

```
HI_S32 HI_UNF_SND_SetSpdifSCMSMode(HI_UNF_SND_E enSound,
HI_UNF_SND_OUTPUTPORT_E enOutPort, HI_UNF_SND_SPDIF_SCMSMODE_E
enSpdifSCMSMode);
```

[Reason]

This API is added to specify the SPDIF SCMS mode.

[Note]

None

[Example]

None

HI_UNF_SND_GetSpdifSCMSMode

[Definition]

```
HI_S32 HI_UNF_SND_GetSpdifSCMSMode(HI_UNF_SND_E enSound,
HI_UNF_SND_OUTPUTPORT_E enOutPort, HI_UNF_SND_SPDIF_SCMSMODE_E
*enSpdifSCMSMode);
```

[Reason]

This API is added to obtain the SPDIF SCMS mode.

[Note]

None

[Example]

None

HI_UNF_SND_SetAllTrackMute

[Definition]

```
HI_S32  HI_UNF_SND_SetAllTrackMute(HI_UNF_SND_E enSound, HI_BOOL bMute);
```

[Reason]

This API is added to set the mute status for all tracks.

[Note]

This API does not apply to the virtual track.

[Example]

None

HI_UNF_SND_GetAllTrackMute

[Definition]

```
HI_S32  HI_UNF_SND_GetAllTrackMute(HI_UNF_SND_E enSound, HI_BOOL *pbMute);
```

[Reason]

This API is added to obtain the mute status of all tracks.

[Note]

This API does not apply to the virtual track.

[Example]

None

HI_UNF_SND_SetTrackChannelMode

[Definition]

```
HI_S32 HI_UNF_SND_SetTrackChannelMode(HI_HANDLE hTrack,
HI_UNF_TRACK_MODE_E enMode);
```

[Reason]

This API is added to set the single-track channel mode.

[Note]

This API does not apply to the virtual track.

[Example]

None

HI_UNF_SND_GetTrackChannelMode

[Definition]

```
HI_S32 HI_UNF_SND_GetTrackChannelMode(HI_HANDLE hTrack,
HI_UNF_TRACK_MODE_E *penMode);
```

[Reason]

This API is added to obtain the single-track channel mode.

[Note]

This API does not apply to the virtual track.

[Example]

None

HI_UNF_SND_GetTrackDelayMs

[Definition]

```
HI_S32 HI_UNF_SND_GetTrackDelayMs(const HI_HANDLE hTrack, HI_U32
*pDelayMs);
```

[Reason]

This API is added to obtain the track delay time.

[Note]

This API does not apply to the virtual track.

[Example]

None

HI_UNF_SND_SetCastAbsWeight

[Definition]

```
HI_S32 HI_UNF_SND_SetCastAbsWeight(HI_HANDLE hCast, const
HI_UNF_SND_ABSGAIN_ATTR_S *pstAbsWeightGain);
```

[Reason]

This API is used to set the cast data volume.

[Note]

None

[Example]

None

HI_UNF_SND_GetCastAbsWeight

[Definition]

```
HI_S32  HI_UNF_SND_GetCastAbsWeight(HI_HANDLE hCast,
HI_UNF_SND_ABSGAIN_ATTR_S *pstAbsWeightGain);
```

[Reason]

This API is used to obtain the cast data volume.

[Note]

None

[Example]

None

HI_UNF_SND_SetCastMute

[Definition]

```
HI_S32   HI_UNF_SND_SetCastMute(HI_HANDLE hCast, HI_BOOL bMute);
```

[Reason]

This API is used to set the cast data mute status.

[Note]

None

[Example]

None

HI_UNF_SND_GetCastMute

[Definition]

```
HI_S32   HI_UNF_SND_GetCastMute(HI_HANDLE hCast, HI_BOOL *pbMute);
```

[Reason]

This API is used to obtain the cast data mute status.

[Note]

None

[Example]

None

2.11 VI

2.11.1 Overview

The UNF 3.2.0 VI module has the following changes compared with UNF 3.1:

The 640 x 480p input of the BT.1120 digital interface is supported.

2.11.1.1 Supporting the 640 x 480p Input of the BT.1120 Digital Interface

The BT.1120 640 x 480p video inputs can be properly received. The data structure is changed as follows:

Data Structure

HI_UNF_VI_INPUT_MODE_E is modified.

2.11.2 Data Structure

2.11.2.1 Modified Data Structure

HI_UNF_VI_INPUT_MODE_E

[Definition]

```
typedef enum hiUNF_VI_INPUT_MODE_E
{
    HI_UNF_VI_MODE_BT656_576I = 0,
    HI_UNF_VI_MODE_BT656_480I,
    HI_UNF_VI_MODE_BT601_576I,
    HI_UNF_VI_MODE_BT601_480I,
    HI_UNF_VI_MODE_BT1120_480P,
    HI_UNF_VI_MODE_BT1120_576P,
    HI_UNF_VI_MODE_BT1120_720P_50,
    HI_UNF_VI_MODE_BT1120_720P_60,
    HI_UNF_VI_MODE_BT1120_1080I_50,
    HI_UNF_VI_MODE_BT1120_1080I_60,
    HI_UNF_VI_MODE_BT1120_1080P_25,
    HI_UNF_VI_MODE_BT1120_1080P_30,
    HI_UNF_VI_MODE_BT1120_1080P_50,
    HI_UNF_VI_MODE_BT1120_1080P_60,
    HI_UNF_VI_MODE_DIGITAL_CAMERA_48,
    HI_UNF_VI_MODE_DIGITAL_CAMERA_57,
    HI_UNF_VI_MODE_DIGITAL_CAMERA_720P_30,
    HI_UNF_VI_MODE_DIGITAL_CAMERA_1080P_30,
    HI_UNF_VI_MODE_BUTT
} HI_UNF_VI_INPUT_MODE_E;
```

After modification:

```
typedef enum hiUNF_VI_INPUT_MODE_E
{
    HI_UNF_VI_MODE_BT656_576I = 0,
    HI_UNF_VI_MODE_BT656_480I,
    HI_UNF_VI_MODE_BT601_576I,
    HI_UNF_VI_MODE_BT601_480I,
    HI_UNF_VI_MODE_BT1120_640X480P,
    HI_UNF_VI_MODE_BT1120_480P,
    HI_UNF_VI_MODE_BT1120_576P,
    HI_UNF_VI_MODE_BT1120_720P_50,
    HI_UNF_VI_MODE_BT1120_720P_60,
    HI_UNF_VI_MODE_BT1120_1080I_50,
    HI_UNF_VI_MODE_BT1120_1080I_60,
    HI_UNF_VI_MODE_BT1120_1080P_25,
    HI_UNF_VI_MODE_BT1120_1080P_30,
    HI_UNF_VI_MODE_BT1120_1080P_50,
    HI_UNF_VI_MODE_BT1120_1080P_60,
    HI_UNF_VI_MODE_DIGITAL_CAMERA_48,
    HI_UNF_VI_MODE_DIGITAL_CAMERA_57,
    HI_UNF_VI_MODE_DIGITAL_CAMERA_720P_30,
```

```
    HI_UNF_VI_MODE_DIGITAL_CAMERA_1080P_30,
    HI_UNF_VI_MODE_BUTT
} HI_UNF_VI_INPUT_MODE_E;
```

[Description]

HI_UNF_VI_MODE_BT1120_640X480P is added for supporting the BT.1120 640 x 480p inputs.

[Note]

None

[Example]

None

2.12 CC

2.12.1 Overview

The UNF 3.2.0 closed caption (CC) module has the following changes compared with UNF 3.1:

- The function of parsing 64 colors defined in the ATSC 708CC standard is added.
- The function of obtaining ARIB CC information is added.
- The function of parsing the font edge style and font edge color is added.
- The SCTE20 standard is supported.

2.12.1.1 Parsing 64 Colors Defined in the ATSC 708CC Standard

The UNF 3.1 supports the parsing of eight colors, while the UNF 3.2.0 supports the parsing of 64 colors. The data structures are changed as follows:

Data Structures

The following data structures are modified:

- HI_UNF_CC_608_CONFIGPARAM_S
- HI_UNF_CC_708_CONFIGPARAM_S
- HI_UNF_CC_COLOR_E

2.12.1.2 Obtaining ARIB CC Information

The ARIB CC information can be obtained, including the CC character code, scrolling mode, time control mode, playback mode, and display mode. The API and data structures are changed as follows:

Data Structure

The following data structures are added:

- HI_UNF_CC_ARIB_ROLLUP_E

- HI_UNF_CC_ARIB_TCS_E
- HI_UNF_CC_ARIB_DF_E
- HI_UNF_CC_ARIB_DMF_E
- HI_UNF_CC_ARIB_TMD_E
- HI_UNF_CC_ARIB_INFONODE_S
- HI_UNF_CC_ARIB_INFO_S

API

HI_UNF_CC_GetARIBCCInfo is added.

2.12.1.3 Parsing the Font Edge Style and Font Edge Color

The font edge style and font edge color information parsed from streams is returned to the users by using the callback function. The data structure is changed as follows:

Data Structure

HI_UNF_CC_TEXT_S is modified.

2.12.1.4 Supporting the SCTE20 Standard

The top/bottom field flag is added to identify the SCTE20 CCs. The data structure is changed as follows:

Data Structure

HI_UNF_CC_USERDATA_S is modified.

2.12.2 Data Structure

2.12.2.1 New Data Structures

HI_UNF_CC_ARIB_ROLLUP_E

[Definition]

```
typedef enum hiUNF_CC_ARIB_ROLLUP_E
{
   HI_UNF_CC_ARIB_NON_ROLLUP,
   HI_UNF_CC_ARIB_ROLLUP,
   HI_UNF_CC_ARIB_ROLLUP_BUTT
}HI_UNF_CC_ARIB_ROLLUP_E;
```

[Reason]

The ARIB CC roll-up type is defined so that useful results are returned when the ARIB CC information is obtained.

[Note]

None

[Example]

None

HI_UNF_CC_ARIB_TCS_E

[Definition]

```
typedef enum hiUNF_CC_ARIB_TCS_E
{
    HI_UNF_CC_ARIB_TCS_8BIT,
    HI_UNF_CC_ARIB_TCS_BUTT
}HI_UNF_CC_ARIB_TCS_E;
```

[Reason]

The ARIB CC character encoding mode is defined so that useful results are returned when the ARIB CC information is obtained.

[Note]

None

[Example]

None

HI_UNF_CC_ARIB_DF_E

[Definition]

```
typedef enum hiUNF_CC_ARIB_DF_E
{
    HI_UNF_CC_ARIB_DF_HORIZONTAL_SD,
    HI_UNF_CC_ARIB_DF_VERTICAL_SD,
    HI_UNF_CC_ARIB_DF_HORIZONTAL_HD,
    HI_UNF_CC_ARIB_DF_VERTICAL_HD,
    HI_UNF_CC_ARIB_DF_HORIZONTAL_WESTERN,
    HI_UNF_CC_ARIB_DF_HORIZONTAL_1920X1080,
    HI_UNF_CC_ARIB_DF_VERTICAL_1920X1080,
    HI_UNF_CC_ARIB_DF_HORIZONTAL_960X540,
    HI_UNF_CC_ARIB_DF_VERTICAL_960X540,
    HI_UNF_CC_ARIB_DF_HORIZONTAL_1280X720,
    HI_UNF_CC_ARIB_DF_VERTICAL_1280X720,
    HI_UNF_CC_ARIB_DF_HORIZONTAL_720X480,
    HI_UNF_CC_ARIB_DF_VERTICAL_720X480,
    HI_UNF_CC_ARIB_DF_BUTT
}HI_UNF_CC_ARIB_DF_E;
```

[Reason]

The ARIB CC display format is defined so that useful results are returned when the ARIB CC information is obtained.

[Note]

None

[Example]

None

HI_UNF_CC_ARIB_DMF_E

[Definition]

```
typedef enum hiUNF_CC_ARIB_DMF_E
{
    HI_UNF_CC_ARIB_DMF_AUTO_AND_AUTO=0x0,
    HI_UNF_CC_ARIB_DMF_AUTO_AND_NOT,
    HI_UNF_CC_ARIB_DMF_AUTO_AND_SELECT,
    HI_UNF_CC_ARIB_DMF_NON_AND_AUTO=0x4,
    HI_UNF_CC_ARIB_DMF_NON_AND_NON,
    HI_UNF_CC_ARIB_DMF_SELECT_AND_AUTO=0x8,
    HI_UNF_CC_ARIB_DMF_SELECT_AND_NON,
    HI_UNF_CC_ARIB_DMF_SELECT_AND_SELECT,
    HI_UNF_CC_ARIB_DMF_SPECIAL_AND_AUTO=0xc,
    HI_UNF_CC_ARIB_DMF_SPECIAL_AND_NON,
    HI_UNF_CC_ARIB_DMF_SPECIAL_AND_SELECT,
    HI_UNF_CC_ARIB_DMF_BUTT
}HI_UNF_CC_ARIB_DMF_E;
```

[Reason]

The ARIB CC display mode is defined so that useful results are returned when the ARIB CC information is obtained.

[Note]

None

[Example]

None

HI_UNF_CC_ARIB_TMD_E

[Definition]

```
typedef enum hiUNF_CC_ARIB_TMD_E
{
    HI_UNF_CC_ARIB_TMD_FREE,
    HI_UNF_CC_ARIB_TMD_REAL_TIME,
    HI_UNF_CC_ARIB_TMD_OFFSET_TIME,
    HI_UNF_CC_ARIB_TMD_BUTT
}HI_UNF_CC_ARIB_TMD_E;
```

[Reason]

The ARIB CC time control mode is defined so that useful results are returned when the ARIB CC information is obtained.

[Note]

None

[Example]

None

HI_UNF_CC_ARIB_INFONODE_S

[Definition]

```
typedef struct hiUNF_CC_ARIB_INFONODE_S
{
    HI_U8 u8LanguageTag;
    HI_UNF_CC_ARIB_DMF_E    enCCAribDMF;
    HI_CHAR    acISO639LanguageCode[4];
    HI_UNF_CC_ARIB_DF_E enCCAribDF;
    HI_UNF_CC_ARIB_TCS_E enCCAribTCS;
    HI_UNF_CC_ARIB_ROLLUP_E enCCAribRollup;
}HI_UNF_CC_ARIB_INFONODE_S;
```

[Reason]

The ARIB CC information structure for a signal language is defined to implement the function of obtaining the ARIB CC information.

[Note]

None

[Example]

None

HI_UNF_CC_ARIB_INFO_S

[Definition]

```
typedef struct hiUNF_CC_ARIB_INFO_S
{
    HI_UNF_CC_ARIB_TMD_E enCCAribTMD;
    HI_U32 u32NumLanguage;
    HI_UNF_CC_ARIB_INFONODE_S stCCAribInfonode[ARIBCC_MAX_LANGUAGE];
}HI_UNF_CC_ARIB_INFO_S;
```

[Reason]

The ARIB CC information structure is defined to implement the function of obtaining the ARIB CC information.

[Note]

None

[Example]

None

2.12.2.2 Modified Data Structures

HI_UNF_CC_608_CONFIGPARAM_S

[Definition]

Before modification:

```
typedef struct hiUNF_CC_608_CONFIGPARAM_S
{
    HI_UNF_CC_608_DATATYPE_E    enCC608DataType;
    HI_UNF_CC_COLOR_E           enCC608TextColor;
    HI_UNF_CC_OPACITY_E         enCC608TextOpac;
    HI_UNF_CC_COLOR_E           enCC608BgColor;
    HI_UNF_CC_OPACITY_E         enCC608BgOpac;
    HI_UNF_CC_FONTSTYLE_E       enCC608FontStyle;
    HI_UNF_CC_DF_E             enCC608DispFormat;
    HI_BOOL                    bLeadingTailingSpace;
}HI_UNF_CC_608_CONFIGPARAM_S;
After modification:
typedef struct hiUNF_CC_608_CONFIGPARAM_S
{
    HI_UNF_CC_608_DATATYPE_E    enCC608DataType;
    HI_U32           u32CC608TextColor;
    HI_UNF_CC_OPACITY_E         enCC608TextOpac;
    HI_U32          u32CC608BgColor;
    HI_UNF_CC_OPACITY_E         enCC608BgOpac;
    HI_UNF_CC_FONTSTYLE_E       enCC608FontStyle;
    HI_UNF_CC_DF_E             enCC608DispFormat;
    HI_BOOL                    bLeadingTailingSpace;
}HI_UNF_CC_608_CONFIGPARAM_S;
```

[Description]

The color values are changed from enumerations to unsigned integers to support 64 or more colors.

[Note]

None

[Example]

None

HI_UNF_CC_708_CONFIGPARAM_S

[Definition]

Before modification:

```
typedef struct hiUNF_CC_708_CONFIGPARAM_S
{
    HI_UNF_CC_708_SERVICE_NUM_E    enCC708ServiceNum;
    HI_UNF_CC_FONTNAME_E       enCC708FontName;
    HI_UNF_CC_FONTSTYLE_E      enCC708FontStyle;
    HI_UNF_CC_FONTSIZE_E       enCC708FontSize;
    HI_UNF_CC_COLOR_E          enCC708TextColor;
    HI_UNF_CC_OPACITY_E        enCC708TextOpac;
    HI_UNF_CC_COLOR_E          enCC708BgColor;
    HI_UNF_CC_OPACITY_E        enCC708BgOpac;
    HI_UNF_CC_COLOR_E          enCC708WinColor;
    HI_UNF_CC_OPACITY_E        enCC708WinOpac;
    HI_UNF_CC_EdgeType_E       enCC708TextEdgeType;
    HI_UNF_CC_COLOR_E          enCC708TextEdgeColor;
    HI_UNF_CC_DF_E            enCC708DispFormat;
} HI_UNF_CC_708_CONFIGPARAM_S;
After modification:
typedef struct hiUNF_CC_708_CONFIGPARAM_S
{
    HI_UNF_CC_708_SERVICE_NUM_E    enCC708ServiceNum;
    HI_UNF_CC_FONTNAME_E       enCC708FontName;
    HI_UNF_CC_FONTSTYLE_E      enCC708FontStyle;
    HI_UNF_CC_FONTSIZE_E       enCC708FontSize;
    HI_U32          u32CC708TextColor;
    HI_UNF_CC_OPACITY_E        enCC708TextOpac;
    HI_U32          u32CC708BgColor;
    HI_UNF_CC_OPACITY_E        enCC708BgOpac;
    HI_U32          u32CC708WinColor;
    HI_UNF_CC_OPACITY_E        enCC708WinOpac;
    HI_UNF_CC_EdgeType_E       enCC708TextEdgeType;
    HI_U32          u32CC708TextEdgeColor;
    HI_UNF_CC_DF_E            enCC708DispFormat;
} HI_UNF_CC_708_CONFIGPARAM_S;
```

[Description]

The color values are changed from enumerations to unsigned integers to support 64 or more colors.

[Note]

None

[Example]

None

HI_UNF_CC_COLOR_E

[Definition]

Before modification:

```
typedef enum hiUNF_CC_COLOR_E
{
    HI_UNF_CC_COLOR_DEFAULT,
    HI_UNF_CC_COLOR_BLACK,
    HI_UNF_CC_COLOR_WHITE,
    HI_UNF_CC_COLOR_RED,
    HI_UNF_CC_COLOR_GREEN,
    HI_UNF_CC_COLOR_BLUE,
    HI_UNF_CC_COLOR_YELLOW,
    HI_UNF_CC_COLOR_MAGENTA,
    HI_UNF_CC_COLOR_CYAN,
    HI_UNF_CC_COLOR_BUTT
}HI_UNF_CC_COLOR_E;
```

After modification:

```
typedef enum hiUNF_CC_COLOR_E
{
    HI_UNF_CC_COLOR_DEFAULT=0x00000000,
    HI_UNF_CC_COLOR_BLACK=0xff000000,
    HI_UNF_CC_COLOR_WHITE=0xffffffff,
    HI_UNF_CC_COLOR_RED=0xffff0000,
    HI_UNF_CC_COLOR_GREEN=0xff00ff00,
    HI_UNF_CC_COLOR_BLUE=0xff0000ff,
    HI_UNF_CC_COLOR_YELLOW=0xffffff00,
    HI_UNF_CC_COLOR_MAGENTA=0xffff00ff,
    HI_UNF_CC_COLOR_CYAN=0xff00ffff,
}HI_UNF_CC_COLOR_E;
```

[Description]

The corresponding ARGB values are assigned to eight color enumerations.

[Note]

None

[Example]

None

HI_UNF_CC_TEXT_S

[Definition]

Before modification:

```
typedef struct hiUNF_CC_TEXT_S
{
    HI_U16              *pu16Text;
    HI_U8               u8TextLen;
    HI_UNF_CC_COLOR_S     stFgColor;
    HI_UNF_CC_COLOR_S     stBgColor;
    HI_U8               u8Justify;
    HI_U8               u8WordWrap;
    HI_UNF_CC_FONTSTYLE_E  enFontStyle;
    HI_UNF_CC_FONTSIZE_E   enFontSize;
} HI_UNF_CC_TEXT_S;
```

After modification:

```
typedef struct hiUNF_CC_TEXT_S
{
    HI_U16              *pu16Text;
    HI_U8               u8TextLen;
    HI_UNF_CC_COLOR_S     stFgColor;
    HI_UNF_CC_COLOR_S     stBgColor;
    HI_UNF_CC_COLOR_S     stEdgeColor;
    HI_U8               u8Justify;
    HI_U8               u8WordWrap;
    HI_UNF_CC_FONTSTYLE_E  enFontStyle;
    HI_UNF_CC_FONTSIZE_E   enFontSize;
    HI_UNF_CC_EdgeType_E    enEdgetype;
} HI_UNF_CC_TEXT_S;
```

[Description]

This data structure is modified to support the function of parsing the font edge style and font edge color.

[Note]

None

[Example]

None

HI_UNF_CC_USERDATA_S

[Definition]

Before modification:

```
typedef struct hiUNF_CC_USERDATA_S
```

```
{
    HI_U8       *pu8userdata;
    HI_U32      u32dataLen;
} HI_UNF_CC_USERDATA_S;
After modification:
typedef struct hiUNF_CC_USERDATA_S
{
    HI_U8       *pu8userdata;
    HI_U32      u32dataLen;
    HI_BOOL     bTopFieldFirst;
} HI_UNF_CC_USERDATA_S;
```

[Description]

The top/bottom field flag is added to support the SCTE20 standard.

[Note]

None

[Example]

None

2.12.3 API

2.12.3.1 New API

HI_UNF_CC_GetARIBCCInfo

[Definition]

```
HI_S32 HI_UNF_CC_GetARIBCCInfo(HI_HANDLE hCC,HI_UNF_CC_ARIB_INFO_S
*pstCCAribInfo);
```

[Reason]

This API is added to obtain the ARIB CC information.

[Note]

Before calling this API, you need to initialize the CC module, create and start an instance, and inject ARIB CC data.

[Example]

None

2.13 Teletext

2.13.1 Overview

The UNF 3.2.0 teletext module has the following changes compared with UNF 3.1:

The function of setting the ISO639 language code is added.

2.13.1.1 Setting the ISO639 Language Code

The decoder sometimes cannot decode data properly because information about the character set for some streams is not transmitted according to the teletext standard. In this case, the ISO639 language code can be set in the decoder when HI_UNF_TTX_SwitchContent is called, so that the decoder can identify the character set of the current stream based on the information. The data structure is changed as follows:

Data Structure

HI_UNF_TTX_CONTENT_PARA_S is modified.

2.13.2 Data Structure

2.13.2.1 Modified Data Structure

HI_UNF_TTX_CONTENT_PARA_S

[Definition]

Before modification:

```
typedef struct hiUNF_TTX_CONTENT_PARA_S
{
    HI_UNF_TTX_TYPE_E     enType;
    HI_UNF_TTX_PAGE_ADDR_S stInitPgAddr;
} HI_UNF_TTX_CONTENT_PARA_S;
```

After modification:

```
typedef struct hiUNF_TTX_CONTENT_PARA_S
{
    HI_UNF_TTX_TYPE_E     enType;
    HI_U32                u32ISO639LanCode;
    HI_UNF_TTX_PAGE_ADDR_S stInitPgAddr;
} HI_UNF_TTX_CONTENT_PARA_S;
```

[Description]

The ISO639 language code is set in the decoder so that the decoder can identify the character set of the current stream based on this information. The ISO639 language code can be parsed when the teletext descriptor is parsed from the program map table (PMT).

[Note]

None

[Example]

None

2.14 HiPlayer

2.14.1 Overview

The UNF 3.2.0 HiPlayer module has the following changes compared with UNF 3.1:

- The 1x rewind and 0.5x slow playback functions are supported.
- The function of setting headers for HTTP packets is added.
- The function of specifying the first segment to be played when the HTTP live streaming (HLS) live play starts is added.

2.14.1.1 Supporting 1x Rewind and 0.5x Slow Playback

Some applications require that streams be played or rewound at a low speed. The data structure is changed as follows:

Data Structure

HI_SVR_PLAYER_PLAY_SPEED_E is modified.

2.14.1.2 Setting Headers for HTTP Packets

Some servers have special requirements on headers of HTTP packets. In this case, applications can set the headers by calling the new API and the HiPlayer attaches the HTTP headers to the HTTP packets to be transmitted to the servers. The data structures are changed as follows:

Data Structure

The following data structures are modified:

- HI_FORMAT_INVOKE_ID_E
- HI_SVR_FORMAT_PARAMETER_S

2.14.1.3 Specifying the First Segment to Be Played When the HLS Live Play Starts

The first segment to be played can be specified to improve the real-time performance of live play. The data structures are changed as follows:

Data Structure

The following data structures are modified:

- HI_FORMAT_INVOKE_ID_E
- HI_SVR_FORMAT_PARAMETER_S

2.14.2 Data Structure

2.14.2.1 Modified Data Structures

HI_SVR_FORMAT_PARAMETER_S

[Definition]

Before modification:

```
typedef struct hiSVR_FORMAT_PARAMETER_S
{
    HI_FORMAT_BUFFER_CONFIG_S stBufConfig;
    HI_S64          s64BufMaxSize;
    HI_U32          u32Cookie;
    HI_FORMAT_HLS_START_MODE_E eHlsStartMode;
    HI_HANDLE       hDRMClient;
    HI_S32          s32DstCodeType;
    HI_VOID         *pArgCharsetMgr;
    HI_U32          u32Userdata;
    HI_U8           *pUserAgent;
    HI_U32          u32LogLevel;
    HI_U8           *pReferer;
    HI_U32          u32NotSupportByteRange;
} HI_SVR_FORMAT_PARAMETER_S;;
```

After modification:

```
typedef struct hiSVR_FORMAT_PARAMETER_S
{
    HI_FORMAT_BUFFER_CONFIG_S stBufConfig;
    HI_S64          s64BufMaxSize;
    HI_CHAR         *pCookie;
    HI_CHAR         *pHeaders;
    HI_FORMAT_HLS_START_MODE_E eHlsStartMode;
    HI_HANDLE       hDRMClient;
    HI_S32          s32DstCodeType;
    HI_VOID         *pArgCharsetMgr;
    HI_U32          u32Userdata;
    HI_U8           *pUserAgent;
    HI_U32          u32LogLevel;
    HI_U8           *pReferer;
    HI_U32          u32NotSupportByteRange;
    HI_S32          s32HlsLiveStartNum;
} HI_SVR_FORMAT_PARAMETER_S;
```

[Description]

The member **pHeaders** is added for storing HTTP headers configured by applications for the HiPlayer.

The member **s32HlsLiveStartNum** is added for setting the first segment to be played when the HLS live play starts.

The member **u32Cookie** is changed to the character string pointer type.

[Note]

None

[Example]

None

HI_SVR_PLAYER_PLAY_SPEED_E

[Definition]

Before modification:

```
typedef enum hiSVR_PLAYER_PLAY_SPEED_E
{
    HI_SVR_PLAYER_PLAY_SPEED_2X_FAST_FORWARD   = 2 *
HI_SVR_PLAYER_PLAY_SPEED_NORMAL,
    HI_SVR_PLAYER_PLAY_SPEED_4X_FAST_FORWARD   = 4 *
HI_SVR_PLAYER_PLAY_SPEED_NORMAL,
    HI_SVR_PLAYER_PLAY_SPEED_8X_FAST_FORWARD   = 8 *
HI_SVR_PLAYER_PLAY_SPEED_NORMAL,
    HI_SVR_PLAYER_PLAY_SPEED_16X_FAST_FORWARD  = 16 *
HI_SVR_PLAYER_PLAY_SPEED_NORMAL,
    HI_SVR_PLAYER_PLAY_SPEED_32X_FAST_FORWARD  = 32 *
HI_SVR_PLAYER_PLAY_SPEED_NORMAL,
    HI_SVR_PLAYER_PLAY_SPEED_64X_FAST_FORWARD  = 64 *
HI_SVR_PLAYER_PLAY_SPEED_NORMAL,
    HI_SVR_PLAYER_PLAY_SPEED_2X_FAST_BACKWARD  = -2 *
HI_SVR_PLAYER_PLAY_SPEED_NORMAL,
    HI_SVR_PLAYER_PLAY_SPEED_4X_FAST_BACKWARD  = -4 *
HI_SVR_PLAYER_PLAY_SPEED_NORMAL,
    HI_SVR_PLAYER_PLAY_SPEED_8X_FAST_BACKWARD  = -8 *
HI_SVR_PLAYER_PLAY_SPEED_NORMAL,
    HI_SVR_PLAYER_PLAY_SPEED_16X_FAST_BACKWARD = -16 *
HI_SVR_PLAYER_PLAY_SPEED_NORMAL,
    HI_SVR_PLAYER_PLAY_SPEED_32X_FAST_BACKWARD = -32 *
HI_SVR_PLAYER_PLAY_SPEED_NORMAL,
    HI_SVR_PLAYER_PLAY_SPEED_64X_FAST_BACKWARD = -64 *
HI_SVR_PLAYER_PLAY_SPEED_NORMAL,
    HI_SVR_PLAYER_PLAY_SPEED_BUTT
} HI_SVR_PLAYER_PLAY_SPEED_E;
```

After modification:

```
typedef enum hiSVR_PLAYER_PLAY_SPEED_E
{
    HI_SVR_PLAYER_PLAY_SPEED_1X2_SLOW_FORWARD  =
HI_SVR_PLAYER_PLAY_SPEED_NORMAL/2,
    HI_SVR_PLAYER_PLAY_SPEED_2X_FAST_FORWARD   = 2 *
HI_SVR_PLAYER_PLAY_SPEED_NORMAL,
    HI_SVR_PLAYER_PLAY_SPEED_4X_FAST_FORWARD   = 4 *
HI_SVR_PLAYER_PLAY_SPEED_NORMAL,
    HI_SVR_PLAYER_PLAY_SPEED_8X_FAST_FORWARD   = 8 *
HI_SVR_PLAYER_PLAY_SPEED_NORMAL,
    HI_SVR_PLAYER_PLAY_SPEED_16X_FAST_FORWARD  = 16 *
HI_SVR_PLAYER_PLAY_SPEED_NORMAL,
    HI_SVR_PLAYER_PLAY_SPEED_32X_FAST_FORWARD  = 32 *
HI_SVR_PLAYER_PLAY_SPEED_NORMAL,
    HI_SVR_PLAYER_PLAY_SPEED_64X_FAST_FORWARD  = 64 *
HI_SVR_PLAYER_PLAY_SPEED_NORMAL,
    HI_SVR_PLAYER_PLAY_SPEED_1X_BACKWARD       = -1 *
HI_SVR_PLAYER_PLAY_SPEED_NORMAL,
    HI_SVR_PLAYER_PLAY_SPEED_2X_FAST_BACKWARD  = -2 *
HI_SVR_PLAYER_PLAY_SPEED_NORMAL,
    HI_SVR_PLAYER_PLAY_SPEED_4X_FAST_BACKWARD  = -4 *
HI_SVR_PLAYER_PLAY_SPEED_NORMAL,
    HI_SVR_PLAYER_PLAY_SPEED_8X_FAST_BACKWARD  = -8 *
HI_SVR_PLAYER_PLAY_SPEED_NORMAL,
    HI_SVR_PLAYER_PLAY_SPEED_16X_FAST_BACKWARD = -16 *
HI_SVR_PLAYER_PLAY_SPEED_NORMAL,
    HI_SVR_PLAYER_PLAY_SPEED_32X_FAST_BACKWARD = -32 *
HI_SVR_PLAYER_PLAY_SPEED_NORMAL,
    HI_SVR_PLAYER_PLAY_SPEED_64X_FAST_BACKWARD = -64 *
HI_SVR_PLAYER_PLAY_SPEED_NORMAL,
    HI_SVR_PLAYER_PLAY_SPEED_BUTT
} HI_SVR_PLAYER_PLAY_SPEED_E;
```

[Description]

The member **HI_SVR_PLAYER_PLAY_SPEED_1X2_SLOW_FORWARD** is added, indicating 0.5x slow playback.

The member **HI_SVR_PLAYER_PLAY_SPEED_1X_BACKWARD** is added, indicating 1x rewind.

[Note]

The 64x fast-forward and rewind are not supported.

