Synway Voice Board

Programmer's Manual

Version 5.4.4.2

Synway Information Engineering Co., Ltd

www.synway.net

Contents

Copyright Declaration

SynCTI Driver Software License Agreement

1. Synway Information Engineering Co., Ltd (hereinafter referred to as 'Synway') owns the copyright of 'this software and its accessories, relative files and archives' (hereinafter referred to as 'this product'). No organization, enterprise, agency or individual may use this product without our authorization.
2. We authorize those who achieve the following two requirements to use this product for free:

   A. Using this product with hardware products purchased from Synway through a legitimate marketing channel;

   B. Promising not to transmit the whole or part of this product to any third party without prior permission from Synway.
3. Any organization, enterprise, agency or individual, except as otherwise subject to the second article of this license agreement, must obtain the written permission from Synway before using this product.
4. The authorized organizations, enterprises or individuals have no right to transfer the authorization.
5. The use of this product indicates that you have fully understood and accepted all terms in this license.

Revision History

Version	Date	Comments
Version 2.0	2001-12	Initial publication
Version 3.0.0.0	2003-07	Significant revision
Version 4.7.3.0	2006-12	Significant revision
Version 4.8.0.0	2007-07	Significant revision
Version 4.8.0.1	2007-08	No modification
Version 4.9.0.0	2007-10	Significant revision
Version 5.0.0.0	2007-12	Significant revision
Version 5.0.1.0	2008-04	Significant revision
Version 5.0.2.0	2008-08	Significant revision
Version 5.0.3.0	2009-01	Significant revision
Version 5.0.4.0	2009-05	Significant revision
Version 5.0.5.0	2009-07	Significant revision
Version 5.1.0.0	2009-11	Significant revision
Version 5.1.1.0	2009-12	Significant revision
Version 5.2.0.0	2010-06	Significant revision
Version 5.2.0.1	2010-08	Significant revision
Version 5.2.0.2	2010-08	Significant revision
Version 5.2.0.3	2010-10	Significant revision
Version 5.3.0.0	2010-12	Significant revision
Version 5.3.0.1	2011-01	Significant revision
Version 5.3.0.2	2011-03	Significant revision
Version 5.3.0.3	2011-04	Significant revision
Version 5.3.0.4	2011-04	Significant revision
Version 5.3.1.0	2011-07	Significant revision
Version 5.3.1.1	2011-10	Significant revision

Version 5.3.1.2	2011-12	Significant revision
Version 5.3.1.3	2012-02	Significant revision
Version 5.3.1.4	2012-04	Significant revision
Version 5.3.1.5	2012-07	Significant revision
Version 5.3.2.0	2012-10	Significant revision
Version 5.3.2.1	2012-12	Significant revision
Version 5.3.2.2	2013-03	Significant revision
Version 5.3.2.3	2013-06	Significant revision
Version 5.3.2.4	2013-08	Significant revision
Version 5.3.2.5	2013-11	Significant revision
Version 5.3.2.6	2014-03	Significant revision
Version 5.3.2.7	2014-07	Significant revision
Version 5.3.3.0	2014-11	Significant revision
Version 5.3.3.1	2015-04	Significant revision
Version 5.3.3.2	2015-11	Significant revision
Version 5.3.4.0	2016-03	Significant revision
Version 5.4.0.0	2016-08	Significant revision
Version 5.4.1.0	2017-01	Significant revision
Version 5.4.2.0	2017-05	Significant revision
Version 5.4.3.0	2018-08	Significant revision
Version 5.4.4.0	2020-04	Significant revision
Version 5.4.4.1	2020-07	Significant revision
Version 5.4.4.2	2021-03	Significant revision

Note: Only major revisions to this manual itself recorded herein. For detailed information about the documentation modification undergone with the upgrade of the SynCTI Driver Program, refer to the corresponding version of *SynCTI Upgrade Manual*.

Preface

Thank you for choosing the Synway boards. SynCTI is a unified driver program for the Synway boards. This manual, as the biggest help file for SynCTI, aims at those software engineers dedicated to API development and application. It can also be provided as a reference book for the salesmen as well as the installation and maintenance technicians who are using products from Synway to set up the CTI application system.

