TCCxxxx Common

API Specification for HD Radio

Rev. 1.50 [A]
2022-04-07

※ The information in this document is subject to change without notice and should not be construed as a commitment by Telechips Inc.

Kindly visit www.telechips.com for more information.

TABLE OF CONTENTS

Contents

Figures

Tables

1 INTRODUCTION

This document describes Application Programming Interface (API) specification for HD Radio solution provided by Telechips. The radio tuner chip uses Silab tuner chip as the default setting.

Table 1.1 Available Chipset

Chipset
TCC803x
TCC803xPE
TCC805x

1.1 Features

- HD Radio Specification
  - HD 1.0
  - HD 1.0 + MRC
  - HD 1.5
  - HD 1.5 + MRC

- Functions
  - Multicasting (Main Program Service (MPS)/Supplemental Program Service (SPS))
  - Program Service Data (PSD)
  - Service Information Guide (SIG)
  - Service Information Service (SIS)
  - Fixed Blending
  - Automatic Audio Alignment (AAA)
  - Large Object Transfer (LOT)
  - Advanced Application Services (AAS)
  - Emergency Alert
  - Maximum Ratio Combining (MRC)
  - Data Services for Background Scan

1.2 Block Diagram

HD Radio solution is provided as header and shared object files.

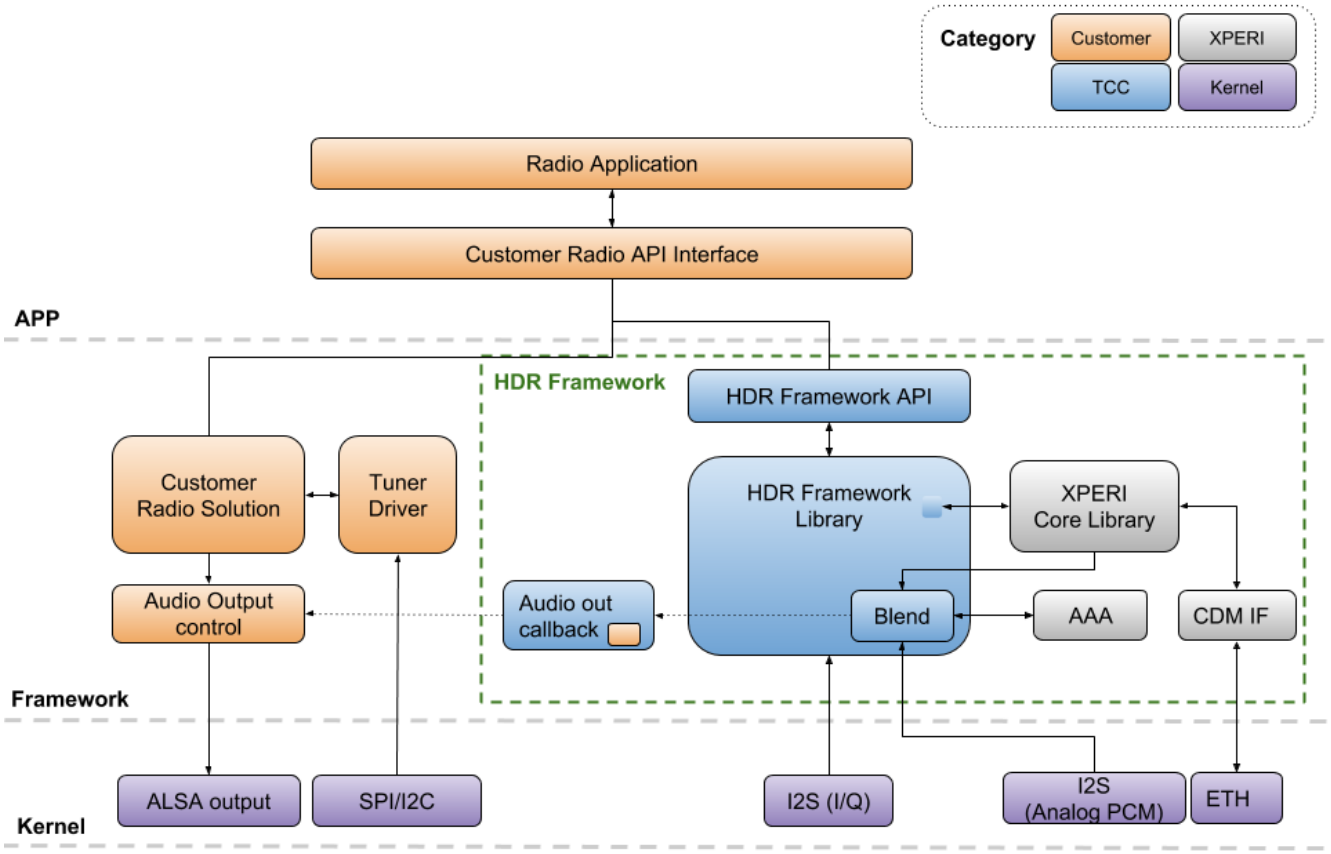

Figure 1.1 Block Diagram of HD Radio Framework

1.3 Package List

HD Radio framework and library are structured as follows:

Table 1.2 List of HD Radio Libraries

Layer	Library	Distributor
HD Radio Framework	libtchdradio.so.1.5.x (V1.5.0 or higher)	Telechips
HD Radio Library	libHDRadio.so.2.6.x (V2.6.10 or higher)	XPERI

2 DEFINITION

2.1 Variable Type

Description

These are the definitions of variable types used in HD Radio framework.

Definition

```
#ifndef U8
typedef unsigned char           U8;
#endif

#ifndef S8
typedef char                    S8;
#endif

#ifndef S8
typedef signed char             S8C;
#endif

#ifndef U16
typedef unsigned short          U16;
#endif

#ifndef S16
typedef short                   S16;
#endif

#ifndef U32
typedef unsigned int            U32;
#endif

#ifndef S32
typedef int                     S32;
#endif

#ifndef U64
typedef unsigned long long      U64;
#endif

#ifndef S64
typedef long long               S64;
#endif

#ifndef F32
typedef float                   F32;
#endif

#ifndef F64
typedef double                  F64;
#endif

#ifndef HDRET
typedef int                     HDRET;
#endif

#ifndef HDBOOL
typedef bool                    HDBOOL;
#endif

#ifndef UP
#define UP                      (1)
#endif
```

```
#ifndef DN
#define DN                      (0)
#endif

#ifndef YES
#define YES                     (1)
#endif

#ifndef NO
#define NO                      (0)
#endif

#ifndef ON
#define ON                      (1)
#endif

#ifndef OFF
#define OFF                     (0)
#endif

#ifndef TRUE
#define TRUE                    (1)
#endif

#ifndef FALSE
#define FALSE                   (0)
#endif

#ifndef pNULL
#define pNULL                   (void*)(0)
#endif
```

Header file

tchdr_types.h

3 ENUMERATION

Table 3.1 describes enumerations used in the HD Radio framework.

Table 3.1 Enumeration (Enum)

Enum Type	Description
eTC_HDR_RET_t	Return value
eTC_HDR_TYPE_t	HD Radio configuration type
eTC_HDR_ID_t	HD Radio ID
eTC_HDR_BAND_t	HD Radio band
eTC_HDR_BBSRC_RATE_t	HD Radio Baseband (BB) input sample rate
eTC_HDR_SIGNAL_STATUS_t	Signal status bitmap
eTC_HDR_AUDIO_MODE_t	Audio output mode
eTC_HDR_PROGRAM_t	HD Radio program number
eTC_HDR_NOTIFY_t	Notification for callback function
eTC_HDR_BLEND_THRESH_SEL_t	Blend threshold configuration
eTC_HDR_BLEND_PARAMS_t	Blend configuration parameters
eTC_HDR_BLEND_ADV_PARAMS_t	Advanced blend configuration parameters
eTC_HDR_THREAD_t	Threads of HD Radio framework
eTC_HDR_PSD_LENGTH_CONFIG_t	List of PSD field/subfield length configuration parameters
eTC_HDR_PSD_CHAR_TYPE_t	Text encoding type for PSD message strings
eTC_HDR_PSD_FIELD_t	List of PSD fields
eTC_HDR_PSD_COMM_SUBFIELD_t	List of comment subfields
eTC_HDR_PSD_UFID_SUBFIELD_t	List of Unique File Identifier (UFID) subfields
eTC_HDR_PSD_COMR_SUBFIELD_t	List of commercial subfields
eTC_HDR_PSD_BITMASK_t	PSD bitmask
eTC_HDR_SIG_STATUS_t	Defines the updated status of service information guide
eTC_HDR_SIG_COMPONENT_TYPE_t	Defines possible service component types
eTC_HDR_SIG_SERVICE_TYPE_t	Defines possible SIG service types
eTC_HDR_SIS_STATUS_t	Defines the updated status of station information service
eTC_HDR_SIS_ALFN_STATUS_t	Defines the updated status of ALFN
eTC_HDR_SIS_TEXT_ENCODING_t	Text encoding type for SIS message strings
eTC_HDR_SIS_DST_SCHEDULE_t	Indicates the region of Daylight Savings Time (DST) schedule
eTC_HDR_SIS_DST_LOCAL_t	Indicates whether Daylight Savings Time (DST) is practiced locally
eTC_HDR_SIS_ACCESS_TYPE_t	SIS access types
eTC_HDR_SIS_BITMASK_t	SIS bitmask
eTC_HDR_AAS_PORT_MODE_t	HD Radio Port ordered type
eTC_HDR_ALERT_TEXT_ENCODING_t	Text encoding type for alert message strings

3.1 eTC_HDR_RET_t

Description

This enum is a return value of function execution result.

Definition

```
typedef enum {
        eTC_HDR_RET_OK   = 0,

        // Common Error
        eTC_HDR_RET_NG_NOT_ENABLED              = -10,
        eTC_HDR_RET_NG_LOCK_MEMORY              = -20,
        eTC_HDR_RET_NG_INIT                     = -21,
        eTC_HDR_RET_NG_BYTESTREAM_OPEN          = -22,
        eTC_HDR_RET_NG_BB_SRC_INIT              = -23,
        eTC_HDR_RET_NG_INSTANCE_INIT            = -24,
        eTC_HDR_RET_NG_CORE_INIT                = -25,
        eTC_HDR_RET_NG_AUD_RESAMPLER_INIT       = -26,
        eTC_HDR_RET_NG_AUD_RESAMPLER_OUTPUT     = -27,
        eTC_HDR_RET_NG_AUD_RESAMPLER_HANDLER    = -28,
        eTC_HDR_RET_NG_SET_BAND                 = -29,
        eTC_HDR_RET_NG_INVALID_PARAMETERS       = -30,
        eTC_HDR_RET_NG_BLEND_CROSSFADE_RESET    = -31,
        eTC_HDR_RET_NG_NOT_YET_INIT             = -32,
        eTC_HDR_RET_NG_NOT_YET_OPEN             = -33,
        eTC_HDR_RET_NG_ALREADY_INIT             = -34,
        eTC_HDR_RET_NG_ALREADY_OPEN             = -35,
        eTC_HDR_RET_NG_INVALID_HDR_TYPE         = -36,
        eTC_HDR_RET_NG_INVALID_IQ_BUFFER        = -37,
        eTC_HDR_RET_NG_EXT_IQ_DRV_OPEN          = -38,
        eTC_HDR_RET_NG_MALLOC                   = -39,
        eTC_HDR_RET_NG_INVALID_BUFFER_POINTER   = -40,
        eTC_HDR_RET_NG_READ_SIZE                = -41,
        eTC_HDR_RET_NG_RSC                      = -42,
        eTC_HDR_RET_NG_MUTEX_INIT               = -43,
        eTC_HDR_RET_NG_MUTEX_DEINIT             = -44,
        eTC_HDR_RET_NG_NULL_POINTER_PARAMETERS  = -45,
        eTC_HDR_RET_NG_NULL_POINTER_MESSAGE     = -46,
        eTC_HDR_RET_NG_INIT_MESSAGE_QUEUE       = -47,
        eTC_HDR_RET_NG_SET_PROGRAM              = -48,
        eTC_HDR_RET_NG_ENABLE_PSD               = -49,
        eTC_HDR_RET_NG_NOT_SUPPORT              = -50,
        eTC_HDR_RET_NG_INVALID_HDR_ID           = -51,
        eTC_HDR_RET_NG_NULL_POINTER_MUTEX       = -53,
        eTC_HDR_RET_NG_GET_PROGRAM              = -54,
        eTC_HDR_RET_NG_GET_DATA                 = -55,
        eTC_HDR_RET_NG_LIB_ERROR                = -56,
        eTC_HDR_RET_NG_NULL_INSTANCE            = -57,
        eTC_HDR_RET_NG_SET_VALUE                = -58,
        eTC_HDR_RET_NG_NOT_YET_CLOSE            = -59,
        eTC_HDR_RET_NG_IQ01IN_XRUN              = -60,
        eTC_HDR_RET_NG_IQ23IN_XRUN              = -61,
        eTC_HDR_RET_NG_IQ_INPUT_DRIVER          = -62,
        eTC_HDR_RET_NG_INVALID_BAND             = -63,
        eTC_HDR_RET_NG_MUTEX_LOCK               = -64,
        eTC_HDR_RET_NG_MUTEX_UNLOCK             = -65,
        eTC_HDR_RET_NG_EVENT_ERROR              = -66,
        eTC_HDR_RET_NG_DEINIT                   = -67,
        eTC_HDR_RET_NG_ALREADY_CLOSE            = -68,

        // DEMOD Error
        eTC_HDR_RET_NG_DEMOD_INVALID_PARAMETERS         = -200,
        eTC_HDR_RET_NG_DEMOD_BUSY                       = -201,
        eTC_HDR_RET_NG_DEMOD_IDLE                       = -202,
        eTC_HDR_RET_NG_DEMOD_NOT_INIT                   = -203,
```

```
        eTC_HDR_RET_NG_DEMOD_INSTANCE_TYPE              = -204,

        // AAS Error
        eTC_HDR_RET_NG_AAS_NOT_FOUND_PORT_OR_SERVICE    = -210,
        eTC_HDR_RET_NG_AAS_RESERVED_PORT_REQ            = -211,
        eTC_HDR_RET_NG_AAS_MAX_AAS_PORTS_ALREADY_ENABLED= -212,
        eTC_HDR_RET_NG_AAS_MAX_LOT_PORTS_ALREADY_ENABLED= -213,
        eTC_HDR_RET_NG_AAS_ALEADY_OPEN_PORT             = -214,
        eTC_HDR_RET_NG_AAS_NOT_FOUND_OBJECT             = -215,
        eTC_HDR_RET_NG_AAS_NO_DATA_AVAILABLE            = -216,
        eTC_HDR_RET_NG_AAS_CORRUPTED_PACKET             = -217,
        eTC_HDR_RET_NG_AAS_NO_COMPLETE_OBJECT           = -218,

        // PSD Error
        eTC_HDR_RET_NG_PSD_INVALID_LENGTH               = -220,

        // SIG Error
        eTC_HDR_RET_NG_SIG_NO_SERVICE                   = -230,
        eTC_HDR_RET_NG_SIG_NO_COMPONENT                 = -231,

        // ALERT Error
        eTC_HDR_RET_NG_ALERT_NO_NEW_MESSAGE             = -240,

        // Unkown Error
        eTC_HDR_RET_NG_UNKNOWN                          = -1000,

}eTC_HDR_RET_t;
```

Elements

Enum	Value	Description
eTC_HDR_RET_OK	0	OK
eTC_HDR_RET_NG_NOT_ENABLED	-1	Enable error
eTC_HDR_RET_NG_LOCK_MEMORY	-20	Lock memory error
eTC_HDR_RET_NG_INIT	-21	Init error
eTC_HDR_RET_NG_BYTESTREAM_OPEN	-22	Byte stream open error
eTC_HDR_RET_NG_BB_SRC_INIT	-23	BB Sample Rate Converter (SRC) init error
eTC_HDR_RET_NG_INSTANCE_INIT	-24	HD Radio instance error
eTC_HDR_RET_NG_CORE_LIB_INIT	-25	Core library init error
eTC_HDR_RET_NG_AUD_RESAMPLER_INIT	-26	Audio resampler init error
eTC_HDR_RET_NG_AUD_RESAMPLER_OUTPUT	-27	Audio resampler output error
eTC_HDR_RET_NG_AUD_RESAMPLER_HANDLER	-28	Audio resampler handler error
eTC_HDR_RET_NG_SET_BAND	-29	Set band error
eTC_HDR_RET_NG_INVALID_PARAMETERS	-30	Invalid parameter error
eTC_HDR_RET_NG_BLEND_CROSSFADE_RESET	-31	Blend cross-fade reset error
eTC_HDR_RET_NG_NOT_YET_INIT	-32	Not init error
eTC_HDR_RET_NG_NOT_YET_OPEN	-33	Not open error
eTC_HDR_RET_NG_ALREADY_INIT	-34	Already init error
eTC_HDR_RET_NG_ALREADY_OPEN	-35	Already open error
eTC_HDR_RET_NG_INVALID_HDR_TYPE	-36	Invalid type error
eTC_HDR_RET_NG_INVALID_IQ_BUFFER	-37	Invalid IQ buffer error
eTC_HDR_RET_NG_EXT_IQ_DRV_OPEN	-38	External IQ driver open error
eTC_HDR_RET_NG_MALLOC	-39	Malloc error
eTC_HDR_RET_NG_INVALID_BUFFER_POINTER	-40	Invalid buffer pointer error
eTC_HDR_RET_NG_READ_SIZE	-41	Read size error
eTC_HDR_RET_NG_RSC	-42	Resource error
eTC_HDR_RET_NG_MUTEX_INIT	-43	Mutex init error
eTC_HDR_RET_NG_MUTEX_DEINIT	-44	Mutex deinit error
eTC_HDR_RET_NG_NULL_POINTER_PARAMETERS	-45	Null pointer parameter error
eTC_HDR_RET_NG_NULL_POINTER_MESSAGE	-46	Null pointer message error
eTC_HDR_RET_NG_INIT_MESSAGE_QUEUE	-47	Init message queue error
eTC_HDR_RET_NG_SET_PROGRAM	-48	Set program error
eTC_HDR_RET_NG_ENABLE_PSD	-49	Enable PSD error
eTC_HDR_RET_NG_NOT_SUPPORT	-50	Not support error
eTC_HDR_RET_NG_INVALID_HDR_ID	-51	Invalid HD Radio ID error

Enum	Value	Description
eTC_HDR_RET_NG_NULL_POINTER_MUTEX	-53	Null pointer mutex
eTC_HDR_RET_NG_GET_PROGRAM	-54	Get program error
eTC_HDR_RET_NG_GET_DATA	-55	Get data error
eTC_HDR_RET_NG_LIB_ERROR	-56	Library error
eTC_HDR_RET_NG_NULL_INSTANCE	-57	Null Instance error
eTC_HDR_RET_NG_SET_VALUE	-58	Set value error
eTC_HDR_RET_NG_NOT_YET_CLOSE	-59	Not close error
eTC_HDR_RET_NG_IQ01IN_XRUN	-60	I2S IQ01 overflow error
eTC_HDR_RET_NG_IQ23IN_XRUN	-61	I2S IQ23 overflow error
eTC_HDR_RET_NG_IQ_INPUT_DRIVER	-62	IQ I2S driver error
eTC_HDR_RET_NG_INVALID_BAND	-63	Invalid band error
eTC_HDR_RET_NG_MUTEX_LOCK	-64	Mutex lock error
eTC_HDR_RET_NG_MUTEX_UNLOCK	-65	Mutex unlock error
eTC_HDR_RET_NG_EVENT_ERROR	-66	Event error
eTC_HDR_RET_NG_DEINIT	-67	Deinit error
eTC_HDR_RET_NG_ALREADY_CLOSE	-68	Already close error
eTC_HDR_RET_NG_DEMOD_INVALID_PARAMETERS	-200	Invalid input parameter (HD Radio DEMOD)
eTC_HDR_RET_NG_DEMOD_BUSY	-201	Busy HD Radio DEMOD
eTC_HDR_RET_NG_DEMOD_IDLE	-202	IDLE mode (HD Radio DEMOD)
eTC_HDR_RET_NG_DEMOD_NOT_INIT	-203	HDR DEMOD is not initialized.
eTC_HDR_RET_NG_DEMOD_INSTANCE_TYPE	-204	HDR DEMOD instance error
eTC_HDR_RET_NG_AAS_NOT_FOUND_PORT_OR_SERVICE	-210	Specified port number or service number is not found.
eTC_HDR_RET_NG_AAS_RESERVED_PORT_REQ	-211	Reserved port number is requested.
eTC_HDR_RET_NG_AAS_MAX_AAS_PORTS_ALREADY_ENABLED	-212	Maximum number of AAS ports is already enabled.
eTC_HDR_RET_NG_AAS_MAX_LOT_PORTS_ALREADY_ENABLED	-213	Maximum number of LOT ports is already enabled.
eTC_HDR_RET_NG_AAS_ALEADY_OPEN_PORT	-214	Specified port number is already opened.
eTC_HDR_RET_NG_AAS_NOT_FOUND_OBJECT	-215	Object not found
eTC_HDR_RET_NG_AAS_NO_DATA_AVAILABLE	-216	No data available
eTC_HDR_RET_NG_AAS_CORRUPTED_PACKET	-217	Packet is corrupted.
eTC_HDR_RET_NG_AAS_NO_COMPLETE_OBJECT	-218	Object is not completed.
eTC_HDR_RET_NG_PSD_INVALID_LENGTH	-220	Invalid length for the specified field
eTC_HDR_RET_NG_SIG_NO_SERVICE	-230	No service found with the specified service number ID
eTC_HDR_RET_NG_SIG_NO_COMPONENT	-231	Service found but no component with that index
eTC_HDR_RET_NG_ALERT_NO_NEW_MESSAGE	-240	No new message received
eTC_HDR_RET_NG_UNKNOWN	-1000	Unknown error

Comments

None

Header file

tchdr_types.h

3.2 eTC_HDR_TYPE_t

Description

This enum data type is a HD radio type. It is used in HD Radio type setting when initializing HD Radio framework.

Definition

```
typedef enum {
      eTC_HDR_TYPE_HD_1p0             = 1,
      eTC_HDR_TYPE_HD_1p5             = 2,
      eTC_HDR_TYPE_HD_1p0_MRC         = 3,
      eTC_HDR_TYPE_HD_1p5_MRC         = 4
}eTC_HDR_TYPE_t;
```

Elements

Enum	Description
eTC_HDR_TYPE_HD_1p0	HD 1.0
eTC_HDR_TYPE_HD_1p5	HD 1.5 (HD 1.0 + Background Scan)
eTC_HDR_TYPE_HD_1p0_MRC	HD 1.0 + MRC
eTC_HDR_TYPE_HD_1p5_MRC	HD 1.5 + MRC

Comments

None

Header file

tchdr_api.h

3.3 eTC_HDR_ID_t

Description

This enum data type is HDR ID.

Definition

```
typedef enum {
      eTC_HDR_ID_MAIN          = 0,
      eTC_HDR_ID_MRC           = 1,
      eTC_HDR_ID_BS            = 2
}eTC_HDR_ID_t;
```

Elements

Enum	Description
eTC_HDR_ID_MAIN	Main (Primary)
eTC_HDR_ID_MRC	MRC
eTC_HDR_ID_BS	Background Scan

Comments

None

Header file

tchdr_api.h

3.4 eTC_HDR_BAND_t

Description

This enum data type is HDR Band.

Definition

```
typedef enum {
      eTC_HDR_FM_BAND          = 0,
      eTC_HDR_AM_BAND          = 1,
      eTC_HDR_IDLE_BAND        = 0xFF
} eTC_HDR_BAND_t;
```

Elements

Enum	Description
eTC_HDR_FM_BAND	Frequency Modulation (FM) Band
eTC_HDR_AM_BAND	Amplitude Modulation (AM) Band
eTC_HDR_IDLE_BAND	Idle Band Only available in tchdr_setTune() API

Comments

None

Header file

tchdr_api.h

3.5 eTC_HDR_BBSRC_RATE_t

Description

This enum data type is HDR BB input sample rate.

Definition

```
typedef enum {
      eTC_HDR_BBSRC_650_KHZ    = 0,
      eTC_HDR_BBSRC_675_KHZ    = 1,
      eTC_HDR_BBSRC_744_KHZ    = 2,
      eTC_HDR_BBSRC_768_KHZ    = 3,
      eTC_HDR_BBSRC_1024_KHZ   = 4,
}eTC_HDR_BBSRC_RATE_t;
```

Elements

Enum	Description
eTC_HDR_BBSRC_650_KHZ	650 kHz Not supported yet
eTC_HDR_BBSRC_675_KHZ	675 kHz Not supported yet
eTC_HDR_BBSRC_744_KHZ	744.1875 kHz
eTC_HDR_BBSRC_768_KHZ	768 kHz Not supported yet
eTC_HDR_BBSRC_1024_KHZ	1024 kHz Not supported yet

Comments

None

Header file

tchdr_api.h

3.6 eTC_HDR_SIGNAL_STATUS_t

Description

This enum data type is a signal status type.

Definition

```
typedef enum {
      eBITMASK_SIGNAL_STATUS_HD_SIGNAL        = 0x01,
      eBITMASK_SIGNAL_STATUS_SIS              = 0x02,
      eBITMASK_SIGNAL_STATUS_SIS_OK           = 0x04,
      eBITMASK_SIGNAL_STATUS_HD_AUDIO         = 0x08
}eTC_HDR_SIGNAL_STATUS_t;
```

Elements

Enum	Description
eBITMASK_SIGNAL_STATUS_HD_SIGNAL	HD Signal Bitmask
eBITMASK_SIGNAL_STATUS_SIS	SIS Bitmask
eBITMASK_SIGNAL_STATUS_SIS_OK	SIS Cyclic Redundancy Check (CRC) OK Bitmask
eBITMASK_SIGNAL_STATUS_HD_AUDIO	HD Audio Bitmask

Comments

None

Header file

tchdr_api.h

3.7 eTC_HDR_AUDIO_MODE_t

Description

This enum data type is an audio mode setting.

Definition

```
typedef enum {
      eTC_HDR_AUDIO_BLEND            = 0,
      eTC_HDR_AUDIO_ANALOG           = 1,
      eTC_HDR_AUDIO_DIGITAL          = 2,
      eTC_HDR_AUDIO_SPLIT            = 3
}eTC_HDR_AUDIO_MODE_t;
```

Elements

Enum	Description
eTC_HDR_AUDIO_BLEND	Blend Audio Output Mode
eTC_HDR_AUDIO_ANALOG	Analog Audio Output Mode
eTC_HDR_AUDIO_DIGITAL	Digital Audio Output Mode
eTC_HDR_AUDIO_SPLIT	For testing

Comments

None

Header file

tchdr_api.h

3.8 eTC_HDR_PROGRAM_t

Description

This enum data type is a program number.

Definition

```
typedef enum {
        eTC_HDR_PROGRAM_HD1             = 0,
        eTC_HDR_PROGRAM_HD2             = 1,
        eTC_HDR_PROGRAM_HD3             = 2,
        eTC_HDR_PROGRAM_HD4             = 3,
        eTC_HDR_PROGRAM_HD5             = 4,
        eTC_HDR_PROGRAM_HD6             = 5,
        eTC_HDR_PROGRAM_HD7             = 6,
        eTC_HDR_PROGRAM_HD8             = 7,
        eTC_HDR_PROGRAM_MAX
}eTC_HDR_PROGRAM_t;
```

Elements

Enum	Description
eTC_HDR_PROGRAM_HD1	Program 1
eTC_HDR_PROGRAM_HD2	Program 2
eTC_HDR_PROGRAM_HD3	Program 3
eTC_HDR_PROGRAM_HD4	Program 4
eTC_HDR_PROGRAM_HD5	Program 5
eTC_HDR_PROGRAM_HD6	Program 6
eTC_HDR_PROGRAM_HD7	Program 7
eTC_HDR_PROGRAM_HD8	Program 8
eTC_HDR_PROGRAM_MAX	-

Comments

None

Header file

tchdr_api.h

3.9 eTC_HDR_NOTIFY_t

Description

This enum data type is a notification type.