[Example]

None

HI_FORMAT_INVOKE_ID_E

[Definition]

Before modification:

```
typedef enum hiFORMAT_INVOKE_ID_E
{
    HI_FORMAT_INVOKE_PRE_CLOSE_FILE = 0x0,
    HI_FORMAT_INVOKE_USER_OPE,
    HI_FORMAT_INVOKE_SET_SUB_LANTYPE,
    HI_FORMAT_INVOKE_SET_SOURCE_CODETYPE,
    HI_FORMAT_INVOKE_SET_DEST_CODETYPE,
    HI_FORMAT_INVOKE_SET_CHARSETMGR_FUNC,
    HI_FORMAT_INVOKE_GET_METADATA,
    HI_FORMAT_INVOKE_GET_THUMBNAIL,
    HI_FORMAT_INVOKE_CHECK_VIDEOSUPPORT,
    HI_FORMAT_INVOKE_SET_COOKIE,
    HI_FORMAT_INVOKE_GET_BANDWIDTH,
    HI_FORMAT_INVOKE_GET_HLS_STREAM_NUM,
    HI_FORMAT_INVOKE_GET_HLS_STREAM_INFO,
    HI_FORMAT_INVOKE_GET_HLS_SEGMENT_INFO,
    HI_FORMAT_INVOKE_GET_PLAYLIST_STREAM_DETAIL_INFO,
    HI_FORMAT_INVOKE_SET_HLS_STREAM,
    HI_FORMAT_INVOKE_SET_HLS_START_MODE,
    HI_FORMAT_INVOKE_SET_BUFFER_CONFIG,
    HI_FORMAT_INVOKE_GET_BUFFER_CONFIG,
    HI_FORMAT_INVOKE_GET_BUFFER_STATUS,
    HI_FORMAT_INVOKE_GET_BUFFER_LAST_PTS,
    HI_FORMAT_INVOKE_SET_BUFFER_MAX_SIZE,
    HI_FORMAT_INVOKE_GET_BUFFER_MAX_SIZE,
    HI_FORMAT_INVOKE_GET_MSG_EVENT,
    HI_FORMAT_INVOKE_GET_FIRST_IFRAME,
    HI_FORMAT_INVOKE_SET_DRM_HDL,
    HI_FORMAT_INVOKE_SET_LOCALTIME,
    HI_FORMAT_INVOKE_SET_USERAGENT,
    HI_FORMAT_INVOKE_SET_REFERER,
    HI_FORMAT_INVOKE_SET_NOT_SUPPORT_BYTERANGE,
    HI_FORMAT_INVOKE_SET_LOG_LEVEL,
    HI_FORMAT_INVOKE_RTMP_RECEIVEAUDIO,
    HI_FORMAT_INVOKE_RTMP_RECEIVEVIDEO,
    HI_FORMAT_INVOKE_SET_BUFFER_UNDERRUN,
    HI_FORMAT_INVOKE_PROTOCOL_USER=100,
    HI_FORMAT_INVOKE_SET_DOLBYRANGEINFO,
```

```
    HI_FORMAT_INVOKE_GET_DOLBYINFO,
    HI_FORMAT_INVOKE_BUTT
} HI_FORMAT_INVOKE_ID_E;
```

After modification:

```
typedef enum hiFORMAT_INVOKE_ID_E
{
    HI_FORMAT_INVOKE_PRE_CLOSE_FILE = 0x0,
    HI_FORMAT_INVOKE_USER_OPE,
    HI_FORMAT_INVOKE_SET_SUB_LANTYPE,
    HI_FORMAT_INVOKE_SET_SOURCE_CODETYPE,
    HI_FORMAT_INVOKE_SET_DEST_CODETYPE,
    HI_FORMAT_INVOKE_SET_CHARSETMGR_FUNC,
    HI_FORMAT_INVOKE_GET_METADATA,
    HI_FORMAT_INVOKE_GET_THUMBNAIL,
    HI_FORMAT_INVOKE_CHECK_VIDEOSUPPORT,
    HI_FORMAT_INVOKE_SET_COOKIE,
    HI_FORMAT_INVOKE_GET_BANDWIDTH,
    HI_FORMAT_INVOKE_GET_HLS_STREAM_NUM,
    HI_FORMAT_INVOKE_GET_HLS_STREAM_INFO,
    HI_FORMAT_INVOKE_GET_HLS_SEGMENT_INFO,
    HI_FORMAT_INVOKE_GET_PLAYLIST_STREAM_DETAIL_INFO,
    HI_FORMAT_INVOKE_SET_HLS_STREAM,
    HI_FORMAT_INVOKE_SET_HLS_START_MODE,
    HI_FORMAT_INVOKE_SET_BUFFER_CONFIG,
    HI_FORMAT_INVOKE_GET_BUFFER_CONFIG,
    HI_FORMAT_INVOKE_GET_BUFFER_STATUS,
    HI_FORMAT_INVOKE_GET_BUFFER_LAST_PTS,
    HI_FORMAT_INVOKE_SET_BUFFER_MAX_SIZE,
    HI_FORMAT_INVOKE_GET_BUFFER_MAX_SIZE,
    HI_FORMAT_INVOKE_GET_MSG_EVENT,
    HI_FORMAT_INVOKE_GET_FIRST_IFRAME,
    HI_FORMAT_INVOKE_SET_DRM_HDL,
    HI_FORMAT_INVOKE_SET_LOCALTIME,
    HI_FORMAT_INVOKE_SET_HEADERS,
    HI_FORMAT_INVOKE_SET_USERAGENT,
    HI_FORMAT_INVOKE_SET_REFERER,
    HI_FORMAT_INVOKE_SET_NOT_SUPPORT_BYTERANGE,
    HI_FORMAT_INVOKE_SET_LOG_LEVEL,
    HI_FORMAT_INVOKE_RTMP_RECEIVEAUDIO,
    HI_FORMAT_INVOKE_RTMP_RECEIVEVIDEO,
    HI_FORMAT_INVOKE_SET_BUFFER_UNDERRUN,
    HI_FORMAT_INVOKE_SET_HLS_LIVE_START_NUM,
    HI_FORMAT_INVOKE_PROTOCOL_USER=100,
```

```
    HI_FORMAT_INVOKE_SET_DOLBYRANGEINFO,
    HI_FORMAT_INVOKE_GET_DOLBYINFO,
    HI_FORMAT_INVOKE_BUTT
} HI_FORMAT_INVOKE_ID_E;
```

[Description]

The member **HI_FORMAT_INVOKE_SET_HEADERS** is added. Some servers have special requirements on headers of HTTP packets. In this case, applications can set the headers to the HiPlayer by calling this API, and the HiPlayer attaches the HTTP headers to the HTTP packets to be transmitted to the servers.

The member **HI_FORMAT_INVOKE_SET_HLS_LIVE_START_NUM** is added for setting the first segment to be played when the HLS live play starts.

[Note]

None

[Example]

None

2.15 HiGo

2.15.1 Overview

The UNF 3.2.0 HiGo module has the following changes compared with UNF 3.1:

The function of filling a rounded rectangle is added.

2.15.1.1 Filling a Rounded Rectangle

The function of filling a rounded rectangle is added. The API is changed as follows:

API

HI_GO_FillRoundRect is added.

2.15.2 API

2.15.2.1 New API

HI_GO_FillRoundRect

[Definition]

```
HI_S32 HI_GO_FillRoundRect(HI_HANDLE Surface, const HI_RECT* pRect,
HI_COLOR Color, HI_S32 s32Radius);
```

[Reason]

The function of filling a rounded rectangle is added.

[Note]

None

[Example]

None

3 Differences Between UNF 3.2.1 and UNF 3.2.0

3.1 AVPLAY

3.1.1 Overview

The UNF 3.2.1 AVPLAY module has the following changes compared with UNF 3.2.0:

The function of playing 3D streams with tile encapsulation is added.

3.1.1.1 Playing 3D Streams with Tile Encapsulation

3D streams with tile encapsulation can be played directly. The data structure is changed as follows:

Data Structure

HI_UNF_VIDEO_FRAME_PACKING_TYPE_E is modified.

3.1.2 Data Structure

3.1.2.1 Modified Data Structure

HI_UNF_VIDEO_FRAME_PACKING_TYPE_E

[Definition]

Before modification:

```
typedef enum hiUNF_VIDEO_FRAME_PACKING_TYPE_E
{
    HI_UNF_FRAME_PACKING_TYPE_NONE,
    HI_UNF_FRAME_PACKING_TYPE_SIDE_BY_SIDE,
    HI_UNF_FRAME_PACKING_TYPE_TOP_AND_BOTTOM,
    HI_UNF_FRAME_PACKING_TYPE_TIME_INTERLACED,
    HI_UNF_FRAME_PACKING_TYPE_BUTT
}HI_UNF_VIDEO_FRAME_PACKING_TYPE_E;
```

After modification:

```
typedef enum hiUNF_VIDEO_FRAME_PACKING_TYPE_E
{
    HI_UNF_FRAME_PACKING_TYPE_NONE,
    HI_UNF_FRAME_PACKING_TYPE_SIDE_BY_SIDE,
    HI_UNF_FRAME_PACKING_TYPE_TOP_AND_BOTTOM,
    HI_UNF_FRAME_PACKING_TYPE_3D_TILE,
    HI_UNF_FRAME_PACKING_TYPE_TIME_INTERLACED,
    HI_UNF_FRAME_PACKING_TYPE_BUTT
}HI_UNF_VIDEO_FRAME_PACKING_TYPE_E;
```

[Reason]

The tile enumeration is added.

[Note]

Streams with the tile encapsulation can be played only in 720p full-screen output mode.

[Example]

```
HI_UNF_VIDEO_FRAME_PACKING_TYPE_E ePackType;
ePackType = HI_UNF_FRAME_PACKING_TYPE_3D_TILE;
HI_UNF_AVPLAY_SetAttr(hAvplay,HI_UNF_AVPLAY_ATTR_ID_FRMPACK_TYPE,&ePackTy
pe);
```

3.2 SPI

3.2.1 Overview

The UNF 3.2.1 SPI module has the following changes compared with UNF 3.2.0:

- The function of opening the specified SPI device is added.
- The function of stopping the specified SPI device is added.
- The function of setting the SPI working mode and related attributes is added.
- The function of obtaining the SPI working mode and related attributes is added.
- The data transmission function is added.
- The data reception function is added.

3.2.1.1 Opening an SPI Device

Data Structure

HI_UNF_SPI_DEV_E is added.

API

HI_UNF_SPI_Open is added.

3.2.1.2 Stopping an SPI Device

Data Structure

HI_UNF_SPI_DEV_E is added.

API

HI_UNF_SPI_Close is added.

3.2.1.3 Setting the SPI Working Mode and Related Attributes

Data Structure

The following data structures are added:

- HI_UNF_SPI_SPO_E
- HI_UNF_SPI_SPO_E
- HI_UNF_SPI_FRF_E
- HI_UNF_SPI_BIGEND_E
- HI_UNF_SPI_ATTR_NM_S
- HI_UNF_SPI_ATTR_EXT_U
- HI_UNF_SPI_ATTR_S

API

HI_UNF_SPI_SetAttr is added.

3.2.1.4 Obtaining the SPI Working Mode and Related Attributes

Data Structure

HI_UNF_SPI_ATTR_S is added.

API

HI_UNF_SPI_SetAttr is added.

3.2.1.5 Transmitting Data

API

HI_UNF_SPI_Read is added.

3.2.1.6 Receiving Data

API

HI_UNF_SPI_Write is added.

3.2.2 Data Structure

3.2.2.1 New Data Structure

HI_UNF_SPI_DEV_E

[Definition]

```
typedef enum hiUNF_SPI_DEV_E
{
    HI_UNF_SPI_DEV_0 =0 ,
    HI_UNF_SPI_DEV_1 = 1,
    HI_UNF_SPI_DEV_BUTT
}HI_UNF_SPI_DEV_E;
```

[Reason]

Some chips have two SPI main controllers.

[Note]

None

[Example]

None

HI_UNF_SPI_SPO_E

[Definition]

```
typedef enum hiUNF_SPI_SPO_E
{
    HI_UNF_SPI_SPO_0 = 0,
    HI_UNF_SPI_SPO_1 = 1
}HI_UNF_SPI_SPO_E;
```

[Reason]

This data structure is added for setting and obtaining the SPI clock output polarity.

[Note]

None

[Example]

None

HI_UNF_SPI_SPH_E

[Definition]

```
typedef enum hiUNF_SPI_SPH_E
{
    HI_UNF_SPI_SPH_0 = 0,
```

```
    HI_UNF_SPI_SPH_1 = 1
}HI_UNF_SPI_SPH_E;
```

[Reason]

This data structure is added for setting and obtaining the SPI clock output phase.

[Note]

None

[Example]

None

HI_UNF_SPI_FRF_E

[Definition]

```
typedef enum hiUNF_SPI_FRF_E
{
    HI_UNF_SPI_FRF_MOTO = 0,
    HI_UNF_SPI_FRF_TI   = 1,
    HI_UNF_SPI_FRF_NM   = 2,
    HI_UNF_SPI_FRF_BUTT = 3
}HI_UNF_SPI_FRF_E;
```

[Reason]

This data structure is added for setting and obtaining the frame format for SPI communication.

[Note]

None

[Example]

None

HI_UNF_SPI_BIGEND_E

[Definition]

```
typedef enum hiUNF_SPI_BIGEND_E
{
    HI_UNF_SPI_BIGEND_LITTLE,
    HI_UNF_SPI_BIGEND_BIG,
}HI_UNF_SPI_BIGEND_E;
```

[Reason]

This data structure is added for setting the big or little endian mode.

[Note]

None

[Example]

None

HI_UNF_SPI_ATTR_MOTO_S

[Definition]

```
typedef struct hiUNF_SPI_ATTR_MOTO_S
{
    HI_UNF_SPI_SPO_E enSpo;
    HI_UNF_SPI_SPH_E enSph;
}HI_UNF_SPI_ATTR_MOTO_S;
```

[Reason]

This data structure is added for setting the polarity and phase of the Motorola SPI protocol clock.

[Note]

None

[Example]

None

HI_UNF_SPI_ATTR_NM_S

[Definition]

```
typedef struct hiUNF_SPI_ATTR_NM_S
{
    HI_BOOL bWaitEn;
    HI_U32 u32Waitval;
}HI_UNF_SPI_ATTR_NM_S;
```

[Reason]

This data structure is added for setting the Microwire SPI protocol wait time.

[Note]

None

[Example]

None

HI_UNF_SPI_ATTR_EXT_U

[Definition]

```
typedef union
{
    HI_UNF_SPI_ATTR_MOTO_S stMoto;
    HI_UNF_SPI_ATTR_NM_S stNm;
```

```
}HI_UNF_SPI_ATTR_EXT_U;
```

[Reason]

This data structure is added for setting dedicated attributes for the Microwire SPI and Motorola SPI protocols.

[Note]

None

[Example]

None

HI_UNF_SPI_ATTR_S

[Definition]

```
typedef struct hiUNF_SPI_ATTR_S
{
    HI_UNF_SPI_DEV_E       enDev;
    HI_U32  u32Baud;
    HI_UNF_SPI_FRF_E       enFrf;
    HI_U32     u32Dss;
    HI_UNF_SPI_BIGEND_E enBigend;
    HI_UNF_SPI_ATTR_EXT_U unExtAttr;
}HI_UNF_SPI_ATTR_S;
```

[Reason]

This data structure is added for setting and obtaining the SPI working mode.

[Note]

None

[Example]

None

3.2.3 API

3.2.3.1 New APIs

HI_UNF_SPI_Open

[Definition]

```
HI_S32 HI_UNF_SPI_Open(HI_UNF_SPI_DEV_E enDev);
```

[Reason]

This API is added as required by the customer.

[Note]

None

[Example]

None

HI_UNF_SPI_Close

[Definition]

```
HI_S32 HI_UNF_SPI_Open(HI_UNF_SPI_DEV_E enDev);
```

[Reason]

This API is added as required by the customer.

[Note]

None

[Example]

None

HI_UNF_SPI_SetAttr

[Definition]

```
HI_S32 HI_UNF_SPI_SetAttr(HI_UNF_SPI_DEV_E enDev, HI_UNF_SPI_ATTR_S
*stAttr);
```

[Reason]

This API is added to set the SPI working mode.

[Note]

None

[Example]

None

HI_UNF_SPI_GetAttr

[Definition]

```
HI_S32 HI_UNF_SPI_SetAttr(HI_UNF_SPI_DEV_E enDev, HI_UNF_SPI_ATTR_S
*stAttr);
```

[Reason]

This API is added to obtain the SPI working mode and related attributes.

[Note]

None

[Example]

None

HI_UNF_SPI_Read

[Definition]

```
HI_S32 HI_UNF_SPI_Read(HI_UNF_SPI_DEV_E enDev, HI_U8 *pu8Read, HI_U32
u32ReadCnt);
```

[Reason]

This API is added to transmit SPI data.

[Note]

None

[Example]

None

HI_UNF_SPI_Write

[Definition]

```
HI_S32 HI_UNF_SPI_Write(HI_UNF_SPI_DEV_E enDev, HI_U8 *pu8Send, HI_U32
u32SendCnt);
```

[Reason]

This API is added to receive SPI data.

[Note]

None

[Example]

None

3.3 Frontend

3.3.1 Overview

The UNF 3.2.1 frontend module has the following changes compared with UNF 3.2.0:

- The sequence of the valid signal and sync signal in TS outputs is adjusted.
- The Hi3137 standby and wakeup function is added.
- The RAFAEL836, MXL608, and MXL214 drivers are supported.

3.3.1.1 Adjusting the Sequence of the Valid Signal and Sync Signal in TS Outputs

The data structure is changed as follows:

Data Structure

HI_UNF_TUNER_OUTPUT_TS_E is modified.

3.3.1.2 Adding the Hi3137 Standby and Wakeup Function

This feature implements the following functions:

- Enables a working Hi3137 to enter the low-power standby mode.
- Wakes the Hi3137 up from standby mode.

The data structure is changed as follows:

Data Structure

HI_UNF_TUNER_DEMOD_STATUS_E is added.

3.3.1.3 Supporting the RAFAEL836/MXL608/MXL214 Drivers

This feature implements the following functions:

- The DVB-T tuner RAFAEL836/MXL608 drivers are supported for adapting to the Hi3137.
- The DVB-C tuner MXL214 driver is supported.

The data structures are changed as follows:

Data Structure

The following data structures are modified:

- HI_UNF_TUNER_DEV_TYPE_E
- HI_UNF_DEMOD_DEV_TYPE_E

3.3.2 Data Structure

3.3.2.1 New Data Structure

HI_UNF_TUNER_DEMOD_STATUS_E

[Definition]

```
typedef enum hiUNF_TUNER_DEMOD_STATUS_E
{
    HI_UNF_TUNER_DEMODE_WAKE_UP = 0,
    HI_UNF_TUNER_DEMODE_STANDBY,
    HI_UNF_TUNER_DEMOD_STATUS_BUTT
}HI_UNF_TUNER_DEMOD_STATUS_E;
```

[Reason]

This data structure is added to support the Hi3137 standby and wakeup function.

[Note]

None

[Example]

See **sample/tuner/tuner_demo.c**.

3.3.2.2 Modified Data Structures

HI_UNF_TUNER_OUTPUT_TS_E

[Definition]

Before modification:

```
typedef enum hiUNF_TUNER_OUTPUT_TS_E
{
    HI_UNF_TUNER_OUTPUT_TSDAT0,
    HI_UNF_TUNER_OUTPUT_TSDAT1,
    HI_UNF_TUNER_OUTPUT_TSDAT2,
    HI_UNF_TUNER_OUTPUT_TSDAT3,
    HI_UNF_TUNER_OUTPUT_TSDAT4,
    HI_UNF_TUNER_OUTPUT_TSDAT5,
    HI_UNF_TUNER_OUTPUT_TSDAT6,
    HI_UNF_TUNER_OUTPUT_TSDAT7,
    HI_UNF_TUNER_OUTPUT_TSVLD,
    HI_UNF_TUNER_OUTPUT_TSSYNC,
    HI_UNF_TUNER_OUTPUT_TSERR,
    HI_UNF_TUNER_OUTPUT_BUTT
} HI_UNF_TUNER_OUTPUT_TS_E;
```

After modification:

```
typedef enum hiUNF_TUNER_OUTPUT_TS_E
{
    HI_UNF_TUNER_OUTPUT_TSDAT0,
    HI_UNF_TUNER_OUTPUT_TSDAT1,
    HI_UNF_TUNER_OUTPUT_TSDAT2,
    HI_UNF_TUNER_OUTPUT_TSDAT3,
    HI_UNF_TUNER_OUTPUT_TSDAT4,
    HI_UNF_TUNER_OUTPUT_TSDAT5,
    HI_UNF_TUNER_OUTPUT_TSDAT6,
    HI_UNF_TUNER_OUTPUT_TSDAT7,
    HI_UNF_TUNER_OUTPUT_TSSYNC,
    HI_UNF_TUNER_OUTPUT_TSVLD,
    HI_UNF_TUNER_OUTPUT_TSERR,
    HI_UNF_TUNER_OUTPUT_BUTT
} HI_UNF_TUNER_OUTPUT_TS_E;
```

[Reason]

The positions of HI_UNF_TUNER_OUTPUT_TSSYNC and HI_UNF_TUNER_OUTPUT_TSVLD are exchanged.

[Note]

None

HI_UNF_TUNER_DEV_TYPE_E

[Definition]

Before modification:

```
typedef enum   hiUNF_TUNER_DEV_TYPE_E
{
    HI_UNF_TUNER_DEV_TYPE_XG_3BL,
    HI_UNF_TUNER_DEV_TYPE_CD1616,
    HI_UNF_TUNER_DEV_TYPE_ALPS_TDAE,
    HI_UNF_TUNER_DEV_TYPE_TDCC,
    HI_UNF_TUNER_DEV_TYPE_TDA18250,
    HI_UNF_TUNER_DEV_TYPE_CD1616_DOUBLE,
    HI_UNF_TUNER_DEV_TYPE_MT2081,
    HI_UNF_TUNER_DEV_TYPE_TMX7070X,
    HI_UNF_TUNER_DEV_TYPE_R820C,
    HI_UNF_TUNER_DEV_TYPE_MXL203,
    HI_UNF_TUNER_DEV_TYPE_AV2011,
    HI_UNF_TUNER_DEV_TYPE_SHARP7903,
    HI_UNF_TUNER_DEV_TYPE_MXL101,
    HI_UNF_TUNER_DEV_TYPE_MXL603,
    HI_UNF_TUNER_DEV_TYPE_IT9170,
    HI_UNF_TUNER_DEV_TYPE_IT9133,
    HI_UNF_TUNER_DEV_TYPE_TDA6651,
    HI_UNF_TUNER_DEV_TYPE_TDA18250B,
    HI_UNF_TUNER_DEV_TYPE_M88TS2022,
    HI_UNF_TUNER_DEV_TYPE_RDA5815,
    HI_UNF_TUNER_DEV_TYPE_MXL254,
    HI_UNF_TUNER_DEV_TYPE_CXD2861,
    HI_UNF_TUNER_DEV_TYPE_SI2147,
    HI_UNF_TUNER_DEV_TYPE_BUTT
} HI_UNF_TUNER_DEV_TYPE_E ;
```

After modification:

```
typedef enum   hiUNF_TUNER_DEV_TYPE_E
{
    HI_UNF_TUNER_DEV_TYPE_XG_3BL,
    HI_UNF_TUNER_DEV_TYPE_CD1616,
    HI_UNF_TUNER_DEV_TYPE_ALPS_TDAE,
    HI_UNF_TUNER_DEV_TYPE_TDCC,
    HI_UNF_TUNER_DEV_TYPE_TDA18250,
    HI_UNF_TUNER_DEV_TYPE_CD1616_DOUBLE,
    HI_UNF_TUNER_DEV_TYPE_MT2081,
```

```
    HI_UNF_TUNER_DEV_TYPE_TMX7070X,
    HI_UNF_TUNER_DEV_TYPE_R820C,
    HI_UNF_TUNER_DEV_TYPE_MXL203,
    HI_UNF_TUNER_DEV_TYPE_AV2011,
    HI_UNF_TUNER_DEV_TYPE_SHARP7903,
    HI_UNF_TUNER_DEV_TYPE_MXL101,
    HI_UNF_TUNER_DEV_TYPE_MXL603,
    HI_UNF_TUNER_DEV_TYPE_IT9170,
    HI_UNF_TUNER_DEV_TYPE_IT9133,
    HI_UNF_TUNER_DEV_TYPE_TDA6651,
    HI_UNF_TUNER_DEV_TYPE_TDA18250B,
    HI_UNF_TUNER_DEV_TYPE_M88TS2022,
    HI_UNF_TUNER_DEV_TYPE_RDA5815,
    HI_UNF_TUNER_DEV_TYPE_MXL254,
    HI_UNF_TUNER_DEV_TYPE_CXD2861,
    HI_UNF_TUNER_DEV_TYPE_SI2147,
    HI_UNF_TUNER_DEV_TYPE_RAFAEL836,
    HI_UNF_TUNER_DEV_TYPE_MXL608,
    HI_UNF_TUNER_DEV_TYPE_MXL214,
    HI_UNF_TUNER_DEV_TYPE_BUTT
} HI_UNF_TUNER_DEV_TYPE_E ;
```

[Reason]

New members are added.

[Note]

RAFAEL836/MXL608 applies to the Hi3137. MXL214 can function as a tuner and a Demod.

[Example]

None

HI_UNF_DEMOD_DEV_TYPE_E

[Definition]

Before modification:

```
typedef enum   hiUNF_DEMOD_DEV_TYPE_E
{
    HI_UNF_DEMOD_DEV_TYPE_NONE,
    HI_UNF_DEMOD_DEV_TYPE_3130I = 0x100,
    HI_UNF_DEMOD_DEV_TYPE_3130E,
    HI_UNF_DEMOD_DEV_TYPE_J83B,
    HI_UNF_DEMOD_DEV_TYPE_AVL6211,
    HI_UNF_DEMOD_DEV_TYPE_MXL101,
    HI_UNF_DEMOD_DEV_TYPE_MN88472,
    HI_UNF_DEMOD_DEV_TYPE_IT9170,
```

```
    HI_UNF_DEMOD_DEV_TYPE_IT9133,
    HI_UNF_DEMOD_DEV_TYPE_3136,
    HI_UNF_DEMOD_DEV_TYPE_3136I,
    HI_UNF_DEMOD_DEV_TYPE_MXL254,
    HI_UNF_DEMOD_DEV_TYPE_CXD2837,
    HI_UNF_DEMOD_DEV_TYPE_3137,
    HI_UNF_DEMOD_DEV_TYPE_BUTT
} HI_UNF_DEMOD_DEV_TYPE_E;
```

After modification:

```
typedef enum   hiUNF_DEMOD_DEV_TYPE_E
{
    HI_UNF_DEMOD_DEV_TYPE_NONE,
    HI_UNF_DEMOD_DEV_TYPE_3130I = 0x100,
    HI_UNF_DEMOD_DEV_TYPE_3130E,
    HI_UNF_DEMOD_DEV_TYPE_J83B,
    HI_UNF_DEMOD_DEV_TYPE_AVL6211,
    HI_UNF_DEMOD_DEV_TYPE_MXL101,
    HI_UNF_DEMOD_DEV_TYPE_MN88472,
    HI_UNF_DEMOD_DEV_TYPE_IT9170,
    HI_UNF_DEMOD_DEV_TYPE_IT9133,
    HI_UNF_DEMOD_DEV_TYPE_3136,
    HI_UNF_DEMOD_DEV_TYPE_3136I,
    HI_UNF_DEMOD_DEV_TYPE_MXL254,
    HI_UNF_DEMOD_DEV_TYPE_CXD2837,
    HI_UNF_DEMOD_DEV_TYPE_3137,
    HI_UNF_DEMOD_DEV_TYPE_MXL214,
    HI_UNF_DEMOD_DEV_TYPE_BUTT
} HI_UNF_DEMOD_DEV_TYPE_E;
```

[Reason]

New members are added.

[Note]

MXL214 can function as a tuner and a Demod.

[Example]

None

4 Differences Between UNF 3.2.2 and UNF 3.2.1

4.1 HiGo

4.1.1 Overview

The UNF 3.2.2 HiGo has the following changes compared with UNF 3.2.1:

The text processing error code is added.

4.1.1.1 Adding the Text Processing Error Code

The text processing error code is added to facilitate fault location and rectification.

Data Structure

HIGO_TEXTOUT_ERR_S is modified.

API

None

4.1.2 Data Structure

4.1.2.1 New Data Structure

None

4.1.2.2 Modified Data Structure

HIGO_TEXTOUT_ERR_S

[Definition]

Before modification:

```
typedef enum
{
    ERR_TEXTOUT_INVRECT = 0,
```

```
    ERR_TEXTOUT_UNSUPPORT_CHARSET,
    ERR_TEXTOUT_ISUSING,
    ERR_TEXTOUT_BUTT
} HIGO_TEXTOUT_ERR_S;
```

After modification:

```
typedef enum
{
    ERR_TEXTOUT_INVRECT = 0,
    ERR_TEXTOUT_UNSUPPORT_CHARSET,
    ERR_TEXTOUT_ISUSING,
    ERR_TEXTOUT_NOIMPLEMENT,
    ERR_TEXTOUT_SHAPE,
    ERR_TEXTOUT_MAX_CHAR,
    ERR_TEXTOUT_CHAR_SET,
    ERR_TEXTOUT_BIDI,
    ERR_TEXTOUT_ERRCODE_MAX = 0x1F,
    ERR_TEXTOUT_INTERNAL,
    ERR_TEXTOUT_BUTT
} HIGO_TEXTOUT_ERR_S;
```

4.2 HiFB

4.2.1 Overview

The UNF 3.2.2 HiFB has the following changes compared with UNF 3.2.1:

The fence sync for display function is added.

4.2.1.1 Adding the API for Fence Sync for Display

This function is added to avoid crack display issues and increase the UI frame rate (smoothness). The following data structure and API are changed:

Data Structure

HIFB_HWC_LAYERINFO_S is added.

API

FBIO_HWC_REFRESH is added.

4.2.2 Data Structure

4.2.2.1 New Data Structure

HIFB_HWC_LAYERINFO_S

[Definition]

```
typedef struct
{
    HIFB_POINT_S stPos;
    HIFB_RECT    stInRect;
    HI_U32       u32LayerAddr;
    HI_U32       u32Stride;
    HI_U32       u32Alpha;
    HI_BOOL      bPreMul;
    HIFB_COLOR_FMT_E eFmt;
    HI_S32 s32AcquireFenceFd;
    HI_S32 s32ReleaseFenceFd;
}HIFB_HWC_LAYERINFO_S;
```

[Reason]

This structure is added to avoid crack display issues and increase the UI frame rate (smoothness).

[Note]

None

[Example]

None

4.2.3 API

4.2.3.1 New API

FBIO_HWC_REFRESH

[Definition]

```
#define FBIO_HWC_REFRESH _IOR(IOC_TYPE_HIFB, 146, HIFB_HWC_LAYERINFO_S*)
```

[Reason]

This API is added to avoid crack display issues and increase the UI frame rate (smoothness).

[Note]

None

[Example]

None

4.3 MCE

4.3.1 Overview

The UNF 3.2.2 MCE has the following changes compared with UNF 3.2.1:

The function of querying whether the playback data is valid is added.

4.3.1.1 Querying Playback Data Validity

This function allows the user to query the playback data validity to determine whether to update the fastplay partition. The data structure is changed as follows:

Data Structure

HI_UNF_MCE_PLAY_PARAM_S is modified.

4.3.2 Data Structure

4.3.2.1 Modified Data Structure

HI_UNF_MCE_PLAY_PARAM_S

[Definition]

Before modification:

```
typedef struct hiUNF_MCE_PLAY_PARAM_S
{
    HI_UNF_MCE_PLAY_TYPE_E         enPlayType;
    HI_BOOL                       bPlayEnable;
    union
    {
        HI_UNF_MCE_DVB_PARAM_S     stDvbParam;
        HI_UNF_MCE_TSFILE_PARAM_S  stTsParam;
        HI_UNF_MCE_ANI_PARAM_S     stAniParam;
    }unParam;
}HI_UNF_MCE_PLAY_PARAM_S;
```

After modification:

```
typedef struct hiUNF_MCE_PLAY_PARAM_S
{
    HI_UNF_MCE_PLAY_TYPE_E         enPlayType;
    HI_BOOL                       bPlayEnable;
    union
    {
        HI_UNF_MCE_DVB_PARAM_S     stDvbParam;
        HI_UNF_MCE_TSFILE_PARAM_S  stTsParam;
        HI_UNF_MCE_ANI_PARAM_S     stAniParam;
```

```
    }unParam;
    HI_BOOL                     bContentValid;
}HI_UNF_MCE_PLAY_PARAM_S;
```

[Description]

The playback data validity is identified so that the user can determine whether to update the fastplay partition.

[Note]

None

[Example]

None

4.4 PDM

4.4.1 Overview

The UNF 3.2.2 PDM has the following changes compared with UNF 3.2.1:

The functions of obtaining and updating the parameters and data of the fastplay backup partition are added.

4.4.1.1 Obtaining and Updating the Parameters and Data of the Fastplay Backup Partition

The APIs are changed as follows:

APIs

The following APIs are added:

- HI_UNF_PDM_GetPlayBakParam
- HI_UNF_PDM_UpdatePlayBakParam
- HI_UNF_PDM_GetPlayBakContent
- HI_UNF_PDM_UpdatePlayBakContent

4.4.2 API

4.4.2.1 New APIs

HI_UNF_PDM_GetPlayBakParam

[Definition]

```
HI_S32  HI_UNF_PDM_GetPlayBakParam(HI_UNF_MCE_PLAY_PARAM_S *pstPlayParam);
```

[Reason]

This API is added for obtaining the parameters of the fastplay backup partition.

[Note]

None

[Example]

None

HI_UNF_PDM_UpdatePlayBakParam

[Definition]

```
HI_S32  HI_UNF_PDM_UpdatePlayBakParam(HI_UNF_MCE_PLAY_PARAM_S
*pstPlayParam);
```

[Reason]

This API is added for updating the parameters of the fastplay backup partition.

[Note]

None

[Example]

None

HI_UNF_PDM_GetPlayBakContent

[Definition]

```
HI_S32  HI_UNF_PDM_GetPlayBakContent(HI_U8 *pu8Content, HI_U32 u32Size);
```

[Reason]

This API is added for obtaining the data of the fastplay backup partition.

[Note]

None

[Example]

None

HI_UNF_PDM_UpdatePlayBakContent

[Definition]

```
HI_S32  HI_UNF_PDM_UpdatePlayBakContent(HI_U8 *pu8Content, HI_U32
u32Size);
```

[Reason]

This API is added for updating the data of the fastplay backup partition.

[Note]

None

[Example]

None

4.5 VO

4.5.1 Overview

The UNF 3.2.2 VO module has the following changes compared with UNF 3.2.1:

The 4K standards are supported.

4.5.1.1 Supporting the 4K Standards

The 4K standards can be configured by calling HI_UNF_DISP_SetFormat. The data structure is changed as follows:

Data Structure

HI_UNF_ENC_FMT_E is modified.

4.5.2 Data Structure

4.5.2.1 Modified Data Structure

HI_UNF_ENC_FMT_E

[Definition]

Before modification:

```
typedef enum hiUNF_ENC_FMT_E
{
    ...
    HI_UNF_ENC_FMT_VESA_2560X1440_60_RB,
    HI_UNF_ENC_FMT_VESA_2560X1600_60_RB,
    HI_UNF_ENC_FMT_BUTT
}HI_UNF_ENC_FMT_E;
```

After modification:

```
typedef enum hiUNF_ENC_FMT_E
{
    ...
    HI_UNF_ENC_FMT_VESA_2560X1440_60_RB,
    HI_UNF_ENC_FMT_VESA_2560X1600_60_RB,
    HI_UNF_ENC_FMT_3840X2160_24 = 0x100,
    HI_UNF_ENC_FMT_3840X2160_25,
    HI_UNF_ENC_FMT_3840X2160_30,
    HI_UNF_ENC_FMT_4096X2160_24,
```

```
    HI_UNF_ENC_FMT_BUTT
}HI_UNF_ENC_FMT_E;
```

[Description]

The 4K standards are added to the standard enumerations.