We offer the SynCTI - a unified driver program and development platform – to serve as the link between the Synway board hardware and software. SynCTI provides a unified API interface for its various board models so that the application system based on certain model can be easily extended to different environments.

This document consists of four chapters.

Chapter 1 introduces the fundamental knowledge about programming for voice boards, covering the structure, classification, system set-up of the Synway boards and the SynCTI driver program, the operation procedure and the precaution on it. Readers will get to know what the board can do and how to enable all board features via API functions after they read through this chapter. We suggest those who are in the first use of our products to read this chapter carefully before programming. You can surely skip over the content in no relation with your application. For example, if your application is for processing analog phone lines, you can ignore the words about digital trunks.

Chapter 2 elaborates the SynCTI's API functions, driver-generated events and relative data structures.

Chapter 3 describes all configuration items in detail for the SynCTI driver program and classifies them by function. Each function has the necessary configuration items as well as the advanced ones which are generally used to meet requirements from a few of particular environments and users.

Chapter 4 tells you how to configure and use the SS7 system.

Although Synway has scrupulously checked through this manual, but cannot guarantee the absence of errors and omissions. We sincerely apologize for any consequent inconvenience brought to you and will be very grateful if you kindly give your advice regarding amendments to this book.

1 Basic Knowledge on Synway Board Programming

1.1 Board Classification (CTI Series)

The Synway boards are divided into two categories: the CTI Series and the REC Series. The former serves as one party of a call in communication while the latter is the call-recording board mainly used for high-impedance recording in parallel. Both drivers for the CTI Series and the REC Series are contained in SynCTI.

The CTI Series includes five subseries:

SubSeries Name	Description
SHT	Use modular structure and analog lines.
SHD	Use digital trunks (E1/T1/J1).
SHV	Voice-alteration boards
SHN	VoIP boards
SHF	Fax boards
HMP	Host media processing

Below is a detailed list of board models in each subseries:

SubSeries	Bus	Board Model	Comments
SHT	USB	SHT-2A/USB	Referred to as 'USB Voice Box' for short
		SHT-4A/USB	
		SHT-2B/USB	
		SHT-4B/USB	
	PCI	SHT-16A-CT/PCI	
		SHT-120A-CT/PCI	Production stopped
		SHT-8B/PCI	
		SHT-8B/PCI/FAX	
		SHT-16B-CT/PCI	
		SHT-16B-CT/PCI/FAX	
		SHT-16B-CT/PCI/MP3	
		SHT-8C/PCI/FAX	
		SHT-8C/PCI/EC	
		SHT-16C-CT/PCI/FAX	
		SHT-16C-CT/PCI/EC	
	PCIe	SHT-16D-CT/PCIe	
	cPCI	SHT-120A-CT/cPCI	Production stopped
		SHT-16B-CT/cPCI	
		SHT-16B-CT/cPCI/FAX	
		SHT-16B-CT/cPCI/MP3	
SHD	PCI	SHD-30A-CT/PCI/SS1	SS1 support
		SHD-30A-CT/PCI/ISDN	SS1, ISDN support
		SHD-30A-CT/PCI/SS7	SS1, ISDN, SS7 support
		SHD-60A-CT/PCI/SS1	SS1 support
		SHD-60A-CT/PCI/ISDN	SS1, ISDN support
		SHD-60A-CT/PCI/SS7	SS1, ISDN, SS7 support
		SHD-120A-CT/PCI/SS1	SS1 support
		SHD-120A-CT/PCI/ISDN	SS1, ISDN support
		SHD-120A-CT/PCI/SS7	SS1, ISDN, SS7 support
		SHD-30B-CT/PCI/SS7/FAX	SS1, ISDN, SS7 support
		SHD-60B-CT/PCI/SS7/FAX	SS1, ISDN, SS7 support