Definition

```
typedef enum
{
	eTC_HDR_NOTIFY_NULL		= 0,
	eTC_HDR_NOTIFY_OPEN		= 101,
	eTC_HDR_NOTIFY_AUDIO_MODE	= 111,
	eTC_HDR_NOTIFY_TUNE		= 112,
	eTC_HDR_NOTIFY_PROGRAM		= 113,
	eTC_HDR_NOTIFY_MUTE		= 114,
	eTC_HDR_NOTIFY_AUDIO_CTRL	= 115,
	eTC_HDR_NOTIFY_PSD		= 121,
	eTC_HDR_NOTIFY_SIS		= 122,
	eTC_HDR_NOTIFY_ALERT		= 125,
	eTC_HDR_NOTIFY_LOT		= 126,
	eTC_HDR_NOTIFY_SIGNAL_STATUS	= 190,
	eTC_HDR_NOTIFY_PTY		= 191,
	eTC_HDR_NOTIFY_END
}eTC_HDR_NOTIFY_t;
```

Elements

Enum	Description
eTC_HDR_NOTIFY_OPEN	HD Radio framework is opened.
eTC_HDR_NOTIFY_AUDIO_MODE	Audio mode
eTC_HDR_NOTIFY_TUNE	Tuner Tune
eTC_HDR_NOTIFY_PROGRAM	For Demo. Deprecated.
eTC_HDR_NOTIFY_MUTE	Audio output mute.
eTC_HDR_NOTIFY_AUDIO_CTRL	Control audio output.
eTC_HDR_NOTIFY_PSD	For Demo. Deprecated.
eTC_HDR_NOTIFY_SIS	For Demo. Deprecated.
eTC_HDR_NOTIFY_ALERT	Emergency Alert
eTC_HDR_NOTIFY_LOT	Large Object Transfer
eTC_HDR_NOTIFY_SIGNAL_STATUS	Signal status
eTC_HDR_NOTIFY_PTY	For Demo. Deprecated.

Comments

eTC_HDR_NOTIFY_SIGNAL_STATUS is an event that notifies the values when the status changes regardless of the execution of the tchdr_getSignalStatus() function.

Header file

tchdr_api.h

3.10 eTC_HDR_BLEND_THRESH_SEL_t

Description

This enum data type is used for the blend threshold configuration.
The blend threshold is a means of controlling the audio quality level, at which the transition between digital and analog audio signals should occur.

With *blend threshold Q1* selected, the receiver conservatively blends to analog at the first indication of digital audio imperfection, limiting the digital coverage area but ensuring a listening experience that is virtually free of channel-induced digital audio artifacts. Conversely, *blend threshold Q4* results in a more aggressive blend strategy in which the digital coverage area is maximized, but the likelihood of audible artifacts increases as well.

Definition

```
typedef enum{
      eBLEND_THRESH_FORCE_ANALOG      = 0,
      eBLEND_THRESH_Q1_THRESH,
      eBLEND_THRESH_Q2_THRESH,
      eBLEND_THRESH_Q3_THRESH,
      eBLEND_THRESH_Q4_THRESH,
      eBLEND_THRESH_RESERVED1,
      eBLEND_THRESH_RESERVED2,
      eBLEND_THRESH_FORCE_DIGITAL,
      eBLEND_THRESH_TOTAL
}eTC_HDR_BLEND_THRESH_SEL_t;
```

Elements

Enum	Description
eBLEND_THRESH_FORCE_ANALOG	No blending - always output analog audio
eBLEND_THRESH_Q1_THRESH	Blend with virtually no audible artifacts (most conservative blending)
eBLEND_THRESH_Q2_THRESH	Blend with rare audible artifacts
eBLEND_THRESH_Q3_THRESH	Blend with infrequent audible artifacts
eBLEND_THRESH_Q4_THRESH	Blend with noticeable audible artifacts (most aggressive blending)
eBLEND_THRESH_FORCE_DIGITAL	No blending - always output digital audio
eBLEND_THRESH_TOTAL	Total number of blend threshold configurations

Comments

None

Header file

tchdr_api.h

3.11 eTC_HDR_BLEND_PARAMS_t

Description

This enum data type is a blend configuration parameter for the tchdr_setBlendParam() and tchdr_getBlendParam().

Definition

```
typedef enum {
      eBLEND_PARAM_FM_MPS_BLEND_THRESH                = 0,
      eBLEND_PARAM_FM_ALL_DIG_BLEND_THRESH,
      eBLEND_PARAM_FM_MPS_AUDIO_SCALING,
      eBLEND_PARAM_FM_ALL_DIG_AUDIO_SCALLING,
      eBLEND_PARAM_FM_MPS_BLEND_RATE,
      eBLEND_PARAM_FM_ALL_DIG_BLEND_RATE,
      eBLEND_PARAM_FM_MPS_DIG_AUDIO_DELAY,
      eBLEND_PARAM_AM_MPS_BLEND_THRESH,
      eBLEND_PARAM_AM_ALL_DIG_BLEND_THRESH,
      eBLEND_PARAM_AM_MPS_AUDIO_SCALING,
      eBLEND_PARAM_AM_ALL_DIG_AUDIO_SCALING,
      eBLEND_PARAM_AM_MPS_DIG_AUDIO_DELAY,
      eBLEND_PARAM_AM_MPS_BLEND_RATE,
      eBLEND_PARAM_AM_ALL_DIG_BLEND_RATE,
      eBLEND_PARAM_D2A_BLEND_HOLDOFF,
      eBLEND_PARAM_BLEND_DECISION,
      eBLEND_PARAM_FM_CDNO_BLEND_DECISION,
      eBLEND_PARAM_AM_CDNO_BLEND_DECISION,
      eBLEND_PARAM_FM_AUDIO_INVERT_PHASE,
      eBLEND_PARAM_AM_AUDIO_INVERT_PHASE,
      eBLEND_PARAM_DISABLE_AUDIO_SCALING
}eTC_HDR_BLEND_PARAMS_t;
```

Elements

Enum	Description
eBLEND_PARAM_FM_MPS_BLEND_THRESH	Blend threshold for FM MPS program
eBLEND_PARAM_FM_ALL_DIG_BLEND_THRESH	Blend threshold for all FM digital programs (MPS/SPS)
eBLEND_PARAM_FM_MPS_AUDIO_SCALING	Level alignment between digital audio and analog audio by scaling down the digital audio of FM Hybrid MPS
eBLEND_PARAM_FM_ALL_DIG_AUDIO_SCALLING	Scale down the audio of all FM digital programs (MPS/SPS)
eBLEND_PARAM_FM_MPS_BLEND_RATE	Control the step size of the FM analog hold duration
eBLEND_PARAM_FM_ALL_DIG_BLEND_RATE	Control the step size of the FM mute hold duration
eBLEND_PARAM_FM_MPS_DIG_AUDIO_DELAY	Delay for fine time alignment between digital audio and analog audio of FM Hybrid MPS
eBLEND_PARAM_AM_MPS_BLEND_THRESH	Blend threshold for AM MPS program
eBLEND_PARAM_AM_ALL_DIG_BLEND_THRESH	Blend threshold for all AM digital programs
eBLEND_PARAM_AM_MPS_AUDIO_SCALING	Level alignment between digital audio and analog audio by scaling down the digital audio of AM Hybrid MPS
eBLEND_PARAM_AM_ALL_DIG_AUDIO_SCALING	Scale down the audio of all AM digital programs (MPS/SPS)
eBLEND_PARAM_AM_MPS_DIG_AUDIO_DELAY	Delay for fine time alignment between digital audio and analog audio of AM Hybrid MPS
eBLEND_PARAM_AM_MPS_BLEND_RATE	Control the step size of the AM analog hold duration
eBLEND_PARAM_AM_ALL_DIG_BLEND_RATE	Control the step size of the AM mute hold duration
eBLEND_PARAM_D2A_BLEND_HOLDOFF	Samples for holding off digital audio when changing digital to analog
eBLEND_PARAM_BLEND_DECISION	To prevent frequency switching the blend from analog to digital, switching the blend is hold off until the value of this parameter becomes the set value of blend decision threshold.
eBLEND_PARAM_FM_CDNO_BLEND_DECISION	The minimum Cd/No required for FM on consecutive frames to allow normal blending
eBLEND_PARAM_AM_CDNO_BLEND_DECISION	The minimum Cd/No required for AM on consecutive frames to allow normal blending
eBLEND_PARAM_FM_AUDIO_INVERT_PHASE	Invert FM digital audio phase
eBLEND_PARAM_AM_AUDIO_INVERT_PHASE	Invert AM digital audio phase
eBLEND_PARAM_DISABLE_AUDIO_SCALING	Disable audio scaling inside the HD Radio library

Comments

Refer to Chapter 4.11 stTC_HDR_BLEND_PARAMS_t for more details on each enum.

Header file

tchdr_api.h

3.12 eTC_HDR_BLEND_ADV_PARAMS_t

Description

This enum data type is an advanced blend configuration parameter for the tchdr_setBlendAdvParam() and tchdr_getBlendAdvParam().

Definition

```
typedef enum {
	eBLEND_ADV_PARAM_RAMP_UP_ENABLED		= 0,
	eBLEND_ADV_PARAM_RAMP_UP_TIME,
	eBLEND_ADV_PARAM_RAMP_DOWN_ENABLED,
	eBLEND_ADV_PARAM_RAMP_DOWN_TIME,
	eBLEND_ADV_PARAM_COMFORT_NOISE_ENABLED,
	eBLEND_ADV_PARAM_COMFORT_NOISE_LEVEL,
	eBLEND_ADV_PARAM_AM_ENH_STREAM_HOLDOFF_ENABLED,
	eBLEND_ADV_PARAM_AM_MPS_ENH_STREAM_HOLDOFF_THRESH,
	eBLEND_ADV_PARAM_ALL_DIG_ENH_STREAM_HOLDOFF_THRESH,
	eBLEND_ADV_PARAM_AM_DIG_AUDIO_BW_MGMT_ENABLED,
	eBLEND_ADV_PARAM_AM_DIG_AUDIO_BLEND_START_BW,
	eBLEND_ADV_PARAM_AM_DIG_AUDIO_MAX_BW,
	eBLEND_ADV_PARAM_AM_DIG_AUDIO_BW_STEP_TIME,
	eBLEND_ADV_PARAM_AM_DIG_AUDIO_BW_STEP_UP_SIZE,
	eBLEND_ADV_PARAM_AM_DIG_AUDIO_BW_STEP_DOWN_SIZE,
	eBLEND_ADV_PARAM_AM_DIG_AUDIO_BW_STEP_THRESHOLD,
	eBLEND_ADV_PARAM_AM_MONO2STEREO_ENABLED,
	eBLEND_ADV_PARAM_AM_MONO2STEREO_START_BW,
	eBLEND_ADV_PARAM_AM_MONO2STEREO_STEP_TIME,
	eBLEND_ADV_PARAM_AM_MONO2STEREO_MAX_SEP
}eTC_HDR_BLEND_ADV_PARAMS_t;
```

Elements

Enum	Description
eBLEND_ADV_PARAM_RAMP_UP_ENABLED	Enable/Disable digital audio ramp up
eBLEND_ADV_PARAM_RAMP_UP_TIME	Time duration to ramp up
eBLEND_ADV_PARAM_RAMP_DOWN_ENABLED	Enable/Disable digital audio ramp down
eBLEND_ADV_PARAM_RAMP_DOWN_TIME	Time duration to ramp down
eBLEND_ADV_PARAM_COMFORT_NOISE_ENABLED	Enable/Disable comfort noise
eBLEND_ADV_PARAM_COMFORT_NOISE_LEVEL	Comfort noise level (dBFS)
eBLEND_ADV_PARAM_AM_ENH_STREAM_HOLDOFF_ENABLED	Enable/Disable enhanced audio stream hold-off
eBLEND_ADV_PARAM_AM_MPS_ENH_STREAM_HOLDOFF_THRESH	C/No threshold for enhanced stream hold-off of hybrid programs
eBLEND_ADV_PARAM_ALL_DIG_ENH_STREAM_HOLDOFF_THRESH	C/No threshold for enhanced stream hold-off of all digital programs
eBLEND_ADV_PARAM_AM_DIG_AUDIO_BW_MGMT_ENABLED	Enable/Disable AM audio bandwidth management
eBLEND_ADV_PARAM_AM_DIG_AUDIO_BLEND_START_BW	Set AM digital audio bandwidth at blend point
eBLEND_ADV_PARAM_AM_DIG_AUDIO_MAX_BW	Maximum bandwidth of the digital audio signal
eBLEND_ADV_PARAM_AM_DIG_AUDIO_BW_STEP_TIME	Step time duration
eBLEND_ADV_PARAM_AM_DIG_AUDIO_BW_STEP_UP_SIZE	Size by which the digital audio bandwidth will decrease for every step during blend transition
eBLEND_ADV_PARAM_AM_DIG_AUDIO_BW_STEP_DOWN_SIZE	Size by which the digital audio bandwidth will increase for every step during blend transition
eBLEND_ADV_PARAM_AM_DIG_AUDIO_BW_STEP_THRESHOLD	C/No threshold to determine whether stepping up or stepping down the bandwidth of digital audio
eBLEND_ADV_PARAM_AM_MONO2STEREO_ENABLED	Enable/Disable mono-to-stereo transition
eBLEND_ADV_PARAM_AM_MONO2STEREO_START_BW	AM mono-to-stereo starting bandwidth
eBLEND_ADV_PARAM_AM_MONO2STEREO_STEP_TIME	AM mono-to-stereo step duration
eBLEND_ADV_PARAM_AM_MONO2STEREO_MAX_SEP	Maximum stereo separation value

Comments

Refer to Chapter 4.12 stTC_HDR_BLEND_ADV_PARAMS_t for more details on each enum.

Header file

tchdr_api.h

3.13 eTC_HDR_THREAD_t

Description

This enum data type defines the threads of the HD Radio framework.

Definition

```
typedef enum {
      eTHREAD_MANAGER   =0,
      eTHREAD_IQINPUT   =1,
      eTHREAD_AUDINPUT  =2,
      eTHREAD_BBINPUT   =3,
      eTHREAD_DEMOD     =4,
      eTHREAD_BLENDING  =5,
      eTHREAD_AUDOUTPUT =6,
      eTHREAD_CMDPROC   =7,
      eTHREAD_LOGGER    =8,
      eTHREAD_MAX       =9
}eTC_HDR_THREAD_t;
```

Elements

Enum	Description
eTHREAD_MANAGER	Manager Thread
eTHREAD_IQINPUT	IQ Input Thread
eTHREAD_AUDINPUT	Audio Input Thread
eTHREAD_BBINPUT	Base-Band Input Thread
eTHREAD_DEMOD	Demodulation Thread
eTHREAD_BLENDING	Blending Audio Thread
eTHREAD_AUDOUTPUT	Audio Output Thread
eTHREAD_CMDPROC	Command Process Thread for CDM I/F
eTHREAD_LOGGER	Logger Thread

Comments

None

Header file

tcrhdr_api.h

3.14 eTC_HDR_PSD_LENGTH_CONFIG_t

Description

This enum data type is a list of PSD field/subfield length configuration parameters. HD Radio library can be configured to truncate the length of some PSD fields/subfields.

Definition

```
typedef enum {
	eTC_HDR_PSD_TITLE_LENGTH_CONFIG,
	eTC_HDR_PSD_ARTIST_LENGTH_CONFIG,
	eTC_HDR_PSD_ALBUM_LENGTH_CONFIG,
	eTC_HDR_PSD_GENRE_LENGTH_CONFIG,
	eTC_HDR_PSD_COMM_SHORT_CONTENT_LENGTH_CONFIG,
	eTC_HDR_PSD_COMM_ACTUAL_TEXT_LENGTH_CONFIG,
	eTC_HDR_PSD_UFID_OWNER_ID_LENGTH_CONFIG,
	eTC_HDR_PSD_COMR_PRICE_STRING_LENGTH_CONFIG,
	eTC_HDR_PSD_COMR_CONTACT_URL_LENGTH_CONFIG,
	eTC_HDR_PSD_COMR_SELLER_NAME_LENGTH_CONFIG,
	eTC_HDR_PSD_COMR_DESCRIPTION_LENGTH_CONFIG,
	eTC_HDR_PSD_XHDR_LENGTH_CONFIG,
	eTC_HDR_PSD_NUM_FIELD_CONFIG
}eTC_HDR_PSD_LENGTH_CONFIG_t;
```

Elements

Enum	Description
eTC_HDR_PSD_TITLE_LENGTH_CONFIG	Title Length Configuration
eTC_HDR_PSD_ARTIST_LENGTH_CONFIG	Artist Length Configuration
eTC_HDR_PSD_ALBUM_LENGTH_CONFIG	Album Length Configuration
eTC_HDR_PSD_GENRE_LENGTH_CONFIG	Genre Length Configuration
eTC_HDR_PSD_COMM_SHORT_CONTENT_LENGTH_CONFIG	Comment Short Content Length Configuration
eTC_HDR_PSD_COMM_ACTUAL_TEXT_LENGTH_CONFIG	Comment Actual Text Length Configuration
eTC_HDR_PSD_UFID_OWNER_ID_LENGTH_CONFIG	UFID Owner Identifier Length Configuration
eTC_HDR_PSD_COMR_PRICE_STRING_LENGTH_CONFIG	Commercial Price String Length Configuration
eTC_HDR_PSD_COMR_CONTACT_URL_LENGTH_CONFIG	Commercial Contact URL Length Configuration
eTC_HDR_PSD_COMR_SELLER_NAME_LENGTH_CONFIG	Commercial Seller Name Length Configuration
eTC_HDR_PSD_COMR_DESCRIPTION_LENGTH_CONFIG	Commercial Description Length Configuration
eTC_HDR_PSD_XHDR_LENGTH_CONFIG	XHDR Length Configuration
eTC_HDR_PSD_NUM_FIELD_CONFIG	Number of field length configurations

Comments

None

Header file

tchdr_psd.h

3.15 eTC_HDR_PSD_CHAR_TYPE_t

Description

This enum data type is a character encoding type for PSD message strings.

Definition

```
typedef enum {
      eTC_HDR_PSD_ISO_IEC_8859_1_1998         = 0,
      eTC_HDR_PSD_ISO_IEC_10646_1_2000        = 1,
      eTC_HDR_PSD_BINARY                      = 0xFF
}eTC_HDR_PSD_CHAR_TYPE_t;
```

Elements

Enum	Description
eTC_HDR_PSD_ISO_IEC_8859_1_1998	8-bit Unicode
eTC_HDR_PSD_ISO_IEC_10646_1_2000	16-bit Unicode
eTC_HDR_PSD_BINARY	Binary

Comments

None

Header file

tchdr_psd.h

3.16 eTC_HDR_PSD_FIELD_t

Description

This enum data type is a PSD field for demo.

Definition

```
typedef enum {
	eTC_HDR_PSD_TITLE		= 0,
	eTC_HDR_PSD_ARTIST,
	eTC_HDR_PSD_ALBUM,
	eTC_HDR_PSD_GENRE,
	eTC_HDR_PSD_COMMENT_LANGUAGE,
	eTC_HDR_PSD_COMMENT_SHORT_CONTENT,
	eTC_HDR_PSD_COMMENT_ACTUAL_TEXT,
	eTC_HDR_PSD_COMMERCIAL_PRICE_STRING,
	eTC_HDR_PSD_COMMERCIAL_VALID_UNTIL,
	eTC_HDR_PSD_COMMERCIAL_CONTACT_URL,
	eTC_HDR_PSD_COMMERCIAL_RECEIVED_AS,
	eTC_HDR_PSD_COMMERCIAL_SELLER_NAME,
	eTC_HDR_PSD_COMMERCIAL_DESCRIPTION,
	eTC_HDR_PSD_XHDR,
	eTC_HDR_PSD_MAX
}eTC_HDR_PSD_FIELD_t;
```

Elements

Enum	Description
eTC_HDR_PSD_TITLE	Title
eTC_HDR_PSD_ARTIST	Artist
eTC_HDR_PSD_ALBUM	Album
eTC_HDR_PSD_GENRE	Genre
eTC_HDR_PSD_COMMENT_LANGUAGE	Comment Language
eTC_HDR_PSD_COMMENT_SHORT_CONTENT	Comment Short Content
eTC_HDR_PSD_COMMENT_ACTUAL_TEXT	Comment Actual Text
eTC_HDR_PSD_COMMERCIAL_PRICE_STRING	Commercial Price String
eTC_HDR_PSD_COMMERCIAL_VALID_UNTIL	Commercial Valid Until
eTC_HDR_PSD_COMMERCIAL_CONTACT_URL	Commercial Contact URL
eTC_HDR_PSD_COMMERCIAL_RECEIVED_AS	Commercial Received As
eTC_HDR_PSD_COMMERCIAL_SELLER_NAME	Commercial Name of Seller
eTC_HDR_PSD_COMMERCIAL_DESCRIPTION	Commercial Description
eTC_HDR_PSD_XHDR	Display Synchronization Trigger
eTC_HDR_PSD_MAX	-

Comments

None

Header file

tchdr_psd.h

3.17 eTC_HDR_PSD_COMM_SUBFIELD_t

Description

This enum data type is a list of supported comment subfields.

Definition

```
typedef enum {
      eTC_HDR_PSD_COMM_LANGUAGE,
      eTC_HDR_PSD_COMM_SHORT_CONTENT,
      eTC_HDR_PSD_COMM_ACTUAL_TEXT,
      eTC_HDR_PSD_NUM_COMM_SUBFIELDS
}eTC_HDR_PSD_COMM_SUBFIELD_t;
```

Elements

Enum	Description
eTC_HDR_PSD_COMM_LANGUAGE	Comment Language
eTC_HDR_PSD_COMM_SHORT_CONTENT	Comment Short Content
eTC_HDR_PSD_COMM_ACTUAL_TEXT	Comment Actual Text
eTC_HDR_PSD_NUM_COMM_SUBFIELDS	Number of comment subfields

Comments

None

Header file

tchdr_psd.h

3.18 eTC_HDR_PSD_UFID_SUBFIELD_t

Description

This enum data type is a list of supported UFID subfields.

Definition

```
typedef enum {
      eTC_HDR_PSD_UFID_OWNER_ID,
      eTC_HDR_PSD_UFID_FILE_ID,
      eTC_HDR_PSD_NUM_UFID_SUBFIELDS,
}eTC_HDR_PSD_UFID_SUBFIELD_t
```

Elements

Enum	Description
eTC_HDR_PSD_UFID_OWNER_ID	UFID owner
eTC_HDR_PSD_UFID_FILE_ID	UFID file ID
eTC_HDR_PSD_NUM_UFID_SUBFIELDS	Number of UFID subfields

Comments

None

Header file

tchdr_psd.h

3.19 eTC_HDR_PSD_COMR_SUBFIELD_t

Description

This enum data type is a list of supported commercial subfields.

Definition

```
typedef enum {
      eTC_HDR_PSD_COMR_PRICE_STRING,
      eTC_HDR_PSD_COMR_VALID_UNTIL,
      eTC_HDR_PSD_COMR_CONTACT_URL,
      eTC_HDR_PSD_COMR_RECEIVED_AS,
      eTC_HDR_PSD_COMR_SELLER_NAME,
      eTC_HDR_PSD_COMR_DESCRIPTION,
      eTC_HDR_PSD_NUM_COMR_SUBFIELDS
}eTC_HDR_PSD_COMR_SUBFIELD_t;
```

Elements

Enum	Description
eTC_HDR_PSD_COMR_PRICE_STRING	Commercial Price String
eTC_HDR_PSD_COMR_VALID_UNTIL	Commercial Valid Until
eTC_HDR_PSD_COMR_CONTACT_URL	Commercial Contact URL
eTC_HDR_PSD_COMR_RECEIVED_AS	Commercial Received As
eTC_HDR_PSD_COMR_SELLER_NAME	Commercial Name of Seller
eTC_HDR_PSD_COMR_DESCRIPTION	Commercial Description
eTC_HDR_PSD_NUM_COMR_SUBFIELDS	Number of commercial subfields

Comments

None

Header file

tchdr_psd.h

3.20 eTC_HDR_PSD_BITMASK_t

Description

This enum data type is PSD bitmask.

Definition

```
typedef enum {
      eBITMASK_PSD_TITLE              = 0x01,
      eBITMASK_PSD_ARTIST             = 0x02,
      eBITMASK_PSD_ALBUM              = 0x04,
      eBITMASK_PSD_GENRE              = 0x08,
      eBITMASK_PSD_COMMENT            = 0x10,
      eBITMASK_PSD_COMMERCIAL         = 0x40,
      eBITMASK_PSD_XHDR               = 0x80
}eTC_HDR_PSD_BITMASK_t;
```

Elements

Enum	Description
eBITMASK_PSD_TITLE	Title
eBITMASK_PSD_ARTIST	Artist
eBITMASK_PSD_ALBUM	Album
eBITMASK_PSD_GENRE	Genre
eBITMASK_PSD_COMMENT	Comment
eBITMASK_PSD_COMMERCIAL	Commercial
eBITMASK_PSD_XHDR	Display Synchronization Trigger

Comments

None

Header file

tchdr_psd.h

3.21 eTC_HDR_SIG_STATUS_t

Description

This enum data type is a SIG status.

Definition

```
typedef enum {
      eTC_HDR_SIG_NO_DATA,
      eTC_HDR_SIG_OLD_DATA,
      eTC_HDR_SIG_NEW_DATA,
      eTC_HDR_SIG_REPEAT_DATA
}eTC_HDR_SIG_STATUS_t
```

Elements

Enum	Description
eTC_HDR_SIG_NO_DATA	No data available
eTC_HDR_SIG_OLD_DATA	Same information since the last read
eTC_HDR_SIG_NEW_DATA	New information available
eTC_HDR_SIG_REPEAT_DATA	Information is updated, but it is the same as the previous information.

Comments

None

Header file

tcrhdr_sig.h

3.22 eTC_HDR_SIG_COMPONENT_TYPE_t

Description

This enum data type is a possible service component type.

Definition

```
typedef enum {
      eTC_HDR_SIG_AUDIO_COMPONENT,
      eTC_HDR_SIG_DATA_COMPONENT,
      eTC_HDR_SIG_NUM_COMPONENT_TYPES
}eTC_HDR_SIG_COMPONENT_TYPE_t;
```

Elements

Enum	Description
eTC_HDR_SIG_AUDIO_COMPONENT	Audio-related component
eTC_HDR_SIG_DATA_COMPONENT	Data-related component
eTC_HDR_SIG_NUM_COMPONENT_TYPES	Number of component types

Comments

None

Header file

tchdr_sig.h

3.23 eTC_HDR_SIG_SERVICE_TYPE_t

Description

This enum data type is a possible SIG service type.

Definition

```
typedef enum {
      eTC_HDR_SIG_AUDIO_SERVICE_TYPE,
      eTC_HDR_SIG_DATA_SERVICE_TYPE,
      eTC_HDR_SIG_NUM_SERVICE_TYPES
}eTC_HDR_SIG_SERVICE_TYPE_t;
```

Elements

Enum	Description
eTC_HDR_SIG_AUDIO_SERVICE_TYPE	Audio-related service
eTC_HDR_SIG_DATA_SERVICE_TYPE	Data-related service
eTC_HDR_SIG_NUM_SERVICE_TYPES	Number of service types

Comments

None

Header file

tchdr_sig.h

3.24 eTC_HDR_SIS_STATUS_t

Description

This enum data type is the updated status of SIS.

Definition

```
typedef enum {
      eTC_HDR_SIS_NO_DATA             = 0,
      eTC_HDR_SIS_OLD_DATA,
      eTC_HDR_SIS_NEW_DATA,
      eTC_HDR_SIS_ERROR
}eTC_HDR_SIS_STATUS_t;
```

Elements

Enum	Description
eTC_HDR_SIS_NO_DATA	No data for requested type is available.
eTC_HDR_SIS_OLD_DATA	Available, but old data. No new data is received since the last request.
eTC_HDR_SIS_NEW_DATA	New data available
eTC_HDR_SIS_ERROR	Error occurs while processing the request

Comments

None

Header file

tchdr_sis.h

3.25 eTC_HDR_SIS_ALFN_STATUS_t

Description

This enum data type is SIS Absolute Layer 1 Frame Number (ALFN) status.

Definition

```
typedef enum {
      eTC_ALFN_VALID,
      eTC_ALFN_ACQUIRING,
      eTC_ALFN_FAILURE,
      eTC_ALFN_INVALID
}eTC_HDR_SIS_ALFN_STATUS_t;
```

Elements

Enum	Description
eTC_ALFN_VALID	Valid data is received.
eTC_ALFN_ACQUIRING	Still waiting for data
eTC_ALFN_FAILURE	Failure occurs
eTC_ALFN_INVALID	Data is received, but it is determined to be invalid.

Comments

None

Header file

tchdr_sis.h

3.26 eTC_HDR_SIS_TEXT_ENCODING_t

Description

This enum data type is a text encoding type for SIS message strings.