[Note]

None

[Example]

None

4.6 Sound

4.6.1 Overview

The UNF 3.2.2 sound module has the following changes compared with UNF 3.2.1:

- The function of setting and obtaining the SPDIF category code is added.
- The function of setting and obtaining the intelligent volume based on the track is added.
- The SPDIF SCMS structure is modified.

4.6.1.1 Setting and Obtaining the SPDIF Category Code

Data Structure

HI_UNF_SND_SPDIF_CATEGORYCODE_E is added.

API

The following APIs are added:

- HI_UNF_SND_SetSpdifCategoryCode
- HI_UNF_SND_GetSpdifCategoryCode

4.6.1.2 Setting and Obtaining the Intelligent Volume Based on the Track

API

The following APIs are added:

- HI_UNF_SND_SetTrackSmartVolume
- HI_UNF_SND_GetTrackSmartVolume

4.6.1.3 Modifying the SPDIF SCMS Structure

Data Structure

HI_UNF_SND_SPDIF_SCMSMODE_E is modified.

4.6.2 Data Structure

4.6.2.1 New Data Structure

HI_UNF_SND_SPDIF_CATEGORYCODE_E

[Definition]

```
typedef enum hiHI_UNF_SND_SPDIF_CATEGORYCODE_E
{
    HI_UNF_SND_SPDIF_CATEGORY_GENERAL = 0x00,
    HI_UNF_SND_SPDIF_CATEGORY_BROADCAST_JP = 0x10,
    HI_UNF_SND_SPDIF_CATEGORY_BROADCAST_USA,
    HI_UNF_SND_SPDIF_CATEGORY_BROADCAST_EU,
    HI_UNF_SND_SPDIF_CATEGORY_PCM_CODEC = 0x20,
    HI_UNF_SND_SPDIF_CATEGORY_DIGITAL_SNDSAMPLER,
    HI_UNF_SND_SPDIF_CATEGORY_DIGITAL_MIXER,
    HI_UNF_SND_SPDIF_CATEGORY_DIGITAL_SNDPROCESSOR,
    HI_UNF_SND_SPDIF_CATEGORY_SRC,
    HI_UNF_SND_SPDIF_CATEGORY_MD = 0x30,
    HI_UNF_SND_SPDIF_CATEGORY_DVD,
    HI_UNF_SND_SPDIF_CATEGORY_SYNTHESISER = 0x40,
    HI_UNF_SND_SPDIF_CATEGORY_MIC,
    HI_UNF_SND_SPDIF_CATEGORY_DAT = 0x50,
    HI_UNF_SND_SPDIF_CATEGORY_DCC,
    HI_UNF_SND_SPDIF_CATEGORY_VCR,
    HI_UNF_SND_SPDIF_CATEGORY_BUTT
} HI_UNF_SND_SPDIF_CATEGORYCODE_E;
```

[Reason]

The SPDIF category code is added.

[Note]

None

[Example]

None

4.6.2.2 Modified Data Structure

HI_UNF_SND_SPDIF_SCMSMODE_E

[Definition]

Before modification:

```
typedef enum hiHI_UNF_SND_SPDIF_SCMSMODE_E
{
```

```
    HI_UNF_SND_SPDIF_SCMSMODE_COPYALLOW,
    HI_UNF_SND_SPDIF_SCMSMODE_COPYONCE,
    HI_UNF_SND_SPDIF_SCMSMODE_COPYPROHIBITED,
    HI_UNF_SND_SPDIF_SCMSMODE_BUTT
} HI_UNF_SND_SPDIF_SCMSMODE_E;
```

After modification:

```
typedef enum hiHI_UNF_SND_SPDIF_SCMSMODE_E
{
    HI_UNF_SND_SPDIF_SCMSMODE_COPYALLOW,
    HI_UNF_SND_SPDIF_SCMSMODE_COPYONCE,
    HI_UNF_SND_SPDIF_SCMSMODE_COPYNOMORE,
    HI_UNF_SND_SPDIF_SCMSMODE_COPYPROHIBITED,
    HI_UNF_SND_SPDIF_SCMSMODE_BUTT
} HI_UNF_SND_SPDIF_SCMSMODE_E;
```

[Description]

The non-copy mode is added.

[Note]

None

[Example]

None

4.6.3 API

4.6.3.1 New APIs

HI_UNF_SND_SetSpdifCategoryCode

[Definition]

```
HI_S32 HI_UNF_SND_SetSpdifCategoryCode(HI_UNF_SND_E enSound,
HI_UNF_SND_OUTPUTPORT_E enOutPort, HI_UNF_SND_SPDIF_CATEGORYCODE_E
enSpdifCategoryCode);
```

[Reason]

This API is added for setting the SPDIF category code.

[Note]

None

[Example]

None

HI_UNF_SND_GetSpdifCategoryCode

[Definition]

```
HI_S32 HI_UNF_SND_GetSpdifCategoryCode(HI_UNF_SND_E enSound,
HI_UNF_SND_OUTPUTPORT_E enOutPort, HI_UNF_SND_SPDIF_CATEGORYCODE_E
*penSpdifCategoryCode);
```

[Reason]

This API is added for obtaining the SPDIF category code.

[Note]

None

[Example]

None

HI_UNF_SND_SetTrackSmartVolume

[Definition]

```
HI_S32 HI_UNF_SND_SetTrackSmartVolume(HI_HANDLE hTrack, HI_BOOL bEnable);
```

[Reason]

This API is added for setting the intelligent volume.

[Note]

None

[Example]

None

HI_UNF_SND_GetTrackSmartVolume

[Definition]

```
HI_S32  HI_UNF_SND_GetAllTrackMute(HI_UNF_SND_E enSound, HI_BOOL *pbMute);
```

[Reason]

This API is added for obtaining the intelligent volume.

[Note]

None

[Example]

None

4.7 SPI

4.7.1 Overview

The UNF 3.2.2 SPI module has the following changes compared with UNF 3.2.1:

The function of controlling the SPI chip select (CS) signal is added.

4.7.1.1 Controlling the SPI CS Signal

Data Structure

- HI_UNF_SPI_DEV_E is added.
- HI_UNF_SPI_ATTR_S is modified.

API

HI_UNF_SPI_ReadExt is added.

4.7.2 Data Structure

4.7.2.1 New Data Structure

HI_UNF_SPI_CFGCS_E

[Definition]

```
typedef enum hiUNF_SPI_CFGCS_E
{
   HI_UNF_SPI_LOGIC_CS =0 ,
   HI_UNF_SPI_GPIO_CS = 1
}HI_UNF_SPI_CFGCS_E;
```

[Reason]

During communications, some SPI flash memories require that the CS signal retain low level when some commands are being sent or data is being received. However, the logic CS cannot meet the requirement. Therefore, the CS signal must be set to GPIO mode to meet the communication requirement.

[Note]

None

[Example]

None

4.7.2.2 Modified Data Structure

HI_UNF_SPI_ATTR_S

[Definition]

Before modification:

```
typedef struct hiUNF_SPI_ATTR_S
{
    HI_UNF_SPI_DEV_E       enDev;
    HI_U32                          u32Baud;
    HI_UNF_SPI_FRF_E       enFrf;
    HI_U32      u32Dss;
    HI_UNF_SPI_BIGEND_E enBigend;
    HI_UNF_SPI_ATTR_EXT_U unExtAttr;
}HI_UNF_SPI_ATTR_S;
```

After modification:

```
typedef struct hiUNF_SPI_ATTR_S
{
    HI_UNF_SPI_DEV_E       enDev;
    HI_UNF_SPI_CFGCS_E csCfg;
    HI_U32                          u32Baud;
    HI_UNF_SPI_FRF_E       enFrf;
    HI_U32      u32Dss;
    HI_UNF_SPI_BIGEND_E enBigend;
    HI_UNF_SPI_ATTR_EXT_U un ExtAttr;
}HI_UNF_SPI_ATTR_S;
```

[Reason]

The member for controlling the SPI CS signal is added.

[Note]

None

[Example]

None

4.7.3 API

4.7.3.1 New API

HI_UNF_SPI_ReadExt

[Definition]

```
HI_S32 HI_UNF_SPI_ReadExt(HI_UNF_SPI_DEV_E enDev, HI_U8 *pu8Send, HI_U32
u32SendCnt, HI_U8 *pu8Read, HI_U32 u32ReadCnt);
```

[Reason]

This API is added as required by the customer.

[Note]

If the CS signal is controlled by the GPIO, and commands are sent by calling HI_UNF_SPI_Write and data is received by calling HI_UNF_SPI_Read, the CS signal retains high level when it is idle, because any SPI flash requires this kind of communication mode. HI_UNF_SPI_ReadExt can be called if some SPI flash requires that the CS signal retain low level to receive data after some commands are sent.

[Example]

None

4.8 PQ

4.8.1 Overview

The UNF 3.2.2 PQ module has the following changes compared with UNF 3.2.1:

- The function of setting and obtaining a single PQ parameter is added.
- The SR demonstration function is added.
- The color enhancement function is added.
- The dynamic contrast enhancement function is added.
- The function of enabling/disabling the PQ algorithm is added.
- The function of enabling/disabling the demo mode is added.

4.8.1.1 Setting and Obtaining a Single PQ Parameter

APIs are added to use the PQ parameters flexibly.

API

The following APIs are added:

- HI_UNF_PQ_GetBrightness
- HI_UNF_PQ_SetBrightness
- HI_UNF_PQ_GetContrast
- HI_UNF_PQ_SetContrast
- HI_UNF_PQ_GetSaturation
- HI_UNF_PQ_SetSaturation
- HI_UNF_PQ_GetHue
- HI_UNF_PQ_SetHue
- HI_UNF_PQ_GetSharpness
- HI_UNF_PQ_SetSharpness

4.8.1.2 Adding the SR Demonstration Function

This function is added to enable 4K outputs for non-4K sources. The data structure and APIs are changed as follows:

Data Structure

HI_UNF_PQ_SR_DEMO_E is added.

API

The following APIs are added:

- HI_UNF_PQ_GetSRMode
- HI_UNF_PQ_SetSRMode

4.8.1.3 Adding the Color Enhancement Function

The data structures and APIs are changed to enhance the color effects and adapt to the UI menu.

Data Structure

The following data structures are added:

- HI_UNF_PQ_COLOR_ENHANCE_E
- HI_UNF_PQ_FLESHTONE_E
- HI_UNF_PQ_SIX_BASE_S
- HI_UNF_PQ_COLOR_SPEC_MODE_E
- HI_UNF_PQ_COLOR_ENHANCE_S

API

The following APIs are added:

- HI_UNF_PQ_GetColorGain
- HI_UNF_PQ_SetColorGain
- HI_UNF_PQ_GetColorEnhanceParam
- HI_UNF_PQ_SetColorEnhanceParam

4.8.1.4 Adding the Dynamic Contrast Enhancement Function

APIs are added to enhance the dynamic image contrast.

API

The following APIs are added:

- HI_UNF_PQ_GetDynamicContrast
- HI_UNF_PQ_SetDynamicContrast

4.8.1.5 Enabling/Disabling the PQ Algorithm

The data structure and APIs are changed to enable/disable the PQ algorithm.

Data Structure

HI_UNF_PQ_MODULE_E is added.

API

The following APIs are added:

- HI_UNF_PQ_SetPQModule
- HI_UNF_PQ_GetPQModule

4.8.1.6 Enabling/Disabling the Demo Mode

The data structure and API are changed to demonstrate the effect on picture quality when the display algorithm is enabled/disabled.

Data Structure

HI_UNF_PQ_DEMO_E is added.

API

HI_UNF_PQ_SetDemo is added.

4.8.2 Data Structure

4.8.2.1 New Data Structure

HI_UNF_PQ_DEMO_E

[Definition]

```
typedef enum hiUNF_PQ_DEMO_E
{
    HI_UNF_PQ_DEMO_SHARPNESS = 0,
    HI_UNF_PQ_DEMO_DCI,
    HI_UNF_PQ_DEMO_COLOR,
    HI_UNF_PQ_DEMO_SR,
    HI_UNF_PQ_DEMO_ALL,

    HI_UNF_PQ_DEMO_BUTT
}  HI_UNF_PQ_DEMO_E;
```

[Reason]

This data structure is added to support the demo mode.

[Note]

None

[Example]

None

HI_UNF_PQ_MODULE_E

[Definition]

```
typedef enum hiUNF_PQ_MODULE_E
{
```

```
    HI_UNF_PQ_MODULE_SHARPNESS = 0,
    HI_UNF_PQ_MODULE_DCI,
    HI_UNF_PQ_MODULE_COLOR,
    HI_UNF_PQ_MODULE_SR,
    HI_UNF_PQ_MODULE_ALL,

    HI_UNF_PQ_MODULE_BUTT
}  HI_UNF_PQ_MODULE_E;
```

[Reason]

This data structure is added to support the function of enabling/disabling the PQ algorithm.

[Note]

None

[Example]

None

HI_UNF_PQ_SR_DEMO_E

[Definition]

```
typedef enum hiUNF_PQ_SR_DEMO_E
{
    HI_UNF_PQ_SR_DISABLE  = 0,
    HI_UNF_PQ_SR_ENABLE_R,
    HI_UNF_PQ_SR_ENABLE_L,
    HI_UNF_PQ_SR_ENABLE_A,

    HI_UNF_PQ_SR_DEMO_BUTT
}HI_UNF_PQ_SR_DEMO_E;
```

[Reason]

This data structure is added to control the area in which the SR algorithm takes effect.

[Note]

None

[Example]

None

HI_UNF_PQ_SIX_BASE_S

[Definition]

```
typedef struct  hiUNF_PQ_SIX_BASE_S
{
   HI_U32  u32Red;
```

```
    HI_U32  u32Green;
    HI_U32  u32Blue;

    HI_U32  u32Cyan;
    HI_U32  u32Magenta;
    HI_U32  u32Yellow;
}HI_UNF_PQ_SIX_BASE_S;
```

[Reason]

This data structure is added to allow customized adjustment of color components.

[Note]

None

[Example]

None

HI_UNF_PQ_FLESHTONE_E

[Definition]

```
typedef enum hiUNF_PQ_FLESHTONE_E
{
    HI_UNF_PQ_FLESHTONE_GAIN_OFF = 0,
    HI_UNF_PQ_FLESHTONE_GAIN_LOW,
    HI_UNF_PQ_FLESHTONE_GAIN_MID,
    HI_UNF_PQ_FLESHTONE_GAIN_HIGH,

    HI_UNF_PQ_FLESHTONE_GAIN_BUTT
}  HI_UNF_PQ_FLESHTONE_E;
```

[Reason]

This data structure is added to allow adjustment of complexion enhancement levels.

[Note]

None

[Example]

None

HI_UNF_PQ_COLOR_ENHANCE_E

[Definition]

```
typedef enum hiUNF_PQ_COLOR_ENHANCE_E
{
    HI_UNF_PQ_COLOR_ENHANCE_FLESHTONE = 0,
    HI_UNF_PQ_COLOR_ENHANCE_SIX_BASE,
```

```
    HI_UNF_PQ_COLOR_ENHANCE_SPEC_COLOR_MODE,
    HI_UNF_PQ_COLOR_ENHANCE_BUTT
} HI_UNF_PQ_COLOR_ENHANCE_E;
```

[Reason]

This data structure is added to allow configuration of color enhancement types.

[Note]

None

[Example]

None

HI_UNF_PQ_COLOR_SPEC_MODE_E

[Definition]

```
typedef enum hiUNF_PQ_COLOR_SPEC_MODE_E
{
    HI_UNF_PQ_COLOR_MODE_RECOMMEND = 0,
    HI_UNF_PQ_COLOR_MODE_BLUE,
    HI_UNF_PQ_COLOR_MODE_GREEN,
    HI_UNF_PQ_COLOR_MODE_BG,
    HI_UNF_PQ_COLOR_MODE_BUTT
} HI_UNF_PQ_COLOR_SPEC_MODE_E;
```

[Reason]

This data structure is added to allow adaptation to color enhancement types.

[Note]

None

[Example]

None

HI_UNF_PQ_COLOR_ENHANCE_S

[Definition]

```
typedef struct  hiUNF_PQ_COLOR_ENHANCE_S
{
    HI_UNF_PQ_COLOR_ENHANCE_E    enColorEnhanceType;
    union
    {
        HI_UNF_PQ_FLESHTONE_E    enFleshtone;
        HI_UNF_PQ_SIX_BASE_S     stSixBase;
        HI_UNF_PQ_COLOR_SPEC_MODE_E   enColorMode;
```

```
    } unColorGain;
} HI_UNF_PQ_COLOR_ENHANCE_S;
```

[Reason]

This data structure is added to provide color enhancement parameters.

[Note]

None

[Example]

None

4.8.3 API

4.8.3.1 New APIs

HI_UNF_PQ_GetBrightness

[Definition]

```
extern HI_S32 HI_UNF_PQ_GetBrightness(HI_UNF_DISP_E enChan, HI_U32
*pu32Brightness);
```

[Reason]

This API is added for obtaining the brightness on the Android UI.

[Note]

None

[Example]

None

HI_UNF_PQ_SetBrightness

[Definition]

```
extern HI_S32 HI_UNF_PQ_SetBrightness(HI_UNF_DISP_E enChan, HI_U32
u32Brightness);
```

[Reason]

This API is added for setting the brightness on the Android UI.

[Note]

The brightness value ranges from 0 to 100.

[Example]

None

HI_UNF_PQ_GetContrast

[Definition]

```
extern HI_S32 HI_UNF_PQ_GetContrast(HI_UNF_DISP_E enChan, HI_U32
*pu32Contrast);
```

[Reason]

This API is added for obtaining the contrast on the Android UI.

[Note]

None

[Example]

None

HI_UNF_PQ_SetContrast

[Definition]

```
extern HI_S32 HI_UNF_PQ_SetContrast(HI_UNF_DISP_E enChan, HI_U32
u32Contrast);
```

[Reason]

This API is added for setting the contrast on the Android UI.

[Note]

The contrast value ranges from 0 to 100.

[Example]

None

HI_UNF_PQ_GetSaturation

[Definition]

```
extern HI_S32 HI_UNF_PQ_GetSaturation(HI_UNF_DISP_E enChan, HI_U32
*pu32Saturation);
```

[Reason]

This API is added for obtaining the saturation on the Android UI.

[Note]

None

[Example]

None

HI_UNF_PQ_SetSaturation

[Definition]

```
extern HI_S32 HI_UNF_PQ_SetSaturation(HI_UNF_DISP_E enChan, HI_U32
u32Saturation);
```

[Reason]

This API is added for setting the saturation on the Android UI.

[Note]

The saturation value ranges from 0 to 100.

[Example]

None

HI_UNF_PQ_GetHue

[Definition]

```
extern HI_S32 HI_UNF_PQ_GetHue(HI_UNF_DISP_E enChan, HI_U32 *pu32Hue);
```

[Reason]

This API is added for obtaining the hue on the Android UI.

[Note]

None

[Example]

None

HI_UNF_PQ_SetHue

[Definition]

```
extern HI_S32 HI_UNF_PQ_SetHue(HI_UNF_DISP_E enChan, HI_U32 u32Hue);
```

[Reason]

This API is added for setting the hue on the Android UI.

[Note]

The hue value ranges from 0 to 100.

[Example]

None

HI_UNF_PQ_GetSRMode

[Definition]

```
extern HI_S32 HI_UNF_PQ_GetSRMode(HI_UNF_DISP_E enChan,
HI_UNF_PQ_SR_DEMO_E *penType);
```

[Reason]

This API is added for obtaining the SR mode on the Android UI.

[Note]

None

[Example]

None

HI_UNF_PQ_SetSRMode

[Definition]

```
extern HI_S32 HI_UNF_PQ_SetSRMode(HI_UNF_DISP_E enChan,
HI_UNF_PQ_SR_DEMO_E enType);
```

[Reason]

This API is added for setting the SR mode on the Android UI.

[Note]

None

[Example]

None

HI_UNF_PQ_GetSharpness

[Definition]

```
extern HI_S32 HI_UNF_PQ_GetSharpness(HI_UNF_DISP_E enChan, HI_U32
*pu32Sharpness);
```

[Reason]

This API is added for obtaining the sharpness on the Android UI.

[Note]

None

[Example]

None

HI_UNF_PQ_SetSharpness

[Definition]

```
extern HI_S32 HI_UNF_PQ_SetSharpness(HI_UNF_DISP_E enChan, HI_U32
u32Sharpness);
```

[Reason]

This API is added for setting the sharpness on the Android UI.

[Note]

The sharpness value ranges from 0 to 100.

[Example]

None

HI_UNF_PQ_GetColorGain

[Definition]

```
extern HI_S32 HI_UNF_PQ_GetColorGain(HI_UNF_DISP_E enChan, HI_U32
*pu32ColorGainLevel);
```

[Reason]

This API is added for obtaining the color gain.

[Note]

None

[Example]

None

HI_UNF_PQ_SetColorGain

[Definition]

```
extern HI_S32 HI_UNF_PQ_SetColorGain(HI_UNF_DISP_E enChan, HI_U32
u32ColorGainLevel);
```

[Reason]

This API is added for setting the color gain.

[Note]

The color gain ranges from 0 to 100.

[Example]

None

HI_UNF_PQ_GetColorEnhanceParam

[Definition]

```
extern HI_S32 HI_UNF_PQ_GetColorEnhanceParam(HI_UNF_PQ_COLOR_ENHANCE_S
*pstColorEnhanceParam);
```

[Reason]

This API is added for obtaining the type and strength of color enhancement on the Android UI.

[Note]

None

[Example]

None

HI_UNF_PQ_SetColorEnhanceParam

[Definition]

```
extern HI_S32 HI_UNF_PQ_SetColorEnhanceParam(HI_UNF_PQ_COLOR_ENHANCE_S
stColorEnhanceParam);
```

[Reason]

This API is added for setting the type and strength of color enhancement on the Android UI.

[Note]

None

[Example]

None

HI_UNF_PQ_GetDynamicContrast

[Definition]

```
extern HI_S32 HI_UNF_PQ_GetDynamicContrast(HI_U32 *pu32DCIlevel);
```

[Reason]

This API is added for obtaining the DCI level on the Android UI.

[Note]

None

[Example]

None

HI_UNF_PQ_SetDynamicContrast

[Definition]

```
extern HI_S32 HI_UNF_PQ_SetDynamicContrast(HI_U32 u32DCIlevel);
```

[Reason]

This API is added for setting the DCI level on the Android UI.

[Note]

The DCI level value ranges from 0 to 100.

[Example]

None

HI_UNF_PQ_SetPQModule

[Definition]

```
extern HI_S32 HI_UNF_PQ_SetPQModule( HI_UNF_PQ_MODULE_E enFlags, HI_U32
u32OnOff);
```

[Reason]

This API is added for enabling/disabling the PQ algorithm on the Android UI.

[Note]

None

[Example]

None

HI_UNF_PQ_GetPQModule

[Definition]

```
extern HI_S32 HI_UNF_PQ_GetPQModule( HI_UNF_PQ_MODULE_E enFlags, HI_U32
*pu32OnOff);
```

[Reason]

This API is added for obtaining the enable status of the PQ algorithm on the Android UI.

[Note]

None

[Example]

None

HI_UNF_PQ_SetDemo

[Definition]

```
extern HI_S32 HI_UNF_PQ_SetDemo( HI_UNF_PQ_DEMO_E enFlags, HI_U32
u32OnOff);
```

[Reason]

This API is added for enabling/disabling the demo mode on the Android UI.

[Note]

None

[Example]

None

5 Differences Between UNF 3.2.3 and UNF 3.2.2

5.1 Demux

5.1.1 Overview

The UNF 3.2.3 Demux has the following changes compared with UNF 3.2.2:

The tag deal feature is supported.

5.1.1.1 Supporting the Tag Deal Feature

The data structures and APIs are changed as follows:

Data Structure

The following data structures are added:

- HI_UNF_DMX_TAG_SYNC_MODE_E
- HI_UNF_DMX_TAG_ATTR_S

API

The following APIs are added:

- HI_UNF_DMX_GetDmxTagAttr
- HI_UNF_DMX_SetDmxTagAttr

5.1.2 Data Structure

5.1.2.1 New Data Structures

HI_UNF_DMX_TAG_SYNC_MODE_E

[Definition]

```
typedef enum hiUNF_DMX_TAG_SYNC_MODE_E
{
```

```
    HI_UNF_DMX_TAG_HEAD_SYNC = 0x0,
    HI_UNF_DMX_NORMAL_HEAD_SYNC = 0x1,
}HI_UNF_DMX_TAG_SYNC_MODE_E;
```

[Reason]

This data structure is added to provide the enumerations of supported tag sync modes.

[Note]

None

[Example]

None

HI_UNF_DMX_TAG_ATTR_S

[Definition]

```
typedef struct hiUNF_DMX_TAG_ATTR_S
{
    HI_U8    au8Tag[MAX_TAG_LENGTH];
    HI_U32   u32TagLen;
    HI_BOOL bEnabled;
    HI_UNF_DMX_TAG_SYNC_MODE_E enSyncMod;
}HI_UNF_DMX_TAG_ATTR_S;
```

[Reason]

This data structure is added to define the configurable attribute parameters of the tag.

[Note]

None

[Example]

None

5.1.3 API

5.1.3.1 New APIs

HI_UNF_DMX_GetDmxTagAttr

[Definition]

```
HI_S32 HI_UNF_DMX_GetDmxTagAttr(HI_U32 u32DmxId, HI_UNF_DMX_TAG_ATTR_S
*pstAttr);
```

[Reason]

This API is added for obtaining the current tag attributes.

[Note]

None

[Example]

None

HI_UNF_DMX_SetDmxTagAttr

[Definition]

```
HI_S32 HI_UNF_DMX_SetDmxTagAttr(HI_U32 u32DmxId, HI_UNF_DMX_TAG_ATTR_S
*pstAttr);
```

[Reason]

This API is added for setting the tag attributes.

[Note]

None

[Example]

None

6 Differences Between UNF 3.2.4 and UNF 3.2.3

6.1 Demux

6.1.1 Overview

The UNF 3.2.4 Demux has the following changes compared with UNF 3.2.3:

The function of controlling the TSO rate by using backpressure when data is output from the RAM port is added.

6.1.1.1 Controlling the TSO Rate by Using Backpressure When Data Is Output from the RAM Port

The data structures are changed as follows:

Data Structure

- HI_UNF_DMX_TSI_ATTACH_TSO_S is added.
- HI_UNF_DMX_INVOKE_TYPE_E is modified.

6.1.2 Data Structure

6.1.2.1 New Data Structure

HI_UNF_DMX_TSI_ATTACH_TSO_S

[Definition]

```
typedef struct hiUNF_DMX_TSI_ATTACH_TSO_S
{
   HI_UNF_DMX_PORT_E      enTSI;
   HI_UNF_DMX_TSO_PORT_E   enTSO;
}HI_UNF_DMX_TSI_ATTACH_TSO_S;
```

[Reason]

This data structure is added for identifying the TSO and TSI interfaces.

[Note]

None

[Example]

None

6.1.2.2 Modified Data Structure

HI_UNF_DMX_INVOKE_TYPE_E

[Definition]

```
typedef enum hiUNF_DMX_INVOKE_TYPE
{
    HI_UNF_DMX_INVOKE_TYPE_CHAN_CC_REPEAT_SET
    HI_UNF_DMX_INVOKE_TYPE_PUSI_SET,
    HI_UNF_DMX_INVOKE_TYPE_TEI_SET,
    HI_UNF_DMX_INVOKE_TYPE_TSI_ATTACH_TSO,
    HI_UNF_DMX_INVOKE_TYPE_BUTT
} HI_UNF_DMX_INVOKE_TYPE_E;
```

[Reason]

The command **HI_UNF_DMX_INVOKE_TYPE_TSI_ATTACH_TSO** supported by HI_UNF_DMX_Invoke is added for binding the TSI input to a TSO.

[Note]

None

[Example]

None

6.2 Frontend

6.2.1 Overview

The UNF 3.2.4 frontend module has the following changes compared with UNF 3.2.3:

- The function of automatically detecting the DVB-T or DVB-T2 signals is added.
- The function of blindly scanning unicable user frequencies is added.

6.2.1.1 Automatically Detecting the DVB-T or DVB-T2 Signals

Data Structure

HI_UNF_TUNER_SAMPLE_DATALEN_E is modified.

6.2.1.2 Blindly Scanning Unicable User Frequencies

Data Structure

HI_UNF_TUNER_SCR_UB_S is added.

API

The following APIs are added:

- HI_UNF_TUNER_GetAgc
- HI_UNF_UNICABLE_ScanAndInstall_UB
- HI_UNF_UNICABLE_GetUBInfo
- HI_UNF_UNICABLE_SetCurUB

6.2.2 Data Structure

6.2.2.1 New Data Structure

HI_UNF_TUNER_SCR_UB_S

[Definition]

```
typedef struct hiUNF_TUNER_SCR_UB_S
{
    HI_U32          u32SCRNo;
    HI_S32          s32CenterFreq;
}HI_UNF_TUNER_SCR_UB_S;
```

[Reason]

This data structure is added for recording the user frequency and center frequency.

[Note]

None

[Example]

See **source/msp/api/frontend/unf_unicable.c**.

6.2.2.2 Modified Data Structure

HI_UNF_TUNER_SIG_TYPE_E

[Definition]

Before modification:

```
typedef enum    hiTUNER_SIG_TYPE_E
{
    HI_UNF_TUNER_SIG_TYPE_CAB = 0,
    HI_UNF_TUNER_SIG_TYPE_SAT,
    HI_UNF_TUNER_SIG_TYPE_DVB_T,
```

```
    HI_UNF_TUNER_SIG_TYPE_DVB_T2,
    HI_UNF_TUNER_SIG_TYPE_ISDB_T,
    HI_UNF_TUNER_SIG_TYPE_ATSC_T,
    HI_UNF_TUNER_SIG_TYPE_DTMB,
    HI_UNF_TUNER_SIG_TYPE_J83B,
    HI_UNF_TUNER_SIG_TYPE_BUTT
} HI_UNF_TUNER_SIG_TYPE_E;
```

After modification:

```
typedef enum   hiTUNER_SIG_TYPE_E
{
    HI_UNF_TUNER_SIG_TYPE_CAB = 1,
    HI_UNF_TUNER_SIG_TYPE_SAT = 2,
    HI_UNF_TUNER_SIG_TYPE_DVB_T = 4,
    HI_UNF_TUNER_SIG_TYPE_DVB_T2 = 8,
    HI_UNF_TUNER_SIG_TYPE_ISDB_T = 16,
    HI_UNF_TUNER_SIG_TYPE_ATSC_T = 32,
    HI_UNF_TUNER_SIG_TYPE_DTMB = 64,
    HI_UNF_TUNER_SIG_TYPE_J83B = 128,
    HI_UNF_TUNER_SIG_TYPE_BUTT
} HI_UNF_TUNER_SIG_TYPE_E;
```

[Reason]

The signal type enumerations are changed to the power of 2 to facilitate bit operations.

[Note]

None

[Example]

See **source/msp/api/frontend/unf_tuner.c**.

6.2.3 APIs

6.2.3.1 New APIs

HI_UNF_TUNER_GetAgc

[Definition]

```
HI_S32 HI_UNF_TUNER_GetAgc(HI_U32 u32TunerId,HI_S32 s32CenterFreq, HI_S32
*ps32Agc);
```

[Reason]

This API is added for obtaining the AGC value required for blind scanning of unicable user frequencies.

[Note]

None

[Example]

None

HI_UNF_UNICABLE_ScanAndInstall_UB

[Definition]

```
HI_S32 HI_UNF_UNICABLE_ScanAndInstall_UB(HI_U32 u32TunerId);
```

[Reason]

This API is added for performing automatic blind scanning of unicable user frequencies.

[Note]

None

[Example]

None

HI_UNF_UNICABLE_GetUBInfo

[Definition]

```
HI_S32 HI_UNF_UNICABLE_GetUBInfo(HI_U32 u32TunerId, HI_UNF_TUNER_SCR_UB_S
*pUBInfo);
```

[Reason]

This API is added for obtaining the results of blind scanning of unicable user frequencies.

[Note]

None

[Example]

None

HI_UNF_UNICABLE_SetCurUB

[Definition]

```
HI_S32 HI_UNF_UNICABLE_SetCurUB(HI_U32 u32TunerId);
```

[Reason]

This API is added for specifying the unicable user frequency to be used.

[Note]

None

[Example]

None

6.3 Sound

6.3.1 Overview

The UNF 3.2.4 sound module has the following changes compared with UNF 3.2.3:

The CODEC StreamInfo structure is modified.

6.3.1.1 Modifying the CODEC StreamInfo Structure

Data Structure

HI_UNF_ACODEC_STREAMINFO_S is modified.

6.3.2 Data Structure

6.3.2.1 Modified Data Structure

HI_UNF_ACODEC_STREAMINFO_S

[Definition]

Before modification:

```
typedef struct hiUNF_ACODEC_STREAMINFO_S
{
    HI_U32   enACodecType;
    HI_U32             enSampleRate;
    HI_UNF_BIT_DEPTH_E   enBitDepth;
}HI_UNF_ACODEC_STREAMINFO_S;
After modification:
typedef struct hiUNF_ACODEC_STREAMINFO_S
{
    HI_U32   enACodecType;
    HI_U32             enSampleRate;
    HI_UNF_BIT_DEPTH_E   enBitDepth;
    HI_U32             u32Channel;
}HI_UNF_ACODEC_STREAMINFO_S;
```

[Description]

The number of channels is increased.

7 Differences Between UNF 3.2.5 and UNF 3.2.4

7.1 AVPLAY

7.1.1 Overview

The UNF 3.2.5 AVPLAY module has the following change compared with UNF 3.2.4:

The audio event callback function type is added.

7.1.1.1 Supporting the Audio Event Callback Function Type

Three AVPLAY events are added: audio information change event, unsupported audio event, and audio frame error event. The change is described as follows:

Data Structure

HI_UNF_AVPLAY_EVENT_E is modified.