		SHD-30C-CT/PCI	E1: SS1, ISDN, SS7 support; T1: ISDN support
		SHD-30C-CT/PCI/FAX	E1: SS1, ISDN, SS7 support; T1: ISDN support
		SHD-60C-CT/PCI	E1: SS1, ISDN, SS7 support; T1: ISDN support
		SHD-60C-CT/PCI/FAX	E1: SS1, ISDN, SS7 support; T1: ISDN support
		SHD-30D-CT/PCI	E1: SS1, ISDN, SS7 support; T1: ISDN support
		SHD-60D-CT/PCI	E1: SS1, ISDN, SS7 support; T1: ISDN support
		SHD-120D-CT/PCI	E1: SS1, ISDN, SS7 support; T1: ISDN support
		SHD-120D-CT/PCI/EC	E1: SS1, ISDN, SS7 support; T1: ISDN support
		SHD-120D-CT/PCI/CAS	SS1 support
		SHD-240D-CT/PCI	E1: SS1, ISDN, SS7 support; T1: ISDN support
		SHD-240D-CT/PCI/EC	E1: SS1, ISDN, SS7 support; T1: ISDN support
		SHD-240D-CT/PCI/CAS	SS1 support
		SHD-30E-CT/PCI(SSW)	E1: SS1, ISDN, SS7 support; T1: ISDN support
		SHD-30E-CT/PCI/FAX(SSW)	E1: SS1, ISDN, SS7 support; T1: ISDN support
		SHD-30E-CT/PCI/EC(SSW)	E1: SS1, ISDN, SS7 support; T1: ISDN support
		SHD-60E-CT/PCI(SSW)	E1: SS1, ISDN, SS7 support; T1: ISDN support
		SHD-60E-CT/PCI/FAX(SSW)	E1: SS1, ISDN, SS7 support; T1: ISDN support
		SHD-60E-CT/PCI/EC(SSW)	E1: SS1, ISDN, SS7 support; T1: ISDN support
		SHD-120E-CT/PCI(SSW)	E1: SS1, ISDN, SS7 support; T1: ISDN support
		SHD-120E-CT/PCI/FAX(SSW)	E1: SS1, ISDN, SS7 support; T1: ISDN support
		SHD-120E-CT/PCI/EC(SSW)	E1: SS1, ISDN, SS7 support; T1: ISDN support
		SHD-240E-CT/PCI(SSW)	E1: SS1, ISDN, SS7 support; T1: ISDN support
		SHD-240E-CT/PCI/FAX(SSW)	E1: SS1, ISDN, SS7 support; T1: ISDN support
		SHD-240E-CT/PCI/EC(SSW)	E1: SS1, ISDN, SS7 support; T1: ISDN support
	PCIe	SHD-30E-CT/PCIe	E1: SS1, ISDN, SS7 support; T1: ISDN support
		SHD-30E-CT/PCIe/FAX	E1: SS1, ISDN, SS7 support; T1: ISDN support
		SHD-30E-CT/PCIe/EC	E1: SS1, ISDN, SS7 support; T1: ISDN support
		SHD-60E-CT/PCIe	E1: SS1, ISDN, SS7 support; T1: ISDN support
		SHD-60E-CT/PCIe/FAX	E1: SS1, ISDN, SS7 support; T1: ISDN support
		SHD-60E-CT/PCIe/EC	E1: SS1, ISDN, SS7 support; T1: ISDN support
		SHD-120E-CT/PCIe	E1: SS1, ISDN, SS7 support; T1: ISDN support
		SHD-120E-CT/PCIe/FAX	E1: SS1, ISDN, SS7 support; T1: ISDN support
		SHD-120E-CT/PCIe/EC	E1: SS1, ISDN, SS7 support; T1: ISDN support
		SHD-240E-CT/PCIe	E1: SS1, ISDN, SS7 support; T1: ISDN support
		SHD-240E-CT/PCIe/FAX	E1: SS1, ISDN, SS7 support; T1: ISDN support
		SHD-240E-CT/PCIe/EC	E1: SS1, ISDN, SS7 support; T1: ISDN support
		SHD-240E-CT/PCIe/VAR	E1: SS1, ISDN, SS7 support; T1: ISDN support; 120 voice-alteration channels support
	cPCI	SHD-30A-CT/cPCI/SS7	SS1, ISDN, SS7 support
		SHD-60A-CT/cPCI/SS7	SS1, ISDN, SS7 support
		SHD-120A-CT/cPCI/SS7	SS1, ISDN, SS7 support
		SHD-240A-CT/cPCI	SS7 support
		SHD-480A-CT/cPCI	SS7 support
		SHD-30B-CT/cPCI/SS7/FAX	SS1, ISDN, SS7 support
		SHD-60B-CT/cPCI/SS7/FAX	SS1, ISDN, SS7 support
		SHD-240S-CT/cPCI	SS7 support
		SHD-480S-CT/cPCI	SS7 support
SHV	PCI	SHV-120A-CT/PCI	
		SHV-240A-CT/PCI	
	cPCI	SHV-240A-CT/cPCI	
SHN	PCI	SHN-32A-CT/PCI	SIP support
		SHN-8B-CT/PCI+	SIP support
		SHN-16B-CT/PCI+	SIP support
		SHN-32B-CT/PCI+	SIP support
		SHN-60B-CT/PCI+	SIP support