Definition

```
typedef enum {
      eTC_HDR_SIS_ISO_IEC_8859_1_1998          = 0,
      eTC_HDR_SIS_ISO_IEC_10646_1_2000         = 4
}eTC_HDR_SIS_TEXT_ENCODING_t;
```

Elements

Enum	Description
eTC_HDR_SIS_ISO_IEC_8859_1_1998	8-bit Unicode
eTC_HDR_SIS_ISO_IEC_10646_1_2000	16-bit Unicode

Comments

None

Header file

tchdr_sis.h

3.27 eTC_HDR_SIS_DST_SCHEDULE_t

Description

This enum data type is a region of DST schedule.

Definition

```
typedef enum {
      eTC_HDR_SIS_DST_SCHED_NONE,
      eTC_HDR_SIS_DST_SCHED_US_CAN,
      eTC_HDR_SIS_DST_SCHED_EU
}eTC_HDR_SIS_DST_SCHEDULE_t;
```

Elements

Enum	Description
eTC_HDR_SIS_DST_SCHED_NONE	None
eTC_HDR_SIS_DST_SCHED_US_CAN	USA/Canada
eTC_HDR_SIS_DST_SCHED_EU	Europe

Comments

None

Header file

tchdr_sis.h

3.28 eTC_HDR_SIS_DST_LOCAL_t

Description

This enum data type indicates whether DST is practiced locally.

Definition

```
typedef enum {
      eTC_HDR_SIS_DST_NOT_PRACTICED,
      eTC_HDR_SIS_DST_PRACTICED
}eTC_HDR_SIS_DST_LOCAL_t;
```

Elements

Enum	Description
eTC_HDR_SIS_DST_NOT_PRACTICED	Not practiced
eTC_HDR_SIS_DST_PRACTICED	Practiced

Comments

None

Header file

tchdr_sis.h

3.29 eTC_HDR_SIS_ACCESS_TYPE_t

Description

This enum data type is a SIS access type.

Definition

```
typedef enum {
      eTC_HDR_SIS_ACCESS_PUBLIC               = 0,
      eTC_HDR_SIS_ACCESS_RESTRICTED,
      eTC_HDR_MAX_SIS_ACCESS_TYPE
}eTC_HDR_SIS_ACCESS_TYPE_t;
```

Elements

Enum	Description
eTC_HDR_SIS_ACCESS_PUBLIC	Public, unrestricted
eTC_HDR_SIS_ACCESS_RESTRICTED	Restricted
eTC_HDR_MAX_SIS_ACCESS_TYPE	-

Comments

None

Header file

tchdr_sis.h

3.30 eTC_HDR_SIS_BITMASK_t

Description

This enum data type is SIS message bitmask for demo.

Definition

```
typedef enum {
      eBITMASK_SIS_STATION_ID                       = 0x00000001,
      eBITMASK_SIS_SHORT_NAME                       = 0x00000002,
      eBITMASK_SIS_LOCATION                         = 0x00000004,
      eBITMASK_SIS_UNIVERSAL_SHORT_NAME             = 0x00000008,
      eBITMASK_SIS_SLOGAN                           = 0x00000010,
      eBITMASK_SIS_STATION_MESSAGE                  = 0x00000020,
      eBITMASK_SIS_STATION_TIME_ZONE                = 0x00000040,
      eBITMASK_SIS_LEAP_SECONDS                     = 0x00000080,
      eBITMASK_SIS_EXCITER_CORE_VERSION             = 0x00000100,
      eBITMASK_SIS_EXCITER_MANUFACTURER_VERSION     = 0x00000200,
      eBITMASK_SIS_IMPORTER_CORE_VERSION            = 0x00000400,
      eBITMASK_SIS_IMPROTER_MANUFACTURER_VERISON    = 0x00000800
}eTC_HDR_SIS_BITMASK_t;
```

Elements

Enum	Description
eBITMASK_SIS_STATION_ID	Station ID
eBITMASK_SIS_SHORT_NAME	Short name
eBITMASK_SIS_LOCATION	Location
eBITMASK_SIS_UNIVERSAL_SHORT_NAME	Universal short name
eBITMASK_SIS_SLOGAN	Slogan
eBITMASK_SIS_STATION_MESSAGE	Station message
eBITMASK_SIS_STATION_TIME_ZONE	Station time zone
eBITMASK_SIS_LEAP_SECONDS	Leap seconds
eBITMASK_SIS_EXCITER_CORE_VERSION	Exciter core version
eBITMASK_SIS_EXCITER_MANUFACTURER_VERSION	Exciter manufacturer version
eBITMASK_SIS_IMPORTER_CORE_VERSION	Importer core version
eBITMASK_SIS_IMPROTER_MANUFACTURER_VERISON	Importer manufacturer version

Comments

None

Header file

tchdr_sis.h

3.31 eTC_HDR_AAS_PORT_MODE_t

Description

This enum data type has the following AAS port types: ordered and non-ordered. In the ordered mode, data packets are ordered within the system before they are sent to the caller. In non-ordered mode, data packets are sent to the caller as they arrive. Even if the port is enabled in non-ordered mode, the caller may use the sequence number provided with the data packets to accomplish the same thing as the ordered mode.

Definition

```
typedef enum {
      eTC_HDR_AAS_PORT_NON_ORDERED     = 0x00,
      eTC_HDR_AAS_PORT_ORDERED         = 0x04
}eTC_HDR_AAS_PORT_MODE_t;
```

Elements

Enum	Description
eTC_HDR_AAS_PORT_NON_ORDERED	Do not order the packets. (First-in-first-out)
eTC_HDR_AAS_PORT_ORDERED	Order the packets based on the sequence number

Comments

This enum is used as the input value of the mode member of stTC_HDR_AAS_PORT_LIST_t.

Header file

tchdr_aas.h

3.32 eTC_HDR_ALERT_TEXT_ENCODING_t

Description

This enum data type is a text encoding type for alert message strings.

Definition

```
typedef enum {
      eTC_HDR_ALERT_ISO_IEC_8859_1_1998       = 0,
      eTC_HDR_ALERT_ISO_IEC_8859_1_1998C      = 1
}eTC_HDR_ALERT_TEXT_ENCODING_t;
```

Elements

Enum	Description
eTC_HDR_ALERT_ISO_IEC_8859_1_1998	8-bit unicode character (default)
eTC_HDR_ALERT_ISO_IEC_8859_1_1998C	ISO/IEC 8859-1:1998 compressed

Comments

None

Header file

tchdr_alert.h

4 STRUCTURE

Table 4.1 describes the structures used in Telechips radio Hardware Abstraction Layer (HAL).

Table 4.1 Definition of Structure Type

Structure Type	Description
stTC_HDR_THREAD_PR_t	HD Radio framework thread priority structure
stTC_HDR_IQ_t	IQ I2S information structure
stTC_HDR_CONF_t	HD Radio configuration structure
stTC_HDR_TUNE_TO_t	Structure used for tuning band/frequency
stTC_HDR_TUNE_INFO_t	Tune information structure
stTC_HDR_SIGNAL_STATUS_t	HD Radio signal status structure
stTC_HDR_PROG_BITMAP_t	HD Radio program bitmask structure
stTC_HDR_STATUS_t	Structure of all HD Radio status
stTC_HDR_PCM_t	PCM data structure
stTC_HDR_PTY_t	Program type for each program
stTC_HDR_BLEND_PARAMS_t	Blend configuration parameters
stTC_HDR_BLEND_ADV_PARAMS_t	Advanced blend configuration parameters
stTC_HDR_AAS_PORT_LIST_t	Define a structure used to specify the list of ports to be enabled/disabled
stTC_HDR_AAS_PACKET_INFO_t	Information associated with a received packet
stTC_HDR_AAS_LOT_OBJECT_LIST_t	List of all complete and incomplete LOT objects
stTC_HDR_AAS_LOT_OBJECT_HEADER_t	LOT object header definition
stTC_HDR_LOT_t	LOT object definition
stTC_HDR_ALERT_MESSAGE_t	Output structure for an HD Radio emergency alert message
stTC_HDR_ALERTS_MSG_STATUS_t	Output structure for current status of message reception
stTC_HDR_PSD_FIELDS_t	PSD field structure
stTC_HDR_PSD_FORM_t	PSD format structure
stTC_HDR_PSD_XHDR_PARAM_t	PSD XHDR parameter field structure
stTC_HDR_PSD_XHDR_FRAME_t	PSD XHDR frame structure
stTC_HDR_PSD_t	PSD information structure
stTC_HDR_SIG_SERVICE_LIST_t	Specify a list of available SIG services
stTC_HDR_SIG_SERVICE_INFO_t	Output data structure for SIG service information
stTC_HDR_SIG_SERVICE_COMPONENT_t	Define service component structure
stTC_HDR_SIS_ENABLED_BASIC_TYPES_t	Structure used for enabling/disabling basic SIS data types
stTC_HDR_SIS_ALFN_t	Absolute Layer 1 Frame Number (ALFN) output data structure
stTC_HDR_SIS_STATION_ID_t	Station ID structure
stTC_HDR_SIS_SHORT_NAME_t	Retrieve the station name
stTC_HDR_SIS_STATION_LOCATION_t	Output data structure for station location
stTC_HDR_SIS_LEAP_SEC_t	Define structure with data related to leap second
stTC_HDR_SIS_STATION_MSG_t	Output data structure for SIS station message
stTC_HDR_SIS_LOCAL_TIME_t	Define structure for retrieving station local time data
stTC_HDR_SIS_UNIV_NAME_t	Output data structure for SIS station universal name
stTC_HDR_SIS_STATION_SLOGAN_t	Output data structure for SIS station slogan
stTC_HDR_SIS_AVAIL_PROGRAMS_t	List of available audio programs reported by SIS
stTC_HDR_SIS_PROGRAM_INFO_t	Audio program information reported by SIS
stTC_HDR_SIS_AVAIL_DATA_SERVICES_t	List of the available data services reported by SIS
stTC_HDR_SIS_DATA_SERVICES_INFO_t	Data services information structure
stTC_HDR_SIS_TX_VER_STR_t	Structure contains exciter core version data
stTC_HDR_SIS_TX_MANUF_VER_t	Structure contains exciter core version string and manufacturer ID
stTC_HDR_SIS_t	SIS information structure

4.1 stTC_HDR_THREAD_PR_t

Description

This structure defines the priority of HD Radio framework threads.

Definition

```
typedef struct{
        S32             policy;
        S32             priority;
}stTC_HDR_THREAD_PR_t;
```

Elements

Member	Description
policy	Thread policy 0: SCHED_OTHER 1: SCHED_FIFO
priority	Thread priority If the policy is 0, the range is from -20 (High Priority) to +19 (Low Priority). If the policy is 1, the range is from 1 (Low Priority) to 99 (High Priority).

Comments

The priority range of SCHED_FIFO (Real-Time) is from 1 to 99, and 99 is the highest priority.
The priority range of SCHED_OTHER (Nice) is from -20 to +19, and -20 is the highest priority.

API	**Nice**		**Real-Time**	
	+19 ⋯ 0	-1 ⋯ -20	1 ⋯ 49	50 ⋯ 99
TOP	**Priority (PR)**			
	39 ⋯ 0		-2 ⋯ -99 -100 (shown as *rt*)	
Kernel	**Fair Scheduling (OTHER)**		**Real-Time Scheduling (FIFO/RR)**	
	139 ⋯ 100		99 ⋯ 1	

Header file

tchdr_api.h

4.2 stTC_HDR_IQ_t

Description

This structure defines the I/Q I2S configuration for HD Radio.

Definition

```
typedef struct {
        U32                     samplingBit;
        eTC_HDR_BBSRC_RATE_t    maxSampleRate;
}stTC_HDR_IQ_t;
```

Elements

Member	Description
samplingBit	IQ I2S sampling bit
maxSampleRate	Maximum sample rate of IQ I2S. Refer to Chapter 3.5 eTC_HDR_BBSRC_RATE_t.

Comments

None

Header file

tchdr_api.h

4.3 stTC_HDR_CONF_t

Description

This structure defines the HD Radio configuration.

Definition

```
typedef struct {
        eTC_HDR_TYPE_t          hdrType;
        stTC_HDR_IQ_t           iq;
        void                    *reserved;
}stTC_HDR_CONF_t;
```

Elements

Member	Description
hdrType	HD Radio type configuration
iq	IQ I2S information structure

Comments

None

Header file

tchdr_api.h

4.4 stTC_HDR_TUNE_TO_t

Description

This structure defines current tune information.

Definition

```
typedef struct {
        eTC_HDR_BAND_t          band;
        U32                     freq;
        eTC_HDR_BBSRC_RATE_t    iqsamplerate;
}stTC_HDR_TUNE_TO_t;
```

Elements

Member	Description
band	Radio band. Refer to Chapter 3.4 eTC_HDR_BAND_t.
freq	Current frequency
iqsamplerate	IQ I2S sample rate. Refer to Chapter 3.5 eTC_HDR_BBSRC_RATE_t.

Comments

None

Header file

tchdr_api.h

4.5 stTC_HDR_TUNE_INFO_t

Description

This structure defines tune information of all tuners. This information is required only when opening the HD radio.

Definition

```
typedef struct {
        stTC_HDR_TUNE_TO_t              main;
        stTC_HDR_TUNE_TO_t              mrc;
        stTC_HDR_TUNE_TO_t              bs;
        stTC_HDR_TUNE_TO_t              reserved;
}stTC_HDR_TUNE_INFO_t;
```

Elements

Member	Description
main	Tune information of main tuner
mrc	Tune information of MRC tuner
bs	Tune information of BS tuner

Comments

None

Header file

tchdr_api.h

4.6 stTC_HDR_SIGNAL_STATUS_t

Description

This structure indicates the default signal status.

Definition

```
typedef struct{
        eTC_HDR_ID_t            hdrID;
        U32                     curPN;
        U32                     acqStatus;
        U32                     cnr;
        U32                     pmap;
        U32                     hybridProgram;
} stTC_HDR_SIGNAL_STATUS_t;
```

Elements

Member	Description
hdrID	HD Radio ID. Refer to Chapter 3.3 eTC_HDR_ID_t.
curPN	Current Program Number (PN)
acqStatus	[3:0] 0: hd_signal_acquired 1: sis_acquired 2: sis_crc_ok 3: digital_audio_acquired
cnr	Carrier to Noise Ratio (CNR)
pmap	Audio available program bitmap
hybridProgram	Hybrid program

Comments

None

Header file

tchdr_api.h

4.7 stTC_HDR_PROG_BITMAP_t

Description

This structure is a bitmap of audio programs.

Definition

```
typedef union {
        struct {
                U8      prog1:1;
                U8      prog2:1;
                U8      prog3:1;
                U8      prog4:1;
                U8      prog5:1;
                U8      prog6:1;
                U8      prog7:1;
                U8      prog8:1;
        }prog;
                U8      all;
}stTC_HDR_PROG_BITMAP_t;
```

Elements

Member	Description
prog1	Bit of program 1
prog2	Bit of program 2
prog3	Bit of program 3
prog4	Bit of program 4
prog5	Bit of program 5
prog6	Bit of program 6
prog7	Bit of program 7
prog8	Bit of program 8
all	Bitmap of all programs

Comments

None

Header file

tchdr_api.h

4.8 stTC_HDR_STATUS_t

Description

This structure indicates all HD Radio status.

Definition

```
typedef struct {
        eTC_HDR_ID_t    hdrID;
        U32             curPN;
        U32             acqStatus;
        U32             audioQualityIndicator;
        U32             cnr;
        U32             digitalAudioGain;
        U32             blendControl;
        U32             pty[eTC_HDR_PROGRAM_MAX];
        U32             curPty;
        U32             pmap;
        U32             chgPmap;
        U32             psm;
        U32             codecMode;
        U32             hybridProgram;
        U32             dsqm;
        U32             rawSnr;
}stTC_HDR_STATUS_t;
```

Elements

Member	Description
hdrID	HD Radio ID. Refer to Chapter 3.3 eTC_HDR_ID_t.
curPN	Current PN
acqStatus	Acquisition status
audioQualityIndicator	Audio quality indicator
cnr	Carrier to Noise Ratio (CNR)
digitalAudioGain	Digital audio gain
blendControl	Blend control
pty[eTC_HDR_PROGRAM_MAX]	Program Type (PTY) eTC_HDR_PROGRAM_MAX is 8 Refer to Chapter 3.8 eTC_HDR_PROGRAM_t.
curPty	Current PTY
pmap	PMAP Bitmap of available programs
chgPmap	Change PMAP Program bitmap for the update field among Program Service Data (PSD)
psm	Primary Service Mode (PSM)
codecMode	Codec mode
hybridProgram	Hybrid program
dsqm	Digital Signal Quality Measurement (DSQM)
rawSnr	Raw Signal to Noise Ratio (SNR)

Comments

None

Header file

tchdr_api.h

4.9 stTC_HDR_PCM_t

Description

This structure is an audio PCM data structure.

Definition

```
typedef struct{
        U16             left;
        U16             right;
}stTC_HDR_PCM_t;
```

Elements

Member	Description
left	Left PCM (16-bit)
right	Right PCM (16-bit)

Comments

None

Header file

tchdr_api.h

4.10 stTC_HDR_PTY_t

Description

This structure is a program type structure for each program.

Definition

```
typedef struct{
        U32     value[eTC_HDR_PROGRAM_MAX];
}stTC_HDR_PTY_t;
```

Elements

Member	Description
value[eTC_HDR_PROGRAM_MAX]	Program Types Array eTC_HDR_PROGRAM_MAX is 8 Refer to Chapter 3.8 eTC_HDR_PROGRAM_t.

Comments

None

Header file

tchdr_api.h

4.11 stTC_HDR_BLEND_PARAMS_t

Description

This is a structure of blend configuration parameters.

Definition

```
typedef struct {
        eTC_HDR_BLEND_THRESH_SEL_t              fm_mps_blend_thresh;
        eTC_HDR_BLEND_THRESH_SEL_t              fm_all_dig_blend_thresh;
        U32                                     fm_mps_audio_scaling;
        U32                                     fm_all_dig_audio_scaling;
        U32                                     fm_mps_blend_rate;
        U32                                     fm_all_dig_blend_rate;
        U32                                     fm_mps_dig_audio_delay;
        eTC_HDR_BLEND_THRESH_SEL_t              am_mps_blend_thresh;
        eTC_HDR_BLEND_THRESH_SEL_               am_all_dig_blend_thresh;
        U32                                     am_mps_audio_scaling;
        U32                                     am_all_dig_audio_scaling;
        U32                                     am_mps_dig_audio_delay;
        U32                                     am_mps_blend_rate;
        U32                                     am_all_dig_blend_rate;
        U32                                     d2a_blend_holdoff;
        HDBOOL                                  blend_decision;
        U32                                     fm_cdno_blend_decision;
        U32                                     am_cdno_blend_decision;
        HDBOOL                                  fm_audio_invert_phase;
        HDBOOL                                  am_audio_invert_phase;
        HDBOOL                                  disable_audio_scaling;
}stTC_HDR_BLEND_PARAMS_t;
```

Elements

Member	Description
fm_mps_blend_thresh	Blend threshold for FM MPS program Set the threshold for determining when to blend digital audio and analog audio for FM Hybrid MPS.
fm_all_dig_blend_thresh	Blend threshold for all FM digital programs (MPS/SPS) Set the threshold for determining when to blend digital audio and mute for all FM digital programs (MPS/SPS).
fm_mps_audio_scaling	Level alignment between digital audio and analog audio by scaling down the digital audio of FM Hybrid MPS Used to align the digital and analog audio levels for FM Hybrid MPS. Digital audio will be scaled down to the level specified by this parameter, where 65335 is a factor of 1. Value ranges from 0 to 65535.
fm_all_dig_audio_scaling	Audio scaling for all FM digital programs (MPS/SPS) Used to scale down the digital audio signal for FM SPS programs as well as all digital MPS programs of FM. Digital audio will be scaled down to the level specified by this parameter, where 65335 is a factor of 1. Value ranges from 0 to 65535.
fm_mps_blend_rate	Control the step size of the FM analog hold duration This parameter configures the hysteresis in the blending process. It controls the step size of the analog hold duration. If the state of the blend line is analog, the blend line cannot be switched to digital again until the digital audio quality remains good for the full period of the analog hold duration. The analog hold duration (in seconds) increases by the value of the Blend Rate parameter (X) each time a blend from digital to analog occurs within the current analog hold duration. The analog hold duration is reset to (1.1*X) seconds after the audio output remains digital for longer than 10*X seconds. The maximum analog hold duration (in seconds) is (1.1*X) + 5*X. That is, the hold time is incremented by a maximum of 5 times. **Example:** When a blend rate is 0x05, Step 1 = 5.5 seconds Step 2 = 10.5 seconds Step 3 = 15.5 seconds

Member	Description
	Step 4 = 20.5 seconds Step 5 = 25.5 seconds Step 6 = 30.5 seconds Value ranges from 3 to 6. Recommended value is 3.
fm_all_dig_blend_rate	Control the step size of the FM mute hold duration. Control the minimum amount of time that the audio output remains muted after loss of digital audio signal. This applies to all FM digital programs (MPS/SPS). Allowed values are 1 and 3 to 6. Recommended value is 1. Refer to fm_mps_blend_rate above.
fm_mps_dig_audio_delay	Delay for fine time alignment between digital audio and analog audio of FM Hybrid MPS Used to perform fine time alignment between digital audio and analog audio, to ensure smooth blending. This parameter is utilized for blending whenever FM Hybrid MPS and the primary sample rate are selected.
am_mps_blend_thresh	Blend threshold for AM MPS program Set the threshold for determining when to blend digital audio and analog audio for AM Hybrid MPS.
am_all_dig_blend_thresh	Blend threshold for all AM digital programs (MPS/SPS) Set the threshold for determining when to blend digital audio and mute for all AM digital programs (MPS/SPS).
am_mps_audio_scaling	Fine audio level alignment for AM hybrid MPS Used to align the digital and analog audio levels for AM Hybrid MPS. Digital audio will be scaled down to the level specified by this parameter, where 65335 is a factor of 1. Value ranges from 0 to 65535.
am_all_dig_audio_scaling	Audio scaling for all AM digital programs (MPS/SPS) Used to scale down the digital audio signal for all AM digital programs (MPS/SPS). Digital audio will be scaled down to the level specified by this parameter, where 65335 is a factor of 1. Value ranges from 0 to 65535.
am_mps_dig_audio_delay	Delay for fine time alignment between digital audio and analog audio of AM Hybrid MPS Used to perform fine time alignment between digital audio and analog audio, to ensure smooth blending. This parameter is utilized for blending whenever AM Hybrid MPS and the primary sample rate are selected. Value ranges from 0 to 16383 audio samples.
am_mps_blend_rate	Control the step size of the AM analog hold duration. Value ranges from 3 to 6. Recommended value is 3. Refer to fm_mps_blend_rate above
am_all_dig_blend_rate	Control the step size of the AM mute hold duration. Allowed values are 1 and 3 to 6. Recommended value is 1. Refer to fm_mps_blend_rate above
d2a_blend_holdoff	Digital-to-Analog blend hold-off Control the delay (ranges from 4 to 21 audio frames) of the digital audio samples by adjusting the size of the blend delay buffer. Blend delay buffer is used to ensure that good audio frames are available for *smooth* blending even if the digital audio is lost. The maximum hold-off for FM is 18 audio frames. Any value exceeding this maximum is limited to 18. **Note:** This parameter is write-protected and can be set during idle mode only.
blend_decision	To prevent frequency switching the blend from analog to digital, switching the blend is hold off until the value of this parameter becomes the set value of blend decision threshold. Blend Decision is the ability of the system to look ahead into the future of the incoming signal and make some decisions about the feasibility of having a good signal for audio decoding. This hysteresis prevents the rapid/frequent blending by requiring the system to have a more stable signal condition. In addition to the audio quality, blend threshold, and blend rate, blend decision takes into account the Cd/No values on consecutive audio frames. Normal blend from analog to digital may be delayed depending on the consecutive audio frame Cd/No values. If the blend line changes the state to play the analog in the enabled state, the blend line cannot be switched to digital until the blend decision threshold becomes the set value of blend decision threshold.

Member	Description
	Note: This parameter applies to only MPS hybrid mode.
fm_cdno_blend_decision	The minimum required Cd/No, in FM, on consecutive frames to allow normal blending. Value ranges from 52 to 60 dB-Hz. Recommended value is 58 dB-Hz. **Note:** This parameter applies to only MPS hybrid mode when blend decision (blend look ahead) is enabled.
am_cdno_blend_decision	The minimum required C/No, in AM, on consecutive frames to allow normal blending. Value ranges from 50 to 70 dB-Hz. Recommended value is 67 dB-Hz. **Note:** This parameter applies to only MPS hybrid mode when blend decision (blend look ahead) is enabled.
fm_audio_invert_phase	Invert FM digital audio phase Setting the flag will invert the FM digital audio phase. Sometimes, it is needed to phase-align analog and digital audio during blending.
am_audio_invert_phase	Invert AM digital audio phase Setting the flag will invert the AM digital audio phase. Sometimes, it is needed to phase-align analog and digital audio during blending.
disable_audio_scaling	Disable audio scaling inside the HD Radio library This overwrites all other audio level modifiers to force digital audio output to be always at full scale. This may be useful for cases when blend level alignment is done outside of the HD Radio library.

Comments

None

Header file

tchdr_api.h

4.12 stTC_HDR_BLEND_ADV_PARAMS_t

Description

This is a structure for advanced blend configuration parameters.

Definition

```
typedef struct {
        HDBOOL          ramp_up_enabled;
        U32             ramp_up_time;
        HDBOOL          ramp_down_enabled;
        U32             ramp_down_time;
        HDBOOL          comfort_noise_enabled;
        S32             comfort_noise_level;
        HDBOOL          am_enh_stream_holdoff_enabled;
        U32             am_mps_enh_stream_holdoff_thresh;
        U32             am_all_dig_enh_stream_holdoff_thresh;
        HDBOOL          am_dig_audio_bw_mgmt_enabled;
        U32             am_dig_audio_blend_start_bw;
        U32             am_dig_audio_max_bw;
        U32             am_dig_audio_bw_step_time;
        U32             am_dig_audio_bw_step_up_size;
        U32             am_dig_audio_bw_step_down_size;
        U32             am_dig_audio_bw_step_threshold;
        HDBOOL          am_mono2stereo_enabled;
        U32             am_mono2stereo_start_bw;
        U32             am_mono2stereo_step_time;
        U32             am_mono2stereo_max_sep;
}stTC_HDR_BLEND_ADV_PARAMS_t;
```

Elements

Member	Description
ramp_up_enabled	Enable/Disable digital audio ramp up. When this feature is enabled, a linear ramp will be applied to the digital audio level whenever audio reception resumes from an outage condition. This is similar to blend crossfade used for mixing the digital audio and analog audio in hybrid mode except for the case that mixing is done with mute (zero) samples. **Note:** This parameter applies to only hybrid SPS or all digital programs.
ramp_up_time	Time duration to ramp up audio This parameter controls the time duration, measured in audio frames (46.4 ms), to ramp up the audio from mute after recovery from an outage. Value ranges from 1 to 16. Recommended value is 16 (743 ms). **Note:** This parameter applies to only hybrid SPS or all digital programs. If the ramp_up_enabled is set to false, this parameter does not affect the ramp up time duration.
ramp_down_enabled	Enable/Disable digital audio ramp down. When this feature is enabled, a linear ramp will be applied to the digital audio level whenever audio reception is lost. This is similar to blend crossfade used for mixing the digital audio and analog audio in hybrid mode except for the case that mixing is done with mute (zero) samples. **Note:** This parameter applies to only hybrid SPS or all digital programs.
ramp_down_time	Time duration to ramp down audio This parameter controls the time duration, measured in audio frames (46.4 ms), to ramp down the audio to mute after reception is lost. Value ranges from 1 to 16. Recommended value is 16 (743 ms). **Note:** This parameter applies to only hybrid SPS or all digital programs. If the ramp_down_enabled is set to false, this parameter does not affect the ramp up time duration.