7.1.2 Data Structure

7.1.2.1 Modified Data Structure

HI_UNF_AVPLAY_EVENT_E

[Definition]

Before modification:

```
typedef enum hiUNF_AVPLAY_EVENT_E
{
    HI_UNF_AVPLAY_EVENT_EOS,
    HI_UNF_AVPLAY_EVENT_STOP,
    HI_UNF_AVPLAY_EVENT_RNG_BUF_STATE,
    HI_UNF_AVPLAY_EVENT_NORM_SWITCH,
    HI_UNF_AVPLAY_EVENT_FRAMEPACKING_CHANGE,
    HI_UNF_AVPLAY_EVENT_NEW_VID_FRAME,
```

```
    HI_UNF_AVPLAY_EVENT_NEW_AUD_FRAME,
    HI_UNF_AVPLAY_EVENT_NEW_USER_DATA,
    HI_UNF_AVPLAY_EVENT_GET_AUD_ES,
    HI_UNF_AVPLAY_EVENT_IFRAME_ERR,
    HI_UNF_AVPLAY_EVENT_SYNC_PTS_JUMP,
    HI_UNF_AVPLAY_EVENT_SYNC_STAT_CHANGE,
    HI_UNF_AVPLAY_EVENT_VID_BUF_STATE,
    HI_UNF_AVPLAY_EVENT_AUD_BUF_STATE,
    HI_UNF_AVPLAY_EVENT_VID_UNSUPPORT,
    HI_UNF_AVPLAY_EVENT_VID_ERR_RATIO,
    HI_UNF_AVPLAY_EVENT_BUTT
} HI_UNF_AVPLAY_EVENT_E;
```

After modification:

```
typedef enum hiUNF_AVPLAY_EVENT_E
{
    HI_UNF_AVPLAY_EVENT_EOS,
    HI_UNF_AVPLAY_EVENT_STOP,
    HI_UNF_AVPLAY_EVENT_RNG_BUF_STATE,
    HI_UNF_AVPLAY_EVENT_NORM_SWITCH,
    HI_UNF_AVPLAY_EVENT_FRAMEPACKING_CHANGE,
    HI_UNF_AVPLAY_EVENT_NEW_VID_FRAME,
    HI_UNF_AVPLAY_EVENT_NEW_AUD_FRAME,
    HI_UNF_AVPLAY_EVENT_NEW_USER_DATA,
    HI_UNF_AVPLAY_EVENT_GET_AUD_ES,
    HI_UNF_AVPLAY_EVENT_IFRAME_ERR,
    HI_UNF_AVPLAY_EVENT_SYNC_PTS_JUMP,
    HI_UNF_AVPLAY_EVENT_SYNC_STAT_CHANGE,
    HI_UNF_AVPLAY_EVENT_VID_BUF_STATE,
    HI_UNF_AVPLAY_EVENT_AUD_BUF_STATE,
    HI_UNF_AVPLAY_EVENT_VID_UNSUPPORT,
    HI_UNF_AVPLAY_EVENT_VID_ERR_RATIO,
    HI_UNF_AVPLAY_EVENT_AUD_INFO_CHANGE,
    HI_UNF_AVPLAY_EVENT_AUD_UNSUPPORT,
    HI_UNF_AVPLAY_EVENT_AUD_FRAME_ERR,
    HI_UNF_AVPLAY_EVENT_BUTT
} HI_UNF_AVPLAY_EVENT_E;
```

[Reason]

The audio information change, unsupported audio, and audio fram error event callback function types are added.

[Note]

None

[Example]

None

7.2 Sound

7.2.1 Overview

The UNF 3.2.5 sound module has the following change compared with UNF 3.2.4:

The PCM delay data structure is modified.

7.2.1.1 Modifying the PCM Delay Data Structure

Data Structure

HI_UNF_I2S_PCMDELAY_E is modified.

7.2.2 Data Structure

7.2.2.1 Modified Data Structure

HI_UNF_I2S_PCMDELAY_E

[Definition]

Before modification:

```
typedef enum hiHI_UNF_I2S_PCMDELAY_E
{
    HI_UNF_I2S_PCM_0_DELAY = 0,
    HI_UNF_I2S_PCM_1_DELAY = 1,
    HI_UNF_I2S_PCM_8_DELAY = 8,
    HI_UNF_I2S_PCM_16_DELAY = 16,
    HI_UNF_I2S_PCM_32_DELAY = 32,
    HI_UNF_I2S_PCM_DELAY_BUTT
} HI_UNF_I2S_PCMDELAY_E;
```

After modification:

```
typedef enum hiHI_UNF_I2S_PCMDELAY_E
{
    HI_UNF_I2S_PCM_0_DELAY = 0,
    HI_UNF_I2S_PCM_1_DELAY = 1,
    HI_UNF_I2S_PCM_8_DELAY = 8,
    HI_UNF_I2S_PCM_16_DELAY = 16,
    HI_UNF_I2S_PCM_17_DELAY = 17,
    HI_UNF_I2S_PCM_24_DELAY = 24,
    HI_UNF_I2S_PCM_32_DELAY = 32,
```

```
    HI_UNF_I2S_PCM_DELAY_BUTT
} HI_UNF_I2S_PCMDELAY_E;
```

[Reason]

The I^2S PCM delay options are added.

[Note]

None

[Example]

None

7.3 VENC

7.3.1 Overview

The UNF 3.2.5 VNEC module has the following change compared with UNF 3.2.4:

The bit rate fluctuation threshold is added.

7.3.1.1 Adding the Bit Rate Fluctuation Threshold

The bit rate may fluctuate when exceptions occur during the bit rate control of the VENC. In this case, the VNEC can control the fluctuation range of the output bit rate by discarding frames internally. The fluctuation threshold can be set to a value ranging from 0−100 or 0xFFFFFFFF. If the threshold is set to **20**, the bit rate control algorithm allows the output bit rate to fluctuate within ±20% of the configured target bit rate. If it is set to **0**, the bit rate control algorithm does not allow any fluctuation of the output bit rate. If it is set to **100**, the bit rate control is loose and the fluctuation range is ±100% of the target bit rate. If it is set to **0xFFFFFFFF**, the VENC controls the bit rate by converging the internal algorithm but not discarding frames. The change is described as follows:

Data Structure

HI_UNF_VENC_CHN_ATTR_S is modified.

7.3.2 Data Structure

7.3.2.1 Modified Data Structure

HI_UNF_VENC_CHN_ATTR_S

[Definition]

Before modification:

```
typedef struct hiUNF_VENC_CHN_ATTR_S
{
    HI_UNF_VCODEC_TYPE_E        enVencType;
    HI_UNF_VCODEC_CAP_LEVEL_E   enCapLevel;
    HI_UNF_H264_PROFILE_E       enVencProfile;
```

```
    HI_U32                      u32Width;
    HI_U32                      u32Height;
    HI_U32                      u32StrmBufSize;
    HI_U32                      u32RotationAngle;
    HI_BOOL                     bSlcSplitEn;
    HI_U32                      u32TargetBitRate;
    HI_U32                      u32TargetFrmRate;
    HI_U32                      u32InputFrmRate;
    HI_U32                      u32MaxQp;
    HI_U32                      u32MinQp;
    HI_BOOL                     bQuickEncode;
    HI_U8                       u8Priority;
    HI_U32                     u32Qlevel;
}HI_UNF_VENC_CHN_ATTR_S;
```

After modification:

```
typedef struct hiUNF_VENC_CHN_ATTR_S
{
    HI_UNF_VCODEC_TYPE_E        enVencType;
    HI_UNF_VCODEC_CAP_LEVEL_E   enCapLevel;
    HI_UNF_H264_PROFILE_E       enVencProfile;
    HI_U32                      u32Width;
    HI_U32                      u32Height;
    HI_U32                      u32StrmBufSize;
    HI_U32                      u32RotationAngle;
    HI_BOOL                     bSlcSplitEn;
    HI_U32                      u32TargetBitRate;
    HI_U32                      u32TargetFrmRate;
    HI_U32                      u32InputFrmRate;
    HI_U32                      u32MaxQp;
    HI_U32                      u32MinQp;
    HI_BOOL                     bQuickEncode;
    HI_U8                       u8Priority;
    HI_U32                     u32Qlevel;
    HI_U32                      u32DriftRateThr;
}HI_UNF_VENC_CHN_ATTR_S;
```

[Reason]

The data structure is modified to allow you to determine whether to discard frames during bit rate control and specify the corresponding bit rate fluctuation range based on the actual scenario.

[Note]

When the threshold is set to **0xFFFFFFFF**, the VENC does not discard frames during bit rate control.

[Example]

None

7.4 Cipher

7.4.1 Overview

The UNF 3.2.5 cipher module has the following changes compared with UNF 3.2.4:

The Irdeto MSR2.2-related specifications are supported.

Supporting the Irdeto MSR2.2-Related Specifications

The changes are described as follows:

Data Structure

HI_UNF_CIPHER_WORK_MODE_E and HI_UNF_CIPHER_HASH_TYPE_E are modified.

7.4.2 Data Structure

7.4.2.1 Modified Data Structure

HI_UNF_CIPHER_CA_TYPE_E

[Definition]

Before modification:

```
typedef enum hiUNF_CIPHER_CA_TYPE_E
{
    HI_UNF_CIPHER_CA_TYPE_R2R = 0x0,
    HI_UNF_CIPHER_CA_TYPE_SP,
    HI_UNF_CIPHER_CA_TYPE_CSA2,
    HI_UNF_CIPHER_CA_TYPE_CSA3,
    HI_UNF_CIPHER_CA_TYPE_MISC,
    HI_UNF_CIPHER_CA_TYPE_GDRM,
    HI_UNF_CIPHER_CA_TYPE_BLPK,
    HI_UNF_CIPHER_CA_TYPE_LPK,
}HI_UNF_CIPHER_CA_TYPE_E;
```

After modification:

```
typedef enum hiUNF_CIPHER_CA_TYPE_E
{
    HI_UNF_CIPHER_CA_TYPE_R2R = 0x0,
    HI_UNF_CIPHER_CA_TYPE_SP,
    HI_UNF_CIPHER_CA_TYPE_CSA2,
```

```
    HI_UNF_CIPHER_CA_TYPE_CSA3,
    HI_UNF_CIPHER_CA_TYPE_MISC,
    HI_UNF_CIPHER_CA_TYPE_GDRM,
    HI_UNF_CIPHER_CA_TYPE_BLPK,
    HI_UNF_CIPHER_CA_TYPE_LPK,
    HI_UNF_CIPHER_CA_TYPE_IRDETO_HCA,
}HI_UNF_CIPHER_CA_TYPE_E;
```

[Reason]

The Irdeto MSR2.2 CBC-MAC calculation function is added.

[Note]

None

[Example]

See command 52 in **sample/advca/ sample_ca_irdeto_msr2.2.c**.

```
52: AES CBC-MAC HIGH-LEVEL CODE AUTHENTICATION
```

HI_UNF_CIPHER_HASH_TYPE_E

[Definition]

Before modification:

```
typedef enum hiHI_UNF_CIPHER_HASH_TYPE_E
{
    HI_UNF_CIPHER_HASH_TYPE_SHA1,
    HI_UNF_CIPHER_HASH_TYPE_SHA256,
    HI_UNF_CIPHER_HASH_TYPE_HMAC_SHA1,
    HI_UNF_CIPHER_HASH_TYPE_HMAC_SHA256,
    HI_UNF_CIPHER_HASH_TYPE_BUTT,
}HI_UNF_CIPHER_HASH_TYPE_E;
```

After modification:

```
typedef enum hiHI_UNF_CIPHER_HASH_TYPE_E
{
    HI_UNF_CIPHER_HASH_TYPE_SHA1,
    HI_UNF_CIPHER_HASH_TYPE_SHA256,
    HI_UNF_CIPHER_HASH_TYPE_HMAC_SHA1,
     HI_UNF_CIPHER_HASH_TYPE_HMAC_SHA256,
     HI_UNF_CIPHER_HASH_TYPE_IRDETO_CBCMAC,
    HI_UNF_CIPHER_HASH_TYPE_BUTT,
}HI_UNF_CIPHER_HASH_TYPE_E;
```

[Reason]

The Irdeto MSR2.2 CBC-MAC calculation function is added.

[Note]

None

[Example]

See the following process:

```
stCipherHashAttr.eShaType = HI_UNF_CIPHER_HASH_TYPE_IRDETO_CBCMAC;
…
HI_UNF_CIPHER_HashInit(…);
HI_UNF_CIPHER_HashUpdate(…);
HI_UNF_CIPHER_HashFinal(…);
```

7.5 KeyLED

7.5.1 Overview

The UNF 3.2.5 KeyLED module has the following changes compared with UNF 3.2.4:

The frequency lock LED on the panel (only the FD650 is supported) can be configured.

7.5.1.1 Configuring the Frequency Lock LED

The frequency lock LED on the panel can be configured. Only the FD650 supports this function.

API

HI_UNF_LED_SetLockPin is added.

7.5.2 API

7.5.2.1 New API

HI_UNF_LED_SetLockPin

[Definition]

```
HI_S32 HI_UNF_LED_SetLockPin(HI_BOOL setLock);
```

[Reason]

This API is added for configuring the frequency lock LED on the panel.

[Note]

This configuration is supported by only the FD650 panel.

[Example]

None

7.6 PDM

7.6.1 Overview

The UNF 3.2.5 PDM module has the following change compared with UNF 3.2.4:

The data structure for HDMI basic configuration parameters is added.

7.6.1.1 Adding the Data Structure for HDMI Basic Configuration Parameters

The changes are described as follows:

Data Structure

HI_UNF_PDM_HDMI_PARAM_S is added.

HI_UNF_PDM_BASEPARAM_TYPE_E is modified.

7.6.2 Data Structure

7.6.2.1 New Data Structure

HI_UNF_PDM_HDMI_PARAM_S

[Definition]

```
typedef struct hiUNF_PDM_HDMI_PARAM_S
{
    HI_U8    *pu8EDID;
    HI_U32   *pu32EDIDLen;
}HI_UNF_PDM_HDMI_PARAM_S;
```

[Reason]

This data structure is added for storing HDMI basic configuration parameters.

[Note]

None

[Example]

None

7.6.2.2 Modified Data Structure

HI_UNF_PDM_BASEPARAM_TYPE_E

[Definition]

Before modification:

```
typedef enum hiUNF_PDM_BASEPARAM_TYPE_E
{
```

```
    HI_UNF_PDM_BASEPARAM_DISP0 = 0,
    HI_UNF_PDM_BASEPARAM_DISP1,
    HI_UNF_PDM_BASEPARAM_SOUND0 = 10,
    HI_UNF_PDM_BASEPARAM_SOUND1,
    HI_UNF_PDM_BASEPARAM_SOUND2,
    HI_UNF_PDM_BASEPARAM_BUTT = 0xFFFF,
}HI_UNF_PDM_BASEPARAM_TYPE_E;
```

After modification:

```
typedef enum hiUNF_PDM_BASEPARAM_TYPE_E
{
    HI_UNF_PDM_BASEPARAM_DISP0 = 0,
    HI_UNF_PDM_BASEPARAM_DISP1,
    HI_UNF_PDM_BASEPARAM_SOUND0 = 10,
    HI_UNF_PDM_BASEPARAM_SOUND1,
    HI_UNF_PDM_BASEPARAM_SOUND2,
    HI_UNF_PDM_BASEPARAM_HDMI = 20,
    HI_UNF_PDM_BASEPARAM_BUTT = 0xFFFF,
}HI_UNF_PDM_BASEPARAM_TYPE_E;
```

[Reason]

The HDMI type is added in the basic parameters.

[Note]

None

[Example]

None

8 Differences Between UNF 3.2.6 and UNF 3.2.5

8.1 VO

8.1.1 Overview

The UNF 3.2.6 VO module has the following change compared with UNF 3.2.5:

The display standards with the refresh rates 23.976, 29.97, and 59.94 are added.

8.1.1.1 Supporting More Display Standards

The display standards with the refresh rates 23.976, 29.97, and 59.94 are added. The change is described as follows:

Data Structure

HI_UNF_ENC_FMT_E is modified.

8.1.2 Data Structure

8.1.2.1 Modified Data Structure

HI_UNF_ENC_FMT_E

[Definition]

Before modification:

```
typedef enum hiUNF_ENC_FMT_E
{
    HI_UNF_ENC_FMT_1080P_60 = 0,     /**<1080p 60 Hz*/
    HI_UNF_ENC_FMT_1080P_50,        /**<1080p 50 Hz*/
    HI_UNF_ENC_FMT_1080P_30,        /**<1080p 30 Hz*/
    HI_UNF_ENC_FMT_1080P_25,        /**<1080p 25 Hz*/
    HI_UNF_ENC_FMT_1080P_24,        /**<1080p 24 Hz*/
```

```
    HI_UNF_ENC_FMT_1080i_60,        /**<1080i 60 Hz*/
    HI_UNF_ENC_FMT_1080i_50,        /**<1080i 50 Hz*/

    HI_UNF_ENC_FMT_720P_60,         /**<720p 60 Hz*/
    HI_UNF_ENC_FMT_720P_50,         /**<720p 50 Hz */

    HI_UNF_ENC_FMT_576P_50,         /**<576p 50 Hz*/
    HI_UNF_ENC_FMT_480P_60,         /**<480p 60 Hz*/

    HI_UNF_ENC_FMT_PAL,            /* B D G H I PAL */
    HI_UNF_ENC_FMT_PAL_N,           /* (N)PAL       */
    HI_UNF_ENC_FMT_PAL_Nc,          /* (Nc)PAL      */

    HI_UNF_ENC_FMT_NTSC,           /* (M)NTSC      */
    HI_UNF_ENC_FMT_NTSC_J,          /* NTSC-J       */
    HI_UNF_ENC_FMT_NTSC_PAL_M,      /* (M)PAL       */

    HI_UNF_ENC_FMT_SECAM_SIN,       /**< SECAM_SIN*/
    HI_UNF_ENC_FMT_SECAM_COS,       /**< SECAM_COS*/

    HI_UNF_ENC_FMT_1080P_24_FRAME_PACKING,
    HI_UNF_ENC_FMT_720P_60_FRAME_PACKING,
    HI_UNF_ENC_FMT_720P_50_FRAME_PACKING,

    HI_UNF_ENC_FMT_861D_640X480_60,
    HI_UNF_ENC_FMT_VESA_800X600_60,
    HI_UNF_ENC_FMT_VESA_1024X768_60,
    HI_UNF_ENC_FMT_VESA_1280X720_60,
    HI_UNF_ENC_FMT_VESA_1280X800_60,
    HI_UNF_ENC_FMT_VESA_1280X1024_60,
    HI_UNF_ENC_FMT_VESA_1360X768_60,
    HI_UNF_ENC_FMT_VESA_1366X768_60,
    HI_UNF_ENC_FMT_VESA_1400X1050_60,
    HI_UNF_ENC_FMT_VESA_1440X900_60,
    HI_UNF_ENC_FMT_VESA_1440X900_60_RB,
    HI_UNF_ENC_FMT_VESA_1600X900_60_RB,
    HI_UNF_ENC_FMT_VESA_1600X1200_60,
    HI_UNF_ENC_FMT_VESA_1680X1050_60,
    HI_UNF_ENC_FMT_VESA_1680X1050_60_RB,
    HI_UNF_ENC_FMT_VESA_1920X1080_60,
    HI_UNF_ENC_FMT_VESA_1920X1200_60,
    HI_UNF_ENC_FMT_VESA_1920X1440_60,
    HI_UNF_ENC_FMT_VESA_2048X1152_60,
    HI_UNF_ENC_FMT_VESA_2560X1440_60_RB,
```

```
    HI_UNF_ENC_FMT_VESA_2560X1600_60_RB,

    HI_UNF_ENC_FMT_3840X2160_24 = 0x100,
    HI_UNF_ENC_FMT_3840X2160_25,
    HI_UNF_ENC_FMT_3840X2160_30,
    HI_UNF_ENC_FMT_4096X2160_24,
    HI_UNF_ENC_FMT_BUTT
}HI_UNF_ENC_FMT_E;
```

After modification:

```
typedef enum hiUNF_ENC_FMT_E
{
    HI_UNF_ENC_FMT_1080P_60 = 0,     /**<1080p 60 Hz*/
    HI_UNF_ENC_FMT_1080P_50,        /**<1080p 50 Hz*/
    HI_UNF_ENC_FMT_1080P_30,        /**<1080p 30 Hz*/
    HI_UNF_ENC_FMT_1080P_25,        /**<1080p 25 Hz*/
    HI_UNF_ENC_FMT_1080P_24,        /**<1080p 24 Hz*/

    HI_UNF_ENC_FMT_1080i_60,        /**<1080i 60 Hz*/
    HI_UNF_ENC_FMT_1080i_50,        /**<1080i 50 Hz*/

    HI_UNF_ENC_FMT_720P_60,         /**<720p 60 Hz*/
    HI_UNF_ENC_FMT_720P_50,         /**<720p 50 Hz */

    HI_UNF_ENC_FMT_576P_50,         /**<576p 50 Hz*/
    HI_UNF_ENC_FMT_480P_60,         /**<480p 60 Hz*/

    HI_UNF_ENC_FMT_PAL,            /* B D G H I PAL */
    HI_UNF_ENC_FMT_PAL_N,           /* (N)PAL       */
    HI_UNF_ENC_FMT_PAL_Nc,          /* (Nc)PAL      */

    HI_UNF_ENC_FMT_NTSC,           /* (M)NTSC      */
    HI_UNF_ENC_FMT_NTSC_J,          /* NTSC-J       */
    HI_UNF_ENC_FMT_NTSC_PAL_M,      /* (M)PAL       */

    HI_UNF_ENC_FMT_SECAM_SIN,       /**< SECAM_SIN*/
    HI_UNF_ENC_FMT_SECAM_COS,       /**< SECAM_COS*/

    HI_UNF_ENC_FMT_1080P_24_FRAME_PACKING,
    HI_UNF_ENC_FMT_720P_60_FRAME_PACKING,
    HI_UNF_ENC_FMT_720P_50_FRAME_PACKING,

    HI_UNF_ENC_FMT_861D_640X480_60,
    HI_UNF_ENC_FMT_VESA_800X600_60,
```

```
    HI_UNF_ENC_FMT_VESA_1024X768_60,
    HI_UNF_ENC_FMT_VESA_1280X720_60,
    HI_UNF_ENC_FMT_VESA_1280X800_60,
    HI_UNF_ENC_FMT_VESA_1280X1024_60,
    HI_UNF_ENC_FMT_VESA_1360X768_60,
    HI_UNF_ENC_FMT_VESA_1366X768_60,
    HI_UNF_ENC_FMT_VESA_1400X1050_60,
    HI_UNF_ENC_FMT_VESA_1440X900_60,
    HI_UNF_ENC_FMT_VESA_1440X900_60_RB,
    HI_UNF_ENC_FMT_VESA_1600X900_60_RB,
    HI_UNF_ENC_FMT_VESA_1600X1200_60,
    HI_UNF_ENC_FMT_VESA_1680X1050_60,
    HI_UNF_ENC_FMT_VESA_1680X1050_60_RB,
    HI_UNF_ENC_FMT_VESA_1920X1080_60,
    HI_UNF_ENC_FMT_VESA_1920X1200_60,
    HI_UNF_ENC_FMT_VESA_1920X1440_60,
    HI_UNF_ENC_FMT_VESA_2048X1152_60,
    HI_UNF_ENC_FMT_VESA_2560X1440_60_RB,
    HI_UNF_ENC_FMT_VESA_2560X1600_60_RB,

    HI_UNF_ENC_FMT_3840X2160_24 = 0x100,
    HI_UNF_ENC_FMT_3840X2160_25,
    HI_UNF_ENC_FMT_3840X2160_30,
    HI_UNF_ENC_FMT_4096X2160_24,

    HI_UNF_ENC_FMT_3840X2160_23_976 = 0x200,
    HI_UNF_ENC_FMT_3840X2160_29_97,
    HI_UNF_ENC_FMT_720P_59_94,
    HI_UNF_ENC_FMT_1080P_59_94,
    HI_UNF_ENC_FMT_1080P_29_97,
    HI_UNF_ENC_FMT_1080P_23_976,
    HI_UNF_ENC_FMT_1080i_59_94,
    HI_UNF_ENC_FMT_BUTT
}HI_UNF_ENC_FMT_E;
```

[Reason]

The display standards with the refresh rates 23.976, 29.97, and 59.94 are added.

[Note]

None

[Example]

None

8.2 Common

8.2.1 Overview

The UNF 3.2.6 common module has the following changes compared with UNF 3.2.5:

The interfaces for obtaining functions supported by the chip are extended.

8.2.1.1 Obtaining Functions Supported by the Chip

The functions supported by the chip (such as the DTS, ADVCA, and Macrovision) and the chip ID can be obtained. The change is described as follows:

Data Structure

HI_hiSYS_CHIP_ATTR_S is modified.

8.2.2 Data Structure

8.2.2.1 Modified Data Structure

HI_hiSYS_CHIP_ATTR_S

[Definition]

Before modification:

```
typedef struct hiSYS_CHIP_ATTR_S
{
    HI_BOOL bDolbySupport;
}HI_SYS_CHIP_ATTR_S;
```

After modification:

```
typedef struct hiSYS_CHIP_ATTR_S
{
    HI_BOOL bDolbySupport;
    HI_BOOL bDTSSupport;
    HI_BOOL bADVCASupport;
    HI_BOOL bMacrovisionSupport;
    HI_U64  u64ChipID;
}HI_SYS_CHIP_ATTR_S;
```

[Reason]

This data structure is extended to support interfaces for obtaining functions supported by the chip.

[Note]

None

[Example]

None

8.3 HiPlayer

8.3.1 Overview

The UNF 3.2.6 HiPlayer module has the following changes compared with UNF 3.2.5:

- The DTS_EXPRESS audio format is supported.
- Seeking by position is supported during the playback of MPEG TS network streams.
- The invoke interfaces are extended.
- The interface for reporting the file information update event is added.
- The function of reporting private data is supported.
- The function of obtaining the window operation handle is added.
- The function of setting the buffer alignment parameter for the VDEC output is added.

8.3.1.1 Supporting the DTS_EXPRESS Audio Format

The DTS_EXPRESS audio format is supported.

Data Structure

HI_FORMAT_AUDIO_TYPE_E is modified.

8.3.1.2 Supporting Seeking by Position During the Playback of MPEG TS Network Streams

Seeking by position is supported during the playback of MPEG TS network streams.

Data Structure

HI_FORMAT_SEEK_MODE_E is added.

8.3.1.3 Extending the Invoke Interfaces

The following functions are extended:

- Obtains the video rating information.
- Obtains the audio rating information.
- Obtains the subtitle rating information.
- Sets the sampling interval for network speed statistics.
- Obtains whether the Demux supports the seek operation when the file from the current URL is played.
- Sets the audio track.
- Sets the seek mode.

Data Structure

HI_FORMAT_INVOKE_ID_E is modified.

8.3.1.4 Reporting the File Information Update Event

The file information update event can be reported.

Data Structure

HI_FORMAT_MSG_TYPE_E and HI_SVR_PLAYER_EVENT_E are modified.

8.3.1.5 Reporting Private Data

Private data can be reported.

Data Structure

HI_FORMAT_MSG_S and HI_SVR_PLAYER_EVENT_E are modified.

8.3.1.6 Obtaining the Window Operation Handle

The window operation handle can be obtained.

Data Structure

HI_SVR_PLAYER_PARAM_S is modified.

8.3.1.7 Setting the Buffer Alignment Parameter for the VDEC Output

The external buffer alignment parameter can be configured.

Data Structure

HI_SVR_PICTURE_S is modified.

8.3.2 Data Structure

8.3.2.1 New Data Structure

HI_FORMAT_SEEK_MODE_E

[Definition]

```
typedef enum hiFORMAT_SEEK_MODE_E
{
    HI_FORMAT_SEEK_MODE_PTS = 0x0,
    HI_FORMAT_SEEK_MODE_POS,
    HI_FORMAT_SEEK_MODE_BUTT
} HI_FORMAT_SEEK_MODE_E;
```

[Reason]

Seeking by position is required when there is no index file for network playback.

[Note]

Seeking by position is not accurate and therefore is used only when there is no index file for network playback. Seeking by PTS is used by default.

[Example]

```
HI_FORMAT_SEEK_MODE_E eMode = HI_FORMAT_SEEK_MODE_POS;
HI_SVR_PLAYER_Invoke(hPlayer,  HI_FORMAT_INVOKE_SET_SEEK_MODE, &eMode);
```

HI_FORMAT_STREAM_INFO_S

[Definition]

```
typedef struct hiFORMAT_STREAM_INFO_S
{
    HI_S32     s32ProgID;
    HI_S32     s32VidID;
    HI_S32     s32AudID;
    HI_S32     s32SubID;
    HI_S64     s64PlayTime;
} HI_FORMAT_STREAM_INFO_S;
```

[Reason]

This data structure is added for the parser to obtain the player information.

[Note]

It is an internal data structure in the HiPlayer, and this function is not used currently.

[Example]

None

HI_SVR_PLAYER_PROC_SEEKINFO_S

[Definition]

```
typedef struct hiSVR_PLAYER_PROC_SEEKINFO_S
{
    HI_U32     u32DoReadSeek;
    HI_U32     u32ReadSeekDone;
    HI_U32     u32DoSeekFrameBinary;
    HI_U32     u32SeekFrameBinaryDone;
    HI_U32     u32DoSeekFrameGeneric;
    HI_U32     u32SeekFrameGenericDone;
    HI_U32     u32DoInitInput;
    HI_U32     u32DoAvioOpenH;
    HI_U32     u32AvioOpenHDone;
    HI_U32     u32DoAvProbeInputBuffer;
    HI_U32     u32AvProbeInputBufferDone;

    HI_U32     u32DoHiSvrFormatSeekPts;
    HI_U32     u32CmdSeek;
    HI_U32     u32DoAvformatOpenInput;
```

```
    HI_U32      u32AvformatOpenInputDone;
    HI_U32      u32DoSvrFormatFindStream;
    HI_U32      u32SvrFormatFindStreamDone;
    HI_U32      u32DoSvrFormatGetFileInfo;
    HI_U32      u32SvrFormatGetFileInfoDone;

} HI_SVR_PLAYER_PROC_SEEKINFO_S;
```

[Reason]

This data structure is added because the invoke interface is modified. It is used for obtaining the seek proc information.

[Note]

None

[Example]

None

HI_SVR_PLAYER_PROC_SWITCHPG_INFOTYPE_E

[Definition]

```
typedef enum hiSVR_PLAYER_PROC_SWITCHPG_INFOTYPE_E
{
    HI_SVR_PLAYER_PROC_DO_STOP=1,
    HI_SVR_PLAYER_PROC_HIMEDIAPLAYER_CONSTRUCT,
    HI_SVR_PLAYER_PROC_SETDATASOURCE,
    HI_SVR_PLAYER_PROC_UNF_AVPLAY_CREATE,
    HI_SVR_PLAYER_PROC_DO_PREPARE,
    HI_SVR_PLAYER_PROC_PREPARE_ASYNC_COMPLETE,
    HI_SVR_PLAYER_PROC_DO_START_ENTER,
    HI_SVR_PLAYER_PROC_PLAYER_STATE_PLAY,
    HI_SVR_PLAYER_PROC_MEDIA_INFO_FIRST_FRAME_TIME,
    HI_SVR_PLAYER_PROC_DO_RESET,
    HI_SVR_PLAYER_PROC_DO_DESTRUCTOR,

} HI_SVR_PLAYER_PROC_SWITCHPG_INFOTYPE_E;
```

[Reason]

This data structure is added because the invoke interface is extended. It provides enumerations of the program switch operations.

[Note]

None

[Example]

None

HI_SVR_PLAYER_PROC_SWITCHPG_S

[Definition]

```
typedef struct hiSVR_PLAYER_PROC_SWITCHPG_S
{
    HI_SVR_PLAYER_PROC_SWITCHPG_INFOTYPE_E eType;
    HI_U32 u32DoStop;
    HI_U32 u32HiMediaPlayerConstruct;
    HI_U32 u32SetDataSource;
    HI_U32 u32DoCreateAVPlay;
    HI_U32 u32DoPrepare;
    HI_U32 u32prepareAsyncComplete;
    HI_U32 u32DoStartEnter;
    HI_U32 u32PlayedEvent;
    HI_U32 u32FirstFrameTime;
    HI_U32 u32DoReset;
    HI_U32 u32DoDestructor;

} HI_SVR_PLAYER_PROC_SWITCHPG_S;
```

[Reason]

This data structure is added because the invoke interface is extended. It is used for obtaining the program switch information.

[Note]

None

[Example]

None

8.3.2.2 Modified Data Structure

HI_FORMAT_AUDIO_TYPE_E

[Definition]

Before modification:

```
typedef enum hiFORMAT_AUDIO_TYPE_E
{
HI_FORMAT_AUDIO_MP2 = 0x000,
……
    HI_FORMAT_AUDIO_BINKAUDIO_RDFT,
    HI_FORMAT_AUDIO_BINKAUDIO_DCT,
    HI_FORMAT_AUDIO_DRA,

    HI_FORMAT_AUDIO_PCM = 0x100,
```

```
    HI_FORMAT_AUDIO_PCM_BLURAY = 0x121,
    HI_FORMAT_AUDIO_ADPCM = 0x130,
……
} HI_FORMAT_AUDIO_TYPE_E;
```

After modification:

```
typedef enum hiFORMAT_AUDIO_TYPE_E
{
HI_FORMAT_AUDIO_MP2 = 0x000,
……
    HI_FORMAT_AUDIO_BINKAUDIO_RDFT,
    HI_FORMAT_AUDIO_BINKAUDIO_DCT,
    HI_FORMAT_AUDIO_DRA,
    HI_FORMAT_AUDIO_DTS_EXPRESS,

    HI_FORMAT_AUDIO_PCM = 0x100,
    HI_FORMAT_AUDIO_PCM_BLURAY = 0x121,
    HI_FORMAT_AUDIO_ADPCM = 0x130,
……
} HI_FORMAT_AUDIO_TYPE_E;
```

[Reason]

This data structure is modified to support the DTS_EXPRESS audio format.

[Note]

The DTS_EXPRESS format is converted into the DTS format during format conversion.