		SHN-120B-CT/PCI+	SIP support
	PCIe	SHN-60B-CT/PCIe+	SIP support
		SHN-120B-CT/PCIe+	SIP support
		SHN-120B-CT/PCIe/VAR	SIP support
		SHN-480C-CT/PCIe	SIP support
SHF	PCI	SHF-2D/PCI	V.34, V.17, V.29, V.27ter, V.8 support
		SHF-4D/PCI	V.34, V.17, V.29, V.27ter, V.8 support
	PCIe	SHF-4D/PCIe	V.34, V.17, V.29, V.27ter, V.8 support
HMP		SYNWAY-HMP	SIP support

1.2 Board Classification (REC Series)

The Synway boards are divided into two categories: the CTI Series and the REC Series. The former serves as one party of a call in communication while the latter is the call-recording board mainly used for high-impedance recording in parallel. Both drivers for the CTI Series and the REC Series are contained in SynCTI.

The REC Series includes five subseries:

SubSeries Name	Description
ATP	Use modular structure, applicable to high-impedance parallel recording of analog phone lines or other analog voice signals.
DTP	Support SS1/ISDN/SS7, applicable to high-impedance parallel recording of digital trunks simultaneously over signaling links and voice channels.
DST	Applicable to high-impedance parallel recording of digital phone lines (2B+D).
SHF	Decode the fax data recorded into voice files and obtain the actual fax image, generally used as accessories for the REC Series boards. The D-type boards support a high fax rate of 33.6k.
IPR	Applicable to the recording of the phones and switches over the VoIP line

Below is a detailed list of board models in each subseries:

SubSeries	Bus	Board Model	Comments
ATP	USB	SHT-2A/USB	Referred to as 'USB Recording Box' for short
		SHT-4A/USB	
		SHT-2B/USB	
		SHT-4B/USB	
	PCI	SHT-8A/PCI	
		SHT-8B/PCI	
		SHT-16A-CT/PCI	
		SHT-16B-CT/PCI	
		SHT-16B-CT/PCI/MP3	Hardware-based MP3 encoding support
		ATP-24A/PCI	
		ATP-24A/PCI+	Hardware-based GSM, MP3 and G.729A encoding support
	PCIe	ATP-24A/PCIe	
		ATP-24A/PCIe+	Hardware-based GSM, MP3 and G.729A encoding support
	cPCI	SHT-16B-CT/cPCI	
		SHT-16B-CT/cPCI/MP3	Hardware-based MP3 encoding support
DTP	PCI	SHD-30A-CT/PCI/FJ	With on-board high-impedance input circuit
		SHD-60A-CT/PCI/FJ	With on-board high-impedance input circuit
		SHD-30B-CT/PCI/FJ	With on-board high-impedance input circuit
		SHD-60B-CT/PCI/FJ	With on-board high-impedance input circuit