Member	Description
comfort_noise_enabled	Enable/Disable comfort noise. When this parameter is enabled, instead of just playing mute after ramp-down is completed, the radio plays comfort noise (which ramps up to the desired value comfort_noise_level). The ramp up and ramp-down duration are set to 4 audio frames (186 ms). **Note:** This parameter applies to only hybrid SPS or all digital programs. If the blend threshold for all digital programs is set to Q7, this parameter is not applied.
comfort_noise_level	Comfort noise level (dBFS) Range is from -100 to 0 dBFS. Recommended value is -48 dBFS. **Note:** This parameter applies to only hybrid SPS or all digital programs. If the blend threshold for all digital programs is set to Q7, this parameter is not applied. If comfort_noise_enabled is set to false, this parameter does not affect the comfort noise level.
am_enh_stream_holdoff_enabled	Enable/Disable enhanced audio stream hold-off. Enhanced stream is combined with core stream to provide stereo with up to 15 kHz of audio bandwidth. If the receiver is near the edge of enhanced coverage so that the enhanced audio cuts in and out, a situation arises similar to analog/digital blending where audio fluctuates between mono and stereo creating a bad user experience. If this parameter is enabled, a hold-off is applied to enhanced audio under the weak signal conditions, until the signal quality exceeds a certain C/No threshold for a predefined period of time. The predefined period of time is initially set to 5 seconds and can be changed up to 25 seconds. **Note:** This parameter applies to only AM hybrid and all AM digital modes.
am_mps_enh_stream_holdoff_thresh	C/No threshold for enhanced stream hold-off of hybrid programs Under weak signal conditions, a hold-off is applied to enhanced audio until the signal quality exceeds a threshold specified by this parameter. Value ranges from 47 to 80 dB-Hz. Recommended value is 72. **Note:** This parameter applies to only AM hybrid and all AM digital modes. If the am_enh_stream_holdoff_enabled is set to false, this parameter does not affect C/No threshold.
am_all_dig_enh_stream_holdoff_thresh	C/No threshold for enhanced stream hold-off of all digital programs Under weak signal conditions, a hold-off is applied to enhanced audio until the signal quality exceeds a threshold specified by this parameter. Value ranges from 47 to 80 dB-Hz. Recommended value is 72. **Note:** This parameter applies to only AM hybrid and all AM digital modes. If the am_enh_stream_holdoff_enabled is set to false, this parameter does not affect C/No threshold.
am_dig_audio_bw_mgmt_enabled	Enable/Disable AM audio bandwidth management. When the system is changed from analog to digital, the audio BW changes from 3.5-5 kHz to 15 kHz. This is a big jump in bandwidth. If the bandwidth becomes wider, it is perceived as the louder sound by you. Frequent blending will cause a poor user experience. Audio BW matches the digital audio bandwidth to analog and then gradually increases it to the maximum level by eliminating abrupt change. **Note:** This parameter applies to only AM hybrid modes.
am_dig_audio_blend_start_bw	Set AM digital audio bandwidth at blend point. At the blend point (transition from Analog to Digital or vice versa), this will be the bandwidth of the digital audio signal. Range is from 0 to 56, where 0: 1.0 kHz 1: 1.25 kHz 2: 1.5 kHz … 56: 15.0 kHz **Note:** This parameter is applied only if the am_dig_audio_bw_mgmt_enabled is set

Member	Description
	to true.
am_dig_audio_max_bw	Maximum bandwidth of the digital audio signal Range is from 0 to 56. Refer to am_dig_audio_blend_start_bw. Recommended value is 56 (15 kHz). **Note:** This parameter is applied only if the am_dig_audio_bw_mgmt_enabled is set to true.
am_dig_audio_bw_step_time	Step time duration If the value of signal strength exceeds the threshold (am_dig_audio_bw_step_threshold) since the timer is started, the system increases the digital audio bandwidth by one step as defined by am_dig_audio_bw_step_up_size. Otherwise, the system decreases the digital audio bandwidth by one step as defined by am_dig_audio_bw_step_down_size. Range is from 0 to 65535, where 0: 0 ms 1: 46.4 ms 2: 92.9 ms … 65535: 3043.4 seconds Recommended value is 87 (4040.28 ms). **Note:** This parameter is applied only if the am_dig_audio_bw_mgmt_enabled is set to true.
am_dig_audio_bw_step_up_size	Step-up size of bandwidth Size by which the digital audio bandwidth decreases for every step (am_dig_audio_bw_step_time) during blend transition Range is from 1 to 56, where 1: 0.25 kHz 2: 0.5 kHz 3: 1.0 kHz … 56: 14.0 kHz Recommended value is 2 (0.5 kHz). **Note:** This parameter is applied only if the am_dig_audio_bw_mgmt_enabled is set to true.
am_dig_audio_bw_step_down_size	Step-down size of bandwidth Size by which the digital audio bandwidth increases for every step (am_dig_audio_bw_step_time) during blend transition. Range is from 1 to 56. Refer to am_dig_audio_bw_step_up_size. Recommended value is 3 (1.0 kHz). **Note:** This parameter is applied only if the am_dig_audio_bw_mgmt_enabled is set to true.
am_dig_audio_bw_step_threshold	C/No threshold for digital audio bandwidth step This parameter determines the carrier-to-noise threshold used in determining whether to step up/down in digital audio bandwidth. Range is from 47 to 80 dB-Hz. Recommended value is 67 dB-Hz. **Note:** This parameter is applied only if the am_dig_audio_bw_mgmt_enabled is set to true.
am_mono2stereo_enabled	Enable/Disable mono-to-stereo transition. On top of gradual audio bandwidth increase, audio can be gradually switched from mono to stereo to further improve the blend experience. **Note:** This parameter is applied only if the am_dig_audio_bw_mgmt_enabled is set to true.
am_mono2stereo_start_bw	AM mono-to-stereo starting bandwidth Set the starting audio bandwidth at which the blend transition from digital audio

Member	Description
	mono to stereo occurs. If the actual BW falls below the BW set by this parameter, the digital audio is switched from stereo to mono. Range is from 0 to 56. Refer to am_dig_audio_blend_start_bw. Recommended value is 16 (5 kHz). **Note:** This parameter is applied only if the am_dig_audio_bw_mgmt_enabled and am_mono2stereo_enabled are set to true.
am_mono2stereo_step_time	AM mono-to-stereo step duration This is the number of audio frames to wait before making the blend adjustment from the next digital mono to stereo. The adjustment occurs only if the current digital audio BW is greater than the value specified by am_mono2stereo_start_bw. Value ranges from 0 to 16. Recommended value is 8 (372 ms).
am_mono2stereo_max_sep	Maximum stereo separation value Range is from 0 to 16, where 0 = Mono and 16 = Full stereo. Recommended value is 16. **Note:** This parameter is applied only if the am_dig_audio_bw_mgmt_enabled and am_mono2stereo_enabled are set to true.

Comments

None

Header file

tchdr_api.h

4.13 stTC_HDR_AAS_PORT_LIST_t

Description

This structure defines the structure used to specify a list of ports to be enabled or disabled.
This is used for enabling or disabling ports and retrieving information on currently enabled ports.

Definition

```
typedef struct {
        struct{
                U16     number;
                U8      mode;
        }port[TC_HDR_AAS_MAX_NUMBER_OF_PORTS];
                U8      num_ports;
}stTC_HDR_AAS_PORT_LIST_t;
```

Elements

Member	Description
number	Port number
mode	Port mode (ordered or non-ordered) Refer to Chapter 3.31 eTC_HDR_AAS_PORT_MODE_t.
port	Store the port list in array
num_ports	Number of AAS ports in the port list

Comments

None

Header file

tchdr_aas.h

4.14 stTC_HDR_AAS_PACKET_INFO_t

Description

This structure is information associated with a received packet.

Definition

```
typedef struct {
        U32             num_packets_avail;
        HDBOOL          overflow_status;
        U32             port_number;
        U32             sequence_number;
        U32             packet_length;
        U32             num_bytes_unread;
}stTC_HDR_AAS_PACKET_INFO_t;
```

Elements

Member	Description
num_packets_avail	Number of packets that are still available on this port (Not including the packet obtained from this structure)
overflow_status	Indicate whether the port queue overflowed. At least one (oldest) packet is deleted.
port_number	Packet port number (16-bit number)
sequence_number	Packet sequence number
packet_length	Packet length in bytes
num_bytes_unread	Number of bytes that are left to be read. Used when the provided buffer is less than the length of packet.

Comments

None

Header file

tchdr_aas.h

4.15 stTC_HDR_AAS_LOT_OBJECT_LIST_t

Description

This structure is a list of all complete and incomplete LOT objects.

Definition

```
typedef struct {
        struct {
                U16             port_number;
                U16             lot_id;
                U8              complete;
        }item[TC_HDR_MAX_NUM_LOT_OBJECTS];
                U32             num_objects;
}stTC_HDR_AAS_LOT_OBJECT_LIST_t;
```

Elements

Member	Description
port_number	Port number of the LOT object
log_id	LOT ID number
complete	Flag indicating if the object is completed (1 - complete)
item	LOT object list array
num_objects	Number of LOT objects

Comments

None

Header file

tchdr_aas.h

4.16 stTC_HDR_AAS_LOT_OBJECT_HEADER_t

Description

This structure is a LOT object header definition. The set of supported LOT file MIME types and their corresponding hash values are listed in the table below.

Definition

```
typedef struct {
        U32             discard_time;
        U32             file_size;
        U32             mime_hash;
        U8              filename[TC_HDR_AAS_LOT_MAX_FILENAME_LENGTH];
        U8              filename_length;
}stTC_HDR_AAS_LOT_OBJECT_HEADER_t;
```

Elements

Member	Description
discard_time	Year, month, day, hour, and minute in Coordinated Universal Time (UTC) After this discard_time, the receiver can discard the object.
file_size	Object size (in bytes)
mime_hash	Hash values of MIME types in LOT file The size of each hash value in the following table is 4 bytes.
filename	File Name (Maximum length is 231 bytes)
filename_length	Length of actual file name

MIME Type	Hash Value
none	0x806FFF30
text/plain	0xBB492AAC
text/enriched	0x7074B716
image/gif	0x6E1D9F04
image/jpeg	0x1E653E9C
Image/png	0x4F328CA0
video/mpeg	0x761FB167
audio/basic	0x06362BAE

Comments

None

Header file

tchdr_aas.h

4.17 stTC_HDR_LOT_t

Description

This structure is a LOT object definition.
This includes the LOT Object Header(stTC_HDR_AAS_LOT_OBJECT_HEADER_t) and the body of the LOT.

Definition

```
typedef struct {
        U32                                     service_number;
        U32                                     app_mime_hash;
        stTC_HDR_AAS_LOT_OBJECT_HEADER_t        header;
        U8*                                     body;
        U32                                     body_bytes_written;
}stTC_HDR_LOT_t;
```

Elements

Member	Description
service_number	Unique number that identifies a service.
app_mime_hash	Application service type for data **MIME Type** / **Hash Value** Station logo / 0xD9C72536 Album Art / 0xBE4B7536
header	LOT object header. Refer to Chapter 4.16 stTC_HDR_AAS_LOT_OBJECT_HEADER_t
body	Pointer to the body buffer (Example: Album art file body)
body_bytes_written	Size of body data (Example: Album art file size)

Comments

None

Header file

tchdr_aas.h

4.18 stTC_HDR_ALERT_MESSAGE_t

Description

This is an output structure for an HD Radio emergency alert message. Payload includes message ID, message control data (CNT (Control Data)), and text string.

Definition

```
typedef struct {
        U32                                     payload_crc;
        U32                                     payload_length;
        U32                                     cnt_length;
        HDBOOL                                  cnt_crc_pass;
        eTC_HDR_ALERT_TEXT_ENCODING_t           text_encoding;
        U32                                     text_length;
        S8*                                     text_message;
        S8                                      payload[TC_HDR_MAX_ALERT_PYALOAD_LENGTH];
}stTC_HDR_ALERT_MESSAGE_t;
```

Elements

Member	Description
payload_crc	7-bit payload CRC value
payload_length	Total payload length (7 to 381 bytes)
cnt_length	CNT length (in byte pairs)
cnt_crc_pass	CNT CRC status. 12-bit CRC that covers all of the CNT bits
text_encoding	Text encoding of the message string
text_length	Length of text string portion of the payload
text_message	Pointer to text message of the emergency alert. Set to NULL if not applicable.
payload	Payload Length

Comments

None

Header file

tchdr_alert.h

4.19 stTC_HDR_ALERTS_MSG_STATUS_t

Description

This is an output structure for current status of message reception.

Definition

```
typedef struct {
        HDBOOL          frame_received;
        HDBOOL          frame0_available;
        HDBOOL          full_message;
        U32             frame_counter;
        U32             message_id;
        U32             payload_crc;
}stTC_HDR_ALERTS_MSG_STATUS_t;
```

Elements

Member	Description
frame_received	Indicate whether any alert message frame (piece) is received.
frame0_available	Indicate whether frame 0 is received. Frame 0 contains information on the contents of the message.
full_message	Indicate whether full message is received.
frame_counter	Total number of frames that are received
message_id	Unique emergency alert message ID. Range from 0 to 255.
payload_crc	7-bit payload CRC value

Comments

None

Header file

tchdr_alert.h

4.20 stTC_HDR_PSD_FIELDS_t

Description

This structure contains a bitmap used to enable/disable PSD fields. Enabled fields are parsed by the HD Radio library and trigger the PSD change flag when new PSD is received.

Definition

```
typedef union {
        struct{
                U8              title:1;
                U8              artist:1;
                U8              album:1;
                U8              genre:1;
                U8              comment:1;
                U8              UFID:1;
                U8              commercial:1;
                U8              XHDR:1;
        }filed;
                U8              all;
}stTC_HDR_PSD_FIELDS_t;
```

Elements

Member	Description
title	Song title
artist	Artist name
album	Album title
genre	Genre
comment	General comments
UFID	Unique File Identifier
commercial	For advertising purposes
XHDR	Image synchronization trigger
all	Used to read/write all fields at once

Comments

None

Header file

tchdr_psd.h

4.21 stTC_HDR_PSD_FORM_t

Description

This is an output structure for a PSD default format.

Definition

```
typedef struct {
        S8                              data[TC_HDR_PSD_MAX_LEN];
        U32                             len;
        eTC_HDR_PSD_CHAR_TYPE_t         charType;
}stTC_HDR_PSD_FORM_t;
```

Elements

Member	Description
data	Data
len	Length
charType	Character Type

Comments

None

Header file

tchdr_psd.h

4.22 stTC_HDR_PSD_XHDR_PARAM_t

Description

This is an output structure for a XHDR parameter field.

Definition

```
typedef struct {
        U8      param_id;
        U8      length;
        U8      value[TC_HDR_MAX_LEN_XHDR_PARAM_VALUES];
        U16     lot_id;
}stTC_HDR_PSD_XHDR_PARAM_t;
```

Elements

<table>
<tr><th>Member</th><th>Description</th></tr>
<tr><td>param_id</td><td>XHDR parameter
<table><tr><th>ParameterID</th><th>Description</th></tr><tr><td>0x00</td><td>Display trigger for image <LOTID></td></tr><tr><td>0x01</td><td>Blank display</td></tr><tr><td>0x02</td><td>Flush memory</td></tr><tr><td>0x03 to 0xFF</td><td>Reserved for future use</td></tr></table></td></tr>
<tr><td>length</td><td>Length of value</td></tr>
<tr><td>value</td><td>Value data (Maximum: 121 bytes)</td></tr>
<tr><td>lot_id</td><td>Assigned Lot ID</td></tr>
</table>

Comments

None

Header file

tchdr_psd.h

4.23 stTC_HDR_PSD_XHDR_FRAME_t

Description

This is an output structure for XHDR. This structure contains stTC_HDR_PSD_XHDR_PARAM_t.

Definition

```
typedef struct {
        U32                             mime_hash;
        stTC_HDR_PSD_XHDR_PARAM_t       params[TC_HDR_MAX_NUM_XHDR_PARAM];
        U32                             numParams;
        U8                              program;
}stTC_HDR_PSD_XHDR_FRAME_t;
```

Elements

Member	Description
mime_hash	Application MIME hash (the same as SIG MIME hash value) Refer to **app_mime_hash** in stTC_HDR_LOT_t.
params	Refer to Chapter 4.22 stTC_HDR_PSD_XHDR_PARAM_t (If it consists of params field without a value, the maximum number of params field is 61.)
numParams	Number of XHDR params fields
program	Number of the currently programs (MPS/SPS)

Comments

None

Header file

tchdr_psd.h

4.24 stTC_HDR_PSD_t

Description

This is an output structure for information on PSD default field. This is used as the callback notification for demo.

Definition

```
typedef struct {
        stTC_HDR_PSD_FORM_t             title;
        stTC_HDR_PSD_FORM_t             artist;
        stTC_HDR_PSD_FORM_t             album;
        stTC_HDR_PSD_FORM_t             genre;

        struct {
            stTC_HDR_PSD_FORM_t         language;
            stTC_HDR_PSD_FORM_t         shortContent;
            stTC_HDR_PSD_FORM_t         actualText;
        }comment;

        struct {
            stTC_HDR_PSD_FORM_t         priceString;
            stTC_HDR_PSD_FORM_t         validUntil;
            stTC_HDR_PSD_FORM_t         contactURL;
            stTC_HDR_PSD_FORM_t         receivedAs;
            stTC_HDR_PSD_FORM_t         sellerName;
            stTC_HDR_PSD_FORM_t         description;
        }commercial;

        stTC_HDR_PSD_XHDR_FRAME_t               xhdr;
}stTC_HDR_PSD_t;
```

Elements

Member	Description
title	Song title
artist	Artist name
album	Album title
genre	Genre
comment	Comment structure
commercial	Commercial structure
xhdr	Display synchronization trigger

Comments

None

Header file

tchdr_psd.h

4.25 stTC_HDR_SIG_SERVICE_LIST_t

Description

This structure contains a list of available services and some additional information on the service.

Note: A change in the receive time does not affect the status.

Definition

```
typedef struct{
        struct {
                U32                             service_number;
                U32                             receive_time;
                eTC_HDR_SIG_STATUS_t            status;
        }item[TC_HDR_MAX_NUM_SIG_SERVICES_PER_STATION];
                U32                             num_services;
}stTC_HDR_SIG_SERVICE_LIST_t;
```

Elements

Member	Description
service_number	Uniquely identify a service
receive_time	ALFN when the service is received.
status	Update status
item	SIG service list array
num_services	Number of services

Comments

None

Header file

tchdr_sig.h

4.26 stTC_HDR_SIG_SERVICE_INFO_t

Description

This is a structure for the output data of SIG service information.

Definition

```
typedef struct {
        eTC_HDR_SIG_SERVICE_TYPE_t      type;
        U32                             service_number;
        U32                             priority;
        U32                             sequence_number;
        eTC_HDR_SIG_STATUS_t            status;
        U32                             receive_time;
        U32                             provider_text_encoding;
        U32                             provider_name_length;
        S8                              provider_name[TC_HDR_MAX_SERVICE_PROVIDER_NAME_LENGTH];
        U32                             display_text_encoding;
        U32                             display_name_length;
        S8                              display_name[TC_HDR_MAX_SERVICE_DISPLAY_NAME_LENGTH];
        U32                             num_components;
}stTC_HDR_SIG_SERVICE_INFO_t;
```

Elements

Member	Description
type	Identify whether it is audio or data service.
service_number	Unique number that identifies a service
priority	Indicate the priority of the specified service.
sequence_number	Indicate updated number on the specified service.
status	Indicate the update status of the service information.
receive_time	Indicate the precise time (ALFN) that the information is received.
provider_text_encoding	Text encoding type used for the service provider name
provider_name_length	Length of the service provider name
provider_name	Service provider name text
display_text_encoding	Text encoding type used for the service display name
display_name_length	Length of service display name
display_name	Service display name text
num_components	Number of service components

Comments

None

Header file

tchdr_sig.h

4.27 stTC_HDR_SIG_SERVICE_COMPONENT_t

Description

This structure defines the service component structure. Each service has at least one component (component 0) which is the anchor component for the service.

Definition

```
typedef struct {
        eTC_HDR_SIG_COMPONENT_TYPE_t     component_type;
        U32                              component_number;
        U32                              channel;
        U32                              content_type;
        U32                              processing;
        U32                              priority;
        U32                              access_rights;
        U32                              mime_hash_value;
        U32                              provider_id;
        U32                              service_id;
        U32                              expanded_id_length;
        U8                               expanded_service_id[TC_HDR_MAX_EXPANDED_SERVICE_ID_SIZE];
}stTC_HDR_SIG_SERVICE_COMPONENT_t;
```

Elements

Member	Description
component_type	Component type, audio, or data
component_number	Component ID number within the service
channel	Correspond to audio program if component type is audio, or correspond to port number if the component type is data.
content_type	If audio defines audio content (Example: News, Talk, Rock, etc.) or if data defines service data type (Example: News, Sports, Traffic, etc.)
processing	If the type of component is audio, this specifies the applied sound experience. The applied sound experience field is applied to further processing of the audio material beyond the channel-related audio encoding and decoding for transport purposes. If the type of component is data, this field specifies the data processing method as follows: ■ 0: Radio Link Subsystem (RLS) Byte Streaming ■ 1: RLS Packet ■ 2: Reserved - Not used ■ 3: LOT - Packet
priority	Indicate the priority of this component within the service record.
access_rights	Data service-only Indication: Indicate whether the data packet is scrambled or not.
mime_hash_value	This hash value indicates the application to which the service component information obtained by this structure may be applied.
provider_id	Part of the unique ID that is provided to identify the source of the service component that is read by this structure. **Note:** This parameter is applicable to data services only.
service_id	Part of the unique ID that is provided to identify the service component that is read by this structure. **Note:** This parameter is applicable to data services only.
expanded_id_length	Length of the entire service identifier Include service provider ID, service ID, and expanded service identifier.
expanded_service_id	The expanded service identifier is available when the service identifier information requires more than 8 bytes that can be used for Service Provider ID and Service ID fields.

Comments

None

Header file

tchdr_sig.h

4.28 stTC_HDR_SIS_ENABLED_BASIC_TYPES_t

Description

This is a structure used for enabling/disabling basic SIS data types. Station ID, Station Short Name, Station Long Name, and Station Location are considered basic SIS data types. All others are extended SIS data.

Definition

```
typedef union {
        struct {
                U8              stationId:1;
                U8              shortName:1;
                U8              location:1;
        }type;
                U8              all;
}stTC_HDR_SIS_ENABLED_BASIC_TYPES_t;
```

Elements

Member	Description
stationId	Station ID
shortName	Station Name (Short Form)
location	Station Location
all	Control all bits at once.

Comments

None

Header file

tchdr_sis.h

4.29 stTC_HDR_SIS_ALFN_t

Description

This is an Absolute Layer 1 Frame Number (ALFN) output data structure.

Definition

```
typedef struct {
        U32                             value;
        eTC_HDR_SIS_ALFN_STATUS_t       status;
}stTC_HDR_SIS_ALFN_t;
```

Elements

Member	Description
value	ALFN value
status	Update status

Comments

None

Header file

tchdr_sis.h

4.30 stTC_HDR_SIS_STATION_ID_t

Description

This is a station ID structure.

Definition

```
typedef struct {
        union {
                struct {
                U32             country_code:10;
                U32             reserved:3;
                U32             fcc_facility_id:19;
                }field;
                U32             all;
        }id;
        eTC_HDR_SIS_STATUS_t    status;
}stTC_HDR_SIS_STATION_ID_t;
```

Elements

Member	Description
country_code	Binary representation of ISO 3166-1-alpha-2 Country Names and Code Elements
reserved	Bits not used
fcc_facility_id	Unique Facility ID assigned by the FCC (USA only)
all	Control all bits at once
status	Update status

Comments

None

Header file

tchdr_sis.h

4.31 stTC_HDR_SIS_SHORT_NAME_t

Description

This is an output data structure for station short name of SIS. The text is null-terminated and length value does not include the null character, matching output of strlen().

Definition

```
typedef struct {
        S8                      text[TC_HDR_SIS_SHORT_NAME_MAX_LEN];
        U32                     length;
        eTC_HDR_SIS_STATUS_t    status;
}stTC_HDR_SIS_SHORT_NAME_t;
```

Elements

Member	Description
text	Short name
length	Length of short name
status	Update status

Comments

None

Header file

tchdr_sis.h

4.32 stTC_HDR_SIS_STATION_LOCATION_t

Description

This is an output data structure for station location. This structure defines the absolute three-dimensional location of the feed point of the broadcast antenna. Location information may be used by the receiver for position determination.

Definition

```
typedef struct {
        S32                     latitude;
        S32                     longitude;
        U32                     altitude;
        eTC_HDR_SIS_STATUS_t    status;
}stTC_HDR_SIS_STATION_LOCATION_t;
```

Elements

Member	Description
latitude	Station Latitude (S19.13 format) Latitude and longitude are both in identical fractional format. The Least Significant Bit (LSB) is equal to 1/8192 degrees. The sign bit indicates the hemisphere, where positive latitude values represent positions north of the equator and negative values represent positions south of the equator. Permissible latitude values are between -90 and +90. Anything outside of these ranges is invalid.
longitude	Station Longitude (S19.13 format) Latitude and longitude are both in identical fractional format. The LSB is equal to 1/8192 degrees. Positive latitude values represent positions north of the equator. Positive longitudes are in the eastern hemisphere. Permissible longitude values are between -180 and +180. Anything outside of these ranges is invalid.
altitude	Altitude of the station Altitude is in units of meters with resolution of 16 meters.
status	Update status

Comments

None

Header file

tchdr_sis.h

4.33 stTC_HDR_SIS_LEAP_SEC_t

Description

This structure defines data related to the leap second. To keep the leap second synchronized with astronomical time, the leap second correction factor is applied to UTC when there is a difference in time by 0.9 seconds between leap second and astronomical time. Since this leap second correction factor is not applied to GPS time, the GPS time and UTC time have diverged slightly over the years. Receivers can calculate GPS time by using the ALFN and then use the correction to calculate UTC.

Definition

```
typedef struct {
        S8                      pending_offset;
        S8                      current_offset;
        U32                     pending_offset_alfn;
        eTC_HDR_SIS_STATUS_t    status;
}stTC_HDR_SIS_LEAP_SEC_t;
```

Elements

Member	Description
pending_offset	Pending leap second offset
current_offset	Current leap second offset
pending_offset_alfn	ALFN of pending leap second offset
status	Update status

Comments

None

Header file

tchdr_sis.h

4.34 stTC_HDR_SIS_STATION_MSG_t

Description

This is an output data structure for station message of SIS. The text is null-terminated and length value does not include the null character, matching output of strlen().

Definition

```
typedef struct {
        S8                              text[TC_HDR_SIS_STATION_MESSAGE_MAX_LEN];
        U32                             length;
        eTC_HDR_SIS_TEXT_ENCODING_t     text_encoding;
        HDBOOL                          high_priority;
        eTC_HDR_SIS_STATUS_t            status;
}stTC_HDR_SIS_STATION_MSG_t;
```

Elements

Member	Description
text	Message buffer
length	Message length
text_encoding	Text encoding type
high_priority	Specify whether the message is sent with high priority.
status	Update status

Comments

None

Header file

tchdr_sis.h

4.35 stTC_HDR_SIS_LOCAL_TIME_t

Description

This defines the structure for retrieving station local time data. Receivers can automatically calculate and display the time of day by using the Local time zone and Daylight Savings Time (DST) information.

Definition

```
typedef struct {
        S32                             utc_offset;
        eTC_HDR_SIS_DST_SCHEDULE_t      dst_schedule;
        eTC_HDR_SIS_DST_LOCAL_t         dst_local;
        HDBOOL                          dst_in_effect;
        eTC_HDR_SIS_STATUS_t            status;
}stTC_HDR_SIS_LOCAL_TIME_t;
```

Elements

Member	Description
utc_offset	Time zone value offset from UTC Store a signed integer in minutes relative to UTC, assuming DST is not in effect.
dst_schedule	Indicate the period over which DST is in effect.
dst_local	Indicate whether DST is practiced locally.
dst_in_effect	Indicate whether DST is in effect within a broad region (Example: country) at a particular time.
status	Update status

Comments

None

Header file

tchdr_sis.h

4.36 stTC_HDR_SIS_UNIV_NAME_t

Description

This is an output data structure for station universal name of SIS. The text is null-terminated and length value does not include the null character, matching output of strlen().

Definition

```
typedef struct {
        S8                              text[TC_HDR_SIS_UNIV_NAME_MAX_LEN];
        U32                             length;
        eTC_HDR_SIS_TEXT_ENCODING_t     text_encoding;
        HDBOOL                          append_fm;
        eTC_HDR_SIS_STATUS_t            status;
}stTC_HDR_SIS_UNIV_NAME_t;
```

Elements

Member	Description
text	Universal name buffer
length	Universal name length
text_encoding	Text encoding type
append_fm	Indicate whether *-FM* should be appended to the short station.
status	Update status

Comments

None

Header file

tchdr_sis.h

4.37 stTC_HDR_SIS_STATION_SLOGAN_t

Description

This is an output data structure for station slogan of SIS. The text is null-terminated and length value does not include the null character, matching output of strlen().