[Example]

```
    case HI_FORMAT_AUDIO_DTS_EXPRESS: \
        (dest) = FORMAT_DTS;
```

HI_FORMAT_INVOKE_ID_E

[Definition]

Before modification:

```
typedef enum hiFORMAT_INVOKE_ID_E
{
    ...
    HI_FORMAT_INVOKE_SET_LOCALTIME,
    HI_FORMAT_INVOKE_SET_HEADERS,
    HI_FORMAT_INVOKE_SET_USERAGENT,
    HI_FORMAT_INVOKE_SET_REFERER,
    HI_FORMAT_INVOKE_SET_NOT_SUPPORT_BYTERANGE,
    HI_FORMAT_INVOKE_SET_LOG_LEVEL,
    HI_FORMAT_INVOKE_RTMP_RECEIVEAUDIO,
```

```
    HI_FORMAT_INVOKE_RTMP_RECEIVEVIDEO,
    HI_FORMAT_INVOKE_SET_BUFFER_UNDERRUN,
    HI_FORMAT_INVOKE_SET_HLS_LIVE_START_NUM,
    HI_FORMAT_INVOKE_SET_STREAM_INFO,
    HI_FORMAT_INVOKE_PROTOCOL_USER=100,
    HI_FORMAT_INVOKE_SET_DOLBYRANGEINFO,
    HI_FORMAT_INVOKE_GET_DOLBYINFO,
    HI_FORMAT_INVOKE_SET_DOLBYDRCMODE,
    HI_FORMAT_INVOKE_SET_EXTERNALFORMAT,
    HI_FORMAT_INVOKE_FIND_BESTSTREAM,
    HI_FORMAT_INVOKE_BUTT
} HI_FORMAT_INVOKE_ID_E;
```

After modification:

```
typedef enum hiFORMAT_INVOKE_ID_E
{
    ...
    HI_FORMAT_INVOKE_SET_LOCALTIME,
    HI_FORMAT_INVOKE_SET_HEADERS,
    HI_FORMAT_INVOKE_SET_USERAGENT,
    HI_FORMAT_INVOKE_SET_REFERER,
    HI_FORMAT_INVOKE_SET_NOT_SUPPORT_BYTERANGE,
    HI_FORMAT_INVOKE_SET_LOG_LEVEL,
    HI_FORMAT_INVOKE_RTMP_RECEIVEAUDIO,
    HI_FORMAT_INVOKE_RTMP_RECEIVEVIDEO,
    HI_FORMAT_INVOKE_SET_BUFFER_UNDERRUN,
    HI_FORMAT_INVOKE_SET_HLS_LIVE_START_NUM,
    HI_FORMAT_INVOKE_SET_STREAM_INFO,
    HI_FORMAT_INVOKE_GET_VIDEO_RATING,
    HI_FORMAT_INVOKE_GET_AUDIO_RATING,
    HI_FORMAT_INVOKE_GET_SUBTITLE_RATING,
    HI_FORMAT_INVOKE_SET_BAND_COLLECT_FREQ_MS,
    HI_FORMAT_INVOKE_GET_SEEKABLE,
    HI_FORMAT_INVOKE_PROTOCOL_USER=100,
    HI_FORMAT_INVOKE_SET_DOLBYRANGEINFO,
    HI_FORMAT_INVOKE_GET_DOLBYINFO,
    HI_FORMAT_INVOKE_SET_DOLBYDRCMODE,
    HI_FORMAT_INVOKE_SET_EXTERNALFORMAT,
    HI_FORMAT_INVOKE_FIND_BESTSTREAM,
    HI_FORMAT_INVOKE_SET_EXTERNAL_AUDIOTRACK,
    HI_FORMAT_INVOKE_SET_SEEK_MODE,
    HI_FORMAT_INVOKE_BUTT
} HI_FORMAT_INVOKE_ID_E;
```

[Reason]

The invoke interfaces for implementing the following functions are extended based on demands and function requirements:

- Obtains the video rating information.
- Obtains the audio rating information.
- Obtains the subtitle rating information.
- Sets the sampling interval for network speed statistics.
- Obtains whether the Demux supports the seek operation when the file from the current URL is played.
- Sets the audio track.
- Sets the seek mode.

[Note]

The following interfaces are not used currently:

- HI_FORMAT_INVOKE_SET_STREAM_INFO
- HI_FORMAT_INVOKE_GET_VIDEO_RATING
- HI_FORMAT_INVOKE_GET_AUDIO_RATING
- HI_FORMAT_INVOKE_GET_SUBTITLE_RATING
- HI_FORMAT_INVOKE_FIND_BESTSTREAM

[Example]

None

HI_FORMAT_MSG_TYPE_E

[Definition]

Before modification:

```
typedef enum hiFORMAT_MSG_TYPE_E
{
    HI_FORMAT_MSG_NONE = 0x0,
    HI_FORMAT_MSG_UNKNOW,
    HI_FORMAT_MSG_DOWNLOAD_FINISH,
    HI_FORMAT_MSG_TIME_OUT,
    HI_FORMAT_MSG_NETWORK,
    HI_FORMAT_MSG_NOT_SUPPORT,
    HI_FORMAT_MSG_DISCONTINUITY_SEEK,
    HI_FORMAT_MSG_DISCONTINUITY_AUD_FORMAT,
    HI_FORMAT_MSG_DISCONTINUITY_VID_FORMAT,
    HI_FORMAT_MSG_USER_PRIVATE = 100,
    HI_FORMAT_MSG_BUTT,
} HI_FORMAT_MSG_TYPE_E;
```

After modification:

```
typedef enum hiFORMAT_MSG_TYPE_E
```

```
{
    HI_FORMAT_MSG_NONE = 0x0,
    HI_FORMAT_MSG_UNKNOW,
    HI_FORMAT_MSG_DOWNLOAD_FINISH,
    HI_FORMAT_MSG_TIME_OUT,
    HI_FORMAT_MSG_NETWORK,
    HI_FORMAT_MSG_NOT_SUPPORT,
    HI_FORMAT_MSG_DISCONTINUITY_SEEK,
    HI_FORMAT_MSG_DISCONTINUITY_AUD_FORMAT,
    HI_FORMAT_MSG_DISCONTINUITY_VID_FORMAT,
    HI_FORMAT_MSG_UPDATE_FILE_INFO,
    HI_FORMAT_MSG_USER_PRIVATE = 100,
    HI_FORMAT_MSG_BUTT,
} HI_FORMAT_MSG_TYPE_E;
```

[Reason]

An enumeration is added for reporting the file information update event.

[Note]

This function is not used currently.

[Example]

None

HI_FORMAT_MSG_S

[Definition]

Before modification:

```
typedef struct hiFORMAT_MSG_S
{
    HI_FORMAT_MSG_TYPE_E     eMsgType;
    HI_FORMAT_BUFFER_STATUS_S stBuffer;
    HI_FORMAT_NET_STATUS_S    stNetStatus;
} HI_FORMAT_MSG_S;
```

After modification:

```
typedef struct hiFORMAT_MSG_S
{
    HI_FORMAT_MSG_TYPE_E     eMsgType;
    HI_FORMAT_BUFFER_STATUS_S stBuffer;
    HI_FORMAT_NET_STATUS_S    stNetStatus;
    HI_VOID  *pPriData;
} HI_FORMAT_MSG_S;
```

[Reason]

This data structure is modified for reporting private data.

[Note]

When the message is HI_FORMAT_MSG_USER_PRIVATE, the private information parameter address is valid.

[Example]

```
(HI_VOID)_SVR_PCTRL_NotifyEvnt(pCtrl, bLock,
HI_SVR_PLAYER_EVENT_USER_PRIVATE,sizeof(stMsg.pPriData),
(HI_U8*)(stMsg.pPriData));
```

HI_SVR_PLAYER_EVENT_E

[Definition]

Before modification:

```
typedef enum hiSVR_PLAYER_EVENT_E
{
    HI_SVR_PLAYER_EVENT_STATE_CHANGED = 0x0,
    HI_SVR_PLAYER_EVENT_SOF,

    HI_SVR_PLAYER_EVENT_EOF,
    HI_SVR_PLAYER_EVENT_PROGRESS,
    HI_SVR_PLAYER_EVENT_STREAMID_CHANGED,
    HI_SVR_PLAYER_EVENT_SEEK_FINISHED,

    HI_SVR_PLAYER_EVENT_CODETYPE_CHANGED,
    HI_SVR_PLAYER_EVENT_DOWNLOAD_PROGRESS,
    HI_SVR_PLAYER_EVENT_BUFFER_STATE,
    HI_SVR_PLAYER_EVENT_FIRST_FRAME_TIME,

    HI_SVR_PLAYER_EVENT_ERROR,
    HI_SVR_PLAYER_EVENT_NETWORK_INFO,
    HI_SVR_PLAYER_EVENT_DOWNLOAD_FINISH,
    HI_SVR_PLAYER_EVENT_BUTT
} HI_SVR_PLAYER_EVENT_E;
```

After modification:

```
typedef enum hiSVR_PLAYER_EVENT_E
{
    HI_SVR_PLAYER_EVENT_STATE_CHANGED = 0x0,
    HI_SVR_PLAYER_EVENT_SOF,

    HI_SVR_PLAYER_EVENT_EOF,
    HI_SVR_PLAYER_EVENT_PROGRESS,
    HI_SVR_PLAYER_EVENT_STREAMID_CHANGED,
```

```
    HI_SVR_PLAYER_EVENT_SEEK_FINISHED,

    HI_SVR_PLAYER_EVENT_CODETYPE_CHANGED,
    HI_SVR_PLAYER_EVENT_DOWNLOAD_PROGRESS,
    HI_SVR_PLAYER_EVENT_BUFFER_STATE,
    HI_SVR_PLAYER_EVENT_FIRST_FRAME_TIME,

    HI_SVR_PLAYER_EVENT_ERROR,
    HI_SVR_PLAYER_EVENT_NETWORK_INFO,
    HI_SVR_PLAYER_EVENT_DOWNLOAD_FINISH,
    HI_SVR_PLAYER_EVENT_UPDATE_FILE_INFO,
    HI_SVR_PLAYER_EVENT_USER_PRIVATE = 100,
    HI_SVR_PLAYER_EVENT_BUTT
} HI_SVR_PLAYER_EVENT_E;
```

[Reason]

This data structure is modified for reporting private data.

[Note]

HI_SVR_PLAYER_EVENT_UPDATE_FILE_INFO:

The file information update event report function is not used currently.

HI_SVR_PLAYER_EVENT_USER_PRIVATE:

When the message is HI_FORMAT_MSG_USER_PRIVATE, the private information parameter address is valid.

[Example]

```
(HI_VOID)_SVR_PCTRL_NotifyEvnt(pCtrl, bLock,
HI_SVR_PLAYER_EVENT_USER_PRIVATE,sizeof(stMsg.pPriData),
(HI_U8*)(stMsg.pPriData));
```

HI_SVR_PLAYER_PARAM_S

[Definition]

Before modification:

```
typedef struct hiSVR_PLAYER_PARAM_S
{
    HI_U32  u32DmxId;
    HI_U32  u32PortId;
    HI_U32  x;
    HI_U32  y;
    HI_U32  w;
    HI_U32  h;
    HI_U32  u32MixHeight;
    HI_HANDLE hAVPlayer;
```

```
    HI_U32   u32SndPort;
    HI_U32   u32Display;
    HI_U32   u32VDecErrCover;
    HI_HANDLE hDRMClient;
} HI_SVR_PLAYER_PARAM_S;
```

After modification:

```
typedef struct hiSVR_PLAYER_PARAM_S
{
    HI_U32  u32DmxId;
    HI_U32  u32PortId;
    HI_U32  x;
    HI_U32  y;
    HI_U32  w;
    HI_U32  h;
    HI_U32  u32MixHeight;
    HI_HANDLE hAVPlayer;
    HI_HANDLE hVSink;
    HI_HANDLE hASink;
    HI_U32   u32SndPort;
    HI_U32   u32Display;
    HI_U32   u32VDecErrCover;
    HI_HANDLE hDRMClient;
    HI_HANDLE hWindow;
} HI_SVR_PLAYER_PARAM_S;
```

[Reason]

The HiPlayer adaptation layer needs to control the window and obtains the window operation handle by using **hWindow**.

[Note]

hVSink and **hASink** apply only to the Android version but not the Linux version.

[Example]

None

HI_SVR_PICTURE_S

[Definition]

Before modification:

```
typedef struct hiSVR_PICTURE_S
{
    HI_U32           u32Width;
    HI_U32           u32Height;
    HI_S64           s64Pts;
```

```
    HI_HANDLE          hBuffer;
    HI_VOID*           priv;
} HI_SVR_PICTURE_S;
```

After modification:

```
typedef struct hiSVR_PICTURE_S
{
    HI_U32             u32Width;
    HI_U32             u32Height;
    HI_U32             u32Stride;
    HI_S64             s64Pts;
    HI_HANDLE          hBuffer;
    HI_VOID*           priv;
} HI_SVR_PICTURE_S;
```

[Reason]

This data structure is modified for aligning the stride of the decoding output buffer with that of the GPU read buffer allocated by GALLOC during VDEC output.

[Note]

This data structure applies only to the Android version but not the Linux version.

[Example]

```
HI_U32 u32Stride = xx;
HI_MPI_AVPLAY_UseExternalBuffer(pstMember->hAVPlay, hBuffers, cnt,
u32BufSize, u32Stride);
```

9 Differences Between UNF 3.2.7 and UNF 3.2.6

9.1 AVPLAY

9.1.1 Overview

The UNF 3.2.7 AVPLAY module has the following change compared with UNF 3.2.6:

The video event callback function type is added.

9.1.1.1 Supporting the Video Event Callback Function Type

The AVPLAY can report the video protocol type error event. The change is described as follows:

Data Structure

HI_UNF_AVPLAY_EVENT_E is modified.

9.1.2 Data Structure

9.1.2.1 Modified Data Structure

HI_UNF_AVPLAY_EVENT_E

[Definition]

Before modification:

```
typedef enum hiUNF_AVPLAY_EVENT_E
{
    HI_UNF_AVPLAY_EVENT_EOS,
    HI_UNF_AVPLAY_EVENT_STOP,
    HI_UNF_AVPLAY_EVENT_RNG_BUF_STATE,
    HI_UNF_AVPLAY_EVENT_NORM_SWITCH,
    HI_UNF_AVPLAY_EVENT_FRAMEPACKING_CHANGE,
    HI_UNF_AVPLAY_EVENT_NEW_VID_FRAME,
```

```
    HI_UNF_AVPLAY_EVENT_NEW_AUD_FRAME,
    HI_UNF_AVPLAY_EVENT_NEW_USER_DATA,
    HI_UNF_AVPLAY_EVENT_GET_AUD_ES,
    HI_UNF_AVPLAY_EVENT_IFRAME_ERR,
    HI_UNF_AVPLAY_EVENT_SYNC_PTS_JUMP,
    HI_UNF_AVPLAY_EVENT_SYNC_STAT_CHANGE,
    HI_UNF_AVPLAY_EVENT_VID_BUF_STATE,
    HI_UNF_AVPLAY_EVENT_AUD_BUF_STATE,
    HI_UNF_AVPLAY_EVENT_VID_UNSUPPORT,
    HI_UNF_AVPLAY_EVENT_VID_ERR_RATIO,
    HI_UNF_AVPLAY_EVENT_AUD_INFO_CHANGE,
    HI_UNF_AVPLAY_EVENT_AUD_UNSUPPORT,
    HI_UNF_AVPLAY_EVENT_AUD_FRAME_ERR,
    HI_UNF_AVPLAY_EVENT_BUTT
} HI_UNF_AVPLAY_EVENT_E;
```

After modification:

```
typedef enum hiUNF_AVPLAY_EVENT_E
{
    HI_UNF_AVPLAY_EVENT_EOS,
    HI_UNF_AVPLAY_EVENT_STOP,
    HI_UNF_AVPLAY_EVENT_RNG_BUF_STATE,
    HI_UNF_AVPLAY_EVENT_NORM_SWITCH,
    HI_UNF_AVPLAY_EVENT_FRAMEPACKING_CHANGE,
    HI_UNF_AVPLAY_EVENT_NEW_VID_FRAME,
    HI_UNF_AVPLAY_EVENT_NEW_AUD_FRAME,
    HI_UNF_AVPLAY_EVENT_NEW_USER_DATA,
    HI_UNF_AVPLAY_EVENT_GET_AUD_ES,
    HI_UNF_AVPLAY_EVENT_IFRAME_ERR,
    HI_UNF_AVPLAY_EVENT_SYNC_PTS_JUMP,
    HI_UNF_AVPLAY_EVENT_SYNC_STAT_CHANGE,
    HI_UNF_AVPLAY_EVENT_VID_BUF_STATE,
    HI_UNF_AVPLAY_EVENT_AUD_BUF_STATE,
    HI_UNF_AVPLAY_EVENT_VID_UNSUPPORT,
    HI_UNF_AVPLAY_EVENT_VID_ERR_RATIO,
    HI_UNF_AVPLAY_EVENT_AUD_INFO_CHANGE,
    HI_UNF_AVPLAY_EVENT_AUD_UNSUPPORT,
    HI_UNF_AVPLAY_EVENT_AUD_FRAME_ERR,
    HI_UNF_AVPLAY_EVENT_VID_ERR_TYPE,
    HI_UNF_AVPLAY_EVENT_BUTT
} HI_UNF_AVPLAY_EVENT_E;
```

[Reason]

A new AVPLAY event (video protocol type error event) is added.

[Note]

None

[Example]

None

9.2 Frontend

9.2.1 Overview

The UNF 3.2.7 frontend module has the following changes compared with UNF 3.2.6:

- The TDA182I5A tuner is supported.
- The name of the ACM/VCM functional UNF interface is optimized.
- The event report mechanism for reporting the blind scan progress and state during blind scanning at unicable user frequencies is added.
- The initial value of the physical layer phase scrambled code can be configured when the DVB-S signal frequency is locked.
- The timeout period can be configured for DVB-T.

9.2.1.1 Supporting the TDA182I5A Tuner

The TDA182I5A tuner is supported. The change is described as follows:

Data Structure

HI_UNF_TUNER_DEV_TYPE_E is modified.

9.2.1.2 Optimizing the Name of the ACM/VCM Functional UNF Interface

The name of the ACM/VCM functional UNF interface is optimized. The changes are described as follows:

Data Structure

- HI_UNF_TUNER_SAT_SIGNALINFO_S is modified.
- HI_UNF_TUNER_CODE_MODULATION_E is added.

9.2.1.3 Supporting the Event Report Mechanism During Blind Scanning at unicable User Frequencies

The event report mechanism during blind scanning at unicable user frequencies is added. The changes are described as follows:

Data Structure

The following data structures are added:

- HI_UNF_TUNER_UNICABLE_SCAN_STATUS_E
- HI_UNF_TUNER_UNICABLE_SCAN_USER_BAND_EVT_E

- HI_UNF_TUNER_UNICABLE_SCAN_USER_BAND_NOTIFY_S
- HI_UNF_TUNER_UNICABLE_SCAN_PARA_S

API

The following APIs are deleted:

- HI_UNF_TUNER_GetAgc
- HI_UNF_UNICABLE_SetCurUB
- HI_UNF_UNICABLE_ScanAndInstall_UB
- HI_UNF_UNICABLE_GetUBInfo

The following APIs are added:

- HI_UNF_TUNER_UNICABLE_ExitScanUserBands
- HI_UNF_TUNER_GetSatIsiID
- HI_UNF_TUNER_GetSatTotalStream
- HI_UNF_TUNER_SetSatIsiID

9.2.1.4 Configuring the Initial Value of the Physical Layer Phase Scrambled Code When the DVB-S Signal Frequency Is Locked

The initial value of the physical layer phase scrambled code can be configured when the DVB-S signal frequency is locked. The change is described as follows:

Data Structure

HI_UNF_SAT_CONNECT_PARA_S is modified.

9.2.1.5 Configuring the Timeout period for DVB-T

The timeout period can be configured for DVB-T.

Data Structure

HI_UNF_TUNER_TER_SCAN_PARA_S is modified.

9.2.2 Data Structure

9.2.2.1 New Data Structure

HI_UNF_TUNER_CODE_MODULATION_E

[Definition]

```
typedef enum hiUNF_TUNER_CODE_MODULATION_E
{
    HI_UNF_TUNER_CODE_MODULATION_VCM_ACM,
    HI_UNF_TUNER_CODE_MODULATION_CCM,
    HI_UNF_TUNER_CODE_MODULATION_MULTISTREAM,
    HI_UNF_TUNER_CODE_MODULATION_BUTT
} HI_UNF_TUNER_CODE_MODULATION_E;
```

[Reason]

The data structure is a member of the HI_UNF_TUNER_SAT_SIGNALINFO_S structure and it describes the code and modulation parameters of satellite signals.

[Note]

None

[Example]

None

HI_UNF_TUNER_UNICABLE_SCAN_STATUS_E

[Definition]

```
typedef enum hiUNF_TUNER_UNICABLE_SCAN_STATUS_E
{
    HI_UNF_TUNER_UNICABLE_SCAN_STATUS_IDLE,
    HI_UNF_TUNER_UNICABLE_SCAN_STATUS_SCANNING,
    HI_UNF_TUNER_UNICABLE_SCAN_STATUS_FINISH,
    HI_UNF_TUNER_UNICABLE_SCAN_STATUS_QUIT,
    HI_UNF_TUNER_UNICABLE_SCAN_STATUS_FAIL,
    HI_UNF_TUNER_UNICABLE_SCAN_STATUS_BUTT
} HI_UNF_TUNER_UNICABLE_SCAN_STATUS_E;
```

[Reason]

This data structure is added to describe the scanning states, such as idle, scanning, and scan completed.

[Note]

None

[Example]

None

HI_UNF_TUNER_UNICABLE_SCAN_USER_BAND_EVT_E

[Definition]

```
typedef enum hiUNF_TUNER_UNICABLE_SCAN_USER_BAND_EVT_E
{
    HI_UNF_TUNER_UNICABLE_SCAN_EVT_STATUS,
    HI_UNF_TUNER_UNICABLE_SCAN_EVT_PROGRESS,
    HI_UNF_TUNER_UNICABLE_SCAN_EVT_BUTT
} HI_UNF_TUNER_UNICABLE_SCAN_USER_BAND_EVT_E;
```

[Reason]

This data structure is added to describe the reported event (state change event or scanning progress change event).

[Note]

None

[Example]

None

HI_UNF_TUNER_UNICABLE_SCAN_USER_BAND_NOTIFY_S

[Definition]

```
typedef union hiUNF_TUNER_UNICABLE_SCAN_USER_BAND_NOTIFY_S
{
   HI_UNF_TUNER_UNICABLE_SCAN_STATUS_E* penStatus;
   HI_U16*   pu16ProgressPercent;
} HI_UNF_TUNER_UNICABLE_SCAN_USER_BAND_NOTIFY_S;
```

[Reason]

This data structure is added to describe the scanning state and scanning progress (percentage).

[Note]

None

[Example]

None

HI_UNF_TUNER_UNICABLE_SCAN_PARA_S

[Definition]

```
typedef struct hiUNF_TUNER_UNICABLE_SCAN_PARA_S
{
   HI_VOID (*pfnEVTNotify)(HI_U32 u32TunerId,
   HI_UNF_TUNER_UNICABLE_SCAN_USER_BAND_EVT_E enEVT,
   HI_UNF_TUNER_UNICABLE_SCAN_USER_BAND_NOTIFY_S *pNotify);
} HI_UNF_TUNER_UNICABLE_SCAN_PARA_S;
```

[Reason]

This data structure is added to describe the function for reporting events.

[Note]

None

[Example]

None

9.2.2.2 Modified Data Structure

HI_UNF_TUNER_DEV_TYPE_E

[Definition]

Before modification:

```
typedef enum   hiUNF_TUNER_DEV_TYPE_E
{
   HI_UNF_TUNER_DEV_TYPE_XG_3BL,
   HI_UNF_TUNER_DEV_TYPE_CD1616,
   HI_UNF_TUNER_DEV_TYPE_ALPS_TDAE,
   HI_UNF_TUNER_DEV_TYPE_TDCC,
   HI_UNF_TUNER_DEV_TYPE_TDA18250,
   HI_UNF_TUNER_DEV_TYPE_CD1616_DOUBLE,
   HI_UNF_TUNER_DEV_TYPE_MT2081,
   HI_UNF_TUNER_DEV_TYPE_TMX7070X,
   HI_UNF_TUNER_DEV_TYPE_R820C,
   HI_UNF_TUNER_DEV_TYPE_MXL203,
   HI_UNF_TUNER_DEV_TYPE_AV2011,
   HI_UNF_TUNER_DEV_TYPE_SHARP7903,
   HI_UNF_TUNER_DEV_TYPE_MXL101,
   HI_UNF_TUNER_DEV_TYPE_MXL603,
   HI_UNF_TUNER_DEV_TYPE_IT9170,
   HI_UNF_TUNER_DEV_TYPE_IT9133,
   HI_UNF_TUNER_DEV_TYPE_TDA6651,
   HI_UNF_TUNER_DEV_TYPE_TDA18250B,
   HI_UNF_TUNER_DEV_TYPE_M88TS2022,
   HI_UNF_TUNER_DEV_TYPE_RDA5815,
   HI_UNF_TUNER_DEV_TYPE_MXL254,
   HI_UNF_TUNER_DEV_TYPE_CXD2861,
   HI_UNF_TUNER_DEV_TYPE_SI2147,
   HI_UNF_TUNER_DEV_TYPE_RAFAEL836,
   HI_UNF_TUNER_DEV_TYPE_MXL608,
   HI_UNF_TUNER_DEV_TYPE_MXL214,
   HI_UNF_TUNER_DEV_TYPE_TDA18280,
   HI_UNF_TUNER_DEV_TYPE_BUTT
} HI_UNF_TUNER_DEV_TYPE_E ;
```

After modification:

```
typedef enum   hiUNF_TUNER_DEV_TYPE_E
{
   HI_UNF_TUNER_DEV_TYPE_XG_3BL,
   HI_UNF_TUNER_DEV_TYPE_CD1616,
   HI_UNF_TUNER_DEV_TYPE_ALPS_TDAE,
```

```
    HI_UNF_TUNER_DEV_TYPE_TDCC,
    HI_UNF_TUNER_DEV_TYPE_TDA18250,
    HI_UNF_TUNER_DEV_TYPE_CD1616_DOUBLE,
    HI_UNF_TUNER_DEV_TYPE_MT2081,
    HI_UNF_TUNER_DEV_TYPE_TMX7070X,
    HI_UNF_TUNER_DEV_TYPE_R820C,
    HI_UNF_TUNER_DEV_TYPE_MXL203,
    HI_UNF_TUNER_DEV_TYPE_AV2011,
    HI_UNF_TUNER_DEV_TYPE_SHARP7903,
    HI_UNF_TUNER_DEV_TYPE_MXL101,
    HI_UNF_TUNER_DEV_TYPE_MXL603,
    HI_UNF_TUNER_DEV_TYPE_IT9170,
    HI_UNF_TUNER_DEV_TYPE_IT9133,
    HI_UNF_TUNER_DEV_TYPE_TDA6651,
    HI_UNF_TUNER_DEV_TYPE_TDA18250B,
    HI_UNF_TUNER_DEV_TYPE_M88TS2022,
    HI_UNF_TUNER_DEV_TYPE_RDA5815,
    HI_UNF_TUNER_DEV_TYPE_MXL254,
    HI_UNF_TUNER_DEV_TYPE_CXD2861,
    HI_UNF_TUNER_DEV_TYPE_SI2147,
    HI_UNF_TUNER_DEV_TYPE_RAFAEL836,
    HI_UNF_TUNER_DEV_TYPE_MXL608,
    HI_UNF_TUNER_DEV_TYPE_MXL214,
    HI_UNF_TUNER_DEV_TYPE_TDA18280,
    HI_UNF_TUNER_DEV_TYPE_TDA182I5A,
    HI_UNF_TUNER_DEV_TYPE_BUTT
} HI_UNF_TUNER_DEV_TYPE_E ;
```

[Reason]

This data structure is modified to support the TDA182I5A tuner.

[Note]

None

[Example]

None

HI_UNF_TUNER_SAT_SIGNALINFO_S

[Definition]

Before modification:

```
typedef struct  hiUNF_TUNER_SAT_SIGNALINFO_S
{
    HI_U32                    u32Freq;
    HI_U32                    u32SymbolRate;
```

```
    HI_UNF_MODULATION_TYPE_E      enModType;
    HI_UNF_TUNER_FE_POLARIZATION_E enPolar;
    HI_UNF_TUNER_FE_FECTYPE_E     enSATType;
    HI_UNF_TUNER_FE_FECRATE_E     enFECRate;
} HI_UNF_TUNER_SAT_SIGNALINFO_S;
```

After modification:

```
typedef struct  hiUNF_TUNER_SAT_SIGNALINFO_S
{
    HI_U32                    u32Freq;
    HI_U32                    u32SymbolRate;
    HI_UNF_MODULATION_TYPE_E      enModType;
    HI_UNF_TUNER_FE_POLARIZATION_E enPolar;
    HI_UNF_TUNER_FE_FECTYPE_E     enSATType;
    HI_UNF_TUNER_FE_FECRATE_E     enFECRate;
    HI_UNF_TUNER_CODE_MODULATION_E enCodeModulation;
} HI_UNF_TUNER_SAT_SIGNALINFO_S;
```

[Reason]

The code and modulation parameters for satellite signals are added.

[Note]

None

[Example]

None

HI_UNF_SAT_CONNECT_PARA_S

[Definition]

Before modification:

```
typedef struct  hiUNF_SAT_CONNECT_PARA_S
{
    HI_U32    u32Freq;
    HI_U32    u32SymbolRate;
    HI_UNF_TUNER_FE_POLARIZATION_E enPolar;
  } HI_UNF_SAT_CONNECT_PARA_S;
```

After modification:

```
typedef struct  hiUNF_SAT_CONNECT_PARA_S
{
    HI_U32    u32Freq;
    HI_U32    u32SymbolRate;
    HI_UNF_TUNER_FE_POLARIZATION_E enPolar;
    HI_U32    u32ScrambleValue;
```

```
} HI_UNF_SAT_CONNECT_PARA_S;
```

[Reason]

The initial value of the physical layer phase scrambled code can be configured when the DVB-S signal frequency is locked.

[Note]

If the scrambled code is not involved when the frequency is locked, this value must be initialized as 0.

[Example]

None

HI_UNF_TUNER_TER_SCAN_PARA_S

[Definition]

Before modification:

```
typedef struct hiUNF_TUNER_TER_SCAN_PARA_S
{
    HI_UNF_TUNER_TER_SCAN_ATTR_S stTer;
    HI_UNF_TUNER_TER_CHANNEL_ATTR_S enChanArray[TER_MAX_TP];
    HI_U32 u32ChanNum;
}HI_UNF_TUNER_TER_SCAN_PARA_S;
```

After modification:

```
typedef struct hiUNF_TUNER_TER_SCAN_PARA_S
{
    HI_UNF_TUNER_TER_SCAN_ATTR_S stTer;
    HI_UNF_TUNER_TER_CHANNEL_ATTR_S enChanArray[TER_MAX_TP];
    HI_U32 u32ChanNum;
    HI_S32 s32TimeOut;
}HI_UNF_TUNER_TER_SCAN_PARA_S;
```

[Reason]

The timeout period can be configured.

[Note]

This parameter is not used currently and will be extended later.

[Example]

None

9.2.3 API

9.2.3.1 New APIs

HI_UNF_TUNER_UNICABLE_ScanUserBands

[Definition]

```
HI_S32 HI_UNF_TUNER_UNICABLE_ScanUserBands(HI_U32 u32TunerId,
HI_UNF_TUNER_UNICABLE_SCAN_PARA_S stScanPara);
```

[Reason]

This API is added for scanning the 950−2150 frequencies to find the user frequency band. It is used to replace the original HI_UNF_UNICABLE_ScanAndInstall_UB.

[Note]

None

[Example]

None

HI_UNF_TUNER_UNICABLE_ExitScanUserBands

[Definition]

```
HI_S32 HI_UNF_TUNER_UNICABLE_ExitScanUserBands(HI_U32 u32TunerId);
```

[Reason]

This API is added for exiting the scan for the user frequency band.

[Note]

None

[Example]

None

HI_UNF_TUNER_UNICABLE_GetUserBandsInfo

[Definition]

```
HI_S32 HI_UNF_TUNER_UNICABLE_GetUserBandsInfo(HI_U32 u32TunerId,
HI_UNF_TUNER_SCR_UB_S **ppUBInfo, HI_U32 *pu32Num);
```

[Reason]

This API is added for obtaining all the scanned user frequency band information. It is used to replace HI_UNF_UNICABLE_GetUBInfo.

[Note]

None

[Example]

None

HI_UNF_TUNER_GetSatTotalStream

[Definition]

```
HI_S32 HI_UNF_TUNER_GetSatTotalStream(HI_U32 u32TunerId, HI_U8
*pu8TotalStream);
```

[Reason]

This API is added for obtaining the number of streams when the frontend transmits VCM signals.

[Note]

None

[Example]

None

HI_UNF_TUNER_GetSatIsiID

[Definition]

```
HI_S32 HI_UNF_TUNER_GetSatIsiID(HI_U32 u32TunerId, HI_U8 u8StreamIndex,
HI_U8 *pu8IsiID);
```

[Reason]

This API is added for obtaining the stream ID based on the stream sequence number when the frontend transmits VCM signals.

[Note]

None

[Example]

None

HI_UNF_TUNER_SetSatIsiID

[Definition]

```
HI_S32 HI_UNF_TUNER_SetSatIsiID(HI_U32 u32TunerId, HI_U8 u8IsiID);
```

[Reason]

This API is added for setting the stream ID to receive the specified VCM stream when the frontend transmits VCM signals and there are multiple streams.

[Note]

None

[Example]

None

9.2.3.2 Deleted APIs

HI_UNF_TUNER_GetAgc

[Definition]

```
HI_S32 HI_UNF_TUNER_GetAgc(HI_U32 u32TunerId,HI_S32 s32CenterFreq, HI_S32
*ps32Agc);
```

[Reason]

The parameters can be ignored.

[Note]

None

[Example]

None

HI_UNF_UNICABLE_SetCurUB

[Definition]

```
HI_S32 HI_UNF_UNICABLE_SetCurUB(HI_U32 u32TunerId);
```

[Reason]

This function is provided in HI_UNF_TUNER_SetLNBConfig.

[Note]

None

[Example]

None

HI_UNF_UNICABLE_ScanAndInstall_UB

[Definition]

```
HI_S32 HI_UNF_UNICABLE_ScanAndInstall_UB(HI_U32 u32TunerId);
```

[Reason]

This API is replaced with HI_UNF_TUNER_UNICABLE_ScanUserBands.

[Note]

None

[Example]

None

HI_UNF_UNICABLE_GetUBInfo

[Definition]

```
HI_S32 HI_UNF_UNICABLE_GetUBInfo(HI_U32 u32TunerId, HI_UNF_TUNER_SCR_UB_S
*pUBInfo);
```

[Reason]

This API is replaced with HI_UNF_TUNER_UNICABLE_GetUserBandsInfo.

[Note]

None

[Example]

None

10 Differences Between UNF 3.2.8 and UNF 3.2.7

10.1 Demux

10.1.1 Overview

The UNF 3.2.8 Demux module has the following change compared with UNF 3.2.7:

The functions of obtaining and setting the descrambler attributes are added.