		DTP-30C/PCI	With on-board high-impedance input circuit
		DTP-30C/PCI+	With on-board high-impedance input circuit
		DTP-60C/PCI	With on-board high-impedance input circuit
		DTP-60C/PCI+	With on-board high-impedance input circuit
		DTP-120C/PCI	With on-board high-impedance input circuit
		DTP-120C/PCI+	With on-board high-impedance input circuit
	PCIe	DTP-30C/PCIe	With on-board high-impedance input circuit
		DTP-30C/PCIe+	With on-board high-impedance input circuit
		DTP-60C/PCIe	With on-board high-impedance input circuit
		DTP-60C/PCIe+	With on-board high-impedance input circuit
		DTP-120C/PCIe	With on-board high-impedance input circuit
		DTP-120C/PCIe+	With on-board high-impedance input circuit
DST	PCI	SHR-16DA-CT/PCI	PCI-X bus support
		SHR-24DA-CT/PCI	PCI-X bus support
		DST-24B/PCI	
		DST-24B/PCI(SSW)	
		DST-24B/PCI+	Hardware-based GSM, MP3 and G.729A encoding support
		DST-24B/PCI+(SSW)	Hardware-based GSM, MP3 and G.729A encoding support
	PCIe	DST-24B/PCIe	
		DST-24B/PCIe+	Hardware-based GSM, MP3 and G.729A encoding support
IPR		SynIPAnalyzer	USB-KEY and NTP support
		SynIPRecorder	USB-KEY and NTP support

1.3 SynCTI Supported CODECs

The following table displays all voice CODECs supported by the Synway boards and their characteristics.

CODEC	Abbreviation	Sampling Rate	bps	bytes/frame	bytes/sec	CODEC Engine	Coding Value
Unsigned 8-bit PCM	PCM8	8000 Hz	64 Kbps	1	8000	Host CPU	1
16-bit linear PCM	PCM16	8000 Hz	128 Kbps	2	16000	Host CPU	-2
A-law	A-law	8000 Hz	64 Kbps	1	8000	On-board DSP	6
μ-law	μ-law	8000 Hz	64 Kbps	1	8000	On-board DSP	7
IMA ADPCM	IMA ADPCM	8000 Hz	32 Kbps	256	4055	On-board DSP	17
VOX	VOX	8000 Hz	32 Kbps	256	4000	On-board DSP	23
MP3	MP3	8000 Hz	8 Kbps	72	1000	On-board DSP Host CPU	85
GSM 6.10	GSM	8000 Hz	13 Kbps	65	1625	Host CPU	49
G.729A	G.729A	8000 Hz	8 Kbps	10	1000	On-board DSP	131

Note:

(1) If the board is marked by '/MP3' in the model, it uses on-board DSP chips as the CODEC engine; otherwise it uses host CPU.

(2) VOX refers to the 'ADPCM' format used by Dialogic.

(3) Due to historical reasons, the coding value 65411 which ever represented G.729A is still valid in early versions and backward compatible. For G.729A recording in new versions, we suggest you use the coding value 131.

1.3.1 Transcoding Functions

ShPcmHandle.dll provides a number of transcoding functions which can be used without the need to initialize the SynCTI driver program first via the call of SsmStartCti. However, after the call of these functions, it is required to

invoke fPcm_Close to release resources. See below for details.