Definition

```
typedef struct {
        S8                              text[TC_HDR_SIS_SLOGAN_MAX_LEN];
        U32                             length;
        eTC_HDR_SIS_TEXT_ENCODING_t     text_encoding;
        eTC_HDR_SIS_STATUS_t            status;
}stTC_HDR_SIS_STATION_SLOGAN_t;
```

Elements

Member	Description
text	Slogan buffer
length	Slogan length
text_encoding	Text encoding type
status	Update status

Comments

None

Header file

tchdr_sis.h

4.38 stTC_HDR_SIS_AVAIL_PROGRAMS_t

Description

This structure is a list of available audio programs reported by SIS.

Definition

```
typedef struct {
        struct {
                eTC_HDR_PROGRAM_t               program_number;
                eTC_HDR_SIS_STATUS_t            status;
        }program[TC_HDR_MAX_NUM_PROGRAMS];
                U32                             program_count;
}stTC_HDR_SIS_AVAIL_PROGRAMS_t;
```

Elements

Member	Description
program_number	Program number
status	Update status
program_count	Total number of available programs

Comments

None

Header file

tchdr_sis.h

4.39 stTC_HDR_SIS_PROGRAM_INFO_t

Description

This structure is audio program information reported by SIS.

Definition

```
typedef struct {
        U32                             program_type;
        U32                             surround_sound;
        eTC_HDR_SIS_ACCESS_TYPE_t       access;
        eTC_HDR_SIS_STATUS_t            status;
}stTC_HDR_SIS_PROGRAM_INFO_t;
```

Elements

Member	Description
program_type	Program type (Example: News, Talk, Information, etc.)
surround_sound	Applied sound experience
access	Program permissions assigned by the broadcaster
status	Update status

Comments

None

Header file

tchdr_sis.h

4.40 stTC_HDR_SIS_AVAIL_DATA_SERVICES_t

Description

This structure is a list of the available data services reported by SIS.

Definition

```
typedef struct {
        struct {
                U32                             type;
                eTC_HDR_SIS_STATUS_t            status;
        }service[TC_HDR_SIS_MAX_NUM_DATA_SERVICES];
                U32                             service_count;
}stTC_HDR_SIS_AVAIL_DATA_SERVICES_t;
```

Elements

Member	Description
type	Data service type ID
status	Data service status (freshness)
service	Data service array
service_count	Number of available data services

Comments

None

Header file

tchdr_sis.h

4.41 stTC_HDR_SIS_DATA_SERVICES_INFO_t

Description

This structure is for the information of data services.

Definition

```
typedef struct {
        struct {
                U16                             service_type;
                U16                             mime_type;
                eTC_HDR_SIS_ACCESS_TYPE_t       access;
                U8                              status;
        }service[TC_HDR_SIS_MAX_NUM_DATA_SERVICES];
                U32                             service_count;
}stTC_HDR_SIS_DATA_SERVICES_INFO_t;
```

Elements

Member	Description
service_type	Indicate the service data type (Example: News, Traffic, Weather, etc.)
mime_type	MIME type hash value specifying the data application program type
access	Program permissions assigned by the broadcaster 0: Public/Unrestricted 1: Restricted
status	Update status 1: Old 2: New (updated)
service_count	Total number of available data services

Comments

None

Header file

tchdr_sis.h

4.42 stTC_HDR_SIS_TX_VER_STR_t

Description

This is a structure that contains exciter core version data. The text string contains ISO/IEC 8859-1 character codes within the range from 32 to 126 only. The string is null-terminated and length value does not include the null character, matching the output of strlen().

Definition

```
typedef struct {
        S8              verstr[TC_HDR_SIS_CORE_VER_STR_MAX_LEN];
        U32             length;
}stTC_HDR_SIS_TX_VER_STR_t;
```

Elements

Member	Description
verstr	Text string containing exciter core version
length	String length

Comments

None

Header file

tchdr_sis.h

4.43 stTC_HDR_SIS_TX_MANUF_VER_t

Description

This is a structure that contains exciter core version string and manufacturer ID. Text string contains ISO/IEC 8859-1 character codes within the range from 32 to 126 only.

Definition

```
typedef struct {
        U8                              right_most_mnf_id;
        U8                              left_most_mnf_id;
        stTC_HDR_SIS_TX_VER_STR_t       version_string;
}stTC_HDR_SIS_TX_MANUF_VER_t;
```

Elements

Member	Description
right_most_mnf_id	Rightmost exciter manufacturer ID character
left_most_mnf_id	Leftmost exciter manufacturer ID character
version_string	Version string

Comments

None

Header file

tchdr_sis.h

4.44 stTC_HDR_SIS_t

Description

This is an output structure for the default information of SIS. This is used as the callback notification for demo.

Definition

```
typedef struct {
        union{
        struct {
                U32                             countryCode:10;
                U32                             reserved:3;
                U32                             facilityID:19;
                }type;
                U32                             all;
        }stationID;

        struct {
                S8                              text[TC_HDR_SIS_SHORT_NAME_MAX_LEN];
                U32                             len;
        }shortName;

        struct {
                S8                              text[TC_HDR_SIS_UNIV_NAME_MAX_LEN];
                U32                             len;
                eTC_HDR_SIS_TEXT_ENCODING_t     charType;
                U32                             appendFm;
        }universalName;

        struct {
                S8                              text[TC_HDR_SIS_SLOGAN_MAX_LEN];
                U32                             len;
                eTC_HDR_SIS_TEXT_ENCODING_t     charType;
        }slogan;
}stTC_HDR_SIS_t;
```

Elements

Member	Description
stationID	Structure of station ID
shortName	Structure of short name
universalName	Structure of universal name
slogan	Structure of slogan

Comments

None

Header file

tchdr_sis.h

5 API FUNCTIONS

Table 5.1 describes API functions of the HD Radio.

Table 5.1 API Functions

Functions	Description
tchdr_init	Initialize HD Radio
tchdr_deinit	De-initialize HD Radio
tchdr_open	Open HD Radio
tchdr_close	Close HD Radio
tchdr_setTune	Set Tune
tchdr_setAudioMode	Set audio mode
tchdr_setProgram	Set program
tchdr_getProgram	Get program
tchdr_getSignalStatus	Get signal status
tchdr_getAllStatus	Get all status
tchdr_enablePsdNotification	Enable PSD notification
tchdr_enableSisNotification	Enable SIS notification
tchdr_enableLotNotification	Enable LOT notification
tchdr_enableAlertNotification	Enable Emergency Alert notification
tchdr_setAudioMute	Set audio mute
tchdr_setAudioCtrl	Control Audio output
tchdr_setAnalogAudioMute	Set analog audio mute
tchdr_setAudioMuteFader	Set fader parameters of audio mute
tchdr_getAudioMuteFader	Get fader parameters of audio mute
tchdr_getAvailablePrograms	Get available program
Tchdr_getProgramType	Get program type of the all program
tchdr_setAutoAudioAlignEnable	Set to enable or disable the automatic audio alignment function
tchdr_setBlendTransitionTime	Set blend transition time
tchdr_setBlendAllParams	Set all blend parameters at once
tchdr_getBlendAllParams	Get all blend parameters at once
tchdr_setBlendParam	Set individual blend parameter
tchdr_getBlendParam	Get individual blend parameter
tchdr_setBlendAllAdvParams	Set all advanced blend parameters at once
tchdr_getBlendAllAdvParams	Get all advanced blend parameters at once
tchdr_setBlendAdvParam	Set individual advanced blend parameter
tchdr_getBlendAdvParam	Get individual advanced blend parameter
tchdr_getFrameworkVersionString	Get HD Radio framework version string
tchdr_getLibraryVersionString	Get HD Radio library version string
tchdr_setThreadsPriority	Set the priority values of the HD Radio framework threads
tchdr_getDefaultThreadsPriority	Get the default priority values of the HD Radio framework threads
Tchdr_getDefaultThreadNicePriority	Get the default nice priority values of the HD Radio framework threads
tchdr_getThreadsPriority	Get the current priority values of the HD Radio framework threads
tchdr_configTunerIQ01Driver	A function to register the IQ01 callback function
tchdr_configTunerIQ23Driver	A function to register the IQ23 callback function
tchdr_configTunerBlendAudioDriver	A function to register the blend audio callback function
tchdr_configTcHdrNotificationCallBack	A function to register the notification callback function
tchdr_configTcHdrAudioQueueCallBack	A function to register the audio queue callback function
tchdr_cb_getIqSampleRate	A callback function to get IQ sample rate
tchdr_cb_setTune	A callback function to set tune
tchdr_aas_enablePorts	Enable specified ports
tchdr_aas_disablePorts	Disable specified ports
tchdr_aas_disableAllPorts	Disable all enabled ports
tchdr_aas_getEnabledPorts	Report information about currently enabled AAS ports
tchdr_aas_getNextPortData	Retrieve the next available packet from any enabled port
tchdr_aas_getPortData	Retrieve the next available packet from the specified port number
tchdr_aas_flushPort	Flush all the data waiting on the specified port number
tchdr_aas_flushAllPorts	Flush all the data waiting on all enabled ports
tchdr_aas_getLotPoolSize	Return total LOT memory pool size (in bytes)
tchdr_aas_getLotSpaceLeft	Return LOT memory (in bytes) left in the pool
tchdr_aas_lotOverflow	Detect if overflow condition occurred
tchdr_aas_enableLotReassembly	Enable the specified port number and initiate LOT reassembly of all LOT objects

Functions	Description
	received on that port
tchdr_aas_disableLotReassembly	Stop LOT reassembly for the specified service number and port number, or all enabled ports for that service number
tchdr_aas_getLotObjectList	Get the current list of all complete and incomplete objects for the specified service number
tchdr_aas_getLotObjectListByName	Get the current list of all complete and incomplete objects for the service number whose file names match the requested file name
tchdr_aas_getLotObjectHeader	Get the header of an object
tchdr_aas_getLotObjectBody	Get the next block of data from the body of the LOT object
tchdr_aas_flushLotObject	Flush an object from LOT memory
tchdr_alert_getMessage	Retrieve the latest emergency alert message
tchdr_alert_getMessageStatus	Provide current status of message reception
tchdr_alert_clearMessageStatus	Clear message status bits
tchdr_psd_getChangedPrograms	Retrieve PSD content change flags
tchdr_psd_clearChangedProgram	Clear PSD content change flag for the specified program
tchdr_psd_enableFields	Specify PSD fields to be processed
tchdr_psd_getEnabledFields	Return currently enabled PSD fields
tchdr_psd_setMaxLength	Truncate a specified PSD field/subfield to a user defined length
tchdr_psd_resetMaxLength	Reset the PSD field/subfield to maximum length
tchdr_psd_getMaxLength	Return the currently set maximum field length
tchdr_psd_getTitle	Get PSD title data
tchdr_psd_getArtist	Get PSD artist data
tchdr_psd_getAlbum	Get PSD album data
tchdr_psd_getGenre	Get PSD genre data
tchdr_psd_getComment	Get PSD comment field/subfield data
tchdr_psd_getUfid	Get PSD UFID field/subfield data
tchdr_psd_getCommercial	Get PSD commercial field/subfield data
tchdr_psd_getXhdr	Get PSD UFID field/subfield data
tchdr_sig_getServiceList	Retrieve a list of available services of the specified type
tchdr_sig_getServiceInfo	Retrieve service information specified by the service number ID
tchdr_sig_getServiceComponent	Retrieve service component data
tchdr_sig_flushAll	Flush all stored information and restarts SIG information retrieval
tchdr_sis_acquired	Return status of SIS data reception
tchdr_sis_crcOk	Instantaneous indication of SIS CRC status
tchdr_sis_enableBasicTypes	Enable user specified SIS basic data types
tchdr_sis_getEnabledBasicTypes	Get the information about enabled basic SIS data types
tchdr_sis_getBlockCount	Return the SIS block count
tchdr_sis_timeGpsLocked	Return status of GPS lock for transmit-site
tchdr_sis_getAlfn	Retrieve Absolute Layer 1 Frame Number (ALFN)
tchdr_sis_getStationID	Retrieve the station ID
tchdr_sis_getStationShortName	Retrieve the station name (short form)
tchdr_sis_getStationLocation	Retrieve station location
tchdr_sis_getLeapSec	Retrieve SIS leap second information
tchdr_sis_getStationMessage	Retrieve SIS station message
tchdr_sis_getLocalTime	Retrieve the SIS local time
tchdr_sis_getUniversalName	Retrieve the universal short name of station
tchdr_sis_getAvailProgramsList	Retrieve a list of available audio programs
tchdr_sis_getStationSlogan	Retrieve the station slogan
tchdr_sis_getProgramInfo	Retrieve specified audio program information
tchdr_sis_getAvailDataServList	Retrieve a list of available data services
tchdr_sis_getAllDataServices	Retrieve the information about all data services
tchdr_sis_getDataServicesType	Retrieve specified data service information
tchdr_sis_getExciterCoreVer	Return pointer to the exciter core version string
tchdr_sis_getExciterManufVer	Return pointer to the exciter manufacturer string
tchdr_sis_getImporterCoreVer	Return pointer to the importer core version string
tchdr_sis_getImporterManufVer	Return pointer to the manufacturer version string
tchdr_sis_flush	Flush out the existing SIS data and start collecting SIS data afresh

5.1 tchdr_init

Description

This function creates a thread for the HD radio service and initializes the related systems.

Definition

```
HDRET tchdr_init(stTC_HDR_CONF_t confSet)
```

Parameters

Parameter	I/O	Description
confSet	I	Initial configuration of HD Radio. Refer to Chapter 4.3 stTC_HDR_CONF_t.

Returns

Return Value	Description
HDRET	Return value Refer to Chapter 3.1 eTC_HDR_RET_t.

Comments

None

5.2 tchdr_deinit

Description

This function terminates the thread for the HD radio service and deinitializes the associated system.

Definition

```
HDRET tchdr_deinit(void)
```

Returns

Return Value	Description
HDRET	Return value Refer to Chapter 3.1 eTC_HDR_RET_t.

Comments

None

5.3 tchdr_open

Description

This function prepares the HD radio functions via the initialization of the IQ and audio drivers.

Definition

```
HDRET tchdr_open(stTC_HDR_TUNE_INFO_t tuneInfo)
```

Parameters

Parameter	I/O	Description
tuneInfo	I	Refer to Chapter 4.5 stTC_HDR_TUNE_INFO_t.

Returns

Return Value	Description
HDRET	Return value Refer to Chapter 3.1 eTC_HDR_RET_t.

Comments

You cannot set eTC_HDR_IDLE_BAND when you open HD Radio framework. At this time, only FM or AM can be selected.

5.4 tchdr_close

Description

This function closes the peripheral drivers and demod.

Definition

```
HDRET tchdr_close(void)
```

Returns

Return Value	Description
HDRET	Return value Refer to Chapter 3.1 eTC_HDR_RET_t.

Comments

None

5.5 tchdr_setTune

Description

This function sets the tune of frequency of HD Radio.

Definition

```
HDRET tchdr_setTune(eTC_HDR_ID_t id, stTC_HDR_TUNE_TO_t tuneTo)
```

Parameters

Parameter	I/O	Description
id	I	Refer to Chapter 3.3 eTC_HDR_ID_t.
tuneTo	I	Refer to Chapter 4.4 stTC_HDR_TUNE_TO_t.

Returns

Return Value	Description
HDRET	Return value Refer to Chapter 3.1 eTC_HDR_RET_t.

Comments

None

5.6 tchdr_setAudioMode

Description

This function sets the audio output source.

Definition

```
HDRET tchdr_setAudioMode(eTC_HDR_AUDIO_MODE_t audioMode)
```

Parameters

Parameter	I/O	Description
audioMode	I	Refer to Chapter 3.7 eTC_HDR_AUDIO_MODE_t.

Returns

Return Value	Description
HDRET	Return value Refer to Chapter 3.1 eTC_HDR_RET_t.

Comments

None

5.7 tchdr_setProgram

Description

This function sets the program.

Definition

```
HDRET tchdr_setProgram(eTC_HDR_ID_t id, eTC_HDR_PROGRAM_t numOfProgram);
```

Parameters

Parameter	I/O	Description
id	I	Refer to Chapter 3.3 eTC_HDR_ID_t.
numOfProgram	I	Refer to Chapter 3.8 eTC_HDR_PROGRAM_t.

Returns

Return Value	Description
HDRET	Return value Refer to Chapter 3.1 eTC_HDR_RET_t.

Comments

None

5.8 tchdr_getProgram

Description

This function gets the current playing program.

Definition

```
HDRET tchdr_getProgram(eTC_HDR_ID_t id, eTC_HDR_PROGRAM_t *numOfProgram)
```

Parameters

Parameter	I/O	Description
id	I	Refer to Chapter 3.3 eTC_HDR_ID_t.
numOfProgram	O	Refer to Chapter 3.8 eTC_HDR_PROGRAM_t.

Returns

Return Value	Description
HDRET	Return value Refer to Chapter 3.1 eTC_HDR_RET_t.

Comments

None

5.9 tchdr_getSignalStatus

Description

This function gets the signal status of the current frequency.

Definition

```
HDRET tchdr_getSignalStatus(eTC_HDR_ID_t id, stTC_HDR_SIGNAL_STATUS_t *dataOut)
```

Parameters

Parameter	I/O	Description
id	I	Refer to Chapter 3.3 eTC_HDR_ID_t.
dataOut	O	Refer to Chapter 4.6 stTC_HDR_SIGNAL_STATUS_t.

Returns

Return Value	Description
HDRET	Return value Refer to Chapter 3.1 eTC_HDR_RET_t.

Comments

None

5.10 tchdr_getAllStatus

Description

This function gets all status of the HD Radio.

Definition

```
HDRET tchdr_getAllStatus(eTC_HDR_ID_t id, stTC_HDR_STATUS_t *dataOut)
```

Parameters

Parameter	I/O	Description
id	I	Refer to Chapter 3.3 eTC_HDR_ID_t.
dataOut	O	Refer to Chapter 4.8 stTC_HDR_STATUS_t.

Returns

Return Value	Description
HDRET	Return value Refer to Chapter 3.1 eTC_HDR_RET_t.

Comments

None

5.11 tchdr_enablePsdNotification

Description

This function enables the callback function to receive notification when PSD basic information changes.

Definition

```
HDRET tchdr_enablePsdNotification(eTC_HDR_ID_t id, U8 progBitmask, U8 psdBitmask, U32 fEn)
```

Parameters

Parameter	I/O	Description
id	I	Refer to Chapter 3.3 eTC_HDR_ID_t.
progBitmask	I	Refer to Chapter 4.7 stTC_HDR_PROG_BITMAP_t.
psdBitmask	I	Refer to Chapter 4.20 stTC_HDR_PSD_FIELDS_t and Chapter 3.20 eTC_HDR_PSD_BITMASK_t.
fEn	I	Enable/Disable notification.

Returns

Return Value	Description
HDRET	Return value Refer to Chapter 3.1 eTC_HDR_RET_t.

Comments

It is the function for **demo** application.

5.12 tchdr_enableSisNotification

Description

This function enables the callback function to receive notification when SIS basic information changes.

Definition

```
HDRET tchdr_enableSisNotification(eTC_HDR_ID_t id, U32 sisBitmask, U32 fEn)
```

Parameters

Parameter	I/O	Description
id	I	Refer to Chapter 3.3 eTC_HDR_ID_t.
sisBitmask	I	Refer to Chapter 3.30 eTC_HDR_SIS_BITMASK_t.
fEn	I	Enable/Disable notification.

Returns

Return Value	Description
HDRET	Return value Refer to Chapter 3.1 eTC_HDR_RET_t.

Comments

It is the function for **demo** application.

5.13 tchdr_enableLotNotification

Description

This function enables the callback function to receive notification when LOT basic information changes.

Definition

```
HDRET tchdr_enableLotNotification(eTC_HDR_ID_t id, U8 progBitmask, U32 fEn);
```

Parameters

Parameter	I/O	Description
id	I	Refer to Chapter 3.3 eTC_HDR_ID_t.
progBitmask	I	bit value of program. 0x01 MPS 0x02 SPS1 0x04 SPS2 0x08 SPS3 0x10 SPS4 0x20 SPS5 0x40 SPS6 0x80 SPS7 0xFF MPS, SPS 1to 7
fEn	I	Enable/Disable notification

Returns

Return Value	Description
HDRET	Return value Refer to Chapter 3.1 eTC_HDR_RET_t.

Comments

None

5.14 tchdr_enableAlertNotification

Description

This function enables the callback function to receive notification when *Emergency Alert* basic information changes.

Definition

```
HDRET tchdr_enableAlertNotification(eTC_HDR_ID_t id, U32 fEn)
```

Parameters

Parameter	I/O	Description
id	I	Refer to Chapter 3.3 eTC_HDR_ID_t.
fEn	I	Enable/Disable notification

Returns

Return Value	Description
HDRET	Return value Refer to Chapter 3.1 eTC_HDR_RET_t.

Comments

None

5.15 tchdr_setAudioMute

Description

This function sets the audio mute.

Definition

```
HDRET tchdr_setAudioMute(U32 fOnOff)
```

Parameters

Parameter	I/O	Description
fOnOff	I	Audio Mute On/Off 0: Mute Off 1: Mute On

Returns

Return Value	Description
HDRET	Return value Refer to Chapter 3.1 eTC_HDR_RET_t.

Comments

None

5.16 tchdr_setAudioCtrl

Description

This function sets the audio output control.

Definition

```
HDRET tchdr_setAudioCtrl(U32 fStartStop)
```

Parameters

Parameter	I/O	Description
fStartStop	I	Start/Stop audio 0: Stop 1: Start

Returns

Return Value	Description
HDRET	Return value Refer to Chapter 3.1 eTC_HDR_RET_t.

Comments

The registered (*pfnAudioQueueCallBack)() function is not called back while the audio output control is in stopped state.

5.17 tchdr_setAnalogAudioMute

Description

This function sets the analog audio mute.

Definition

```
HDRET tchdr_setAnalogAudioMute(U32 fOnOff)
```

Parameters

Parameter	I/O	Description
fOnOff	I	Mute On/Off 0: Mute Off 1: Mute On

Returns

Return Value	Description
HDRET	Return value Refer to Chapter 3.1 eTC_HDR_RET_t.

Comments

None

5.18 tchdr_setAudioMuteFader

Description

This function sets the fader parameters of the audio mute.

Definition

```
HDRET tchdr_setAudioMuteFader(U32 enable, U32 fadein_ms, U32 fadeout_ms)
```

Parameters

Parameter	I/O	Description
enable	I	Enable audio mute fader on/off. It is enabled by default. 0: Mute fader Off 1: Mute fader On
fadein_ms	I	Fade-in time (ms) The default value is 100 ms. The range is from 0 to 1000.
fadeout_ms	I	Fade-out time (ms) The default value is 100 ms. The range is from 0 to 1000.

Returns

Return Value	Description
HDRET	Return value Refer to Chapter 3.1 eTC_HDR_RET_t.

Comments

Refer to Figure 5.1.

Figure 5.1 Fade-in Time and Fade-out Time of Audio Mute

5.19 tchdr_getAudioMuteFader

Description

This function gets the fader parameters of the audio mute.

Definition

```
HDRET tchdr_getAudioMuteFader(U32 *enable, U32 *fadein_ms, U32 *fadeout_ms)
```

Parameters

Parameter	I/O	Description
enable	O	Pointer to enable audio mute fader on/off 0: Mute fader Off 1: Mute fader On
fadein_ms	O	Pointer to fade-in time (ms)
fadeout_ms	O	Pointer to fade-out time (ms)

Returns

Return Value	Description
HDRET	Return value Refer to Chapter 3.1 eTC_HDR_RET_t.

Comments

None

5.20 tchdr_getAvailablePrograms

Description

This function gets a bitmap of available audio programs.

Definition

```
HDRET tchdr_getAvailablePrograms(eTC_HDR_ID_t id, stTC_HDR_PROG_BITMAP_t *availablePrograms)
```

Parameters

Parameter	I/O	Description
id	I	Refer to Chapter 3.3 eTC_HDR_ID_t.
availablePrograms	I	Refer to Chapter 4.7 stTC_HDR_PROG_BITMAP_t.

Returns

Return Value	Description
HDRET	Return value Refer to Chapter 3.1 eTC_HDR_RET_t.

Comments

None

5.21 tchdr_getProgramType

Description

This function gets the type of each program.

Definition

```
HDRET tchdr_getProgramType(eTC_HDR_ID_t id, stTC_HDR_PTY_t *pty);
```

Parameters

Parameter	I/O	Description
id	I	Refer to Chapter 3.3 eTC_HDR_ID_t.
*pty	O	Refer to Chapter 4.10 stTC_HDR_PTY_t.

Returns

Return Value	Description
HDRET	Return value Refer to Chapter 3.1 eTC_HDR_RET_t.

Comments

None

5.22 tchdr_setAutoAudioAlignEnable

Description

This function is set to enable or disable the automatic audio alignment function.

Definition

```
HDRET tchdr_setAutoAudioAlignEnable(U32 fEnable);
```

Parameters

Parameter	I/O	Description
fEnable	I	Control the Automatic Audio Alignment (AAA) function. 0: Disable AAA 1: Enable AAA

Returns

Return Value	Description
HDRET	Return value Refer to Chapter 3.1 eTC_HDR_RET_t.

Comments

None

5.23 tchdr_setBlendTransitionTime

Description

This function configures blend crossfade transition time. The new value is applied to the next blend crossfade execution, in which case the next crossfade has a new transition time.

Definition

```
HDRET tchdr_setBlendTransitionTime(U32 time)
```

Parameters

Parameter	I/O	Description
time	I	Duration of blend measured in audio frames (2048 pcm samples)

Returns

Return Value	Description
HDRET	Return value Refer to Chapter 3.1 eTC_HDR_RET_t.

Comments

Note: Do not change the transition time during blend crossfade. If you change the transition time during the blend crossfade, an error may occur.

5.24 tchdr_setBlendAllParams

Description

This function sets all blend parameters at once.

Definition

```
HDRET tchdr_setBlendAllParams(stTC_HDR_BLEND_PARAMS_t params)
```

Parameters

Parameter	I/O	Description
params	I	New blend parameters Refer to Chapter 3.11 eTC_HDR_BLEND_PARAMS_t.

Returns

Return Value	Description
HDRET	Return value Refer to Chapter 3.1 eTC_HDR_RET_t.

Comments

Note: Some parameters are write-protected and can be modified only in idle mode.

5.25 tchdr_getBlendAllParams

Description

This function gets all blend parameters at once.

Definition

```
HDRET tchdr_getBlendAllParams(stTC_HDR_BLEND_PARAMS_t *params)
```

Parameters

Parameter	I/O	Description
params	I	Pointer to blend parameters Refer to Chapter 3.11 eTC_HDR_BLEND_PARAMS_t.

Returns

Return Value	Description
HDRET	Return value Refer to Chapter 3.1 eTC_HDR_RET_t.

Comments

None

5.26 tchdr_setBlendParam

Description

This function sets an individual blend parameter.

Definition

```
HDRET tchdr_setBlendParam(eTC_HDR_BLEND_PARAMS_t param, U32 param_value)
```

Parameters

Parameter	I/O	Description
param	I	Name of the parameter from eTC_HDR_BLEND_PARAMS_t Refer to Chapter 3.11 eTC_HDR_BLEND_PARAMS_t.
param_value	I	Value to set each parameter

Returns

Return Value	Description
HDRET	Return value Refer to Chapter 3.1 eTC_HDR_RET_t.

Comments

None

5.27 tchdr_getBlendParam

Description

This function gets an individual blend parameter.
The default output is U32 type, so allocate a minimum of 4 bytes for the result.

Definition

```
HDRET tchdr_getBlendParam(eTC_HDR_BLEND_PARAMS_t param, U32 *param_value)
```

Parameters

Parameter	I/O	Description
param	I	Name of the parameter from eTC_HDR_BLEND_PARAMS_t Refer to Chapter 3.11 eTC_HDR_BLEND_PARAMS_t.
param_value	O	Pointer to the output value (must be at least 4 bytes long)

Returns

Return Value	Description
HDRET	Return value Refer to Chapter 3.1 eTC_HDR_RET_t.

Comments

Refer to the following example:

- tchdr_getBlendParam(eBLEND_PARAM_FM_MPS_BLEND_THRESH, &fmMpsBlendThresh);

5.28 tchdr_setBlendAllAdvParams

Description

This function sets all advanced blend parameters at once.

Definition

```
HDRET tchdr_setBlendAllAdvParams(stTC_HDR_BLEND_ADV_PARAMS_t params)
```

Parameters

Parameter	I/O	Description
params	I	New advanced blend parameters Refer to Chapter 3.12 eTC_HDR_BLEND_ADV_PARAMS_t.

Returns

Return Value	Description
HDRET	Return value Refer to Chapter 3.1 eTC_HDR_RET_t.