10.1.1.1 Obtaining and Setting the Descrambler Attributes

The descrambler attributes can be configured and obtained. The changes are described as follows:

API

The following APIs are added:

- HI_UNF_DMX_GetDescramblerAttr
- HI_UNF_DMX_SetDescramblerAttr

10.1.2 API

10.1.2.1 New APIs

HI_UNF_DMX_GetDescramblerAttr

[Definition]

```
HI_S32 HI_UNF_DMX_GetDescramblerAttr(HI_HANDLE hKey,
HI_UNF_DMX_DESCRAMBLER_ATTR_S *pstAttr);
```

[Reason]

This API is added for dynamically obtaining the descrambler attributes.

[Note]

None

[Example]

None

HI_UNF_DMX_SetDescramblerAttr

[Definition]

```
HI_S32 HI_UNF_DMX_SetDescramblerAttr(HI_HANDLE hKey,
HI_UNF_DMX_DESCRAMBLER_ATTR_S *pstAttr)
```

[Reason]

This API is added for dynamically configuring the descrambler attributes.

[Note]

None

[Example]

None

10.2 ADVCA

10.2.1 Overview

The UNF 3.2.8 ADVCA module has the following change compared with UNF 3.2.7:

The chip can boot from the SPI NAND flash and the SD card.

10.2.1.1 Supporting Two More Types of Flash Memories from Which the System Boots

The change is described as follows:

Data Structure

HI_UNF_ADVCA_FLASH_TYPE_E is modified.

10.2.2 Data Structure

10.2.2.1 Modified Data Structure

HI_UNF_ADVCA_FLASH_TYPE_E

[Definition]

Before modification:

```
typedef enum hiUNF_ADVCA_FLASH_TYPE_E
{
    HI_UNF_ADVCA_FLASH_TYPE_SPI = 0,
```

```
    HI_UNF_ADVCA_FLASH_TYPE_NAND,
    HI_UNF_ADVCA_FLASH_TYPE_NOR,
    HI_UNF_ADVCA_FLASH_TYPE_EMMC,
    HI_UNF_ADVCA_FLASH_TYPE_BUTT
}HI_UNF_ADVCA_FLASH_TYPE_E;
```

After modification:

```
typedef enum hiUNF_ADVCA_FLASH_TYPE_E
{
    HI_UNF_ADVCA_FLASH_TYPE_SPI = 0,
    HI_UNF_ADVCA_FLASH_TYPE_NAND,
    HI_UNF_ADVCA_FLASH_TYPE_NOR,
    HI_UNF_ADVCA_FLASH_TYPE_EMMC,
    HI_UNF_ADVCA_FLASH_TYPE_SPI_NAND,
    HI_UNF_ADVCA_FLASH_TYPE_SD,
    HI_UNF_ADVCA_FLASH_TYPE_BUTT
}HI_UNF_ADVCA_FLASH_TYPE_E;;
```

[Reason]

This data structure is modified to allow the chip to boot from the SPI NAND flash and the SD card.

[Note]

None

[Example]

See **sample/advca/sample_ca_opensecboot.c** and **sample/advca/sample_ca_get_otp_fuse.c**.

10.3 Frontend

10.3.1 Overview

The UNF 3.2.8 frontend module has the following change compared with UNF 3.2.7:

The number of supported motor rotation steps increases from 2 to 10.

10.3.1.1 Supporting Eight More Motor Rotation Steps

Eight more motor rotation steps are supported. The changes are described as follows:

Data Structure

HI_UNF_TUNER_DISEQC_MOVE_TYPE_E is modified.

HI_UNF_TUNER_DISEQC_RUN_S is added.

API

HI_UNF_TUNER_DISEQC_RunStep is added.

10.3.2 Data Structure

10.3.2.1 New Data Structure

HI_UNF_TUNER_DISEQC_RUN_S

[Definition]

```
typedef struct hiUNF_TUNER_DISEQC_RUN_S
{
    HI_UNF_TUNER_DISEQC_LEVEL_E    enLevel;
    HI_UNF_TUNER_DISEQC_MOVE_DIR_E  enDir;
    HI_U32  u32RunningSteps;
} HI_UNF_TUNER_DISEQC_RUN_S;
```

[Reason]

This data structure is added to support the configuration of the motor rotation step in HI_UNF_TUNER_DISEQC_RunStep.

[Note]

None

[Example]

None

10.3.2.2 Modified Data Structure

HI_UNF_TUNER_DISEQC_MOVE_TYPE_E

[Definition]

Before modification:

```
typedef enum hiUNF_TUNER_DISEQC_MOVE_TYPE_E
{
    HI_UNF_TUNER_DISEQC_MOVE_STEP_SLOW,
    HI_UNF_TUNER_DISEQC_MOVE_STEP_FAST,
    HI_UNF_TUNER_DISEQC_MOVE_CONTINUE,
    HI_UNF_TUNER_DISEQC_MOVE_TYPE_BUTT
} HI_UNF_TUNER_DISEQC_MOVE_TYPE_E;
```

After modification:

```
typedef enum hiUNF_TUNER_DISEQC_MOVE_TYPE_E
{
    HI_UNF_TUNER_DISEQC_MOVE_STEP_SLOW,
    HI_UNF_TUNER_DISEQC_MOVE_STEP_SLOW1,
```

```
    HI_UNF_TUNER_DISEQC_MOVE_STEP_SLOW2,
    HI_UNF_TUNER_DISEQC_MOVE_STEP_SLOW3,
    HI_UNF_TUNER_DISEQC_MOVE_STEP_SLOW4,
    HI_UNF_TUNER_DISEQC_MOVE_STEP_FAST,
    HI_UNF_TUNER_DISEQC_MOVE_STEP_FAST1,
    HI_UNF_TUNER_DISEQC_MOVE_STEP_FAST2,
    HI_UNF_TUNER_DISEQC_MOVE_STEP_FAST3,
    HI_UNF_TUNER_DISEQC_MOVE_STEP_FAST4,
    HI_UNF_TUNER_DISEQC_MOVE_CONTINUE,
    HI_UNF_TUNER_DISEQC_MOVE_TYPE_BUTT
} HI_UNF_TUNER_DISEQC_MOVE_TYPE_E;
```

[Reason]

Eight motor rotation steps are added:

- HI_UNF_TUNER_DISEQC_MOVE_STEP_SLOW: rotate one step at a time.
- HI_UNF_TUNER_DISEQC_MOVE_STEP_SLOW1: rotate two steps at a time.
- HI_UNF_TUNER_DISEQC_MOVE_STEP_SLOW2: rotate three steps at a time.
- HI_UNF_TUNER_DISEQC_MOVE_STEP_SLOW3: rotate four steps at a time.
- HI_UNF_TUNER_DISEQC_MOVE_STEP_SLOW4: rotate five steps at a time.
- HI_UNF_TUNER_DISEQC_MOVE_STEP_FAST: rotate six steps at a time.
- HI_UNF_TUNER_DISEQC_MOVE_STEP_FAST1: rotate seven steps at a time.
- HI_UNF_TUNER_DISEQC_MOVE_STEP_FAST2: rotate eight steps at a time.
- HI_UNF_TUNER_DISEQC_MOVE_STEP_FAST3: rotate nine steps at a time.
- HI_UNF_TUNER_DISEQC_MOVE_STEP_FAST4: rotate ten steps at a time.
- HI_UNF_TUNER_DISEQC_MOVE_CONTINUE: rotate continuously.

[Note]

None

[Example]

None

10.3.3 API

10.3.3.1 New API

HI_UNF_TUNER_DISEQC_RunStep

[Definition]

```
HI_S32 HI_UNF_TUNER_DISEQC_RunStep(HI_U32 u32TunerId,
HI_UNF_TUNER_DISEQC_RUN_S* pstPara);
```

[Reason]

This API is added for setting the motor rotation step. This API will gradually replace HI_UNF_TUNER_DISEQC_Move because it facilitates the extension of steps.

[Note]

None

[Example]

None

11 Differences Between UNF 3.2.9 and UNF 3.2.8

11.1 Demux

11.1.1 Overview

The UNF 3.2.9 Demux module has the following changes compared with UNF 3.2.8:

- The function of obtaining the index and recorded data synchronously is added.
- The function of obtaining the number of tag ports is added.
- Extended ports are defined.

11.1.1.1 Obtaining the Index and Recorded Data Synchronously

Data Structure

The following data structures are added:

- DMX_MAX_IDX_ACQUIRED_EACH_TIME
- HI_UNF_DMX_REC_DATA_INDEX_S

API

The following APIs are added:

- HI_UNF_DMX_AcquireRecDataAndIndex
- HI_UNF_DMX_ReleaseRecDataAndIndex

11.1.1.2 Obtaining the Number of Tag Ports

Data Structure

HI_UNF_DMX_CAPABILITY_S is modified.

11.1.1.3 Defining Extended Ports

HI_UNF_DMX_PORT_E is modified.

11.1.2 Data Structures

11.1.2.1 New Data Structures

DMX_MAX_IDX_ACQUIRED_EACH_TIME

[Definition]

```
#define DMX_MAX_IDX_ACQUIRED_EACH_TIME  256
```

[Reason]

This data structure is added for defining the maximum number of indexes that can be obtained each time HI_UNF_DMX_AcquireRecDataAndIndex is called.

[Note]

None

[Example]

None

HI_UNF_DMX_REC_DATA_INDEX_S

[Definition]

```
typedef struct hiUNF_DMX_REC_DATA_INDEX_S
{
    HI_U32 u32IdxNum;
    HI_U32 u32RecDataCnt;
    HI_UNF_DMX_REC_INDEX_S stIndex[DMX_MAX_IDX_ACQUIRED_EACH_TIME];
    HI_UNF_DMX_REC_DATA_S  stRecData[2];
} HI_UNF_DMX_REC_DATA_INDEX_S;
```

[Reason]

This data structure is added for defining the index and recorded data that can be obtained by calling HI_UNF_DMX_AcquireRecDataAndIndex.

[Note]

None

[Example]

None

11.1.2.2 Modified Data Structures

HI_UNF_DMX_CAPABILITY_S

[Definition]

Before modification:

```
typedef struct hiUNF_DMX_CAPABILITY_S
```

```
{
    HI_U32 u32IFPortNum;
    HI_U32 u32TSIPortNum;
    HI_U32 u32TSOPortNum;
    HI_U32 u32RamPortNum;
    HI_U32 u32DmxNum;
    HI_U32 u32ChannelNum;
    HI_U32 u32AVChannelNum;
    HI_U32 u32FilterNum;
    HI_U32 u32KeyNum;
    HI_U32 u32RecChnNum;
} HI_UNF_DMX_CAPABILITY_S;
```

After modification:

```
typedef struct hiUNF_DMX_CAPABILITY_S
{
    HI_U32 u32IFPortNum;
    HI_U32 u32TSIPortNum;
    HI_U32 u32TSOPortNum;
    HI_U32 u32RamPortNum;
    HI_U32 u32DmxNum;
    HI_U32 u32ChannelNum;
    HI_U32 u32AVChannelNum;
    HI_U32 u32FilterNum;
    HI_U32 u32KeyNum;
    HI_U32 u32RecChnNum;
    HI_U32 u32TagPortNum;
} HI_UNF_DMX_CAPABILITY_S;
```

[Reason]

The **u32TagPortNum** field is added to define the number of tag ports obtained when HI_UNF_DMX_GetCapability is called.

[Note]

None

[Example]

None

HI_UNF_DMX_PORT_E

[Definition]

Before modification:

```
typedef enum hiUNF_DMX_PORT_E
{
```

```
    HI_UNF_DMX_PORT_IF_0 = 0x0,
    HI_UNF_DMX_PORT_IF_1,
    HI_UNF_DMX_PORT_IF_2,
    HI_UNF_DMX_PORT_IF_3,
    HI_UNF_DMX_PORT_IF_4,
    HI_UNF_DMX_PORT_IF_5,
    HI_UNF_DMX_PORT_IF_6,
    HI_UNF_DMX_PORT_IF_7,
    HI_UNF_DMX_PORT_TSI_0 = 0x20,
    HI_UNF_DMX_PORT_TSI_1,
    HI_UNF_DMX_PORT_TSI_2,
    HI_UNF_DMX_PORT_TSI_3,
    HI_UNF_DMX_PORT_TSI_4,
    HI_UNF_DMX_PORT_TSI_5,
    HI_UNF_DMX_PORT_TSI_6,
    HI_UNF_DMX_PORT_TSI_7,
    HI_UNF_DMX_PORT_RAM_0 = 0x80,
    HI_UNF_DMX_PORT_RAM_1,
    HI_UNF_DMX_PORT_RAM_2,
    HI_UNF_DMX_PORT_RAM_3,
    HI_UNF_DMX_PORT_RAM_4,
    HI_UNF_DMX_PORT_RAM_5,
    HI_UNF_DMX_PORT_RAM_6,
    HI_UNF_DMX_PORT_RAM_7,
    HI_UNF_DMX_PORT_BUTT
} HI_UNF_DMX_PORT_E;
```

After modification:

```
typedef enum hiUNF_DMX_PORT_E
{
    HI_UNF_DMX_PORT_IF_0 = 0x0,
    HI_UNF_DMX_PORT_IF_1,
    HI_UNF_DMX_PORT_IF_2,
    HI_UNF_DMX_PORT_IF_3,
    HI_UNF_DMX_PORT_IF_4,
    HI_UNF_DMX_PORT_IF_5,
    HI_UNF_DMX_PORT_IF_6,
    HI_UNF_DMX_PORT_IF_7,
    HI_UNF_DMX_PORT_IF_8,
    HI_UNF_DMX_PORT_IF_9,
    HI_UNF_DMX_PORT_IF_10,
    HI_UNF_DMX_PORT_IF_11,
    HI_UNF_DMX_PORT_IF_12,
    HI_UNF_DMX_PORT_IF_13,
```

```
    HI_UNF_DMX_PORT_IF_14,
    HI_UNF_DMX_PORT_IF_15,
    HI_UNF_DMX_PORT_TSI_0 = 0x20,
    HI_UNF_DMX_PORT_TSI_1,
    HI_UNF_DMX_PORT_TSI_2,
    HI_UNF_DMX_PORT_TSI_3,
    HI_UNF_DMX_PORT_TSI_4,
    HI_UNF_DMX_PORT_TSI_5,
    HI_UNF_DMX_PORT_TSI_6,
    HI_UNF_DMX_PORT_TSI_7,
    HI_UNF_DMX_PORT_TSI_8,
    HI_UNF_DMX_PORT_TSI_9,
    HI_UNF_DMX_PORT_TSI_10,
    HI_UNF_DMX_PORT_TSI_11,
    HI_UNF_DMX_PORT_TSI_12,
    HI_UNF_DMX_PORT_TSI_13,
    HI_UNF_DMX_PORT_TSI_14,
    HI_UNF_DMX_PORT_TSI_15,
    HI_UNF_DMX_PORT_RAM_0 = 0x80,
    HI_UNF_DMX_PORT_RAM_1,
    HI_UNF_DMX_PORT_RAM_2,
    HI_UNF_DMX_PORT_RAM_3,
    HI_UNF_DMX_PORT_RAM_4,
    HI_UNF_DMX_PORT_RAM_5,
    HI_UNF_DMX_PORT_RAM_6,
    HI_UNF_DMX_PORT_RAM_7,
    HI_UNF_DMX_PORT_RAM_8,
    HI_UNF_DMX_PORT_RAM_9,
    HI_UNF_DMX_PORT_RAM_10,
    HI_UNF_DMX_PORT_RAM_11,
    HI_UNF_DMX_PORT_RAM_12,
    HI_UNF_DMX_PORT_RAM_13,
    HI_UNF_DMX_PORT_RAM_14,
    HI_UNF_DMX_PORT_RAM_15,
    HI_UNF_DMX_PORT_BUTT
} HI_UNF_DMX_PORT_E;
```

[Reason]

The numbers of IF, TSI, and RAM ports are extended.

[Note]

None

[Example]

None

11.1.3 API

11.1.3.1 New APIs

HI_UNF_DMX_AcquireRecDataAndIndex

[Definition]

```
HI_S32  HI_UNF_DMX_AcquireRecDataAndIndex(HI_HANDLE hRecChn,
HI_UNF_DMX_REC_DATA_INDEX_S* pstRecDataIdx);
```

[Reason]

This API is added for obtaining the index data and recorded data synchronously.

[Note]

None

[Example]

None

HI_UNF_DMX_ReleaseRecDataAndIndex

[Definition]

```
HI_S32  HI_UNF_DMX_ReleaseRecDataAndIndex(HI_HANDLE hRecChn,
HI_UNF_DMX_REC_DATA_INDEX_S* pstRecDataIdx);
```

[Reason]

This API is added for releasing the index data and recorded data synchronously.

[Note]

None

[Example]

None

11.2 Sound

11.2.1 Overview

The UNF 3.2.9 sound module has the following change compared with UNF 3.2.8:

The Karaoke function is added.

11.2.1.1 Supporting the Karaoke Function

The Karaoke low-delay solution is implemented. The track type structure is modified, and the APIs for setting and obtaining the delay time are added.

Data Structure

HI_UNF_SND_TRACK_TYPE_E is modified.

API

The following APIs are added:

- HI_UNF_SND_SetLowLatency
- HI_UNF_SND_GetLowLatency

11.2.2 Data Structure

11.2.2.1 Modified Data Structure

HI_UNF_SND_TRACK_TYPE_E

[Definition]

Before modification:

```
typedef enum hiHI_UNF_SND_TRACK_TYPE_E
{
    HI_UNF_SND_TRACK_TYPE_MASTER = 0,
    HI_UNF_SND_TRACK_TYPE_SLAVE,
    HI_UNF_SND_TRACK_TYPE_VIRTUAL,
    HI_UNF_SND_TRACK_TYPE_BUTT
} HI_UNF_SND_TRACK_TYPE_E;
```

After modification:

```
typedef enum hiHI_UNF_SND_TRACK_TYPE_E
{
    HI_UNF_SND_TRACK_TYPE_MASTER = 0,
    HI_UNF_SND_TRACK_TYPE_SLAVE,
    HI_UNF_SND_TRACK_TYPE_VIRTUAL,
    HI_UNF_SND_TRACK_TYPE_LOWLATENCY,
    HI_UNF_SND_TRACK_TYPE_BUTT
} HI_UNF_SND_TRACK_TYPE_E;
```

[Reason]

The low-delay track is added.

[Note]

The low-delay track cannot be bound to the AVPLAY. It supports only the 16-bit dual-channel audio with the sampling rate of 48 kHz. Only one low-delay track is supported.

[Example]

None

11.2.3 APIs

11.2.3.1 New APIs

HI_UNF_SND_SetLowLatency

[Definition]

```
HI_S32 HI_UNF_SND_SetLowLatency(HI_UNF_SND_E enSound,
HI_UNF_SND_OUTPUTPORT_E eOutPort, HI_U32 u32LatecnyMs);
```

[Reason]

This API is added for setting the delay based on the sound device.

[Note]

eOutPort is the output port, which can be configured as required.

u32LatecnyMs ranges from 10 to 40.

[Example]

None

HI_UNF_SND_GetLowLatency

[Definition]

```
HI_S32 HI_UNF_SND_GetLowLatency(HI_UNF_SND_E enSound,
HI_UNF_SND_OUTPUTPORT_E eOutPort, HI_U32 *p32LatecnyMs);
```

[Reason]

This API is added for obtaining the delay based on the sound device.

[Note]

eOutPort is the output port, which can be configured as required.

[Example]

None

11.3 Display

11.3.1 Overview

The UNF 3.2.9 display module has the following change compared with UNF 3.2.8:

The function of setting the standards of the HD and SD channels in the same-source scenario at the same time by using the same API is added.

11.3.1.1 Setting the Standards of the HD and SD Channels at the Same Time

The standards of the HD and SD channels can be configured at the same time by using the same API. This optimizes the standard switching operation. The changes are described as follows:

Data Structure

HI_UNF_DISP_ISOGENY_ATTR_S is added.

API

HI_UNF_DISP_SetIsogenyAttr is added.

11.3.2 Data Structure

11.3.2.1 New Data Structure

HI_UNF_DISP_ISOGENY_ATTR_S

[Definition]

```
typedef struct  hiUNF_DISP_ISOGENY_ATTR_S
{
   HI_UNF_DISP_E        enDisp;
   HI_UNF_ENC_FMT_E      enFormat;
}HI_UNF_DISP_ISOGENY_ATTR_S ;
```

[Reason]

This data structure is added so that the standards of the HD and SD channels can be configured by using one API, which reduces the standard switch time and optimizes the display effect when the standard is switched.

[Note]

This data structure is reserved for future extension.

[Example]

None

11.3.3 API

11.3.3.1 New API

HI_UNF_DISP_SetIsogenyAttr

[Definition]

```
HI_S32 HI_UNF_DISP_SetIsogenyAttr(const HI_UNF_DISP_ISOGENY_ATTR_S
*pstIsogeny, const HI_U32 u32ChannelNum);
```

[Reason]

This API is added for setting the standards of the HD and SD channels at the same time. This optimizes the standard switching operation.

- The standard configuration time is reduced by one or several hundreds of milliseconds compared with the scenario when the SD and HD standards are separately configured.
- This alleviates the issue that the screen flickers for the HDMI output when the standard is switched.

[Note]

This API cannot be used at the same time with the original standard switch API HI_UNF_DISP_SetFormat.

[Example]

See the usage of HI_UNF_DISP_SetFormat.

11.4 VO

11.4.1 Overview

The UNF 3.2.9 VO module has the following change compared with UNF 3.2.8:

The display formats with the resolutions of 3840 x 2160 and 4096 x 2160 are added.

11.4.1.1 Supporting More Display Formats

The display formats with the resolutions of 3840 x 2160 and 4096 x 2160 are added. The change is described as follows:

Data Structure

HI_UNF_ENC_FMT_E is modified.

11.4.2 Data Structure

11.4.2.1 Modified Data Structure

HI_UNF_ENC_FMT_E

[Definition]

Before modification:

```
typedef enum hiUNF_ENC_FMT_E
{
    HI_UNF_ENC_FMT_1080P_60 = 0,     /**<1080p 60 Hz*/
    HI_UNF_ENC_FMT_1080P_50,        /**<1080p 50 Hz*/
    HI_UNF_ENC_FMT_1080P_30,        /**<1080p 30 Hz*/
    HI_UNF_ENC_FMT_1080P_25,        /**<1080p 25 Hz*/
    HI_UNF_ENC_FMT_1080P_24,        /**<1080p 24 Hz*/
```

```
HI_UNF_ENC_FMT_1080i_60,        /**<1080i 60 Hz*/
HI_UNF_ENC_FMT_1080i_50,        /**<1080i 50 Hz*/

HI_UNF_ENC_FMT_720P_60,         /**<720p 60 Hz*/
HI_UNF_ENC_FMT_720P_50,         /**<720p 50 Hz */

HI_UNF_ENC_FMT_576P_50,         /**<576p 50 Hz*/
HI_UNF_ENC_FMT_480P_60,         /**<480p 60 Hz*/

HI_UNF_ENC_FMT_PAL,            /* B D G H I PAL */
HI_UNF_ENC_FMT_PAL_N,           /* (N)PAL       */
HI_UNF_ENC_FMT_PAL_Nc,          /* (Nc)PAL      */

HI_UNF_ENC_FMT_NTSC,           /* (M)NTSC      */
HI_UNF_ENC_FMT_NTSC_J,          /* NTSC-J       */
HI_UNF_ENC_FMT_NTSC_PAL_M,      /* (M)PAL       */

HI_UNF_ENC_FMT_SECAM_SIN,       /**< SECAM_SIN*/
HI_UNF_ENC_FMT_SECAM_COS,       /**< SECAM_COS*/

HI_UNF_ENC_FMT_1080P_24_FRAME_PACKING,
HI_UNF_ENC_FMT_720P_60_FRAME_PACKING,
HI_UNF_ENC_FMT_720P_50_FRAME_PACKING,

HI_UNF_ENC_FMT_861D_640X480_60,
HI_UNF_ENC_FMT_VESA_800X600_60,
HI_UNF_ENC_FMT_VESA_1024X768_60,
HI_UNF_ENC_FMT_VESA_1280X720_60,
HI_UNF_ENC_FMT_VESA_1280X800_60,
HI_UNF_ENC_FMT_VESA_1280X1024_60,
HI_UNF_ENC_FMT_VESA_1360X768_60,
HI_UNF_ENC_FMT_VESA_1366X768_60,
HI_UNF_ENC_FMT_VESA_1400X1050_60,
HI_UNF_ENC_FMT_VESA_1440X900_60,
HI_UNF_ENC_FMT_VESA_1440X900_60_RB,
HI_UNF_ENC_FMT_VESA_1600X900_60_RB,
HI_UNF_ENC_FMT_VESA_1600X1200_60,
HI_UNF_ENC_FMT_VESA_1680X1050_60,
HI_UNF_ENC_FMT_VESA_1680X1050_60_RB,
HI_UNF_ENC_FMT_VESA_1920X1080_60,
HI_UNF_ENC_FMT_VESA_1920X1200_60,
HI_UNF_ENC_FMT_VESA_1920X1440_60,
HI_UNF_ENC_FMT_VESA_2048X1152_60,
```

```
    HI_UNF_ENC_FMT_VESA_2560X1440_60_RB,
    HI_UNF_ENC_FMT_VESA_2560X1600_60_RB,

    HI_UNF_ENC_FMT_3840X2160_24 = 0x100,
    HI_UNF_ENC_FMT_3840X2160_25,
    HI_UNF_ENC_FMT_3840X2160_30,
    HI_UNF_ENC_FMT_4096X2160_24,

    HI_UNF_ENC_FMT_3840X2160_23_976,
    HI_UNF_ENC_FMT_3840X2160_29_97,
    HI_UNF_ENC_FMT_720P_59_94,
    HI_UNF_ENC_FMT_1080P_59_94,
    HI_UNF_ENC_FMT_1080P_29_97,
    HI_UNF_ENC_FMT_1080P_23_976,
    HI_UNF_ENC_FMT_1080i_59_94,
    HI_UNF_ENC_FMT_BUTT
}HI_UNF_ENC_FMT_E;
```

After modification:

```
typedef enum hiUNF_ENC_FMT_E
{
    HI_UNF_ENC_FMT_1080P_60 = 0,     /**<1080p 60 Hz*/
    HI_UNF_ENC_FMT_1080P_50,        /**<1080p 50 Hz*/
    HI_UNF_ENC_FMT_1080P_30,        /**<1080p 30 Hz*/
    HI_UNF_ENC_FMT_1080P_25,        /**<1080p 25 Hz*/
    HI_UNF_ENC_FMT_1080P_24,        /**<1080p 24 Hz*/

    HI_UNF_ENC_FMT_1080i_60,        /**<1080i 60 Hz*/
    HI_UNF_ENC_FMT_1080i_50,        /**<1080i 50 Hz*/

    HI_UNF_ENC_FMT_720P_60,         /**<720p 60 Hz*/
    HI_UNF_ENC_FMT_720P_50,         /**<720p 50 Hz */

    HI_UNF_ENC_FMT_576P_50,         /**<576p 50 Hz*/
    HI_UNF_ENC_FMT_480P_60,         /**<480p 60 Hz*/

    HI_UNF_ENC_FMT_PAL,            /* B D G H I PAL */
    HI_UNF_ENC_FMT_PAL_N,          /* (N)PAL       */
    HI_UNF_ENC_FMT_PAL_Nc,         /* (Nc)PAL      */

    HI_UNF_ENC_FMT_NTSC,           /* (M)NTSC      */
    HI_UNF_ENC_FMT_NTSC_J,         /* NTSC-J       */
    HI_UNF_ENC_FMT_NTSC_PAL_M,      /* (M)PAL       */
```

```
    HI_UNF_ENC_FMT_SECAM_SIN,       /**< SECAM_SIN*/
    HI_UNF_ENC_FMT_SECAM_COS,       /**< SECAM_COS*/

    HI_UNF_ENC_FMT_1080P_24_FRAME_PACKING,
    HI_UNF_ENC_FMT_720P_60_FRAME_PACKING,
    HI_UNF_ENC_FMT_720P_50_FRAME_PACKING,

    HI_UNF_ENC_FMT_861D_640X480_60,
    HI_UNF_ENC_FMT_VESA_800X600_60,
    HI_UNF_ENC_FMT_VESA_1024X768_60,
    HI_UNF_ENC_FMT_VESA_1280X720_60,
    HI_UNF_ENC_FMT_VESA_1280X800_60,
    HI_UNF_ENC_FMT_VESA_1280X1024_60,
    HI_UNF_ENC_FMT_VESA_1360X768_60,
    HI_UNF_ENC_FMT_VESA_1366X768_60,
    HI_UNF_ENC_FMT_VESA_1400X1050_60,
    HI_UNF_ENC_FMT_VESA_1440X900_60,
    HI_UNF_ENC_FMT_VESA_1440X900_60_RB,
    HI_UNF_ENC_FMT_VESA_1600X900_60_RB,
    HI_UNF_ENC_FMT_VESA_1600X1200_60,
    HI_UNF_ENC_FMT_VESA_1680X1050_60,
    HI_UNF_ENC_FMT_VESA_1680X1050_60_RB,
    HI_UNF_ENC_FMT_VESA_1920X1080_60,
    HI_UNF_ENC_FMT_VESA_1920X1200_60,
    HI_UNF_ENC_FMT_VESA_1920X1440_60,
    HI_UNF_ENC_FMT_VESA_2048X1152_60,
    HI_UNF_ENC_FMT_VESA_2560X1440_60_RB,
    HI_UNF_ENC_FMT_VESA_2560X1600_60_RB,

    HI_UNF_ENC_FMT_3840X2160_24 = 0x100,
    HI_UNF_ENC_FMT_3840X2160_25,
    HI_UNF_ENC_FMT_3840X2160_30,
    HI_UNF_ENC_FMT_3840X2160_50,
    HI_UNF_ENC_FMT_3840X2160_60,

    HI_UNF_ENC_FMT_4096X2160_24,
    HI_UNF_ENC_FMT_4096X2160_25,
    HI_UNF_ENC_FMT_4096X2160_30,
    HI_UNF_ENC_FMT_4096X2160_50,
    HI_UNF_ENC_FMT_4096X2160_60,

    HI_UNF_ENC_FMT_3840X2160_23_976,
    HI_UNF_ENC_FMT_3840X2160_29_97,
    HI_UNF_ENC_FMT_720P_59_94,
```

```
    HI_UNF_ENC_FMT_1080P_59_94,
    HI_UNF_ENC_FMT_1080P_29_97,
    HI_UNF_ENC_FMT_1080P_23_976,
    HI_UNF_ENC_FMT_1080i_59_94,
    HI_UNF_ENC_FMT_BUTT
}HI_UNF_ENC_FMT_E;
```

[Reason]

The display formats with the resolutions of 3840 x 2160 and 4096 x 2160 are added.

[Note]

None

[Example]

None

11.5 HDMI

11.5.1 Overview

The UNF 3.2.9 HDMI module has the following changes compared with UNF 3.2.8:

- The YCbCr420 pixel format is supported.
- More color spaces are supported.
- More color space capability sets are supported.

11.5.1.1 Supporting the YCbCr420 Pixel Format

Data Structure

HI_UNF_HDMI_VIDEO_MODE_E is modified.

11.5.1.2 Supporting More Color Spaces

Data Structure

HI_UNF_HDMI_COLORSPACE_E is modified.

11.5.1.3 Supporting More Color Space Capability Sets

Data Structure

HI_UNF_EDID_COLORIMETRY_S is modified.

11.5.2 Data Structures

11.5.2.1 Modified Data Structures

HI_UNF_HDMI_VIDEO_MODE_E

[Definition]

Before modification:

```
typedef enum hiUNF_HDMI_VIDEO_MODE
{
    HI_UNF_HDMI_VIDEO_MODE_RGB444,
    HI_UNF_HDMI_VIDEO_MODE_YCBCR422,
    HI_UNF_HDMI_VIDEO_MODE_YCBCR444,
    HI_UNF_HDMI_VIDEO_MODE_BUTT
}HI_UNF_HDMI_VIDEO_MODE_E;
```

After modification:

```
typedef enum hiUNF_HDMI_VIDEO_MODE
{
    HI_UNF_HDMI_VIDEO_MODE_RGB444,
    HI_UNF_HDMI_VIDEO_MODE_YCBCR422,
    HI_UNF_HDMI_VIDEO_MODE_YCBCR444,
    HI_UNF_HDMI_VIDEO_MODE_YCBCR420,
    HI_UNF_HDMI_VIDEO_MODE_BUTT
}HI_UNF_HDMI_VIDEO_MODE_E;
```

[Reason]

This data structure is modified to support the YCbCr420 pixel format.

[Note]

None

[Example]

None

HI_UNF_HDMI_COLORSPACE_E

[Definition]

Before modification:

```
typedef enum hiUNF_HDMI_COLORSPACE_E
{
    HDMI_COLORIMETRY_NO_DATA,
    HDMI_COLORIMETRY_ITU601,
    HDMI_COLORIMETRY_ITU709,
    HDMI_COLORIMETRY_EXTENDED,
```

```
    HDMI_COLORIMETRY_XVYCC_601,
    HDMI_COLORIMETRY_XVYCC_709,
} HI_UNF_HDMI_COLORSPACE_E;
```

After modification:

```
typedef enum hiUNF_HDMI_COLORSPACE_E
{
    HDMI_COLORIMETRY_NO_DATA,
    HDMI_COLORIMETRY_ITU601,
    HDMI_COLORIMETRY_ITU709,
    HDMI_COLORIMETRY_EXTENDED,
    HDMI_COLORIMETRY_XVYCC_601,
    HDMI_COLORIMETRY_XVYCC_709,
    HDMI_COLORIMETRY_S_YCC_601,
    HDMI_COLORIMETRY_ADOBE_YCC_601,
    HDMI_COLORIMETRY_ADOBE_RGB,
    HDMI_COLORIMETRY_2020_CONST_LUMINOUS,
    HDMI_COLORIMETRY_2020_NON_CONST_LUMINOUS,
} HI_UNF_HDMI_COLORSPACE_E;
```

[Reason]

This data structure is modified to support the S_YCC_601, ADOBE_YCC_601, ADOBE_RGB, BT2020cYCC, BT2020RGB, and BT2020YCC color spaces.