Functions	Description
fPcm_Mp3ConvertALaw	Transcodes MP3 files which are recorded by the Synway board to A-law files.
fPcm_Mp3ConvertULaw	Transcodes MP3 files to μ-law files.
fPcm_AdpcmToAlaw	Transcodes IMA ADPCM files which are recorded by the Synway board to A-law files.
fPcm_AdpcmToGsm	Transcodes IMA ADPCM files to GSM 6.10 files.
fPcm_AdpcmToMp3	Transcodes IMA ADPCM files to MP3 files.
fPcm_AlawToUlaw	Transcodes A-Law files recorded by the Synway board to μ-Law files.
fPcm_UlawToAlaw	Transcodes μ-Law files recorded by the Synway board to A-Law files.
fPcm_MemAdpcmToALAW	Transcodes the IMA ADPCM data which are recorded by the Synway board and saved in the buffer to A-law format.
fPcm_MemAdpcmToULAW	Transcodes the IMA ADPCM data which are recorded by the Synway board and saved in the buffer to μ-law format.
fPcm_AlawConvertPcm16	Transcodes A-law files to 16-bit linear PCM format.
fPcm_ALawConvertPcm8	Transcodes A-law files to unsigned 8-bit PCM format.
fPCM_AlawConvertGC8	Transcodes A-law files to G.729A files.
fPcm_ALawConvertMp3	Transcodes A-law files to MP3 files.
fPcm_ULawConvertMp3	Transcodes μ-Law files to MP3 files.
fPcm_ALawToVox	Transcodes A-Law files to vox files.
fPcm_MemAlawToPcm8	Transcodes A-law data saved in the buffer to unsigned 8-bit PCM format.
fPcm_MemAlawToPcm16 fPcm_MemMp3ToPcm16	Transcodes A-law/MP3 data saved in the buffer to unsigned 16-bit PCM format.
fPcm_MemGSMToPcm16	Transcodes GSM data saved in the buffer to 16-bit PCM format.
fPcm_MemGSMToPcm8	Transcodes GSM data saved in the buffer to unsigned 8-bit PCM format.
fPcm_MemGSMToUlaw	Transcodes GSM data saved in the buffer to μ-law format.
fPcm_Pcm16ConvertALaw	Transcodes 16-bit linear PCM files to A-law files.
fPcm_MemPcm8ToAlaw	Transcodes unsigned 8-bit PCM data saved in the buffer to A-law format.
fPcm_MemAlawToUlaw	Transcodes A-Law data saved in the buffer to μ-law format.
fPcm_MemUlawtoAlaw	Transcodes μ-Law data saved in the buffer to A-Law format.
fPcm_MemPcm16ToAlaw	Transcodes unsigned 16-bit PCM data saved in the buffer to A-law format.
fPcm_GC8Convert	Tanscodes G.729A files to other formats.
fPcm_Vox6KTo8K fPcm_Vox8KTo6K	Enable the sampling rate of VOX files to convert between 6K and 8K.
fPcm_ULawConvertGSM	Transcodes μ-Law files to GSM files.
fPCM_Pcm16ConvertG729A	Transcodes 16-bit linear PCM files to G.729A files.
fPcm_Pcm8ConvertGSM	Transcodes 8-bit PCM files to GSM files.
fPcm_GSMToMp3	Transcodes GSM files to MP3 files.
fPcm_G729AConvert	Tanscodes G.729A files to other formats.
fPCM_MemULawToG729A	Tanscodes μ-Law data saved in the buffer to G.729A format.
fPcm_MemG729AToPcm8 fPcm_MemG729AToPcm16	Converts the G.729A data saved in the buffer area to the PCM16 or PCM8 format
fPcm_NotchFilter_ULAW	Filters the specific frequency from a μ-Law formatted voice file which is recorded by the Synway board using the notch filtering algorithm
fPcm_ALawToAdpcm	Converts the A-law formatted file recorded by Synway boards to be IMA

	ADPCM formatted file
fPcm_MemPcm16ToGSMEx	Converts the 16-bit PCM formatted data stored in the buffer area to be the GSM formatted data

1.3.2 Board Supported CODECs

All CODECs supported by the Synway boards in voice playback and recording operation are shown as follows:

Series	Board Model	PCM8		PCM16		A-law		μ-law		IMA ADPCM		VOX		MP3		GSM		G.729A	
		DEC	COD	DEC	COD	DEC	COD	DEC	COD	DEC	COD	DEC	COD	DEC	COD	DEC	COD	DEC	COD
SHT	SHT-2A/USB	☆	☆	☆	☆	√	√	√	√	—	—	√	√	★	★	★	★	★	—
	SHT-4A/USB	☆	☆	☆	☆	√	√	√	√	—	—	√	√	★	★	★	★	★	—
	SHT-2B/USB	☆	☆	☆	☆	√	√	√	√	—	—	√	√	★	★	★	★	★	—
	SHT-4B/USB	☆	☆	☆	☆	√	√	√	√	—	—	√	√	★	★	★	★	★	—
	SHT-16A-CT/PCI	☆	☆	☆	☆	√	√	√	√	√	√	☆	—	★	★	★	★	★	—
	SHT-120A-CT/PCI	☆	☆	☆	☆	√	√	√	√	√	√	☆	—	★	★	★	★	★	—
	SHT-8B/PCI	☆	☆	☆	☆	√	√	√	√	√	√	☆	—	★	★	★	★	★	—
	SHT-8B/PCI/FAX	☆	☆	☆	☆	√	√	√	√	√	√	☆	—	★	★	★	★	★	—
	SHT-16B-CT/PCI	☆	☆	☆	☆	√	√	√	√	√	√	☆	—	★	★	★	★	★	—
	SHT-16B-CT/PCI/FAX	☆	☆	☆	☆	√	√	√	√	√	√	☆	—	★	★	√	√	★	—
	SHT-16B-CT/PCI/MP3	☆	☆	☆	☆	√	√	√	√	√	√	☆	—	★	√	★	★	★	—
	SHT-8C/PCI/FAX	☆	☆	☆	☆	√	√	√	√	√	√	☆	—	★	★	★	★	★	—
	SHT-8C/PCI/EC	☆	☆	☆	☆	√	√	√	√	√	√	☆	—	★	★	★	★	★	—
	SHT-16C-CT/PCI/FAX	☆	☆	☆	☆	√	√	√	√	√	√	☆	—	★	★	★	★	★	—
	SHT-16C-CT/PCI/EC	☆	☆	☆	☆	√	√	√	√	√	√	☆	—	★	★	★	★	★	—
	SHT-120A-CT/cPCI	☆	☆	☆	☆	√	√	√	√	√	√	☆	—	★	★	★	★	★	—
	SHT-16B-CT/cPCI	☆	☆	☆	☆	√	√	√	√	√	√	☆	—	★	★	★	★	★	—
	SHT-16B-CT/cPCI/FAX	☆	☆	☆	☆	√	√	√	√	√	√	☆	—	★	★	★	★	★	—
	SHT-16B-CT/cPCI/MP3	☆	☆	☆	☆	√	√	√	√	√	√	☆	—	★	√	★	★	★	—
	SHT-16D-CT/PCIe	☆	☆	☆	☆	√	√	√	√	√	√	☆	—	★	★	★	★	★	—
SHD	SHD-30A-CT/PCI/SS1	☆	☆	☆	☆	√	√	√	√	√	√	☆	—	★	★	★	★	★	—
	SHD-30A-CT/PCI/ISDN	☆	☆	☆	☆	√	√	√	√	√	√	☆	—	★	★	★	★	★	—
	SHD-30A-CT/PCI/SS7	☆	☆	☆	☆	√	√	√	√	√	√	☆	—	★	★	★	★	★	—
	SHD-60A-CT/PCI/SS1	☆	☆	☆	☆	√	√	√	√	√	√	☆	—	★	★	★	★	★	—
	SHD-60A-CT/PCI/ISDN	☆	☆	☆	☆	√	√	√	√	√	√	☆	—	★	★	★	★	★	—
	SHD-60A-CT/PCI/SS7	☆	☆	☆	☆	√	√	√	√	√	√	☆	—	★	★	★	★	★	—
	SHD-120A-CT/PCI/SS1	☆	☆	☆	☆	√	√	√	√	√	√	☆	—	★	★	★	★	★	—
	SHD-120A-CT/PCI/ISDN	☆	☆	☆	☆	√	√	√	√	√	√	☆	—	★	★	★	★	★	—
	SHD-120A-CT/PCI/SS7	☆	☆	☆	☆	√	√	√	√	√	√	☆	—	★	★	★	★	★	—
	SHD-30B-CT/PCI/SS7/FAX	☆	☆	☆	☆	√	√	√	√	√	√	☆	—	★	★	★	★	★	—
	SHD-60B-CT/PCI/SS7/FAX	☆	☆	☆	☆	√	√	√	√	√	√	☆	—	★	★	★	★	★	—
	SHD-30C-CT/PCI	☆	☆	☆	☆	√	√	√	√	√	√	☆	—	★	★	★	★	★	—
	SHD-30C-CT/PCI/FAX	☆	☆	☆	☆	√	√	√	√	√	√	☆	—	★	★	★	★	★	—
	SHD-60C-CT/PCI	☆	☆	☆	☆	√	√	√	√	√	√	☆	—	★	★	★	★	★	—
	SHD-60C-CT/PCI/FAX	☆	☆	☆	☆	√	√	√	√	√	√	☆	—	★	★	★	★	★	—
	SHD-30D-CT/PCI	☆	☆	☆	☆	√	√	√	√	√	√	☆	—	★	★	★	★	★	—
	SHD-60D-CT/PCI	☆	☆	☆	☆	√	√	√	√	√	√	☆	—	★	★	★	★	★	—
	SHD-120D-CT/PCI	☆	☆	☆	☆	√	√	√	√	√	√	☆	—	★	★	★	★	★	—
	SHD-120D-CT/PCI/EC	☆	☆	☆	☆	√	√	√	√	√	√	☆	—	★	★	★	★	★	—
	SHD-120D-CT/PCI/CAS	☆	☆	☆	☆	√	√	√	√	√	√	☆	—	★	★	★	★	★	—
	SHD-240D-CT/PCI	☆	☆	☆	☆	√	√	√	√	√	√	☆	—	★	★	★	★	★	—
	SHD-240D-CT/PCI/EC	☆	☆	☆	☆	√	√	√	√	√	√	☆	—	★	★	★	★	★	—
	SHD-240D-CT/PCI/CAS	☆	☆	☆	☆	√	√	√	√	√	√	☆	—	★	★	★	★	★	—
	SHD-30E-CT/PCI(SSW)	☆	☆	☆	☆	√	√	√	√	√	√	☆	—	★	★	★	★	★	—
	SHD-30E-CT/PCI/FAX(SSW)	☆	☆	☆	☆	√	√	√	√	√	√	☆	—	★	★	★	★	★	—