Comments

None

5.29 tchdr_getBlendAllAdvParams

Description

This function gets all advanced blend parameters at once.

Definition

```
HDRET tchdr_getBlendAllAdvParams(stTC_HDR_BLEND_ADV_PARAMS_t *params)
```

Parameters

Parameter	I/O	Description
params	I	Pointer to advanced blend parameters Refer to Chapter 3.12 eTC_HDR_BLEND_ADV_PARAMS_t.

Returns

Return Value	Description
HDRET	Return value Refer to Chapter 3.1 eTC_HDR_RET_t.

Comments

None

5.30 tchdr_setBlendAdvParam

Description

This function sets an individual advanced blend parameter.

Definition

```
HDRET tchdr_setBlendAdvParam(eTC_HDR_BLEND_ADV_PARAMS_t param, U32 param_value)
```

Parameters

Parameter	I/O	Description
param	I	Name of the parameter from eTC_HDR_BLEND_ADV_PARAMS_t Refer to Chapter 3.12 eTC_HDR_BLEND_ADV_PARAMS_t.
param_value	I	Value to set each parameter

Returns

Return Value	Description
HDRET	Return value Refer to Chapter 3.1 eTC_HDR_RET_t.

Comments

None

5.31 tchdr_getBlendAdvParam

Description

This function gets an individual advanced blend parameter.
The default output is U32, so allocate a minimum of 4 bytes for the result.

Definition

```
HDRET tchdr_getBlendAdvParam(eTC_HDR_BLEND_ADV_PARAMS_t param, U32 *param_value)
```

Parameters

Parameter	I/O	Description
param	I	Name of the parameter from eTC_HDR_BLEND_ADV_PARAMS_t Refer to Chapter 3.12 eTC_HDR_BLEND_ADV_PARAMS_t.
param_value	O	Pointer to the output value

Returns

Return Value	Description
HDRET	Return value Refer to Chapter 3.1 eTC_HDR_RET_t.

Comments

Refer to the following example:

- ret = tchdr_getBlendAdvParam(eBLEND_ADV_PARAM_RAMP_UP_ENABLED, &rampUpEnabledVal);

5.32 tchdr_getFrameworkVersionString

Description

This function returns a pointer to a character string containing the HD framework version.

Definition

```
const S8 *tchdr_getFrameworkVersionString(void)
```

Returns

Return Value	Description
const S8 *	Pointer to a character string

Comments

None

5.33 tchdr_getLibraryVersionString

Description

This function returns a pointer to a character string containing the HD library version.

Definition

```
const S8 *tchdr_getLibraryVersionString(void)
```

Returns

Return Value	Description
const S8 *	Pointer to a character string

Comments

None

5.34 tchdr_setThreadPriority

Description

This function sets the priority values of the HD Radio framework threads. The scheduler of the HD Radio framework threads allows selecting real-time or other policy by user setting.

Definition

```
HDRET tchdr_setThreadPriority(eTC_HDR_THREAD_t thread, stTC_HDR_THREAD_PR_t userprio)
```

Parameters

Parameter	I/O	Description
thread	I	An enumeration input argument that selects thread Refer to Chapter 3.13 eTC_HDR_THREAD_t.
userprio	I	A structure input argument that sets the policy and priority for the selected thread If policy is 0, it is SCHED_OTHER (Nice). If policy is 1, it is SCHED_FIFO (Real-time). The value range of real-time priority is from 1 (Low) to 99 (High). The value range of nice priority is from -20 (High) to +19 (Low). Refer to Chapter 4.1 stTC_HDR_THREAD_PR_t.

API	Nice		Real-time	
	+19 (Low) ⋯0	-1 ⋯ -20 (High)	1 (Low) ⋯ 49	50⋯ 99 (High)
Top	**Priority (PR)**			
	39 ⋯ 0		-2 ⋯ -99 -100 (shown as *rt*)	

Returns

Return Value	Description
HDRET	Return value Refer to Chapter 3.1 eTC_HDR_RET_t.

Comments

Before the thread is initialized by tchdr_init function, you should set the priority of the thread. If you do not set the priority of threads, the threads are created with the default real-time priority values. The priority default values can be obtained through the tchdr_getDefaultThreadPriority() function.

Note: It is recommended to use the default priority values. The problem that occurs after changing the priority values is not guaranteed.

5.35 tchdr_getDefaultThreadPriority

Description

This function gets the default policy and priority values of the HD Radio framework threads.

Definition

```
HDRET tchdr_getDefaultThreadPriority(eTC_HDR_THREAD_t thread, stTC_HDR_THREAD_PR_t *userprio)
```

Parameters

Parameter	I/O	Description
thread	I	An enumeration input argument that selects thread. Refer to Chapter 3.13 eTC_HDR_THREAD_t.
userprio	O	A structure argument that gets the default policy and priority for the selected thread Default policy and priority are provided as the recommended real-time (SCHED_FIFO) scheduler. Refer to Chapter 4.1 stTC_HDR_THREAD_PR_t.

Returns

Return Value	Description
HDRET	Return value Refer to Chapter 3.1 eTC_HDR_RET_t.

Comments

None

5.36 tchdr_getDefaultThreadNicePriority

Description

This function gets the default policy and priority values based on the *SCHED_OTHER* scheduler of the HD radio framework threads.

Definition

```
HDRET tchdr_getDefaultThreadNicePriority(eTC_HDR_THREAD_t thread, stTC_HDR_THREAD_PR_t *userprio)
```

Parameters

Parameter	I/O	Description
thread	I	An enumeration input argument that selects thread. Refer to Chapter 3.13 eTC_HDR_THREAD_t.
userprio	O	A structure argument that gets the default policy and priority for the selected thread Default policy and priority are provided as the *SCHED_OTHER* scheduler. Refer to Chapter 4.1 stTC_HDR_THREAD_PR_t.

Returns

Return Value	Description
HDRET	Return value Refer to Chapter 3.1 eTC_HDR_RET_t.

Comments

None

5.37 tchdr_getThreadPriority

Description

This function gets the current policy and priority values of the HD Radio framework threads.

Definition

```
HDRET tchdr_getThreadPriority(eTC_HDR_THREAD_t thread, stTC_HDR_THREAD_PR_t *userprio)
```

Parameters

Parameter	I/O	Description
thread	I	An enumeration input argument that selects thread Refer to Chapter 3.13 eTC_HDR_THREAD_t.
userprio	O	A structure argument that gets the policy and priority for the selected thread. Refer to Chapter 4.1 stTC_HDR_THREAD_PR_t.

Returns

Return Value	Description
HDRET	Return value Refer to Chapter 3.1 eTC_HDR_RET_t.

Comments

None

5.38 tchdr_configTunerIQ01Driver

Description

This function registers an IQ01 driver callback function.

Definition

```
void tchdr_configTunerIQ01Driver(S32(*pfnIQ01DrvOpen)(void),
                                 S32(*pfnIQ01DrvClose)(void),
                                 S32(*pfnIQ01DrvStart)(S32, S32, S32),
                                 S32(*pfnIQ01DrvStop)(void),
                                 S32(*pfnIQ01DrvRead)(S8*, S32));
```

Parameters

Parameter	I/O	Description
pfnIQ01DrvOpen	I	A function pointer to open the I/Q01 I2S driver
pfnIQ01DrvClose	I	A function pointer to close the I/Q01 I2S driver
pfnIQ01DrvStart	I	A function pointer to start the I/Q01 I2S driver
pfnIQ01DrvStop	I	A function pointer to stop the I/Q01 I2S driver
pfnIQ01DrvRead	I	A function pointer to read the I/Q01 I2S driver

Comments

I/Q0: Primary tuner I and Q data. I2S DIN[1:0]
I/Q1: Secondary tuner I and Q data. I2S DIN[3:2]

Example Code

```
tchdr_configTunerIQ01Driver(open_func, close_func, setparams_func, start_func, stop_func, read_func);
```

5.39 tchdr_configTunerIQ23Driver

Description

This function registers an IQ23 driver callback function.

Definition

```
void tchdr_configTunerIQ23Driver(S32(*pfnIQ23DrvOpen)(void),
                                 S32(*pfnIQ23DrvClose)(void),
                                 S32(*pfnIQ23DrvStart)(S32, S32, S32),
                                 S32(*pfnIQ23DrvStop)(void),
                                 S32(*pfnIQ23DrvRead)(S8*, S32));
```

Parameters

Parameter	I/O	Description
pfnIQ23DrvOpen	I	A function pointer to open the I/Q23 I2S driver
pfnIQ23DrvClose	I	A function pointer to close the I/Q23 I2S driver
pfnIQ23DrvStart	I	A function pointer to start the I/Q23 I2S driver
pfnIQ23DrvStop	I	A function pointer to stop the I/Q23 I2S driver
pfnIQ23DrvRead	I	A function pointer to read the I/Q23 I2S driver

Comments

I/Q2: Tertiary tuner I and Q data. I2S DIN[1:0]
I/Q3: Quaternary tuner I and Q data. I2S DIN[3:2] (Not supported yet)

Example Code

```
tchdr_configTunerIQ23Driver(open_func, close_func, setparams_func, start_func, stop_func, read_func);
```

5.40 tchdr_configTunerBlendAudioDriver

Description

This function registers a blend audio input driver callback function.

Definition

```
void tchdr_configTunerBlendAudioDriver(S32(*pfnBlendAudioDrvOpen)(void),
                                       S32(*pfnBlendAudioDrvClose)(void),
                                       S32(*pfnBlendAudioDrvStart)(S32, S32, S32, S32),
                                       S32(*pfnBlendAudioDrvStop)(void),
                                       S32(*pfnBlendAudioDrvRead)(S8*, S32));
```

Parameters

Parameter	I/O	Description
pfnBlendAudioDrvOpen	I	A function pointer to open the blend audio I2S driver
pfnBlendAudioDrvClose	I	A function pointer to close the blend audio I2S driver
pfnBlendAudioDrvStart	I	A function pointer to start the blend audio I2S driver
pfnBlendAudioDrvStop	I	A function pointer to stop the blend audio I2S driver
pfnBlendAudioDrvRead	I	A function pointer to read the blend audio I2S driver

Comments

None

Example Code

```
tchdr_configTunerBlendAudioDriver(open_func,   close_func,   setparams_func,   start_func,   stop_func,
read_func);
```

5.41 tchdr_configTcHdrNotificationCallBack

Description

This function registers a notification callback function.

Definition

```
void tchdr_configTcHdrNotificationCallBack(void(*pfnNotificationCallBack)(U32 notifyID, const U32
*pArg, void* const *pData, S32 errorCode))
```

Parameters

Parameter	I/O	Description
pfnNotificationCallBack	I	A callback function to notify the result and data of executing the HD Radio API Refer to Chapter 6.1 pfnNotificationCallBack.

Comments

None

5.42 tchdr_configTcHdrAudioQueueCallBack

Description

This function registers an audio queue callback function.

Definition

```
void tchdr_configTcHdrAudioQueueCallBack(void(*pfnAudioQueueCallBack)(void *pOutBuf, S32 frame, U32 samplerate))
```

Parameters

Parameter	I/O	Description
pfnAudioQueueCallBack	I	A callback function for audio queue Refer to Chapter 6.2 pfnAudioQueueCallBack.

Comments

Whenever an audio queue occurs, the registered callback function is called.

5.43 tchdr_cb_getIqSampleRate

Description

This function gets sampling rate of IQ for CDM interface.

Definition

```
HDRET tchdr_cb_getIqSampleRate(U32 ntuner)
```

Parameters

Parameter	I/O	Description
ntuner	I	Tuner number

Returns

Return Value	Description
HDRET	Return value Refer to Chapter 3.1 eTC_HDR_RET_t.

Comments

None

5.44 tchdr_cb_setTune

Description

This function sets the tuner frequency for CDM interface.

Definition

```
HDRET tchdr_cb_setTune(eTC_HDR_BAND_t band, U32 freq, U32 instance_number)
```

Parameters

Parameter	I/O	Description
band	I	FM/AM/IDLE band Refer to Chapter 3.4 eTC_HDR_BAND_t.
freq	I	Frequency (kHz)
instance_number	I	HD Radio instance number

Returns

Return Value	Description
HDRET	Return value Refer to Chapter 3.1 eTC_HDR_RET_t.

Comments

None

5.45 tchdr_aas_enablePorts

Description

This function enables the specified ports.

Definition

```
HDRET tchdr_aas_enablePorts(eTC_HDR_ID_t id, const stTC_HDR_AAS_PORT_LIST_t *port_list)
```

Parameters

Parameter	I/O	Description
id	I	Refer to Chapter 3.3 eTC_HDR_ID_t.
port_list	I	Refer to Chapter 4.13 stTC_HDR_AAS_PORT_LIST_t.

Returns

Return Value	Description
HDRET	Return value Refer to Chapter 3.1 eTC_HDR_RET_t.

Comments

None

5.46 tchdr_aas_disablePorts

Description

This function disables the specified ports.

Definition

```
HDRET tchdr_aas_disablePorts(eTC_HDR_ID_t id, const stTC_HDR_AAS_PORT_LIST_t *port_list)
```

Parameters

Parameter	I/O	Description
id	I	Refer to Chapter 3.3 eTC_HDR_ID_t.
port_list	I	Refer to Chapter 4.13 stTC_HDR_AAS_PORT_LIST_t.

Returns

Return Value	Description
HDRET	Return value Refer to Chapter 3.1 eTC_HDR_RET_t.

Comments

None

5.47 tchdr_aas_disableAllPorts

Description

This function disables all enabled ports.

Definition

```
HDRET tchdr_aas_disableAllPorts(eTC_HDR_ID_t id)
```

Parameters

Parameter	I/O	Description
id	I	Refer to Chapter 3.3 eTC_HDR_ID_t.

Returns

Return Value	Description
HDRET	Return value Refer to Chapter 3.1 eTC_HDR_RET_t.

Comments

None

5.48 tchdr_aas_getEnabledPorts

Description

This function reports information about currently enabled AAS ports.

Definition

```
HDRET tchdr_aas_getEnabledPorts(eTC_HDR_ID_t id, stTC_HDR_AAS_PORT_LIST_t* port_list)
```

Parameters

Parameter	I/O	Description
id	I	Refer to Chapter 3.3 eTC_HDR_ID_t.
port_list	O	Refer to Chapter 4.13 stTC_HDR_AAS_PORT_LIST_t.

Returns

Return Value	Description
HDRET	Return value Refer to Chapter 3.1 eTC_HDR_RET_t.

Comments

None

5.49 tchdr_aas_getNextPortData

Description

This function gets the next available packet from any enabled port.

Definition

```
HDRET tchdr_aas_getNextPortData(eTC_HDR_ID_t id, stTC_HDR_AAS_PACKET_INFO_t *packet_info, U8*
packet_buffer, U32 buffer_size, U32 * bytes_written)
```

Parameters

Parameter	I/O	Description
id	I	Refer to Chapter 3.3 eTC_HDR_ID_t.
packet_info	O	Pointer to the packet information structure. Must be allocated by the caller. Refer to Chapter 4.14 stTC_HDR_AAS_PACKET_INFO_t.
packet_buffer	O	Pointer to the packet output buffer. Must be allocated by the caller.
buffer_size	I	Size of the output buffer provided
bytes_written	O	Number of bytes written to the buffer

Returns

Return Value	Description
HDRET	Return value Refer to Chapter 3.1 eTC_HDR_RET_t.

Comments

None

5.50 tchdr_aas_getPortData

Description

This function gets the available packet from the specified port number.

Definition

```
HDRET tchdr_aas_getPortData(eTC_HDR_ID_t id, U32 port_number, stTC_HDR_AAS_PACKET_INFO_t* packet_info,
U8* packet_buffer, U32 buffer_size, U32 *bytes_written)
```

Parameters

Parameter	I/O	Description
id	I	Refer to Chapter 3.3 eTC_HDR_ID_t.
port_number	I	Port number of requested data
packet_info	O	Pointer to the packet information structure. Must be allocated by the caller
packet_buffer	O	Pointer to the packet output buffer. Must be allocated by the caller
buffer_size	I	Size of the output buffer provided
bytes_written	O	Number of bytes written in the buffer

Returns

Return Value	Description
HDRET	Return value Refer to Chapter 3.1 eTC_HDR_RET_t.

Comments

None

5.51 tchdr_aas_flushPort

Description

This function flushes all data waiting on the specified port number.

Definition

```
HDRET tchdr_aas_flushPort(eTC_HDR_ID_t id, U32 port_number)
```

Parameters

Parameter	I/O	Description
id	I	Refer to Chapter 3.3 eTC_HDR_ID_t.
port_number	I	Specified port number to flush data

Returns

Return Value	Description
HDRET	Return value Refer to Chapter 3.1 eTC_HDR_RET_t.

Comments

None

5.52 tchdr_aas_flushAllPorts

Description

This function flushes all data waiting on all enabled ports.

Definition

```
HDRET tchdr_aas_flushAllPorts(eTC_HDR_ID_t id)
```

Parameters

Parameter	I/O	Description
id	I	Refer to Chapter 3.3 eTC_HDR_ID_t.

Returns

Return Value	Description
HDRET	Return value Refer to Chapter 3.1 eTC_HDR_RET_t.

Comments

None

5.53 tchdr_aas_getLotPoolSize

Description

This function returns the total size of LOT memory pool in bytes.

Definition

```
HDRET tchdr_aas_getLotPoolSize(eTC_HDR_ID_t id, U32 * mem_pool_size)
```

Parameters

Parameter	I/O	Description
id	I	Refer to Chapter 3.3 eTC_HDR_ID_t.
mem_pool_size	O	Size of memory pool

Returns

Return Value	Description
HDRET	Return value Refer to Chapter 3.1 eTC_HDR_RET_t.

Comments

None

5.54 tchdr_aas_getLotSpaceLeft

Description

This function returns LOT memory (in bytes) left in the pool.

Definition

```
HDRET tchdr_aas_getLotSpaceLeft(eTC_HDR_ID_t id, U32 *mem_left)
```

Parameters

Parameter	I/O	Description
id	I	Refer to Chapter 3.3 eTC_HDR_ID_t.
mem_left	O	Size of LOT memory left in the pool

Returns

Return Value	Description
HDRET	Return value Refer to Chapter 3.1 eTC_HDR_RET_t.

Comments

None

5.55 tchdr_aas_lotOverflow

Description

This function detects if an overflow condition occurred.

Definition

```
HDRET tchdr_aas_lotOverflow(eTC_HDR_ID_t id, HDBOOL *overFlow)
```

Parameters

Parameter	I/O	Description
id	I	Refer to Chapter 3.3 eTC_HDR_ID_t.
overFlow	O	Condition value of overflow

Returns

Return Value	Description
HDRET	Return value Refer to Chapter 3.1 eTC_HDR_RET_t.

Comments

None

5.56 tchdr_aas_enableLotReassembly

Description

This function enables the specified port number and initiates LOT reassembly of all LOT objects received on that port.

Definition

```
HDRET tchdr_aas_enableLotReassembly(eTC_HDR_ID_t id, U32 service_number, U32 port_number)
```

Parameters

Parameter	I/O	Description
id	I	Refer to Chapter 3.3 eTC_HDR_ID_t.
service_number	I	Specified service number
port_number	I	Specified port number

Returns

Return Value	Description
HDRET	Return value Refer to Chapter 3.1 eTC_HDR_RET_t.

Comments

None

5.57 tchdr_aas_disableLotReassembly

Description

This function stops LOT reassembly for the specified service number and port number, or all enabled ports for that service number.

Definition

```
HDRET tchdr_aas_disableLotReassembly(eTC_HDR_ID_t id, U32 service_number, U32 port_number)
```

Parameters

Parameter	I/O	Description
id	I	Refer to Chapter 3.3 eTC_HDR_ID_t.
service_number	I	Specify the service number that should be disabled
port_number	I	Specify the port number to be disabled. Use port number 0 to disable all ports associated with the specified service number

Returns

Return Value	Description
HDRET	Return value Refer to Chapter 3.1 eTC_HDR_RET_t.

Comments

None

5.58 tchdr_aas_getLotObjectList

Description

This function gets the current list of all complete and incomplete objects for the specified service number.

Definition

```
HDRET tchdr_aas_getLotObjectList(eTC_HDR_ID_t id, U32 service_number, stTC_HDR_AAS_LOT_OBJECT_LIST_t *
object_list)
```

Parameters

Parameter	I/O	Description
id	I	Refer to Chapter 3.3 eTC_HDR_ID_t.
service_number	I	Specified service number
object_list	O	Refer to Chapter 4.15 stTC_HDR_AAS_LOT_OBJECT_LIST_t. Store the list of LOT objects.

Returns

Return Value	Description
HDRET	Return value Refer to Chapter 3.1 eTC_HDR_RET_t.

Comments

None

5.59 tchdr_aas_getLotObjectListByName

Description

This function gets the current list of all complete and incomplete objects for the service number whose file names match the requested file name.

Definition

```
HDRET tchdr_aas_getLotObjectListByName(eTC_HDR_ID_t id, U32 service_number, const S8* filename,
stTC_HDR_AAS_LOT_OBJECT_LIST_t* object_list)
```

Parameters

Parameter	I/O	Description
id	I	Refer to Chapter 3.3 eTC_HDR_ID_t.
service_number	I	Specified service number
filename	I	Specify the Null-terminated filename of the objects
object_list	O	Refer to Chapter 4.15 stTC_HDR_AAS_LOT_OBJECT_LIST_t. Store the list of complete and incomplete objects.

Returns

Return Value	Description
HDRET	Return value Refer to Chapter 3.1 eTC_HDR_RET_t.

Comments

None

5.60 tchdr_aas_getLotObjectHeader

Description

This function gets the header of an object.

Definition

```
HDRET tchdr_aas_getLotObjectHeader(eTC_HDR_ID_t id, U32 port_number, U32 object_lot_id,
stTC_HDR_AAS_LOT_OBJECT_HEADER_t* object_header)
```

Parameters

Parameter	I/O	Description
id	I	Refer to Chapter 3.3 eTC_HDR_ID_t.
port_number	I	Specify the port number
object_lot_id	I	Specify the object LOT ID
object_header	O	Refer to Chapter 4.16 stTC_HDR_AAS_LOT_OBJECT_HEADER_t. Pointer to the structure, where the header should be stored

Returns

Return Value	Description
HDRET	Return value Refer to Chapter 3.1 eTC_HDR_RET_t.

Comments

None

5.61 tchdr_aas_getLotObjectBody

Description

This function gets the next block of data from the body of the LOT object.

Definition

```
HDRET tchdr_aas_getLotObjectBody(eTC_HDR_ID_t id, U32 port_number, U32 object_lot_id, U8* buffer, U32
buffer_size, U32 * bytes_written)
```

Parameters

Parameter	I/O	Description
id	I	Refer to Chapter 3.3 eTC_HDR_ID_t.
port_number	I	Specify the port number
object_lot_id	I	Specify the object LOT ID
buffer	O	Pointer to the buffer where the object body should be stored
buffer_size	O	Size of the buffer
bytes_written	O	Actual number of bytes written to the buffer

Returns

Return Value	Description
HDRET	Return value. Refer to Chapter 3.1 eTC_HDR_RET_t for value lower than 0. 1: End of file 0: Not end of file

Comments

None

5.62 tchdr_aas_flushLotObject

Description

This function flushes an object from LOT memory.

Definition

```
HDRET tchdr_aas_flushLotObject(eTC_HDR_ID_t id, U32 port_number, U32 object_lot_id)
```

Parameters

Parameter	I/O	Description
id	I	Refer to Chapter 3.3 eTC_HDR_ID_t.
port_number	I	Specify the port number
object_lot_id	I	Specify the object LOT ID

Returns

Return Value	Description
HDRET	Return value Refer to Chapter 3.1 eTC_HDR_RET_t.

Comments

None

5.63 tchdr_alert_getMessage

Description

This function gets the latest emergency alert message.

Definition

```
HDRET tchdr_alert_getMessage(eTC_HDR_ID_t id, stTC_HDR_ALERT_MESSAGE_t* message)
```

Parameters

Parameter	I/O	Description
id	I	Refer to Chapter 3.3 eTC_HDR_ID_t.
message	O	Pointer to emergency alert message

Returns

Return Value	Description
HDRET	Return value Refer to Chapter 3.1 eTC_HDR_RET_t.

Comments

None

5.64 tchdr_alert_getMessageStatus

Description

This function provides current status of message reception.

Definition

```
HDRET tchdr_alert_getMessageStatus(eTC_HDR_ID_t id, stTC_HDR_ALERTS_MSG_STATUS_t* status)
```

Parameters

Parameter	I/O	Description
id	I	Refer to Chapter 3.3 eTC_HDR_ID_t.
status	O	Pointer to emergency alert status

Returns

Return Value	Description
HDRET	Return value Refer to Chapter 3.1 eTC_HDR_RET_t.

Comments

None

5.65 tchdr_alert_clearMessageStatus

Description

This function clears message status bits.

Definition

```
HDRET tchdr_alert_clearMessageStatus(eTC_HDR_ID_t id)
```

Parameters

Parameter	I/O	Description
id	I	Refer to Chapter 3.3 eTC_HDR_ID_t.

Returns

Return Value	Description
HDRET	Return value Refer to Chapter 3.1 eTC_HDR_RET_t.

Comments

None

5.66 tchdr_psd_getChangedPrograms

Description

This function retrieves PSD content change flags.

Definition

```
HDRET tchdr_psd_getChangedPrograms(eTC_HDR_ID_t id, stTC_HDR_PROG_BITMAP_t* bitmap)
```

Parameters

Parameter	I/O	Description
id	I	Refer to Chapter 3.3 eTC_HDR_ID_t.
bitmap	O	Refer to Chapter 4.7 stTC_HDR_PROG_BITMAP_t. Bitfield data structure with flags set to 1 for programs for which PSD is changed

Returns

Return Value	Description
HDRET	Return value Refer to Chapter 3.1 eTC_HDR_RET_t.

Comments

None

5.67 tchdr_psd_clearChangedProgram

Description

This function clears PSD content change flag for the specified program.

Definition

```
HDRET tchdr_psd_clearChangedProgram(eTC_HDR_ID_t id, eTC_HDR_PROGRAM_t program_number)
```

Parameters

Parameter	I/O	Description
id	I	Refer to Chapter 3.3 eTC_HDR_ID_t.
program_number	I	Clear flag for this program. Refer to Chapter 3.8 eTC_HDR_PROGRAM_t.

Returns

Return Value	Description
HDRET	Return value Refer to Chapter 3.1 eTC_HDR_RET_t.

Comments

None

5.68 tchdr_psd_enableFields

Description

This function specifies PSD fields to be processed.

Definition

```
HDRET tchdr_psd_enableFields(eTC_HDR_ID_t id, stTC_HDR_PSD_FIELDS_t enabled_fields)
```

Parameters

Parameter	I/O	Description
id	I	Refer to Chapter 3.3 eTC_HDR_ID_t.
enabled_fields	I	Refer to Chapter 4.20 stTC_HDR_PSD_FIELDS_t. Specify PSD fields to be enabled.

Returns

Return Value	Description
HDRET	Return value Refer to Chapter 3.1 eTC_HDR_RET_t.

Comments

None

5.69 tchdr_psd_getEnabledFields

Description

This function returns currently enabled PSD fields.

Definition

```
HDRET tchdr_psd_getEnabledFields(eTC_HDR_ID_t id, stTC_HDR_PSD_FIELDS_t *fields)
```

Parameters

Parameter	I/O	Description
id	I	Refer to Chapter 3.3 eTC_HDR_ID_t.
fields	I	Refer to Chapter 4.20 stTC_HDR_PSD_FIELDS_t. Currently enabled PSD fields

Returns

Return Value	Description
HDRET	Return value Refer to Chapter 3.1 eTC_HDR_RET_t.

Comments

None

5.70 tchdr_psd_setMaxLength

Description

This function truncates the specified PSD field/subfield to a user-defined length.

Definition

```
HDRET tchdr_psd_setMaxLength(eTC_HDR_ID_t id, eTC_HDR_PSD_LENGTH_CONFIG_t config, U32 length)
```

Parameters

Parameter	I/O	Description
id	I	Refer to Chapter 3.3 eTC_HDR_ID_t.
config	I	Refer to Chapter 3.14 eTC_HDR_PSD_LENGTH_CONFIG_t. Specify the PSD field/subfield to be truncated.
length	I	New user-defined maximum length of the field/subfield

Returns

Return Value	Description
HDRET	Return value Refer to Chapter 3.1 eTC_HDR_RET_t.

Comments

None

5.71 tchdr_psd_resetMaxLength

Description

This function resets the PSD field/subfield to maximum length.