[Note]

None

[Example]

None

HI_UNF_EDID_COLORIMETRY_S

[Definition]

Before modification:

```
typedef struct hiUNF_EDID_COLORIMETRY_S
{
    HI_BOOL    bxvYCC601     ;
    HI_BOOL    bxvYCC709     ;
} HI_UNF_EDID_COLORIMETRY_S;
```

After modification:

```
typedef struct hiUNF_EDID_COLORIMETRY_S
{
    HI_BOOL    bxvYCC601     ;
    HI_BOOL    bxvYCC709     ;
```

```
    HI_BOOL    bsYCC601      ;
    HI_BOOL    bAdobleYCC601  ;
    HI_BOOL    bAdobleRGB     ;
    HI_BOOL    bBT2020cYCC    ;
    HI_BOOL    bBT2020YCC     ;
    HI_BOOL    bBT2020RGB     ;
} HI_UNF_EDID_COLORIMETRY_S;
```

[Reason]

This data structure is modified to support the S_YCC_601, ADOBE_YCC_601, ADOBE_RGB, BT2020cYCC, BT2020RGB, and BT2020YCC color space capability sets.

[Note]

None

[Example]

None

11.6 Common

11.6.1 Overview

The UNF 3.2.9 common module has the following change compared with UNF 3.2.8:

The function of converting the 64-bit chip ID into the 32-bit chip ID is added.

11.6.1.1 Converting the 64-Bit Chip ID into the 32-Bit Chip ID

HI_SYS_CRC32 is added for converting the 64-bit chip ID into the 32-bit chip ID.

API

HI_SYS_CRC32 is added.

11.6.2 API

11.6.2.1 New API

HI_SYS_CRC32

[Definition]

```
HI_S32 HI_SYS_CRC32(HI_U8 *pu8Src, HI_U32 u32SrcLen, HI_U32 *pu32Dst);
```

[Reason]

This API is added for converting the 64-bit chip ID into the 32-bit chip ID.

[Note]

The first parameter is the buffer pointer to the input data, that is, pointer to the 64-bit chip ID. The second parameter is the buffer length, and the third parameter is the buffer pointer to the converted 32-bit chip ID.

[Example]

None

11.7 Cipher

11.7.1 Overview

The UNF 3.2.9 cipher module has the following change compared with UNF 3.2.8:

The function of encrypting and decrypting data by using the STB_ROOT_KEY is added.

11.7.1.1 Encrypting and Decrypting Data by Using the STB_ROOT_KEY

Data can be encrypted and decrypted by using the STB_ROOT_KEY. The change is described as follows:

Data Structure

HI_UNF_CIPHER_CA_TYPE_E is modified.

API

None

11.7.2 Data Structure

11.7.2.1 Modified Data Structure

HI_UNF_CIPHER_CA_TYPE_E

[Definition]

Before modification:

```
typedef enum hiUNF_CIPHER_CA_TYPE_E
{
    HI_UNF_CIPHER_CA_TYPE_R2R   = 0x0,
    HI_UNF_CIPHER_CA_TYPE_SP,
    HI_UNF_CIPHER_CA_TYPE_CSA2,
    HI_UNF_CIPHER_CA_TYPE_CSA3,
    HI_UNF_CIPHER_CA_TYPE_MISC,
    HI_UNF_CIPHER_CA_TYPE_GDRM,
    HI_UNF_CIPHER_CA_TYPE_BLPK,
    HI_UNF_CIPHER_CA_TYPE_LPK,
    HI_UNF_CIPHER_CA_TYPE_IRDETO_HCA,
}HI_UNF_CIPHER_CA_TYPE_E;
```

After modification:

```
typedef enum hiUNF_CIPHER_CA_TYPE_E
{
    HI_UNF_CIPHER_CA_TYPE_R2R   = 0x0,
    HI_UNF_CIPHER_CA_TYPE_SP,
    HI_UNF_CIPHER_CA_TYPE_CSA2,
    HI_UNF_CIPHER_CA_TYPE_CSA3,
    HI_UNF_CIPHER_CA_TYPE_MISC,
    HI_UNF_CIPHER_CA_TYPE_GDRM,
    HI_UNF_CIPHER_CA_TYPE_BLPK,
    HI_UNF_CIPHER_CA_TYPE_LPK,
    HI_UNF_CIPHER_CA_TYPE_IRDETO_HCA,
    HI_UNF_CIPHER_CA_TYPE_STBROOTKEY,
    HI_UNF_CIPHER_CA_TYPE_BUTT
}HI_UNF_CIPHER_CA_TYPE_E;
```

[Reason]

This data structure is modified to support the function of encrypting and decrypting data by using the STB_ROOT_KEY.

[Note]

None

[Example]

See **sample\cipher\sample_anticopy.c**.

11.8 CC

11.8.1 Overview

The UNF 3.2.9 CC module has the following change compared with UNF 3.2.8:

The function of transmitting the even and odd field data simultaneously to the VO for output is added.

11.8.1.1 Transmitting the Even and Odd Field Data Simultaneously to the VO for Output

The even and odd field data in a frame can be transmitted simultaneously to the VO. The change is described as follows:

API

HI_UNF_CC_VBI_CB_FN is modified.

11.8.2 API

11.8.2.1 Modified API

HI_UNF_CC_VBI_CB_FN

[Definition]

Before modification:

```
typedef HI_S32 (*HI_UNF_CC_VBI_CB_FN)(HI_U32 u32UserData,
HI_UNF_CC_VBI_DADA_S *pstVBIDataField);
```

After modification:

```
typedef HI_S32 (*HI_UNF_CC_VBI_CB_FN)(HI_U32 u32UserData,
HI_UNF_CC_VBI_DADA_S *pstVBIOddDataField1,HI_UNF_CC_VBI_DADA_S
*pstVBIEvenDataField2);
```

[Reason]

In the VBI callback function, the even and odd field data is transmitted simultaneously so that the even and odd field data is in the same frame.

[Note]

None

[Example]

None

11.9 PMOC

11.9.1 Overview

The UNF 3.2.9 PMOC module has the following change compared with UNF 3.2.8:

More modules are added to the list of modules that can be separately controlled during intelligent standby.

11.9.1.1 Adding More Modules to the List of Modules that Can Be Separately Controlled During Intelligent Standby

Some module enumerations are added. You can determine whether to power off these modules separately during intelligent standby.

Data Structure

HI_UNF_PMOC_HOLD_MOD_E is modified.

11.9.2 Data Structure

11.9.2.1 Modified Data Structure

HI_UNF_PMOC_HOLD_MOD_E

[Definition]

Before modification:

```
typedef enum hiUNF_PMOC_HOLD_MOD_E
{
    HI_UNF_PMOC_HOLD_ETH = 0x0001,
    HI_UNF_PMOC_HOLD_WIFI = 0x0002,
    HI_UNF_PMOC_HOLD_USB = 0x0004,
    HI_UNF_PMOC_HOLD_TUNER = 0x0008,
    HI_UNF_PMOC_HOLD_DEMUX = 0x0010,
    HI_UNF_PMOC_HOLD_SDIO = 0x0020,
    HI_UNF_PMOC_HOLD_BUTT
}HI_UNF_PMOC_HOLD_MOD_E;
```

After modification:

```
typedef enum hiUNF_PMOC_HOLD_MOD_E
{
    HI_UNF_PMOC_HOLD_ETH = 0x0001,
    HI_UNF_PMOC_HOLD_WIFI = 0x0002,
    HI_UNF_PMOC_HOLD_USB = 0x0004,
    HI_UNF_PMOC_HOLD_TUNER = 0x0008,
    HI_UNF_PMOC_HOLD_DEMUX = 0x0010,
    HI_UNF_PMOC_HOLD_SDIO = 0x0020,
    HI_UNF_PMOC_HOLD_SCI = 0x0040,
    HI_UNF_PMOC_HOLD_VENC = 0x0080,
    HI_UNF_PMOC_HOLD_PNG = 0x0100,
    HI_UNF_PMOC_HOLD_JPGE = 0x0200,
    HI_UNF_PMOC_HOLD_JPEG = 0x0400,
    HI_UNF_PMOC_HOLD_WDG = 0x0800,
    HI_UNF_PMOC_HOLD_HDMI = 0x1000,
    HI_UNF_PMOC_HOLD_VO = 0x2000,
    HI_UNF_PMOC_HOLD_DISP = 0x4000,
    HI_UNF_PMOC_HOLD_AO = 0x8000,
    HI_UNF_PMOC_HOLD_AI = 0x10000,
    HI_UNF_PMOC_HOLD_ADSP = 0x20000,
    HI_UNF_PMOC_HOLD_CIPHER = 0x40000,
    HI_UNF_PMOC_HOLD_VDEC = 0x80000,
    HI_UNF_PMOC_HOLD_VPSS = 0x100000,
    HI_UNF_PMOC_HOLD_OTP = 0x200000,
```

```
    HI_UNF_PMOC_HOLD_TDE = 0x400000,
    HI_UNF_PMOC_HOLD_I2C = 0x800000,
    HI_UNF_PMOC_HOLD_GPIO = 0x1000000,
    HI_UNF_PMOC_HOLD_BUTT = 0x80000000,
}HI_UNF_PMOC_HOLD_MOD_E;
```

[Reason]

This data structure is modified to separately power off modules such as the display module during intelligent standby.

[Note]

The enumerations indicate the modules that are not powered off during intelligent standby. If you want to power off a certain module, you need to transfer the ORed enumeration logic of all other modules as a parameter.

[Example]

None

11.10 HiPlayer

11.10.1 Overview

The UNF 3.2.9 HiPlayer module has the following changes compared with UNF 3.2.8:

- The invoke interfaces are extended.
- The function of obtaining the seek information and program switch time in the proc information is added.
- The fence sync API is added.

11.10.1.1 Extending the Invoke Interfaces

The following functions are extended:

- Obtains the PTS of the last frame in the vide buffer.
- Obtains the PTS of the last frame in the audio buffer.
- Sets the stream recording function by using proc.
- Obtains seek information in the proc information.
- Obtains the program switch information in the proc information.

Data Structure

HI_FORMAT_INVOKE_ID_E is modified.

11.10.1.2 Obtaining the Seek Information and Program Switch Time in the Proc Information

The seek information and program switch time can be obtained in the proc information.

Data Structure

The following data structures are added:

- HI_SVR_PLAYER_PROC_SEEKINFO_S
- HI_SVR_PLAYER_PROC_SWITCHPG_INFOTYPE_E
- HI_SVR_PLAYER_PROC_SWITCHPG_S

11.10.1.3 Supporting the Fence Sync API

The fence sync API is added.

Data Structure

HI_SVR_PLAYER_PROC_SEEKINFO_S is modified.

API

HI_SVR_VSINK_CheckFence is added.

11.10.2 Data Structure

11.10.2.1 New Data Structures

HI_SVR_PLAYER_PROC_SEEKINFO_S

[Definition]

```
typedef struct hiSVR_PLAYER_PROC_SEEKINFO_S
{
    HI_U32      u32DoReadSeek;
    HI_U32      u32ReadSeekDone;
    HI_U32      u32DoSeekFrameBinary;
    HI_U32      u32SeekFrameBinaryDone;
    HI_U32      u32DoSeekFrameGeneric;
    HI_U32      u32SeekFrameGenericDone;
    HI_U32      u32DoInitInput;
    HI_U32      u32DoAvioOpenH;
    HI_U32      u32AvioOpenHDone;
    HI_U32      u32DoAvProbeInputBuffer;
    HI_U32      u32AvProbeInputBufferDone;

    HI_U32      u32DoHiSvrFormatSeekPts;
    HI_U32      u32CmdSeek;
    HI_U32      u32DoAvformatOpenInput;
    HI_U32      u32AvformatOpenInputDone;
    HI_U32      u32DoSvrFormatFindStream;
    HI_U32      u32SvrFormatFindStreamDone;
    HI_U32      u32DoSvrFormatGetFileInfo;
```

```
    HI_U32     u32SvrFormatGetFileInfoDone;

} HI_SVR_PLAYER_PROC_SEEKINFO_S;
```

[Reason]

This data structure is added for obtaining the proc information about the seek operation.

[Note]

None

[Example]

None

HI_SVR_PLAYER_PROC_SWITCHPG_INFOTYPE_E

[Definition]

```
typedef enum hiSVR_PLAYER_PROC_SWITCHPG_INFOTYPE_E
{
    HI_SVR_PLAYER_PROC_DO_STOP=1,
    HI_SVR_PLAYER_PROC_HIMEDIAPLAYER_CONSTRUCT,
    HI_SVR_PLAYER_PROC_SETDATASOURCE,
    HI_SVR_PLAYER_PROC_UNF_AVPLAY_CREATE,
    HI_SVR_PLAYER_PROC_DO_PREPARE,
    HI_SVR_PLAYER_PROC_PREPARE_ASYNC_COMPLETE,
    HI_SVR_PLAYER_PROC_DO_START_ENTER,
    HI_SVR_PLAYER_PROC_PLAYER_STATE_PLAY,
    HI_SVR_PLAYER_PROC_MEDIA_INFO_FIRST_FRAME_TIME,
    HI_SVR_PLAYER_PROC_DO_RESET,
    HI_SVR_PLAYER_PROC_DO_DESTRUCTOR,
HI_SVR_PLAYER_ATTR_BUTT
} HI_SVR_PLAYER_PROC_SWITCHPG_INFOTYPE_E;
```

[Reason]

This data structure is added for defining the program switch enumerations.

[Note]

None

[Example]

None

HI_SVR_PLAYER_PROC_SWITCHPG_S

[Definition]

```
typedef struct hiSVR_PLAYER_PROC_SWITCHPG_S
{
```

```
    HI_SVR_PLAYER_PROC_SWITCHPG_INFOTYPE_E eType;
    HI_U32 u32DoStop;
    HI_U32 u32HiMediaPlayerConstruct;
    HI_U32 u32SetDataSource;
    HI_U32 u32DoCreateAVPlay;
    HI_U32 u32DoPrepare;
    HI_U32 u32prepareAsyncComplete;
    HI_U32 u32DoStartEnter;
    HI_U32 u32PlayedEvent;
    HI_U32 u32FirstFrameTime;
    HI_U32 u32DoReset;
    HI_U32 u32DoDestructor;

} HI_SVR_PLAYER_PROC_SWITCHPG_S;
```

[Reason]

This data structure is added for obtaining the program switch information.

[Note]

None

[Example]

None

11.10.2.2 Modified Data Structures

HI_FORMAT_INVOKE_ID_E

[Definition]

Before modification:

```
typedef enum hiFORMAT_INVOKE_ID_E
{
    ...
    HI_FORMAT_INVOKE_SET_EXTERNALFORMAT,
    HI_FORMAT_INVOKE_FIND_BESTSTREAM,
    HI_FORMAT_INVOKE_SET_EXTERNAL_AUDIOTRACK,
    HI_FORMAT_INVOKE_SET_SEEK_MODE,
    HI_FORMAT_INVOKE_BUTT
} HI_FORMAT_INVOKE_ID_E;
```

After modification:

```
typedef enum hiFORMAT_INVOKE_ID_E
{
    ...
    HI_FORMAT_INVOKE_SET_EXTERNALFORMAT,
```

```
    HI_FORMAT_INVOKE_FIND_BESTSTREAM,
    HI_FORMAT_INVOKE_SET_EXTERNAL_AUDIOTRACK,
    HI_FORMAT_INVOKE_SET_SEEK_MODE,
    HI_FORMAT_INVOKE_GET_LAST_VIDBUF_PTS,
    HI_FORMAT_INVOKE_GET_LAST_AUDBUF_PTS,
    HI_FORMAT_INVOKE_SET_RECORD_STREAM,
    HI_FORMAT_INVOKE_GET_TRACE_SEEK,
    HI_FORMAT_INVOKE_GET_SWITCH_PG_TIME,
    HI_FORMAT_INVOKE_BUTT
} HI_FORMAT_INVOKE_ID_E;
```

[Reason]

The invoke interfaces for implementing the following functions are extended based on demands and function requirements:

- Obtains the PTS of the last frame in the vide buffer.
- Obtains the PTS of the last frame in the audio buffer.
- Sets the stream recording function by using proc.
- Obtains seek information in the proc information.
- Obtains the program switch information in the proc information.

[Note]

None

[Example]

None

HI_SVR_PLAYER_PROC_SEEKINFO_S

[Definition]

Before modification:

```
typedef enum hiSVR_VSINK_CMD_E
{
    HI_SVR_VSINK_SET_DIMENSIONS,
    HI_SVR_VSINK_SET_FORMAT,
    HI_SVR_VSINK_SET_PICNB,
    HI_SVR_VSINK_WRITE_PIC,
    HI_SVR_VSINK_SET_CROP,
    HI_SVR_VSINK_GET_MINBUFNB,
} HI_SVR_VSINK_CMD_E;
```

After modification:

```
typedef enum hiSVR_VSINK_CMD_E
{
    HI_SVR_VSINK_SET_DIMENSIONS,
    HI_SVR_VSINK_SET_FORMAT,
```

```
    HI_SVR_VSINK_SET_PICNB,
    HI_SVR_VSINK_WRITE_PIC,
    HI_SVR_VSINK_SET_CROP,
    HI_SVR_VSINK_GET_MINBUFNB,
    HI_SVR_VSINK_CHECK_FENCE,
} HI_SVR_VSINK_CMD_E;
```

[Reason]

This data structure is modified to support the enumeration variable corresponding to the fence sync API.

[Note]

This data structure applies only to the Android version but not the Linux version.

[Example]

None

11.10.3 API

11.10.3.1 New API

HI_SVR_VSINK_CheckFence

[Definition]

```
static inline HI_S32 HI_SVR_VSINK_CheckFence(HI_SVR_VSINK_S* vsink,
       HI_SVR_PICTURE_S* pic)
```

[Reason]

This API is added for checking the fence sync status.

[Note]

This data structure applies only to the Android version but not the Linux version.

[Example]

None

11.11 PVR

11.11.1 Overview

The UNF 3.2.9 PVR module has the following change compared with UNF 3.2.8:

The slow backward playback mode is defined.

11.11.1.1 Defining the Slow Backward Playback Mode

Data Structure

HI_UNF_PVR_PLAY_STATE_E is modified.

API

None

11.11.2 Data Structure

11.11.2.1 Modified Data Structure

HI_UNF_PVR_PLAY_STATE_E

[Definition]

Before modification:

```
typedef enum hiUNF_PVR_PALY_STATE_E
{
    HI_UNF_PVR_PLAY_STATE_INVALID,
    HI_UNF_PVR_PLAY_STATE_INIT,
    HI_UNF_PVR_PLAY_STATE_PLAY,
    HI_UNF_PVR_PLAY_STATE_PAUSE,
    HI_UNF_PVR_PLAY_STATE_FF,
    HI_UNF_PVR_PLAY_STATE_FB,
    HI_UNF_PVR_PLAY_STATE_SF,
    HI_UNF_PVR_PLAY_STATE_STEPF,
    HI_UNF_PVR_PLAY_STATE_STEPB,
    HI_UNF_PVR_PLAY_STATE_STOP,
    HI_UNF_PVR_PLAY_STATE_BUTT
} HI_UNF_PVR_PLAY_STATE_E;
```

After modification:

```
typedef enum hiUNF_PVR_PALY_STATE_E
{
    HI_UNF_PVR_PLAY_STATE_INVALID,
    HI_UNF_PVR_PLAY_STATE_INIT,
    HI_UNF_PVR_PLAY_STATE_PLAY,
    HI_UNF_PVR_PLAY_STATE_PAUSE,
    HI_UNF_PVR_PLAY_STATE_FF,
    HI_UNF_PVR_PLAY_STATE_FB,
    HI_UNF_PVR_PLAY_STATE_SF,
    HI_UNF_PVR_PLAY_STATE_SB,
    HI_UNF_PVR_PLAY_STATE_STEPF,
```

```
    HI_UNF_PVR_PLAY_STATE_STEPB,
    HI_UNF_PVR_PLAY_STATE_STOP,
    HI_UNF_PVR_PLAY_STATE_BUTT
} HI_UNF_PVR_PLAY_STATE_E;
```

[Reason]

This data structure is modified to complete the playback status of the PVR.

[Note]

The PVR playback status is completed, but the function is not supported currently.

[Example]

None

11.12 Frontend

11.12.1 Overview

The UNF 3.2.9 frontend module has the following changes compared with UNF 3.2.8:

- The AV2018 and SI2144 tuners are supported.
- The function of configuring the initial phase of the scrambled code at the physical layer is added.

11.12.1.1 Supporting the AV2018 and SI2144 Tuners

The AV2018 and SI2144 tuners are supported. The change is described as follows:

Data Structure

HI_UNF_TUNER_DEV_TYPE_E is modified.

11.12.1.2 Configuring the Initial Phase of the Scrambled Code at the Physical Layer

The initial phase of the scrambled code at the physical layer can be configured. The change is described as follows:

API

HI_UNF_TUNER_SetScramble is added.

11.12.2 Data Structure

11.12.2.1 Modified Data Structure

HI_UNF_TUNER_DEV_TYPE_E

[Definition]

Before modification:

```
typedef enum    hiUNF_TUNER_DEV_TYPE_E
{
    HI_UNF_TUNER_DEV_TYPE_XG_3BL,
    HI_UNF_TUNER_DEV_TYPE_CD1616,
    HI_UNF_TUNER_DEV_TYPE_ALPS_TDAE,
    HI_UNF_TUNER_DEV_TYPE_TDCC,
    HI_UNF_TUNER_DEV_TYPE_TDA18250,
    HI_UNF_TUNER_DEV_TYPE_CD1616_DOUBLE,
    HI_UNF_TUNER_DEV_TYPE_MT2081,
    HI_UNF_TUNER_DEV_TYPE_TMX7070X,
    HI_UNF_TUNER_DEV_TYPE_R820C,
    HI_UNF_TUNER_DEV_TYPE_MXL203,
    HI_UNF_TUNER_DEV_TYPE_AV2011,
    HI_UNF_TUNER_DEV_TYPE_SHARP7903,
    HI_UNF_TUNER_DEV_TYPE_MXL101,
    HI_UNF_TUNER_DEV_TYPE_MXL603,
    HI_UNF_TUNER_DEV_TYPE_IT9170,
    HI_UNF_TUNER_DEV_TYPE_IT9133,
    HI_UNF_TUNER_DEV_TYPE_TDA6651,
    HI_UNF_TUNER_DEV_TYPE_TDA18250B,
    HI_UNF_TUNER_DEV_TYPE_M88TS2022,
    HI_UNF_TUNER_DEV_TYPE_RDA5815,
    HI_UNF_TUNER_DEV_TYPE_MXL254,
    HI_UNF_TUNER_DEV_TYPE_CXD2861,
    HI_UNF_TUNER_DEV_TYPE_SI2147,
    HI_UNF_TUNER_DEV_TYPE_RAFAEL836,
    HI_UNF_TUNER_DEV_TYPE_MXL608,
    HI_UNF_TUNER_DEV_TYPE_MXL214,
    HI_UNF_TUNER_DEV_TYPE_TDA18280,
    HI_UNF_TUNER_DEV_TYPE_TDA182I5A,
    HI_UNF_TUNER_DEV_TYPE_BUTT
} HI_UNF_TUNER_DEV_TYPE_E ;
```

After modification:

```
typedef enum    hiUNF_TUNER_DEV_TYPE_E
{
    HI_UNF_TUNER_DEV_TYPE_XG_3BL,
    HI_UNF_TUNER_DEV_TYPE_CD1616,
    HI_UNF_TUNER_DEV_TYPE_ALPS_TDAE,
    HI_UNF_TUNER_DEV_TYPE_TDCC,
    HI_UNF_TUNER_DEV_TYPE_TDA18250,
    HI_UNF_TUNER_DEV_TYPE_CD1616_DOUBLE,
    HI_UNF_TUNER_DEV_TYPE_MT2081,
```

```
    HI_UNF_TUNER_DEV_TYPE_TMX7070X,
    HI_UNF_TUNER_DEV_TYPE_R820C,
    HI_UNF_TUNER_DEV_TYPE_MXL203,
    HI_UNF_TUNER_DEV_TYPE_AV2011,
    HI_UNF_TUNER_DEV_TYPE_SHARP7903,
    HI_UNF_TUNER_DEV_TYPE_MXL101,
    HI_UNF_TUNER_DEV_TYPE_MXL603,
    HI_UNF_TUNER_DEV_TYPE_IT9170,
    HI_UNF_TUNER_DEV_TYPE_IT9133,
    HI_UNF_TUNER_DEV_TYPE_TDA6651,
    HI_UNF_TUNER_DEV_TYPE_TDA18250B,
    HI_UNF_TUNER_DEV_TYPE_M88TS2022,
    HI_UNF_TUNER_DEV_TYPE_RDA5815,
    HI_UNF_TUNER_DEV_TYPE_MXL254,
    HI_UNF_TUNER_DEV_TYPE_CXD2861,
    HI_UNF_TUNER_DEV_TYPE_SI2147,
    HI_UNF_TUNER_DEV_TYPE_RAFAEL836,
    HI_UNF_TUNER_DEV_TYPE_MXL608,
    HI_UNF_TUNER_DEV_TYPE_MXL214,
    HI_UNF_TUNER_DEV_TYPE_TDA18280,
    HI_UNF_TUNER_DEV_TYPE_TDA182I5A,
    HI_UNF_TUNER_DEV_TYPE_SI2144,
    HI_UNF_TUNER_DEV_TYPE_AV2018,
    HI_UNF_TUNER_DEV_TYPE_BUTT
} HI_UNF_TUNER_DEV_TYPE_E ;
```

[Reason]

This data structure is modified to support the drivers for the SI2144 and AV2018 tuners.

[Note]

None

[Example]

None

11.12.3 API

11.12.3.1 New API

HI_UNF_TUNER_SetScramble

[Definition]

```
HI_S32 HI_UNF_TUNER_SetScramble(HI_U32 u32TunerId,HI_U32 u32N);
```

[Reason]

This API is added for setting the initial phase of the scrambled code at the physical layer.

[Note]

None

[Example]

None

12 Differences Between UNF 3.2.10 and UNF 3.2.9

12.1 ADVCA

12.1.1 Overview

The ADVCA module of UNF 3.2.10 has the following changes compared with that of UNF 3.2.9:

- New key ladder types are added.
- The Panaccess advanced CA chip is supported.
- The function of setting and obtaining the secure boot enable flag is added.
- The function of setting and obtaining the type of flash memory from which the system boots is added.
- The function of setting and obtaining multiple lock flag bits is added.

12.1.1.1 Adding New Ladder Types

Three types of key ladders (BLPK, LPK, and IRDETO HCA) are added. The data structure is changed as follows:

Data Structure

HI_UNF_ADVCA_CA_TYPE_E is modified.

12.1.1.2 Supporting the Panaccess Advanced CA Chip

The data structure is changed as follows:

Data Structure

HI_UNF_ADVCA_VENDORID_E is modified.

12.1.1.3 Setting and Obtaining the Secure Boot Enable Flag

This feature allows you to enable the secure boot function or to check whether the secure boot is enabled. The data structure is changed as follows:

Data Structure

HI_UNF_ADVCA_OTP_FUSE_E is modified.

12.1.1.4 Setting and Obtaining the Type of Flash Memory from Which the System Boots

The data structure is changed as follows:

Data Structures

- HI_UNF_ADVCA_OTP_BOOT_FLASH_TYPE_ATTR_S is added.
- HI_UNF_ADVCA_OTP_FUSE_E is modified.

12.1.1.5 Setting and Obtaining Multiple Lock Flag Bits

The function of setting and obtaining eight types of lock flags (RSA KEY LOCK, STBSN LOCK, MSID LOCK, VERSIONID LOCK, OEM ROOTKEY LOCK, R2R ROOTKEY LOCK, JTAG KEY LOCK, and TZ AREA LOCK) are added. The data structure is changed as follows:

Data Structure

HI_UNF_ADVCA_OTP_FUSE_E is modified.

12.1.2 Data Structures

12.1.2.1 New Data Structure

HI_UNF_ADVCA_OTP_BOOT_FLASH_TYPE_ATTR_S

[Definition]

```
typedef struct hiUNF_ADVCA_OTP_BOOT_FLASH_TYPE_ATTR_S

{

   HI_BOOL bBootSelCtrl;   /**<0--the boot flash type is defined by
chipset pin, 1--the boot flash type is defined by OTP value*/

   HI_UNF_ADVCA_FLASH_TYPE_E enFlashType; /**<Boot flash type, only valid
when bBootSelCtrl is 1*/

}HI_UNF_ADVCA_OTP_BOOT_FLASH_TYPE_ATTR_S;
```

[Reason]

This data structure is added for setting and obtaining the type of flash memory from which the system boots.

[Note]

None

[Example]

None

12.1.2.2 Modified Data Structures

HI_UNF_ADVCA_CA_TYPE_E

[Definition]

Before modification:

```
typedef enum hiUNF_ADVCA_CA_TYPE_E
{
    HI_UNF_ADVCA_CA_TYPE_R2R      = 0x0,
    HI_UNF_ADVCA_CA_TYPE_SP       = 0x1,
    HI_UNF_ADVCA_CA_TYPE_CSA2     = 0x1,
    HI_UNF_ADVCA_CA_TYPE_CSA3     = 0x1,
    HI_UNF_ADVCA_CA_TYPE_MISC     = 0x2,
    HI_UNF_ADVCA_CA_TYPE_GDRM     = 0x3,
}HI_UNF_ADVCA_CA_TYPE_E;
```

After modification:

```
typedef enum hiUNF_ADVCA_CA_TYPE_E
{
    HI_UNF_ADVCA_CA_TYPE_R2R   = 0x0,
    HI_UNF_ADVCA_CA_TYPE_SP,
    HI_UNF_ADVCA_CA_TYPE_CSA2,
    HI_UNF_ADVCA_CA_TYPE_CSA3,
    HI_UNF_ADVCA_CA_TYPE_MISC,
    HI_UNF_ADVCA_CA_TYPE_GDRM,
    HI_UNF_ADVCA_CA_TYPE_BLPK,
    HI_UNF_ADVCA_CA_TYPE_LPK,
    HI_UNF_ADVCA_CA_TYPE_IRDETO_HCA,
}HI_UNF_ADVCA_CA_TYPE_E;
```

[Reason]

This data structure is modified to add three types of key ladders.

[Note]

None

[Example]

None

HI_UNF_ADVCA_VENDORID_E

[Definition]

Before modification:

```
typedef enum hiUNF_ADVCA_VENDORID_E
{
```

```
    HI_UNF_ADVCA_NULL      = 0x00,
    HI_UNF_ADVCA_NAGRA     = 0x01,
    HI_UNF_ADVCA_IRDETO    = 0x02,
    HI_UNF_ADVCA_CONAX     = 0x03,
    HI_UNF_ADVCA_SUMA      = 0x05,
    HI_UNF_ADVCA_NOVEL     = 0x06,
    HI_UNF_ADVCA_VERIMATRIX = 0x07,
    HI_UNF_ADVCA_CTI       = 0x08,
    HI_UNF_ADVCA_COMMONDCA  = 0x0b,
    HI_UNF_ADVCA_DCAS      = 0x0c,
    HI_UNF_ADVCA_VENDORIDE_BUTT
}HI_UNF_ADVCA_VENDORID_E;
```

After modification:

```
typedef enum hiUNF_ADVCA_VENDORID_E
{
    HI_UNF_ADVCA_NULL      = 0x00,
    HI_UNF_ADVCA_NAGRA     = 0x01,
    HI_UNF_ADVCA_IRDETO    = 0x02,
    HI_UNF_ADVCA_CONAX     = 0x03,
    HI_UNF_ADVCA_SUMA      = 0x05,
    HI_UNF_ADVCA_NOVEL     = 0x06,
    HI_UNF_ADVCA_VERIMATRIX = 0x07,
    HI_UNF_ADVCA_CTI       = 0x08,
    HI_UNF_ADVCA_COMMONCA   = 0x0b,
    HI_UNF_ADVCA_DCAS      = 0x0c,
    HI_UNF_ADVCA_PANACCESS  = 0x0e,
    HI_UNF_ADVCA_VENDORIDE_BUTT
}HI_UNF_ADVCA_VENDORID_E;
```

[Reason]

This data structure is modified to support the Panaccess advanced CA chip.