Definition

```
HDRET tchdr_psd_resetMaxLength(eTC_HDR_ID_t id, eTC_HDR_PSD_LENGTH_CONFIG_t config)
```

Parameters

Parameter	I/O	Description
id	I	Refer to Chapter 3.3 eTC_HDR_ID_t.
config	I	Refer to Chapter 3.14 eTC_HDR_PSD_LENGTH_CONFIG_t. Specify the PSD field/subfield to be truncated.

Returns

Return Value	Description
HDRET	Return value Refer to Chapter 3.1 eTC_HDR_RET_t.

Comments

None

5.72 tchdr_psd_getMaxLength

Description

This function returns the currently set maximum field length.

Definition

```
HDRET tchdr_psd_getMaxLength(eTC_HDR_ID_t id, eTC_HDR_PSD_LENGTH_CONFIG_t config, U32 *max_length)
```

Parameters

Parameter	I/O	Description
id	I	Refer to Chapter 3.3 eTC_HDR_ID_t.
config	I	Refer to Chapter 3.14 eTC_HDR_PSD_LENGTH_CONFIG_t. Specify the PSD field/subfield.
max_length	O	Current maximum length of the requested field/subfield

Returns

Return Value	Description
HDRET	Return value Refer to Chapter 3.1 eTC_HDR_RET_t.

Comments

None

5.73 tchdr_psd_getTitle

Description

This function gets PSD title data.

Definition

```
HDRET tchdr_psd_getTitle(eTC_HDR_ID_t id, eTC_HDR_PROGRAM_t program_number, stTC_HDR_PSD_FORM_t* title)
```

Parameters

Parameter	I/O	Description
id	I	Refer to Chapter 3.3 eTC_HDR_ID_t.
program_number	I	Refer to Chapter 3.8 eTC_HDR_PROGRAM_t. Specify the program number.
title	O	Refer to Chapter 4.21 stTC_HDR_PSD_FORM_t. Pointer to output data

Returns

Return Value	Description
HDRET	Return value Refer to Chapter 3.1 eTC_HDR_RET_t.

Comments

None

5.74 tchdr_psd_getArtist

Description

This function gets PSD artist data.

Definition

```
HDRET tchdr_psd_getArtist(eTC_HDR_ID_t id, eTC_HDR_PROGRAM_t program_number, stTC_HDR_PSD_FORM_t* artist)
```

Parameters

Parameter	I/O	Description
id	I	Refer to Chapter 3.3 eTC_HDR_ID_t.
program_number	I	Refer to Chapter 3.8 eTC_HDR_PROGRAM_t. Specify the program number.
artist	O	Refer to Chapter 4.21 stTC_HDR_PSD_FORM_t. Pointer to output data

Returns

Return Value	Description
HDRET	Return value Refer to Chapter 3.1 eTC_HDR_RET_t.

Comments

None

5.75 tchdr_psd_getAlbum

Description

This function gets PSD album data.

Definition

```
HDRET tchdr_psd_getAlbum(eTC_HDR_ID_t id, eTC_HDR_PROGRAM_t program_number, stTC_HDR_PSD_FORM_t* album)
```

Parameters

Parameter	I/O	Description
id	I	Refer to Chapter 3.3 eTC_HDR_ID_t.
program_number	I	Refer to Chapter 3.8 eTC_HDR_PROGRAM_t. Specify the program number.
album	O	Refer to Chapter 4.21 stTC_HDR_PSD_FORM_t. Pointer to output data

Returns

Return Value	Description
HDRET	Return value Refer to Chapter 3.1 eTC_HDR_RET_t.

Comments

None

5.76 tchdr_psd_getGenre

Description

This function gets PSD genre data.

Definition

```
HDRET tchdr_psd_getGenre(eTC_HDR_ID_t id, eTC_HDR_PROGRAM_t program_number, stTC_HDR_PSD_FORM_t* genre)
```

Parameters

Parameter	I/O	Description
id	I	Refer to Chapter 3.3 eTC_HDR_ID_t.
program_number	I	Refer to Chapter 3.8 eTC_HDR_PROGRAM_t. Specify the program number.
genre	O	Refer to Chapter 4.21 stTC_HDR_PSD_FORM_t. Pointer to output data

Returns

Return Value	Description
HDRET	Return value Refer to Chapter 3.1 eTC_HDR_RET_t.

Comments

None

5.77 tchdr_psd_getComment

Description

This function gets PSD comment field/subfield data.

Definition

```
HDRET tchdr_psd_getComment(eTC_HDR_ID_t id, eTC_HDR_PROGRAM_t program_number,
eTC_HDR_PSD_COMM_SUBFIELD_t subfield, stTC_HDR_PSD_FORM_t* comment)
```

Parameters

Parameter	I/O	Description
id	I	Refer to Chapter 3.3 eTC_HDR_ID_t.
program_number	I	Refer to Chapter 3.8 eTC_HDR_PROGRAM_t. Specify the program number.
subfield	I	Refer to Chapter 3.17 eTC_HDR_PSD_COMM_SUBFIELD_t. Specify requested subfield.
comment	O	Refer to Chapter 4.21 stTC_HDR_PSD_FORM_t. Pointer to output data

Returns

Return Value	Description
HDRET	Return value Refer to Chapter 3.1 eTC_HDR_RET_t.

Comments

None

5.78 tchdr_psd_getUfid

Description

This function gets PSD UFID field/subfield data.

Definition

```
HDRET tchdr_psd_getUfid(eTC_HDR_ID_t id, eTC_HDR_PROGRAM_t program_number, U32 ufid_num,
eTC_HDR_PSD_UFID_SUBFIELD_t subfield, stTC_HDR_PSD_FORM_t* ufid)
```

Parameters

Parameter	I/O	Description
id	I	Refer to Chapter 3.3 eTC_HDR_ID_t.
program_number	I	Refer to Chapter 3.8 eTC_HDR_PROGRAM_t. Specify the program number.
ufid_num	I	Specify UFID frame.
subfield	I	Refer to Chapter 3.17 eTC_HDR_PSD_COMM_SUBFIELD_t. Specify requested subfield.
ufid	O	Refer to Chapter 4.21 stTC_HDR_PSD_FORM_t. Pointer to output data

Returns

Return Value	Description
HDRET	Return value Refer to Chapter 3.1 eTC_HDR_RET_t.

Comments

None

5.79 tchdr_psd_getCommercial

Description

This function gets PSD commercial field/subfield data.

Definition

```
HDRET tchdr_psd_getCommercial(eTC_HDR_ID_t id, eTC_HDR_PROGRAM_t program_number,
eTC_HDR_PSD_COMR_SUBFIELD_t subfield, stTC_HDR_PSD_FORM_t* commercial)
```

Parameters

Parameter	I/O	Description
id	I	Refer to Chapter 3.3 eTC_HDR_ID_t.
program_number	I	Refer to Chapter 3.8 eTC_HDR_PROGRAM_t. Specify the program number.
subfield	I	Refer to Chapter 3.17 eTC_HDR_PSD_COMM_SUBFIELD_t. Specify requested subfield.
commercial	O	Refer to Chapter 4.21 stTC_HDR_PSD_FORM_t. Pointer to output data

Returns

Return Value	Description
HDRET	Return value Refer to Chapter 3.1 eTC_HDR_RET_t.

Comments

None

5.80 tchdr_psd_getXhdr

Description

This function gets PSD XHDR field/subfield data.

Definition

```
HDRET tchdr_psd_getXhdr(eTC_HDR_ID_t id, eTC_HDR_PROGRAM_t program_number, U32 xhdr_num,
stTC_HDR_PSD_FORM_t* xhdr)
```

Parameters

Parameter	I/O	Description
id	I	Refer to Chapter 3.3 eTC_HDR_ID_t.
program_number	I	Refer to Chapter 3.8 eTC_HDR_PROGRAM_t. Specify the program number
xhdr_num	I	Specify UFID frame
xhdr	O	Refer to Chapter 4.21 stTC_HDR_PSD_FORM_t. Pointer to output data

Returns

Return Value	Description
HDRET	Return value Refer to Chapter 3.1 eTC_HDR_RET_t.

Comments

None

5.81 tchdr_sig_getServiceList

Description

This function gets a list of available services of the specified type.

Definition

```
HDRET tchdr_sig_getServiceList(eTC_HDR_ID_t id, eTC_HDR_SIG_SERVICE_TYPE_t service_type,
stTC_HDR_SIG_SERVICE_LIST_t* service_list);
```

Parameters

Parameter	I/O	Description
id	I	Refer to Chapter 3.3 eTC_HDR_ID_t.
service_type	I	Refer to Chapter 3.23 eTC_HDR_SIG_SERVICE_TYPE_t. Specify type (audio/data) of service.
service_list	O	Refer to Chapter 4.25 stTC_HDR_SIG_SERVICE_LIST_t. Pointer to the audio services output data structure

Returns

Return Value	Description
HDRET	Return value Refer to Chapter 3.1 eTC_HDR_RET_t.

Comments

None

5.82 tchdr_sig_getServiceInfo

Description

This function gets service information specified by the service number ID.

Definition

```
HDRET tchdr_sig_getServiceInfo(eTC_HDR_ID_t id, U32 service_number, stTC_HDR_SIG_SERVICE_INFO_t*
service_info);
```

Parameters

Parameter	I/O	Description
id	I	Refer to Chapter 3.3 eTC_HDR_ID_t.
service_number	I	Service number ID
service_info	O	Refer to Chapter 4.26 stTC_HDR_SIG_SERVICE_INFO_t. Service information

Returns

Return Value	Description
HDRET	Return value Refer to Chapter 3.1 eTC_HDR_RET_t.

Comments

None

5.83 tchdr_sig_getServiceComponent

Description

This function gets service component data.

Definition

```
HDRET tchdr_sig_getServiceComponent(eTC_HDR_ID_t id, U32 service_number, U32 component_index,
stTC_HDR_SIG_SERVICE_COMPONENT_t* component);
```

Parameters

Parameter	I/O	Description
id	I	Refer to Chapter 3.3 eTC_HDR_ID_t.
service_number	I	Service number ID
component_index	I	Index of the service component as it is stored in memory
component	O	Refer to Chapter 4.27 stTC_HDR_SIG_SERVICE_COMPONENT_t. Pointer to the service component output data structure

Returns

Return Value	Description
HDRET	Return value Refer to Chapter 3.1 eTC_HDR_RET_t.

Comments

None

5.84 tchdr_sig_flushAll

Description

This function flushes all stored information and restarts SIG information retrieval.

Definition

```
HDRET tchdr_sig_flushAll(eTC_HDR_ID_t id);
```

Parameters

Parameter	I/O	Description
id	I	Refer to Chapter 3.3 eTC_HDR_ID_t.

Returns

Return Value	Description
HDRET	Return value Refer to Chapter 3.1 eTC_HDR_RET_t.

Comments

None

5.85 tchdr_sis_acquired

Description

This function gets status of SIS data reception.

Definition

```
HDRET tchdr_sis_acquired(eTC_HDR_ID_t id, HDBOOL *acquired);
```

Parameters

Parameter	I/O	Description
id	I	Refer to Chapter 3.3 eTC_HDR_ID_t.
acquired	O	Status of SIS data reception True: Received SIS messages False: Not received SIS messages

Returns

Return Value	Description
HDRET	Return value Refer to Chapter 3.1 eTC_HDR_RET_t.

Comments

None

5.86 tchdr_sis_crcOk

Description

This function instantaneously indicates the CRC status of SIS.

Definition

```
HDRET tchdr_sis_crcOk(eTC_HDR_ID_t id, HDBOOL *crc);
```

Parameters

Parameter	I/O	Description
id	I	Refer to Chapter 3.3 eTC_HDR_ID_t.
crc	O	True: Last SIS message CRC is OK. False: Last SIS message CRC failed.

Returns

Return Value	Description
HDRET	Return value Refer to Chapter 3.1 eTC_HDR_RET_t.

Comments

None

5.87 tchdr_sis_enableBasicTypes

Description

This function enables user specified SIS basic data types.

Definition

```
HDRET tchdr_sis_enableBasicTypes(eTC_HDR_ID_t id, stTC_HDR_SIS_ENABLED_BASIC_TYPES_t types);
```

Parameters

Parameter	I/O	Description
id	I	Refer to Chapter 3.3 eTC_HDR_ID_t.
types	O	Refer to Chapter 4.28 stTC_HDR_SIS_ENABLED_BASIC_TYPES_t. Bitmap list of basic SIS information to be enabled

Returns

Return Value	Description
HDRET	Return value Refer to Chapter 3.1 eTC_HDR_RET_t.

Comments

None

5.88 tchdr_sis_getEnabledBasicTypes

Description

This function gets information about enabled basic SIS data types.

Definition

```
HDRET tchdr_sis_getEnabledBasicTypes(eTC_HDR_ID_t id, stTC_HDR_SIS_ENABLED_BASIC_TYPES_t*
enabled_types);
```

Parameters

Parameter	I/O	Description
id	I	Refer to Chapter 3.3 eTC_HDR_ID_t.
enabled_types	O	Refer to Chapter 4.28 stTC_HDR_SIS_ENABLED_BASIC_TYPES_t. Pointer to the storage of enabled types provided by the caller

Returns

Return Value	Description
HDRET	Return value Refer to Chapter 3.1 eTC_HDR_RET_t.

Comments

None

5.89 tchdr_sis_getBlockCount

Description

This function gets the SIS block count.

Definition

```
HDRET tchdr_sis_getBlockCount(eTC_HDR_ID_t id, U32 *count);
```

Parameters

Parameter	I/O	Description
id	I	Refer to Chapter 3.3 eTC_HDR_ID_t.
count	O	SIS block count

Returns

Return Value	Description
HDRET	Return value Refer to Chapter 3.1 eTC_HDR_RET_t.

Comments

None

5.90 tchdr_sis_timeGpsLocked

Description

This function returns the status of GPS lock for transmit-site.

Definition

```
HDRET tchdr_sis_timeGpsLocked(eTC_HDR_ID_t id, HDBOOL *locked);
```

Parameters

Parameter	I/O	Description
id	I	Refer to Chapter 3.3 eTC_HDR_ID_t.
locked	O	Flags of GPS lock

Returns

Return Value	Description
HDRET	Return value Refer to Chapter 3.1 eTC_HDR_RET_t.

Comments

None

5.91 tchdr_sis_getAlfn

Description

This function gets ALFN.

Definition

```
HDRET tchdr_sis_getAlfn(eTC_HDR_ID_t id, stTC_HDR_SIS_ALFN_t* alfn);
```

Parameters

Parameter	I/O	Description
id	I	Refer to Chapter 3.3 eTC_HDR_ID_t.
alfn	O	Refer to Chapter 4.29 stTC_HDR_SIS_ALFN_t. Pointer to ALFN output data structure

Returns

Return Value	Description
HDRET	Return value Refer to Chapter 3.1 eTC_HDR_RET_t.

Comments

None

5.92 tchdr_sis_getStationID

Description

This function gets the station ID.

Definition

```
HDRET tchdr_sis_getStationID(eTC_HDR_ID_t id, stTC_HDR_SIS_STATION_ID_t* station_id);
```

Parameters

Parameter	I/O	Description
id	I	Refer to Chapter 3.3 eTC_HDR_ID_t.
station_id	O	Refer to Chapter 4.30 stTC_HDR_SIS_STATION_ID_t. Pointer to station ID

Returns

Return Value	Description
HDRET	Return value Refer to Chapter 3.1 eTC_HDR_RET_t.

Comments

None

5.93 tchdr_sis_getStationShortName

Description

This function gets the station name (short form).

Definition

```
HDRET tchdr_sis_getStationShortName(eTC_HDR_ID_t id, stTC_HDR_SIS_SHORT_NAME_t* short_name);
```

Parameters

Parameter	I/O	Description
id	I	Refer to Chapter 3.3 eTC_HDR_ID_t.
short_name	O	Refer to Chapter 4.31 stTC_HDR_SIS_SHORT_NAME_t. Pointer to the station short name output

Returns

Return Value	Description
HDRET	Return value Refer to Chapter 3.1 eTC_HDR_RET_t.

Comments

None

5.94 tchdr_sis_getStationLocation

Description

This function gets the station location.

Definition

```
HDRET tchdr_sis_getStationLocation(eTC_HDR_ID_t id, stTC_HDR_SIS_STATION_LOCATION_t* location);
```

Parameters

Parameter	I/O	Description
id	I	Refer to Chapter 3.3 eTC_HDR_ID_t.
location	O	Refer to Chapter 4.32 stTC_HDR_SIS_STATION_LOCATION_t. Pointer to the station location output

Returns

Return Value	Description
HDRET	Return value Refer to Chapter 3.1 eTC_HDR_RET_t.

Comments

None

5.95 tchdr_sis_getLeapSec

Description

This function gets SIS leap second information.

Definition

```
HDRET tchdr_sis_getLeapSec(eTC_HDR_ID_t id, stTC_HDR_SIS_LEAP_SEC_t* leap_sec);
```

Parameters

Parameter	I/O	Description
id	I	Refer to Chapter 3.3 eTC_HDR_ID_t.
leap_sec	O	Refer to Chapter 4.33 stTC_HDR_SIS_LEAP_SEC_t. Pointer to the output data structure

Returns

Return Value	Description
HDRET	Return value Refer to Chapter 3.1 eTC_HDR_RET_t.

Comments

None

5.96 tchdr_sis_getStationMessage

Description

This function gets the SIS station message.

Definition

```
HDRET tchdr_sis_getStationMessage(eTC_HDR_ID_t id, stTC_HDR_SIS_STATION_MSG_t* station_msg);
```

Parameters

Parameter	I/O	Description
id	I	Refer to Chapter 3.3 eTC_HDR_ID_t.
station_msg	O	Refer to Chapter 4.34 stTC_HDR_SIS_STATION_MSG_t. Pointer to the output data structure

Returns

Return Value	Description
HDRET	Return value Refer to Chapter 3.1 eTC_HDR_RET_t.

Comments

None

5.97 tchdr_sis_getLocalTime

Description

This function gets the SIS local time.

Definition

```
HDRET tchdr_sis_getLocalTime(eTC_HDR_ID_t id, stTC_HDR_SIS_LOCAL_TIME_t* local_time);
```

Parameters

Parameter	I/O	Description
id	I	Refer to Chapter 3.3 eTC_HDR_ID_t.
local_time	O	Refer to Chapter 4.35 stTC_HDR_SIS_LOCAL_TIME_t. Pointer to the output data structure

Returns

Return Value	Description
HDRET	Return value Refer to Chapter 3.1 eTC_HDR_RET_t.

Comments

None

5.98 tchdr_sis_getUniversalName

Description

This function gets the universal short name of station.

Definition

```
HDRET tchdr_sis_getUniversalName(eTC_HDR_ID_t id, stTC_HDR_SIS_UNIV_NAME_t* univ_name);
```

Parameters

Parameter	I/O	Description
id	I	Refer to Chapter 3.3 eTC_HDR_ID_t.
univ_name	O	Refer to Chapter 4.36 stTC_HDR_SIS_UNIV_NAME_t. Pointer to the output data structure

Returns

Return Value	Description
HDRET	Return value Refer to Chapter 3.1 eTC_HDR_RET_t.

Comments

None

5.99 tchdr_sis_getAvailProgramsList

Description

This function gets a list of available audio programs.

Definition

```
HDRET tchdr_sis_getAvailProgramsList(eTC_HDR_ID_t id, stTC_HDR_SIS_AVAIL_PROGRAMS_t*
available_programs);
```

Parameters

Parameter	I/O	Description
id	I	Refer to Chapter 3.3 eTC_HDR_ID_t.
available_programs	O	Refer to Chapter 4.38 stTC_HDR_SIS_AVAIL_PROGRAMS_t. Pointer to the output data structure

Returns

Return Value	Description
HDRET	Return value Refer to Chapter 3.1 eTC_HDR_RET_t.

Comments

None

5.100 tchdr_sis_getStationSlogan

Description

This function gets the station slogan.

Definition

```
HDRET tchdr_sis_getStationSlogan(eTC_HDR_ID_t id, stTC_HDR_SIS_STATION_SLOGAN_t* slogan);
```

Parameters

Parameter	I/O	Description
id	I	Refer to Chapter 3.3 eTC_HDR_ID_t.
slogan	O	Refer to Chapter 4.37 stTC_HDR_SIS_STATION_SLOGAN_t. Pointer to the output data structure

Returns

Return Value	Description
HDRET	Return value Refer to Chapter 3.1 eTC_HDR_RET_t.

Comments

None

5.101 tchdr_sis_getProgramInfo

Description

This function gets specified audio program information.

Definition

```
HDRET tchdr_sis_getProgramInfo(eTC_HDR_ID_t id, eTC_HDR_PROGRAM_t program_number,
stTC_HDR_SIS_PROGRAM_INFO_t* program_info);
```

Parameters

Parameter	I/O	Description
id	I	Refer to Chapter 3.3 eTC_HDR_ID_t.
program_number	I	Refer to Chapter 3.8 eTC_HDR_PROGRAM_t.. Specify the program number requested.
program_info	O	Refer to Chapter 4.39 stTC_HDR_SIS_PROGRAM_INFO_t. Pointer to the output data

Returns

Return Value	Description
HDRET	Return value Refer to Chapter 3.1 eTC_HDR_RET_t.

Comments

None

5.102 tchdr_sis_getAvailDataServList

Description

This function gets a list of available data services.

Definition

```
HDRET tchdr_sis_getAvailDataServList(eTC_HDR_ID_t id, stTC_HDR_SIS_AVAIL_DATA_SERVICES_t*
available_services);
```

Parameters

Parameter	I/O	Description
id	I	Refer to Chapter 3.3 eTC_HDR_ID_t.
available_services	O	Refer to Chapter 4.40 stTC_HDR_SIS_AVAIL_DATA_SERVICES_t. Pointer to the output data

Returns

Return Value	Description
HDRET	Return value Refer to Chapter 3.1 eTC_HDR_RET_t.

Comments

None

5.103 tchdr_sis_getAllDataServices

Description

This function gets information about all data services.

Definition

```
HDRET tchdr_sis_getAllDataServices(eTC_HDR_ID_t id, stTC_HDR_SIS_DATA_SERVICES_INFO_t*
data_services_info);
```

Parameters

Parameter	I/O	Description
id	I	Refer to Chapter 3.3 eTC_HDR_ID_t.
data_services_info	O	Refer to Chapter 4.41 stTC_HDR_SIS_DATA_SERVICES_INFO_t. Pointer to the output data

Returns

Return Value	Description
HDRET	Return value Refer to Chapter 3.1 eTC_HDR_RET_t.

Comments

None

5.104 tchdr_sis_getDataServicesType

Description

This function gets the specified data service information.

Definition

```
HDRET tchdr_sis_getDataServicesType(eTC_HDR_ID_t id, U32 service_type,
stTC_HDR_SIS_DATA_SERVICES_INFO_t* data_services_info);
```

Parameters

Parameter	I/O	Description
id	I	Refer to Chapter 3.3 eTC_HDR_ID_t.
service_type	I	Service type
data_services_info	O	Refer to Chapter 4.41 stTC_HDR_SIS_DATA_SERVICES_INFO_t. Pointer to the output data

Returns

Return Value	Description
HDRET	Return value Refer to Chapter 3.1 eTC_HDR_RET_t.

Comments

None

5.105 tchdr_sis_getExciterCoreVer

Description

This function gets a pointer to the exciter core version string.

Definition

```
HDRET tchdr_sis_getExciterCoreVer(eTC_HDR_ID_t id, stTC_HDR_SIS_TX_VER_STR_t* version_string);
```

Parameters

Parameter	I/O	Description
id	I	Refer to Chapter 3.3 eTC_HDR_ID_t.
version_string	O	Refer to Chapter 4.42 stTC_HDR_SIS_TX_VER_STR_t. Pointer to the output data

Returns

Return Value	Description
HDRET	Return value Refer to Chapter 3.1 eTC_HDR_RET_t.

Comments

None

5.106 tchdr_sis_getExciterManufVer

Description

This function gets a pointer to the exciter manufacturer string.

Definition

```
HDRET tchdr_sis_getExciterManufVer(eTC_HDR_ID_t id, stTC_HDR_SIS_TX_MANUF_VER_t* version_struct);
```

Parameters

Parameter	I/O	Description
id	I	Refer to Chapter 3.3 eTC_HDR_ID_t.
version_struct	O	Refer to Chapter 4.43 stTC_HDR_SIS_TX_MANUF_VER_t. Pointer to the output data

Returns

Return Value	Description
HDRET	Return value Refer to Chapter 3.1 eTC_HDR_RET_t.

Comments

None

5.107 tchdr_sis_getImporterCoreVer

Description

This function gets a pointer to the importer core version string.

Definition

```
HDRET tchdr_sis_getImporterCoreVer(eTC_HDR_ID_t id, stTC_HDR_SIS_TX_VER_STR_t* version_string);
```

Parameters

Parameter	I/O	Description
id	I	Refer to Chapter 3.3 eTC_HDR_ID_t.
version_string	O	Refer to Chapter 4.42 stTC_HDR_SIS_TX_VER_STR_t. Pointer to the output data

Returns

Return Value	Description
HDRET	Return value Refer to Chapter 3.1 eTC_HDR_RET_t.

Comments

None

5.108 tchdr_sis_getImporterManufVer

Description

This function gets a pointer to the manufacturer version string.

Definition

```
HDRET tchdr_sis_getImporterManufVer(eTC_HDR_ID_t id, stTC_HDR_SIS_TX_MANUF_VER_t* version_struct);
```

Parameters

Parameter	I/O	Description
id	I	Refer to Chapter 3.3 eTC_HDR_ID_t.
version_struct	O	Refer to Chapter 4.43 stTC_HDR_SIS_TX_MANUF_VER_t. Pointer to the output data

Returns

Return Value	Description
HDRET	Return value Refer to Chapter 3.1 eTC_HDR_RET_t.

Comments

None

5.109 tchdr_sis_flush

Description

This function flushes out the existing SIS data and starts collecting SIS data afresh.

Definition

```
HDRET tchdr_sis_flush(eTC_HDR_ID_t id);
```

Parameters

Parameter	I/O	Description
id	I	Refer to Chapter 3.3 eTC_HDR_ID_t.

Returns

Return Value	Description
HDRET	Return value Refer to Chapter 3.1 eTC_HDR_RET_t.

Comments

None

6 CALLBACK FUNCTIONS

This chapter describes callback functions regarding the results of running the HD Radio framework API.

Table 6.1 Callback Function

Name	Description
void(*pfnNotificationCallBack)	Notify the result of executing the HD Radio framework APIs.
void(*pfnAudioQueueCallBack)	Notify the audio queue.

6.1 pfnNotificationCallBack

Description

This is a callback function to notify the result and data of executing the HD Radio framework APIs.

Definition

```
void(*pfnNotificationCallBack)(U32 notifyID, U32 *pArg, void **pData, S32 errorCode)
```

Parameters

Parameter	Description
notifyID	Generated event notification ID Refer to Chapter 3.9 eTC_HDR_NOTIFY_t.
pArg	Data associated with the event
pData	Data pointer associated with the event
errorCodes	Event error code

Notifications

Notification ID	Result	Description
eTC_HDR_NOTIFY_OPEN	None	*HD Radio* framework is opened.
eTC_HDR_NOTIFY_AUDIO_MODE	pArg	[0]: Audio mode 0: Blending audio 1: Analog audio 2: Digital audio 3: Split audio (Left: Digital right audio, Right: Analog right audio)
eTC_HDR_NOTIFY_TUNE	pArg	[0]: HD Radio ID 0: Main 1: MRC 2: Back Scan (BS) Refer to Chapter 3.3 eTC_HDR_ID_t. [1]: Band changing flag 0: Not changed 1: Changed [2]: Frequency changing flag 0: Not changed 1: Changed [3]: Sample rate changing flag 0: Not changed 1: Changed
eTC_HDR_NOTIFY_MUTE	pArg	[0]: Mute status 0: Off 1: On
eTC_HDR_NOTIFY_SIGNAL_STATUS	pData	*(pData+0): Signal status Refer to Chapter 4.6 stTC_HDR_SIGNAL_STATUS_t.

Comments

None

Header file

None

Example Code

Refer to tchdradiocui_getTcHdrNotificationCallBack() function in tchdr_cui_if.c (src\example) file.

6.2 pfnAudioQueueCallBack

Description

This is a callback function to notify an audio queue.

Definition

```
void(*pfnAudioQueueCallBack)(void *pOutBuf, S32 frame, U32 samplerate)
```

Parameters

Parameter	Description
*pOutBuf	Audio output PCM buffer pointer This is a stTC_HDR_PCM_t array pointer.
frame	Audio output PCM frame size
samplerate	Audio output PCM sample rate The value is fixed as 44100 Hz.

Comments

When the audio output data is accumulated, this callback function is called periodically. Write the received PCM data to the audio device.

Header file

tchdr_callback_conf.h

7 SEQUENCE DIAGRAM

The recommended sequence of HD Radio framework is as follows:

1. Initialize the tuner based on the specification of customer application.
2. Get the information about modulation mode and I/Q I2S sample rate from the tuner.
3. The operation sequence of HD Radio framework is as follows:
   Init → Open → Tune → Operation of customer's application → Close → Deinit

7.1 Init

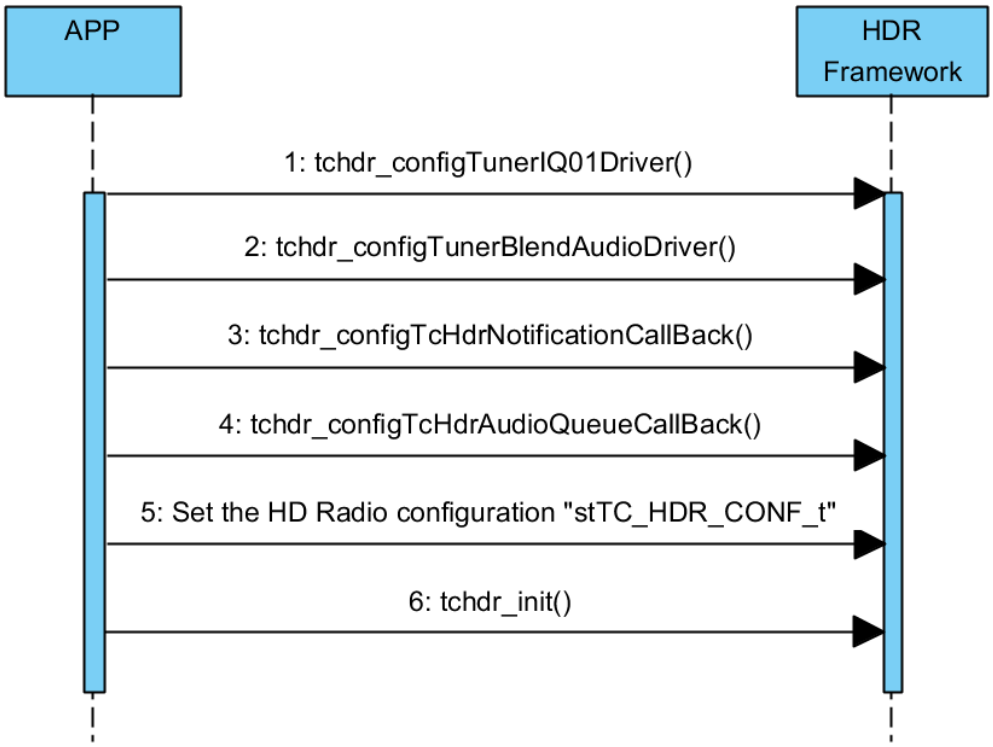

Figure 7.1 Diagram of Initializing HD Radio Framework

7.2 Open

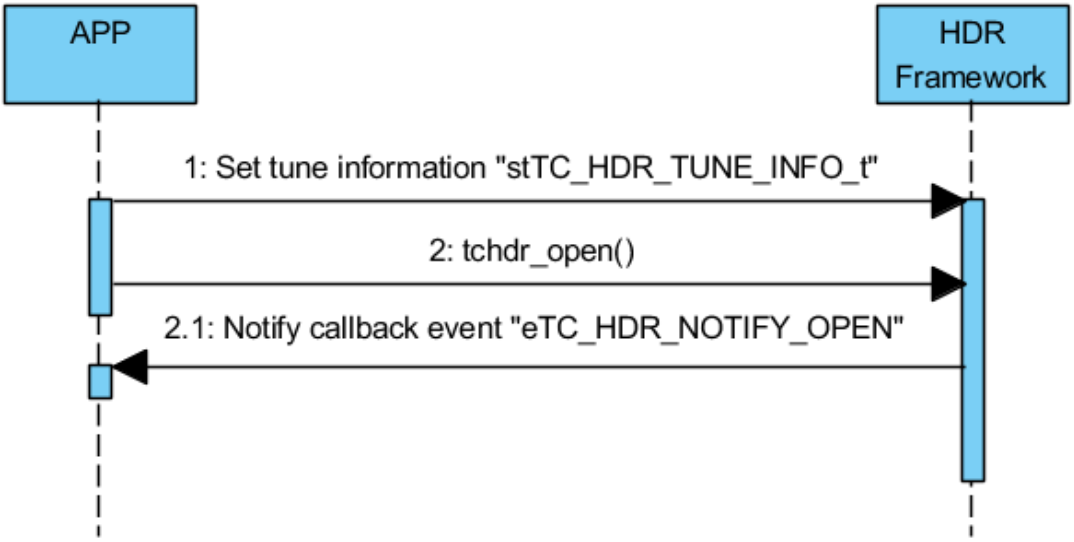

Figure 7.2 Diagram of Opening HD Radio Framework

7.3 Tune

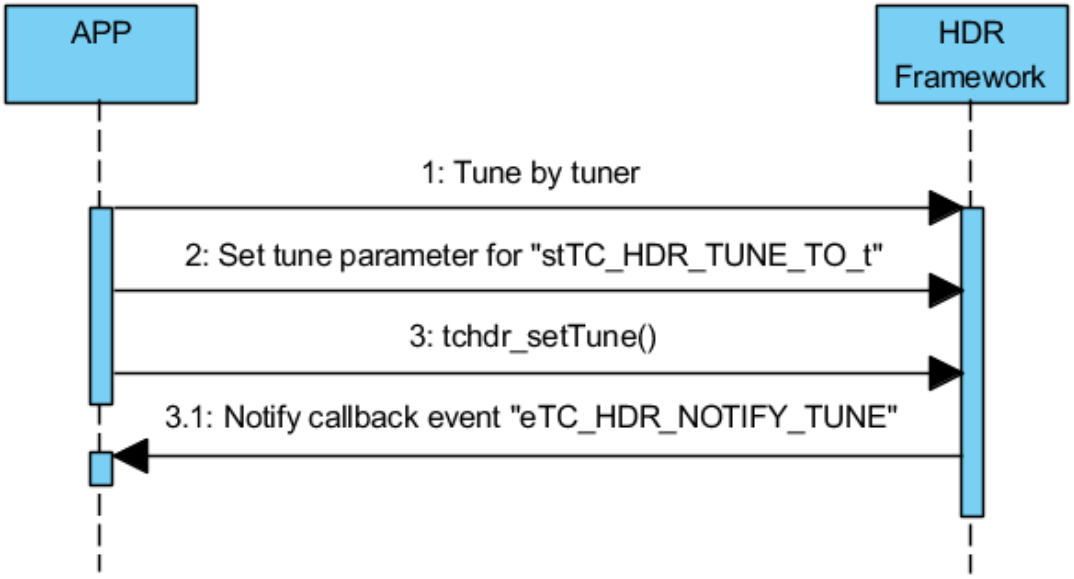

Figure 7.3 Diagram of Tuning HD Radio Framework

7.4 Close

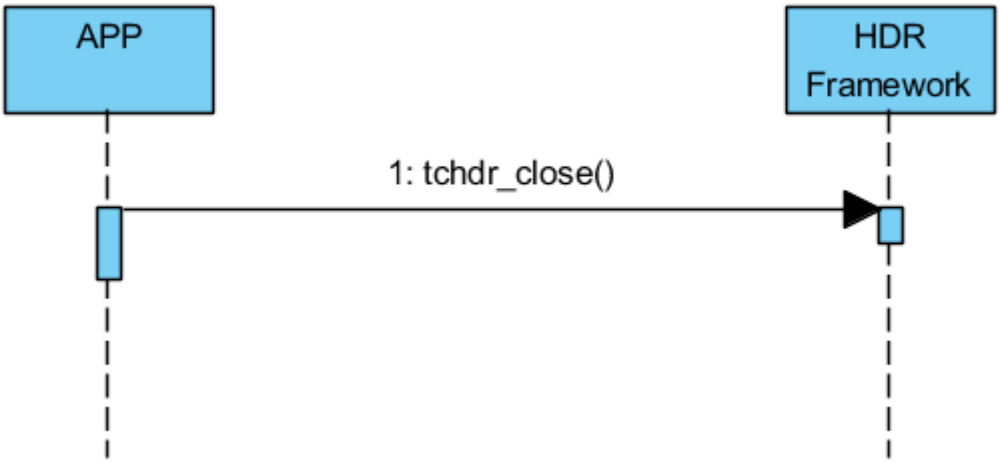

Figure 7.4 Diagram of Closing HD Radio Framework

7.5 Deinit

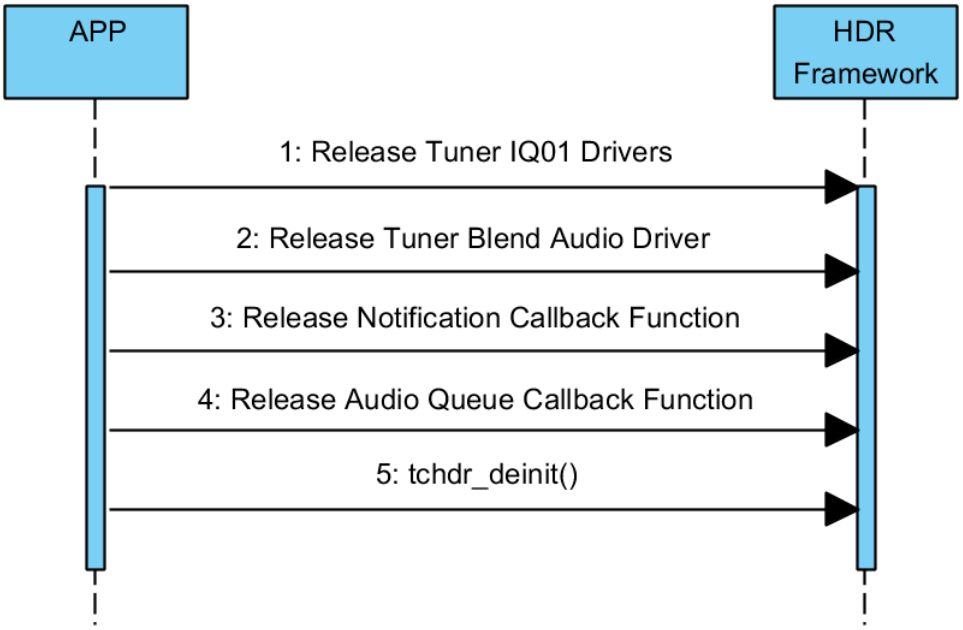

Figure 7.5 Diagram of Deinitializing HD Radio Framework

8 REFERENCES

[1] Contact Telechips for more details: sales@telechips.com

9 REVISION HISTORY

Rev. 1.50: 2021-04-07

- Updated
  - Chapter 1.3:
    - Table 1.2: HD Radio Framework
  - Chapter2.1: Variable Type
    - Deleted NULL definition
  - Chapter 3: Table 3.1
  - Chapter 4: Table 4.1
  - Chapter 5: Table 5.1
  - Chapter 5.24: tchdr_setBlendAllParams
    - Changed structure pointer input to structure input
  - Chapter5.28:tchdr_setBlendAllAdvParams
    - Changed structure pointer input to structure input
  - Chapter5.41: tchdr_configTcHdrNotificationCallBack
    - Changed some parameter type to const
  - Chapter 5.45: tchdr_aas_enablePorts
    - Changed port_list parameter type to const
  - Chapter5.46: tchdr_aas_disablePorts
    - Changed port_list parameter type to const

- Deleted
  - Chapter 5.66: tchdr_alert_getToneAudio

Rev. 1.41: 2021-11-02

- Updated
  - Chapter 5.35: tchdr_getDefaultThreadPriority
    - Changed function return argument from void to HDRET
  - Chapter 5.36: tchdr_getDefaultThreadNicePrioritytchdr_setThreadPriority
    - Changed function return argument from void to HDRET
  - Chapter 5.37: tchdr_getThreadPriority
    - Changed function return argument from void to HDRET

Rev. 1.40: 2021-10-22

- Updated
  - Chapter 1: Table 1.1 Available Chipset
    - Deleted TCC802x, TCC897x
  - Chapter 4: Table 4.1 Definition of Structure Type
    - Changed from stTC_HDR_PSD_XHDR_BODY_t to stTC_HDR_PSD_XHDR_PARAM_t
    - Changed from stTC_HDR_PSD_XHDR_t to stTC_HDR_PSD_XHDR_PARAM_t
  - Chapter 4.1: stTC_HDR_THREAD_PR_t
    - Changed to a single thread priority structure
  - Chapter 4.8: stTC_HDR_STATUS_t
    - Changed from pty[8] to pty[eTC_HDR_PROGRAM_MAX]
  - Chapter 4.22: stTC_HDR_PSD_XHDR_PARAM_t
    - Changed from stTC_HDR_PSD_XHDR_PARAM_t to stTC_HDR_PSD_XHDR_PARAM_t
  - Chapter 4.24: stTC_HDR_PSD_t
    - Changed from stTC_HDR_PSD_XHDR_t to stTC_HDR_PSD_XHDR_FRAME_t
  - Chapter 4.42: stTC_HDR_SIS_TX_VER_STR_t
    - Changed from string to verstr
  - Chapter 4.44: stTC_HDR_SIS_t
    - Changed text[] of shortName from U8 to S8
    - Changed text[] of universalName from U8 to S8
  - Chapter 5.34: tchdr_setThreadPriority
    - Changed function input structure arguments
  - Chapter 5.35: tchdr_getDefaultThreadPriority
    - Changed function input structure arguments

- Chapter 5.37: tchdr_getThreadPriority
  - Changed function input structure arguments

- Added
  - Chapter 3.13: eTC_HDR_THREAD_t
  - Chapter 4.10: stTC_HDR_PTY_t
  - Chapter 4.23: stTC_HDR_PSD_XHDR_FRAME_t
  - Chapter 5.21: tchdr_getProgramType
  - Chapter 5.36: tchdr_getDefaultThreadNicePriority

- Deleted
  - Chapter 4.23 stTC_HDR_PSD_XHDR_t

Rev. 1.31: 2021-07-26

- Updated
  - Chapter 1: Table 1.2: Library version
  - Chapter 5.13 tchdr_enableLotNotification
    - Changed service_number to progBitmask

Rev. 1.30: 2021-05-12

- Updated
  - Chapter 1: Table 1.1
  - Chapter 3.1: eTC_HDR_RET_t enum
    - Added 'eTC_HDR_RET_NG_AAS_NO_COMPLETE_OBJECT = -218,'
  - Chapter 3.9: eTC_HDR_NOTIFY_t enum
    - Added 'eTC_HDR_NOTIFY_ALERT = 125'
    - Added 'eTC_HDR_NOTIFY_LOT = 126'
  - Chapter 3.16: eTC_HDR_PSD_FIELD_t enum
    - Added 'eTC_HDR_PSD_XHDR'
  - Chapter 3.20: eTC_HDR_PSD_BITMASK_t enum
    - Added 'eBITMASK_PSD_XHDR = 0x80'
  - Chapter 4: Table 4.1
  - Chapter 4.16: eTC_HDR_PSD_FIELD_t structure
    - Added 'Image/png' in mime_hash
  - ~~Chapter 4.23: stTC_HDR_PSD_t structure~~
    - ~~Added 'stTC_HDR_PSD_XHDR_t xhdr'~~
  - Chapter 4.31: stTC_HDR_SIS_SHORT_NAME_t structure
    - Changed 'eTC_HDR_SIS_SHORT_NAME_t' to 'stTC_HDR_SIS_SHORT_NAME_t'
  - Chapter 5: Table 5.1
  - Chapter 5.86: Description of parameters
  - Chapter 5.93: Definition, description of parameters
  - Chapter 6.1: Description
    - Changed 'Example Code'
  - Chapter 7.2: Figure 7.2
  - Chapter 7.3: Figure 7.3

- Added
  - Chapter 4.17: stTC_HDR_LOT_t structure
  - Chapter 4.22: stTC_HDR_PSD_XHDR_PARAM_t structure
  - Chapter ~~Chapter 4.23: stTC_HDR_PSD_t structure~~
  - Chapter 5.13: tchdr_enableLotNotification() API function
  - Chapter 5.14: tchdr_enableAlertNotification() API function

Rev. 1.20: 2021-01-26

- Updated
  - Chapter 1.3: Table 1.2
  - Chapter 3.1: eTC_HDR_RET_t enum
    - Changed 'eTC_HDR_RET_NG_NOT_ENABLED' value from '-1' to '-10'

- Added 'eTC_HDR_RET_NG_EVENT_ERROR = -66,'
- Added 'eTC_HDR_RET_NG_DEINIT = -67,'
- Added 'eTC_HDR_RET_NG_ALREADY_CLOSE = -68,'

- Chapter 4.7: stTC_HDR_PROG_BITMAP_t structure
  - Added internal structure name 'prog'
- Chapter 4.20: stTC_HDR_PSD_FIELDS_t structure
  - Added internal structure name 'field'
- Chapter 4.28: stTC_HDR_SIS_ENABLED_BASIC_TYPES_t structure
  - Added internal structure name 'type'
- Chapter 4.30: stTC_HDR_SIS_STATION_ID_t structure
  - Added internal structure name 'field' and 'id'
- Chapter 4.44: stTC_HDR_SIS_t structure
  - Added internal structure name 'type' of the stationID
- Chapter 5.6: tchdr_setAudioMode function
  - Changed parameter name from 'mode' to 'audioMode'
- Chapter 5.24: tchdr_setBlendAllParams function
  - Deleted the const type of the parameter
- Chapter 5.26: tchdr_setBlendParam function
  - Changed the 2nd parameter name from 'value' to 'param_value'
- Chapter 5.27: tchdr_getBlendParam function
  - Changed the 2nd parameter name from '*value' to '*param_value'
- Chapter 5.28: tchdr_setBlendAllAdvParams function
  - Deleted the const type of the parameter
- Chapter 5.30: tchdr_setBlendAdvParam function
  - Changed the 2nd parameter name from 'value' to 'param_value'
- Chapter 5.31: tchdr_getBlendAdvParam function
  - Changed the 2nd parameter name from '*value' to '*param_value'
- Chapter 5.45: tchdr_aas_enablePorts function
  - Deleted the const type of the 2nd parameter
- Chapter 5.46: tchdr_aas_disablePorts function
  - Deleted the const type of the 2nd parameter

■ Changed
- Document title changed from "TCCxxxx ~~Automotive~~ Common-API Specification for HD Radio"

Rev. 1.11: 2020-08-28

■ Updated
- Chapter 3.1: Header file
- Chapter 3.18 ~ Chapter 3.30: Header file
- Chapter 3.31 eTC_HDR_AAS_PORT_MODE_t: Definition names

Rev. 1.10: 2020-08-26

■ Updated
- Chapter 1: Table 1.1
- Chapter 3.1: 'eTC_HDR_RET_t'
  - Changed the 'eTC_HDR_RET_NG_AUDIO_RESAMPLER_INIT' to the 'eTC_HDR_RET_NG_AUD_RESAMPLER_INIT'
  - Changed the 'eTC_HDR_RET_NG_AUDIO_RESAMPLER_OUTPUT' to the 'eTC_HDR_RET_NG_AUD_RESAMPLER_OUTPUT'
  - Changed the 'eTC_HDR_RET_NG_AUDIO_RESAMPLER_HANDLER' to the 'eTC_HDR_RET_NG_AUD_RESAMPLER_HANDLER'
  - Added 'eTC_HDR_RET_NG_IQ01IN_XRUN = -60,'
  - Added 'eTC_HDR_RET_NG_IQ23IN_XRUN = -61,'
  - Added 'eTC_HDR_RET_NG_IQ_INPUT_DRIVER = -62,'
  - Added 'eTC_HDR_RET_NG_INVALID_BAND = -63,'
  - Added 'eTC_HDR_RET_NG_MUTEX_LOCK = -64,'
  - Added 'eTC_HDR_RET_NG_MUTEX_UNLOCK = -65,'
  - Changed the 'eTC_HDR_RET_NG_AAS_MAX_NUM_OF_AAS_PORTS_ALREADY_ENABLED' to the 'eTC_HDR_RET_NG_AAS_MAX_AAS_PORTS_ALREADY_ENABLED'
  - Changed the 'eTC_HDR_RET_NG_AAS_MAX_NUM_OF_LOT_PORTS_ALREADY_ENABLED' to the 'eTC_HDR_RET_NG_AAS_MAX_LOT_PORTS_ALREADY_ENABLED'
- Chapter 7.1: Figure 7.1

- Chapter 7.5: Figure 7.5

- Added
  - Chapter 2.1 Variable Type
  - Chapter 5.22 tchdr_setAutoAudioAlignEnable
  - Chapter 5.42 tchdr_configTcHdrAudioQueueCallBack
  - Chapter 6.2 pfnAudioQueueCallBack

- Deleted
  - Chapter 5.19 tchdr_getBlendAudioStreamOutput

Rev. 1.09: 2020-02-08

- 3.1: Update the 'eTC_HDR_RET_t'

Rev. 1.08: 2019-10-25

- Delete the ' tchdr_setBlendDigitalAudioDelay()' API function
- Delete the ' tchdr_getBlendDigitalAudioDelay()' API function
- 5.13: Change a mute API name to 'tchdr_setAudioMute()' from 'tchdr_setMute()'
- 5.16: Add the ' tchdr_setAudioMuteFader()' API function
- 5.17: Add the ' tchdr_getAudioMuteFader()' API function

Rev. 1.07: 2019-09-23

- 3.4: Add 'eTC_HDR_IDLE_BAND' in the 'eTC_HDR_BAND_t' enum
- 3.5: Change enum value in the 'eTC_HDR_BBSRC_RATE_t'
- 4.1: Add the 'stTC_HDR_THREAD_PR_t' structure
- Delete the 'tchdr_releaseTunerIQ01Driver()'
- Can be released with ' tchdr_configTunerIQ01Driver()'
- Delete the 'tchdr_releaseTunerIQ23Driver()'
- Can be released with ' tchdr_configTunerIQ23Driver()'
- Delete the 'tchdr_releaseTunerBlendAudioDriver()'
- Can be released with 'tchdr_configTunerBlendAudioDriver()'
- 5.28: Add the ' tchdr_setThreadPriority()' API function
- 5.29: Add the ' tchdr_getDefaultThreadPriority()' API function
- 5.30: Add the ' tchdr_getThreadPriority()' API function
- 5.37: Change the arguments of the 'tchdr_cb_setTune()' API function
- 7.2: Change deinit sequence diagram

Rev. 1.06: 2019-09-03

- 3.6: Fix the 'eTC_HDR_SIGNAL_STATUS_t'

Rev. 1.05: 2019-08-07

- 3.1:Update the 'eTC_HDR_RET_t'
- 5.54: Update the return value of the ' tchdr_aas_getLotObjectBody()' API function
- 5.15: Add the ' tchdr_setAnalogAudioMute()' API function
- 3.5: Add the ' eTC_HDR_BBSRC_RATE_t' enum
- 4.1: Replace the float type of 'stTC_HDR_IQ_t' with 'eTC_HDR_BBSRC_RATE_t'
- 4.3: Replace the float type of ' stTC_HDR_TUNE_TO_t' with 'eTC_HDR_BBSRC_RATE_t'

Rev. 1.04: 2019-06-03

- 5.14: Add the 'tchdr_setAudioCtrl() 'API function

Rev. 1.03: 2019-05-31

- Delete the 'eTC_HDR_DATA_STATUS_t ' enum
- Delete the 'eTC_HDR_CHAR_TYPE_t' enum
- 3.1: Change 'eTC_HDR_RET_NG_GET_PROGRAME' to 'eTC_HDR_RET_NG_GET_PROGRAM'
- Reinforce the explanations for chapter 3 and 4.

Rev. 1.02: 2019-05-20

- 3.9: Change the 'eTC_HDR_Notify_t' to 'eTC_HDR_NOTIFY_t'
- 3.7: Delete the 'eTC_HDR_IDLE_BAND' of the 'eTC_HDR_BAND_t' enum
- Delete the 'tchdr_setBlendDigitalAudioDelay()'
- Delete the 'tchdr_getBlendDigitalAudioDelay()'
- Add all blend parameters setting
  - 3.13: Add the 'eTC_HDR_BLEND_THRESH_SEL_t'
  - 3.14: Add the 'eTC_HDR_BLEND_PARAMS_t'
  - 3.15: Add the 'eTC_HDR_BLEND_ADV_PARAMS_t'
  - 4.9: Add the 'stTC_HDR_BLEND_PARAMS_t'
  - 4.10: Add the 'stTC_HDR_BLEND_ADV_PARAMS_t'
  - 5.16: Add the 'tchdr_setBlendTransitionTime()' API function
  - 5.17: Add the 'tchdr_setBlendAllParams()' API function
  - 5.18: Add the 'tchdr_getBlendAllParams()' API function
  - 5.19: Add the 'tchdr_setBlendParam()' API function
  - 5.20: Add the 'tchdr_getBlendParam()' API function
  - 5.21: Add the 'tchdr_setBlendAllAdvParams()' API function
  - 5.22: Add the 'tchdr_getBlendAllAdvParams()' API function
  - 5.23: Add the 'tchdr_setBlendAdvParam()' API function
  - 5.24: Add the 'tchdr_getBlendAdvParam()' API function

Rev. 1.01: 2019-05-07

- 3.9: Update the 'eTC_HDR_Notify_t'.
- 6: Add information on the call back function and notification for user event.

Rev. 1.00: 2019-03-30

- First version release

DISCLAIMER

All information and data contained in this material are without any commitment, are not to be considered as an offer for conclusion of a contract, nor shall they be construed as to create any liability. Any new issue of this material invalidates previous issues. Product availability and delivery are exclusively subject to our respective order confirmation form; the same applies to orders based on development samples delivered. By this publication, Telechips, Inc. does not assume responsibility for patent infringements or other rights of third parties that may result from its use.

Further, Telechips, Inc. reserves the right to revise this publication and to make changes to its content, at any time, without obligation to notify any person or entity of such revisions or changes.
No part of this publication may be reproduced, photocopied, stored on a retrieval system, or transmitted without the express written consent of Telechips, Inc.
This product is designed for general purpose, and accordingly customer be responsible for all or any of intellectual property licenses required for actual application. Telechips, Inc. does not provide any indemnification for any intellectual properties owned by third party.
Telechips, Inc. can not ensure that this application is the proper and sufficient one for any other purposes but the one explicitly expressed herein. Telechips, Inc. is not responsible for any special, indirect, incidental or consequential damage or loss whatsoever resulting from the use of this application for other purposes.

COPYRIGHT STATEMENT

Important Notice

This material may include technology owned by the 3rd party licensor and the technology may be subject to its associated licenses. It is solely customer's responsibility to identify and comply with such licenses. No other licenses are granted or implied by Telechips with making available this material.

For customers who use licensed Codec ICs and/or licensed codec firmware of mp3:
"Supply of this product does not convey a license nor imply any right to distribute content created with this product in revenue-generating broadcast systems (terrestrial. Satellite, cable and/or other distribution channels), streaming applications(via internet, intranets and/or other networks), other content distribution systems(pay-audio or audio-on-demand applications and the like) or on physical media(compact discs, digital versatile discs, semiconductor chips, hard drives, memory cards and the like). An independent license for such use is required. For details, please visit http://mp3licensing.com".

For customers who use other firmware of mp3:
"Supply of this product does not convey a license under the relevant intellectual property of Thomson and/or Fraunhofer Gesellschaft nor imply any right to use this product in any finished end user or ready-to-use final product. An independent license for such use is required. For details, please visit http://mp3licensing.com".

For customers who use Digital Wave DRA solution:
"Supply of this implementation of DRA technology does not convey a license nor imply any right to this implementation in any finished end-user or ready-to-use terminal product. An independent license for such use is required."

For customers who use DTS technology:
"This product made under license to certain U.S. patents and/or foreign counterparts."
"© 1996 – 2011 DTS, Inc. All rights reserved."

For customers who use Dolby technology:
"Supply of this Implementation of Dolby technology does not convey a license nor imply a right under any patent, or any other industrial or intellectual property right of Dolby Laboratories, to use this Implementation in any finished end-user or ready-to-use final product. It is hereby notified that a license for such use is required from Dolby Laboratories."

For customers who use Google technology:
"Copyright © 2013 Google Inc. All rights reserved."

For customers who use MS technology:
"This product is subject to certain intellectual property rights of Microsoft and cannot be used or distributed further without the appropriate license(s) from Microsoft."