[Note]

None

[Example]

None

HI_UNF_ADVCA_OTP_FUSE_E

[Definition]

Before modification:

```
typedef enum hiUNF_ADVCA_OTP_FUSE_E
{
```

```
    HI_UNF_ADVCA_OTP_NULL = 0,
    HI_UNF_ADVCA_OTP_SECURE_BOOT_ACTIVATION,
    HI_UNF_ADVCA_OTP_BOOT_DECRYPTION_ACTIVATION,
    HI_UNF_ADVCA_OTP_SELF_BOOT_DEACTIVATION,
    HI_UNF_ADVCA_OTP_DDR_WAKEUP_DEACTIVATION,
    HI_UNF_ADVCA_OTP_CSA2_KL_LEVEL_SEL,
    HI_UNF_ADVCA_OTP_R2R_KL_LEVEL_SEL,
    HI_UNF_ADVCA_OTP_SP_KL_LEVEL_SEL,
    HI_UNF_ADVCA_OTP_CSA3_KL_LEVEL_SEL,
    HI_UNF_ADVCA_OTP_LP_DEACTIVATION,
    HI_UNF_ADVCA_OTP_CSA2_CW_HARDONLY_ACTIVATION,
    HI_UNF_ADVCA_OTP_SP_CW_HARDONLY_ACTIVATION,
    HI_UNF_ADVCA_OTP_CSA3_CW_HARDONLY_ACTIVATION,
    HI_UNF_ADVCA_OTP_CSA2_KL_DEACTIVATION,
    HI_UNF_ADVCA_OTP_SP_KL_DEACTIVATION,
    HI_UNF_ADVCA_OTP_CSA3_KL_DEACTIVATION,
    HI_UNF_ADVCA_OTP_MISC_KL_DEACTIVATION,
    HI_UNF_ADVCA_OTP_GOOGLE_KL_DEACTIVATION,
    HI_UNF_ADVCA_OTP_DCAS_KL_DEACTIVATION,
    HI_UNF_ADVCA_OTP_DDR_SCRAMBLE_ACTIVATION,
    HI_UNF_ADVCA_OTP_GLOBAL_LOCK_ACTIVATION,
    HI_UNF_ADVCA_OTP_RUNTIME_CHECK_ACTIVATION,
    HI_UNF_ADVCA_OTP_DDR_WAKEUP_CHECK_ACTIVATION,
    HI_UNF_ADVCA_OTP_VERSION_ID_CHECK_ACTIVATION,
    HI_UNF_ADVCA_OTP_BOOT_MSID_CHECK_ACTIVATION,
    HI_UNF_ADVCA_OTP_JTAG_MODE,
    HI_UNF_ADVCA_OTP_JTAG_READ_DEACTIVATION,
    HI_UNF_ADVCA_OTP_R2R_ROOTKEY,
    HI_UNF_ADVCA_OTP_SP_ROOTKEY,
    HI_UNF_ADVCA_OTP_CSA3_ROOTKEY,
    HI_UNF_ADVCA_OTP_MISC_ROOTKEY,
    HI_UNF_ADVCA_OTP_OEM_ROOTKEY,
    HI_UNF_ADVCA_OTP_ESCK_ROOTKEY,
    HI_UNF_ADVCA_OTP_JTAG_KEY,
    HI_UNF_ADVCA_OTP_CHIP_ID,
    HI_UNF_ADVCA_OTP_MARKET_SEGMENT_ID,
    HI_UNF_ADVCA_OTP_VERSION_ID,
    HI_UNF_ADVCA_OTP_MISC_KL_LEVEL_SEL,
    HI_UNF_ADVCA_OTP_VMX_BL_FUSE, /
    HI_UNF_ADVCA_OTP_IRDETO_ITCSA3_ACTIVATION,
    HI_UNF_ADVCA_OTP_BOOTINFO_DEACTIVATION,
    HI_UNF_ADVCA_OTP_ITCSA3_IMLB,
    HI_UNF_ADVCA_OTP_FUSE_BUTT
}HI_UNF_ADVCA_OTP_FUSE_E;
```

After modification:

```
typedef enum hiUNF_ADVCA_OTP_FUSE_E
{
    HI_UNF_ADVCA_OTP_NULL = 0,
    HI_UNF_ADVCA_OTP_SECURE_BOOT_ACTIVATION,
    HI_UNF_ADVCA_OTP_BOOT_DECRYPTION_ACTIVATION,
    HI_UNF_ADVCA_OTP_SELF_BOOT_DEACTIVATION,
    HI_UNF_ADVCA_OTP_DDR_WAKEUP_DEACTIVATION,
    HI_UNF_ADVCA_OTP_CSA2_KL_LEVEL_SEL,
    HI_UNF_ADVCA_OTP_R2R_KL_LEVEL_SEL,
    HI_UNF_ADVCA_OTP_SP_KL_LEVEL_SEL,
    HI_UNF_ADVCA_OTP_CSA3_KL_LEVEL_SEL,
    HI_UNF_ADVCA_OTP_LP_DEACTIVATION,
    HI_UNF_ADVCA_OTP_CSA2_CW_HARDONLY_ACTIVATION,
    HI_UNF_ADVCA_OTP_SP_CW_HARDONLY_ACTIVATION,
    HI_UNF_ADVCA_OTP_CSA3_CW_HARDONLY_ACTIVATION,
    HI_UNF_ADVCA_OTP_CSA2_KL_DEACTIVATION,
    HI_UNF_ADVCA_OTP_SP_KL_DEACTIVATION,
    HI_UNF_ADVCA_OTP_CSA3_KL_DEACTIVATION,
    HI_UNF_ADVCA_OTP_MISC_KL_DEACTIVATION,
    HI_UNF_ADVCA_OTP_GOOGLE_KL_DEACTIVATION,
    HI_UNF_ADVCA_OTP_DCAS_KL_DEACTIVATION,
    HI_UNF_ADVCA_OTP_DDR_SCRAMBLE_ACTIVATION,
    HI_UNF_ADVCA_OTP_GLOBAL_LOCK_ACTIVATION,
    HI_UNF_ADVCA_OTP_RUNTIME_CHECK_ACTIVATION,
    HI_UNF_ADVCA_OTP_DDR_WAKEUP_CHECK_ACTIVATION,
    HI_UNF_ADVCA_OTP_VERSION_ID_CHECK_ACTIVATION,
    HI_UNF_ADVCA_OTP_BOOT_MSID_CHECK_ACTIVATION,
    HI_UNF_ADVCA_OTP_JTAG_MODE,
    HI_UNF_ADVCA_OTP_JTAG_READ_DEACTIVATION,
    HI_UNF_ADVCA_OTP_CSA2_ROOTKEY,
    HI_UNF_ADVCA_OTP_R2R_ROOTKEY,
    HI_UNF_ADVCA_OTP_SP_ROOTKEY,
    HI_UNF_ADVCA_OTP_CSA3_ROOTKEY,
    HI_UNF_ADVCA_OTP_MISC_ROOTKEY,
    HI_UNF_ADVCA_OTP_OEM_ROOTKEY,
    HI_UNF_ADVCA_OTP_ESCK_ROOTKEY,
    HI_UNF_ADVCA_OTP_JTAG_KEY,
    HI_UNF_ADVCA_OTP_CHIP_ID,
    HI_UNF_ADVCA_OTP_MARKET_SEGMENT_ID,
    HI_UNF_ADVCA_OTP_VERSION_ID,
    HI_UNF_ADVCA_OTP_MISC_KL_LEVEL_SEL,
    HI_UNF_ADVCA_OTP_VMX_BL_FUSE,
```

```
    HI_UNF_ADVCA_OTP_IRDETO_ITCSA3_ACTIVATION,
    HI_UNF_ADVCA_OTP_BOOTINFO_DEACTIVATION,
    HI_UNF_ADVCA_OTP_ITCSA3_IMLB,
    HI_UNF_ADVCA_OTP_SECURE_BOOT_ACTIVATION_ONLY,
    HI_UNF_ADVCA_OTP_BOOT_FLASH_TYPE,
    HI_UNF_ADVCA_OTP_RSA_KEY_LOCK_FLAG,
    HI_UNF_ADVCA_OTP_STBSN_LOCK_FLAG,
    HI_UNF_ADVCA_OTP_MSID_LOCK_FLAG,
    HI_UNF_ADVCA_OTP_VERSIONID_LOCK_FLAG,
    HI_UNF_ADVCA_OTP_OEM_ROOTKEY_LOCK_FLAG,
    HI_UNF_ADVCA_OTP_R2R_ROOTKEY_LOCK_FLAG,
    HI_UNF_ADVCA_OTP_JTAG_KEY_LOCK_FLAG,
    HI_UNF_ADVCA_OTP_TZ_AREA_LOCK_FLAG,
    HI_UNF_ADVCA_OTP_FUSE_BUTT
}HI_UNF_ADVCA_OTP_FUSE_E;
```

[Reason]

This data structure is modified for setting and obtaining the following attributes:

- Secure boot enable flag
- Type of flash memory from which the system boots
- Multiple lock flag bits

[Note]

None

[Example]

None

12.2 Common

12.2.1 Overview

The UNF 3.2.10 common module has the following changes compared with UNF 3.2.9:

- The function of obtaining the chip package type is added.
- The sub version numbers of Hi3716M V410 and Hi3716M V420 are supported.

12.2.1.1 Obtaining the Chip Package Type

The data structure and API are changed as follows:

Data Structure

HI_CHIP_PACKAGE_TYPE_E is added.

API

HI_SYS_GetChipPackageType is added.

12.2.1.2 Supporting the Sub Version Numbers of Hi3716M V410 and Hi3716M V420

The data structure is changed as follows:

Data Structure

HI_CHIP_VERSION_E is modified.

12.2.2 Data Structures

12.2.2.1 New Data Structure

HI_CHIP_PACKAGE_TYPE_E

[Definition]

```
typedef enum
{
    HI_CHIP_PACKAGE_TYPE_BGA_15_15 = 0,
    HI_CHIP_PACKAGE_TYPE_BGA_16_16,
    HI_CHIP_PACKAGE_TYPE_BGA_19_19,
    HI_CHIP_PACKAGE_TYPE_BGA_23_23,
    HI_CHIP_PACKAGE_TYPE_BGA_31_31,
    HI_CHIP_PACKAGE_TYPE_QFP_216,
    HI_CHIP_PACKAGE_TYPE_BUTT
} HI_CHIP_PACKAGE_TYPE_E;
```

[Reason]

This data structure is added for the new feature of enumerating chip package types.

[Note]

None

[Example]

None

12.2.2.2 Modified Data Structure

HI_CHIP_VERSION_E

[Definition]

Before modification:

```
typedef enum hiCHIP_VERSION_E
{
```

```
    HI_CHIP_VERSION_V100 = 0x100,
    HI_CHIP_VERSION_V101 = 0x101,
    HI_CHIP_VERSION_V200 = 0x200,
    HI_CHIP_VERSION_V300 = 0x300,
    HI_CHIP_VERSION_V400 = 0x400,
    HI_CHIP_VERSION_BUTT
}HI_CHIP_VERSION_E;
```

After modification:

```
typedef enum hiCHIP_VERSION_E
{
    HI_CHIP_VERSION_V100 = 0x100,
    HI_CHIP_VERSION_V101 = 0x101,
    HI_CHIP_VERSION_V200 = 0x200,
    HI_CHIP_VERSION_V300 = 0x300,
    HI_CHIP_VERSION_V400 = 0x400,
    HI_CHIP_VERSION_V410 = 0x410,
    HI_CHIP_VERSION_V420 = 0x420,
    HI_CHIP_VERSION_BUTT
}HI_CHIP_VERSION_E;
```

[Reason]

This data structure is modified to support the sub version numbers of Hi3716M V410 and Hi3716M V420.

[Note]

None

[Example]

None

12.2.3 API

12.2.3.1 New API

HI_SYS_GetChipPackageType

[Definition]

```
HI_S32 HI_SYS_GetChipPackageType(HI_CHIP_PACKAGE_TYPE_E *penPackageType);
```

[Reason]

This API is added for the new feature of obtaining the chip package type.

[Note]

None

[Example]

None

12.3 VENC

12.3.1 Overview

The UNF 3.2.10 VNEC module has the following change compared with UNF 3.2.9:

The function of setting the slice size is added.

12.3.1.1 Setting the Slice Size

In the slice encoding mode for video encoding, the slice size can be set by using the new attribute **u32SplitSize**, which must be greater than or equal to 512 bytes. The data structure is changed as follows:

Data Structure

HI_UNF_VENC_CHN_ATTR_S is modified.

12.3.2 Data Structure

12.3.2.1 Modified Data Structure

HI_UNF_VENC_CHN_ATTR_S

[Definition]

Before modification:

```
typedef struct hiUNF_VENC_CHN_ATTR_S
{
    HI_UNF_VCODEC_TYPE_E       enVencType;
    HI_UNF_VCODEC_CAP_LEVEL_E   enCapLevel;
    HI_UNF_H264_PROFILE_E      enVencProfile;
    HI_U32                  u32Width;
    HI_U32                  u32Height;
    HI_U32                  u32StrmBufSize;
    HI_U32                  u32RotationAngle;
    HI_BOOL                 bSlcSplitEn;
    HI_U32                  u32TargetBitRate;
    HI_U32                  u32TargetFrmRate;
    HI_U32                  u32InputFrmRate;
    HI_U32                  u32Gop;
    HI_U32                  u32MaxQp;
    HI_U32                  u32MinQp;
    HI_BOOL                 bQuickEncode;
    HI_U8                   u8Priority;
    HI_U32                  u32Qlevel;
```

```
    HI_U32                  u32DriftRateThr;
}HI_UNF_VENC_CHN_ATTR_S;
```

After modification:

```
typedef struct hiUNF_VENC_CHN_ATTR_S
{
    HI_UNF_VCODEC_TYPE_E       enVencType;
    HI_UNF_VCODEC_CAP_LEVEL_E   enCapLevel;
    HI_UNF_H264_PROFILE_E      enVencProfile;
    HI_U32                  u32Width;
    HI_U32                  u32Height;
    HI_U32                  u32StrmBufSize;
    HI_U32                  u32RotationAngle;
    HI_BOOL                 bSlcSplitEn;
    HI_U32                  u32SplitSize;
    HI_U32                  u32TargetBitRate;
    HI_U32                  u32TargetFrmRate;
    HI_U32                  u32InputFrmRate;
    HI_U32                  u32Gop;
    HI_U32                  u32MaxQp;
    HI_U32                  u32MinQp;
    HI_BOOL                 bQuickEncode;
    HI_U8                   u8Priority;
    HI_U32                  u32Qlevel;
    HI_U32                  u32DriftRateThr;
}HI_UNF_VENC_CHN_ATTR_S;
```

[Reason]

This data structure is modified to support the new feature of setting the slice size as required.

[Note]

- According to the H.264 protocol, certain margin must be reserved for the slice size. Because of the VEDU limitation, the reserved margin must allow the configured slice size plus 2304 bytes in the worst scenario. Typically, the actual output slice size during tests is the configured slice size plus 100 bytes or less.
- The configuration is invalid when the slice encoding mode is disabled.

[Example]

None

12.4 PVR

12.4.1 Overview

The UNF 3.2.10 PVR module has the following changes compared with UNF 3.2.9:

- The function of adding a recording PID is added.
- The function of deleting a recording PID is added.
- The function of deleting all recording PIDs is added.

12.4.1.1 Adding a Recording PID

A PID channel is created and bound to the recording channel. The API is changed as follows:

API

HI_UNF_PVR_RecAddPID is added.

12.4.1.2 Deleting a Recording PID

A PID channel is destroyed and unbound from the recording channel. The API is changed as follows:

API

HI_UNF_PVR_RecDelPID is added.

12.4.1.3 Deleting All Recording PIDs

All PID channels bound to the recording channels are destroyed. The API is changed as follows:

API

sHI_UNF_PVR_RecDelAllPID is added.

12.4.2 APIs

12.4.2.1 New APIs

HI_UNF_PVR_RecAddPID

[Definition]

```
HI_S32 HI_UNF_PVR_RecAddPID(HI_U32 u32ChnID, HI_U32 u32Pid);
```

[Reason]

This API is added to support the Demux to record multiple programs. If an extra program needs to be recorded based on the existing recording channels, this API can be used to add the PID of the new program to the existing recording channels.

[Note]

The addition of a PID at the frequency different from that of the recording channel is not supported.

[Example]

See **sample/common/hi_adp_pvr.c**.

HI_UNF_PVR_RecDelPID

[Definition]

```
HI_S32 HI_UNF_PVR_RecDelPID(HI_U32 u32ChnID, HI_U32 u32Pid);
```

[Reason]

This API is used to unbind the PID channel from the recording channel and end the recording of this program.

[Note]

None

[Example]

See **sample/common/hi_adp_pvr.c**.

sHI_UNF_PVR_RecDelAllPID

[Definition]

```
HI_S32 HI_UNF_PVR_RecDelAllPID(HI_U32 u32ChnID);
```

[Reason]

This API is used to delete all the PID channels bound to the recording channel and end the recording of all programs on this recording channel.

[Note]

None

[Example]

See **sample/common/hi_adp_pvr.c**.

12.5 PQ

12.5.1 Overview

The UNF 3.2.10 PQ module has the following changes compared with UNF 3.2.9:

- The denoising function is added.
- New demo display mode is added.
- The function of setting the brightness, hue, contrast, and saturation separately for video and picture is added.
- The function of setting the TNR, DEI, and DBM demo modes is added.

12.5.1.1 Denoising

The PQ module integrates the TNR algorithm. The data structures and APIs are changed as follows:

Data Structures

- HI_UNF_PQ_MODULE_E is modified.
- HI_UNF_PQ_DEMO_E is modified.

APIs

- HI_UNF_PQ_GetNR is added.
  HI_UNF_PQ_SetNR is added.

12.5.1.2 Adding New Demo Display Mode

This function is added to present the demo display modes in rolled/fixed way, with strengthened left/right algorithm. The data structure and APIs are changed as follows:

Data Structure

HI_UNF_PQ_DEMO_MODE_E is added.

APIs

- HI_UNF_PQ_SetDemoMode is added.
- HI_UNF_PQ_GetDemoMode is added.

12.5.1.3 Setting the Brightness, Hue, Contrast, and Saturation Separately for Video and Picture

This function is added to set the image effect for the video layer without affecting the effect of the graphical user interface. The data structures and APIs are changed as follows:

Data Structures

- HI_UNF_PQ_IMAGE_TYPE_E is added.
- HI_UNF_PQ_IMAGE_PARAM_S_S is added.

APIs

- HI_UNF_PQ_SetBasicImageParam is added.
- HI_UNF_PQ_GetBasicImageParam is added.

12.5.1.4 Setting TNR, DEI, and DBM Demo Modes

This function is added to compare the display effects of TNR, DEI, and DBM demo modes. The data structure is changed as follows:

Data Structure

HI_UNF_PQ_DEMO_E is modified.

12.5.2 Data Structures

12.5.2.1 New Data Structures

HI_UNF_PQ_DEMO_MODE_E

[Definition]

```
typedef enum hiUNF_PQ_DEMO_MODE_E
{
    HI_UNF_PQ_DEMO_MODE_FIXED_R,
    HI_UNF_PQ_DEMO_MODE_FIXED_L,
    HI_UNF_PQ_DEMO_MODE_SCROLL_R,
    HI_UNF_PQ_DEMO_MODE_SCROLL_L,

    HI_UNF_PQ_DEMO_MODE_BUTT
}  HI_UNF_PQ_DEMO_MODE_E;
```

[Reason]

This data structures is added to display the demo in different modes.

[Note]

None

[Example]

See **sample/pq/sample_pq_v2.c**.

HI_UNF_PQ_IMAGE_TYPE_E

[Definition]

```
typedef enum hiUNF_PQ_IMAGE_TYPE_E
{
    HI_UNF_PQ_IMAGE_GRAPH = 0,
    HI_UNF_PQ_IMAGE_VIDEO,
    HI_UNF_PQ_IMAGE_BUTT
} HI_UNF_PQ_IMAGE_TYPE_E;
```

[Reason]

This data structures is added to enable a graphics layer or video layer for image parameter setting.

[Note]

None

[Example]

See **sample/pq/sample_pq_v2.c**.

HI_UNF_PQ_IMAGE_PARAM_S

[Definition]

```
typedef struct hiUNF_PQ_IMAGE_PARAM_S
{
```

```
    HI_U32      u32Brightness;
    HI_U32      u32Contrast;
    HI_U32      u32Hue;
    HI_U32      u32Saturation;
} HI_UNF_PQ_IMAGE_PARAM_S;
```

[Reason]

This data structure is added to set the image brightness, hue, contrast, and saturation.

[Note]

The values for the image brightness, hue, contrast, and saturation range from 0 to100.

[Example]

See **sample/pq/sample_pq_v2.c**.

12.5.2.2 Modified Data Structures

HI_UNF_PQ_DEMO_E

[Definition]

Before modification:

```
typedef enum hiUNF_PQ_DEMO_E
{
    HI_UNF_PQ_DEMO_SHARPNESS = 0,
    HI_UNF_PQ_DEMO_DCI,
    HI_UNF_PQ_DEMO_COLOR,
    HI_UNF_PQ_DEMO_SR,
    HI_UNF_PQ_DEMO_ALL,

    HI_UNF_PQ_DEMO_BUTT
}  HI_UNF_PQ_DEMO_E;
```

After modification:

```
typedef enum hiUNF_PQ_DEMO_E
{
    HI_UNF_PQ_DEMO_SHARPNESS = 0,
    HI_UNF_PQ_DEMO_DCI,
    HI_UNF_PQ_DEMO_COLOR,
    HI_UNF_PQ_DEMO_SR,
    HI_UNF_PQ_DEMO_TNR,
```

```
    HI_UNF_PQ_DEMO_DEI,
    HI_UNF_PQ_DEMO_DBM,
    HI_UNF_PQ_DEMO_ALL,

    HI_UNF_PQ_DEMO_BUTT
} HI_UNF_PQ_DEMO_E;
```

[Reason]

This function is added for setting TNR, DEI, and DBM demo modes and viewing the effect of enabling/disabling the TNR, DEI, and DBM algorithm.

[Note]

None

[Example]

See **sample/pq/sample_pq_v2.c**.

HI_UNF_PQ_MODULE_E

[Definition]

Before modification:

```
typedef enum hiUNF_PQ_MODULE_E
{
    HI_UNF_PQ_MODULE_SHARPNESS = 0,
    HI_UNF_PQ_MODULE_DCI,
    HI_UNF_PQ_MODULE_COLOR,
    HI_UNF_PQ_MODULE_SR,
   HI_UNF_PQ_MODULE_ALL,

    HI_UNF_PQ_MODULE_BUTT
} HI_UNF_PQ_MODULE_E;
```

After modification:

```
typedef enum hiUNF_PQ_MODULE_E
{
    HI_UNF_PQ_MODULE_SHARPNESS = 0,
    HI_UNF_PQ_MODULE_DCI,
    HI_UNF_PQ_MODULE_COLOR,
    HI_UNF_PQ_MODULE_SR,
```

```
    HI_UNF_PQ_MODULE_TNR,
    HI_UNF_PQ_MODULE_ALL,

    HI_UNF_PQ_MODULE_BUTT
} HI_UNF_PQ_MODULE_E;
```

[Reason]

This data structure is modified for enabling/disabling the TNR algorithm.

[Note]

None

[Example]

See **sample/pq/sample_pq_v2.c**.

12.5.3 APIs

12.5.3.1 New APIs

HI_UNF_PQ_GetNR

[Definition]

```
HI_S32 HI_UNF_PQ_GetNR(HI_UNF_DISP_E enChan, HI_U32* pu32NRLevel);
```

[Reason]

This API is added to obtain the NR strength.

[Note]

None

[Example]

See **sample/pq/sample_pq_v2.c**.

HI_UNF_PQ_SetNR

[Definition]

```
HI_S32 HI_UNF_PQ_SetNR(HI_UNF_DISP_E enChan, HI_U32 u32NRLevel);
```

[Reason]

This API is added to stet the NR strength.

[Note]

The NR strength value ranges from 0 to 100.

[Example]

See **sample/pq/sample_pq_v2.c**.

HI_UNF_PQ_SetDemoMode

[Definition]

```
extern HI_S32 HI_UNF_PQ_SetDemoMode( HI_UNF_DISP_E enChan,
HI_UNF_PQ_DEMO_MODE_E enMode);
```

[Reason]

This API is added to set the demo display mode.

[Note]

None

[Example]

See **sample/pq/sample_pq_v2.c**.

HI_UNF_PQ_GetDemoMode

[Definition]

```
extern HI_S32 HI_UNF_PQ_GetDemoMode( HI_UNF_DISP_E enChan,
HI_UNF_PQ_DEMO_MODE_E* penMode);
```

[Reason]

This API is added to obtain the demo display mode.

[Note]

None

[Example]

See sample/pq/sample_pq_v2.c.

HI_UNF_PQ_SetBasicImageParam

[Definition]

```
xtern HI_S32 HI_UNF_PQ_SetBasicImageParam(HI_UNF_PQ_IMAGE_TYPE_E enType,
HI_UNF_DISP_E enChan, HI_UNF_PQ_IMAGE_PARAM_S stParam);
```

[Reason]

This API is added to set the brightness, hue, contrast, and saturation for images or videos separately.

[Note]

The values for the brightness, hue, contrast, and saturation range from 0 to100.

[Example]

See **sample/pq/sample_pq_v2.c**.

HI_UNF_PQ_GetBasicImageParam

[Definition]

```
extern HI_S32 HI_UNF_PQ_GetBasicImageParam(HI_UNF_PQ_IMAGE_TYPE_E enType,
HI_UNF_DISP_E enChan, HI_UNF_PQ_IMAGE_PARAM_S* pstParam);
```

[Reason]

This API is added to obtain the brightness, hue, contrast, and saturation for images or videos.

[Note]

None

[Example]

See **sample/pq/sample_pq_v2.c**.

12.6 Subtitle

12.6.1 Overview

The UNF 3.2.10 subtitle module has the following changes compared with UNF 3.2.9:

The function of setting the maximum synchronization deviation time is added.

12.6.1.1 Setting the Maximum Synchronization Deviation Time

An API is added for setting the maximum synchronization deviation time to process the great difference between the displayed subtitle PTS and the video PTS. The unit is ms. The change is as follows:

API

HI_UNF_SO_SetMaxInterval is added.

12.6.2 API

12.6.2.1 New API

HI_UNF_SO_SetMaxInterval

[Definition]

```
HI_S32 HI_UNF_SO_SetMaxInterval(HI_HANDLE handle, HI_U32 u32IntervalMs )
```

[Reason]

For some special streams during tests, the parsed subtitle PTS differs significantly from the video PTS, and therefore the subtitles in this format cannot be output properly. This API is added to set the maximum PTS difference. If the PTS difference of received data exceeds the threshold, the data is output immediately.

[Note]

None

[Example]

None

12.7 Teletext

12.7.1 Overview

The UNF 3.2.10 Teletext module has the following changes compared with UNF 3.2.9:

The function of setting the maximum synchronization deviation time is added.

12.7.1.1 Setting the Maximum Synchronization Deviation Time

An API is added for setting the maximum synchronization deviation time to process the great difference between the displayed Teletext PTS and video PTS. The unit is ms. The change is as follows:

Data Structure

None

API

HI_UNF_TTX_SetMaxInterval is added.

12.7.2 API

12.7.2.1 New API

HI_UNF_TTX_SetMaxInterval

[Definition]

```
HI_S32 HI_UNF_TTX_SetMaxInterval(HI_HANDLE hTTX, HI_U32 u32IntervalMs )
```

[Reason]

There is a default 10s deviation value in the Teletext module, that is, if the difference between the Teletext PTS to be displayed and the video PTS exceeds 10s, data is output immediately. This processing is relatively rigid because data may be output immediately even if the PTS can be synchronized, which is not as expected. Therefore, this API is added for the upper layer for setting the synchronization deviation flexibly.

[Note]

None

[Example]

None

12.8 Frontend

12.8.1 Overview

The UNF 3.2.10 Frontend module has the following change compared with UNF 3.2.9:

- The MXL251, HIFDVBC100, and HIFJ83B100 components are supported.
- The function of obtaining full-band spectrum data is added.

12.8.1.1 Supporting MXL251, HIFDVBC100, and HIFJ83B100 Components

The data structures are changed as follows:

Data Structures

- 错误！未找到引用源。 is modified.
- 错误！未找到引用源。 is modified.

12.8.1.2 Obtaining Full-Band Spectrum Data

The change is as follows:

API

HI_UNF_TUNER_GetTunerPowerSpectrumData is added.

12.8.2 Data Structures

12.8.2.1 Modified Data Structures

HI_UNF_TUNER_DEV_TYPE_E

[Definition]

Before modification:

```
typedef enum   hiUNF_TUNER_DEV_TYPE_E
{
    HI_UNF_TUNER_DEV_TYPE_XG_3BL,
    HI_UNF_TUNER_DEV_TYPE_CD1616,
    HI_UNF_TUNER_DEV_TYPE_ALPS_TDAE,
    HI_UNF_TUNER_DEV_TYPE_TDCC,
    HI_UNF_TUNER_DEV_TYPE_TDA18250,
    HI_UNF_TUNER_DEV_TYPE_CD1616_DOUBLE,
    HI_UNF_TUNER_DEV_TYPE_MT2081,
    HI_UNF_TUNER_DEV_TYPE_TMX7070X,
    HI_UNF_TUNER_DEV_TYPE_R820C,
    HI_UNF_TUNER_DEV_TYPE_MXL203,
    HI_UNF_TUNER_DEV_TYPE_AV2011,
    HI_UNF_TUNER_DEV_TYPE_SHARP7903,
    HI_UNF_TUNER_DEV_TYPE_MXL101,
```

```
    HI_UNF_TUNER_DEV_TYPE_MXL603,
    HI_UNF_TUNER_DEV_TYPE_IT9170,
    HI_UNF_TUNER_DEV_TYPE_IT9133,
    HI_UNF_TUNER_DEV_TYPE_TDA6651,
    HI_UNF_TUNER_DEV_TYPE_TDA18250B,
    HI_UNF_TUNER_DEV_TYPE_M88TS2022,
    HI_UNF_TUNER_DEV_TYPE_RDA5815,
    HI_UNF_TUNER_DEV_TYPE_MXL254,
    HI_UNF_TUNER_DEV_TYPE_CXD2861,
    HI_UNF_TUNER_DEV_TYPE_SI2147,
    HI_UNF_TUNER_DEV_TYPE_RAFAEL836,
    HI_UNF_TUNER_DEV_TYPE_MXL608,
    HI_UNF_TUNER_DEV_TYPE_MXL214,
    HI_UNF_TUNER_DEV_TYPE_TDA18280,
    HI_UNF_TUNER_DEV_TYPE_TDA182I5A,
    HI_UNF_TUNER_DEV_TYPE_SI2144,
    HI_UNF_TUNER_DEV_TYPE_AV2018,
    HI_UNF_TUNER_DEV_TYPE_BUTT
} HI_UNF_TUNER_DEV_TYPE_E ;
```

After modification:

```
typedef enum   hiUNF_TUNER_DEV_TYPE_E
{
    HI_UNF_TUNER_DEV_TYPE_XG_3BL,
    HI_UNF_TUNER_DEV_TYPE_CD1616,
    HI_UNF_TUNER_DEV_TYPE_ALPS_TDAE,
    HI_UNF_TUNER_DEV_TYPE_TDCC,
    HI_UNF_TUNER_DEV_TYPE_TDA18250,
    HI_UNF_TUNER_DEV_TYPE_CD1616_DOUBLE,
    HI_UNF_TUNER_DEV_TYPE_MT2081,
    HI_UNF_TUNER_DEV_TYPE_TMX7070X,
    HI_UNF_TUNER_DEV_TYPE_R820C,
    HI_UNF_TUNER_DEV_TYPE_MXL203,
    HI_UNF_TUNER_DEV_TYPE_AV2011,
```

```
    HI_UNF_TUNER_DEV_TYPE_SHARP7903,
    HI_UNF_TUNER_DEV_TYPE_MXL101,
    HI_UNF_TUNER_DEV_TYPE_MXL603,
    HI_UNF_TUNER_DEV_TYPE_IT9170,
    HI_UNF_TUNER_DEV_TYPE_IT9133,
    HI_UNF_TUNER_DEV_TYPE_TDA6651,
    HI_UNF_TUNER_DEV_TYPE_TDA18250B,
    HI_UNF_TUNER_DEV_TYPE_M88TS2022,
    HI_UNF_TUNER_DEV_TYPE_RDA5815,
    HI_UNF_TUNER_DEV_TYPE_MXL254,
    HI_UNF_TUNER_DEV_TYPE_CXD2861,
    HI_UNF_TUNER_DEV_TYPE_SI2147,
    HI_UNF_TUNER_DEV_TYPE_RAFAEL836,
    HI_UNF_TUNER_DEV_TYPE_MXL608,
    HI_UNF_TUNER_DEV_TYPE_MXL214,
    HI_UNF_TUNER_DEV_TYPE_TDA18280,
    HI_UNF_TUNER_DEV_TYPE_TDA182I5A,
    HI_UNF_TUNER_DEV_TYPE_SI2144,
    HI_UNF_TUNER_DEV_TYPE_AV2018,
    HI_UNF_TUNER_DEV_TYPE_MXL251,
    HI_UNF_TUNER_DEV_TYPE_BUTT
} HI_UNF_TUNER_DEV_TYPE_E ;
```

[Reason]

This data structure is modified for supporting the MXL251.

[Note]

The MXL251 is a full-band tuner component with the Demod functions.

[Example]

None

HI_UNF_DEMOD_DEV_TYPE_E

[Definition]

Before modification:

```
typedef enum   hiUNF_DEMOD_DEV_TYPE_E
```

```
{
    HI_UNF_DEMOD_DEV_TYPE_NONE,
    HI_UNF_DEMOD_DEV_TYPE_3130I = 0x100,
    HI_UNF_DEMOD_DEV_TYPE_3130E,
    HI_UNF_DEMOD_DEV_TYPE_J83B,
    HI_UNF_DEMOD_DEV_TYPE_AVL6211,
    HI_UNF_DEMOD_DEV_TYPE_MXL101,
    HI_UNF_DEMOD_DEV_TYPE_MN88472,
    HI_UNF_DEMOD_DEV_TYPE_IT9170,
    HI_UNF_DEMOD_DEV_TYPE_IT9133,
    HI_UNF_DEMOD_DEV_TYPE_3136,
    HI_UNF_DEMOD_DEV_TYPE_3136I,
    HI_UNF_DEMOD_DEV_TYPE_MXL254,
    HI_UNF_DEMOD_DEV_TYPE_CXD2837,
    HI_UNF_DEMOD_DEV_TYPE_3137,
    HI_UNF_DEMOD_DEV_TYPE_MXL214,
    HI_UNF_DEMOD_DEV_TYPE_TDA18280,
    HI_UNF_DEMOD_DEV_TYPE_HIFDVBC100,
    HI_UNF_DEMOD_DEV_TYPE_HIFJ83B100,
    HI_UNF_DEMOD_DEV_TYPE_BUTT
} HI_UNF_DEMOD_DEV_TYPE_E;
```

After modification:

```
typedef enum   hiUNF_DEMOD_DEV_TYPE_E
{
    HI_UNF_DEMOD_DEV_TYPE_NONE,
    HI_UNF_DEMOD_DEV_TYPE_3130I = 0x100,
    HI_UNF_DEMOD_DEV_TYPE_3130E,
    HI_UNF_DEMOD_DEV_TYPE_J83B,
    HI_UNF_DEMOD_DEV_TYPE_AVL6211,
    HI_UNF_DEMOD_DEV_TYPE_MXL101,
    HI_UNF_DEMOD_DEV_TYPE_MN88472,
    HI_UNF_DEMOD_DEV_TYPE_IT9170,
    HI_UNF_DEMOD_DEV_TYPE_IT9133,
```

```
    HI_UNF_DEMOD_DEV_TYPE_3136,
    HI_UNF_DEMOD_DEV_TYPE_3136I,
    HI_UNF_DEMOD_DEV_TYPE_MXL254,
    HI_UNF_DEMOD_DEV_TYPE_CXD2837,
    HI_UNF_DEMOD_DEV_TYPE_3137,
    HI_UNF_DEMOD_DEV_TYPE_MXL214,
    HI_UNF_DEMOD_DEV_TYPE_TDA18280,
    HI_UNF_DEMOD_DEV_TYPE_HIFDVBC100,
    HI_UNF_DEMOD_DEV_TYPE_HIFJ83B100,
    HI_UNF_DEMOD_DEV_TYPE_MXL251,
    HI_UNF_DEMOD_DEV_TYPE_BUTT
} HI_UNF_DEMOD_DEV_TYPE_E;
```

[Reason]

This data structure is modified for supporting three more Demod components.

[Note]

The MXL251 is a full-band tuner component with the Demod functions. HIFDVBC100 and HIFJ83B100 are the internal Demod of a chip.

[Example]

None

12.8.3 API

12.8.3.1 New API

HI_UNF_TUNER_GetTunerPowerSpectrumData

[Definition]

```
HI_S32 HI_UNF_TUNER_GetTunerPowerSpectrumData(HI_U32 u32TunerId, HI_U32
u32freqStartInHz,HI_U32 u32freqStepInHz,HI_U32 u32numOfFreqSteps,HI_S16
*ps16powerData);
```

[Reason]

This API is added for obtaining full-band spectrum data.

[Note]

None

[Example]

None