	SHD-30E-CT/PCI/EC(SSW)	☆	☆	☆	☆	√	√	√	√	√	√	☆	—	★	★	★	★	★	—
	SHD-60E-CT/PCI(SSW)	☆	☆	☆	☆	√	√	√	√	√	√	☆	—	★	★	★	★	★	—
	SHD-60E-CT/PCI/FAX(SSW)	☆	☆	☆	☆	√	√	√	√	√	√	☆	—	★	★	★	★	★	—
	SHD-60E-CT/PCI/EC(SSW)	☆	☆	☆	☆	√	√	√	√	√	√	☆	—	★	★	★	★	★	—
	SHD-120E-CT/PCI(SSW)	☆	☆	☆	☆	√	√	√	√	√	√	☆	—	★	★	★	★	★	—
	SHD-120E-CT/PCI/FAX(SSW)	☆	☆	☆	☆	√	√	√	√	√	√	☆	—	★	★	★	★	★	—
	SHD-120E-CT/PCI/EC(SSW)	☆	☆	☆	☆	√	√	√	√	√	√	☆	—	★	★	★	★	★	—
	SHD-240E-CT/PCI(SSW)	☆	☆	☆	☆	√	√	√	√	√	√	☆	—	★	★	★	★	★	—
	SHD-240E-CT/PCI/FAX(SSW)	☆	☆	☆	☆	√	√	√	√	√	√	☆	—	★	★	★	★	★	—
	SHD-240E-CT/PCI/EC(SSW)	☆	☆	☆	☆	√	√	√	√	√	√	☆	—	★	★	★	★	★	—
	SHD-30E-CT/PCIe	☆	☆	☆	☆	√	√	√	√	√	√	☆	—	★	★	★	★	★	—
	SHD-30E-CT/PCIe/FAX	☆	☆	☆	☆	√	√	√	√	√	√	☆	—	★	★	★	★	★	—
	SHD-30E-CT/PCIe/EC	☆	☆	☆	☆	√	√	√	√	√	√	☆	—	★	★	★	★	★	—
	SHD-60E-CT/PCIe	☆	☆	☆	☆	√	√	√	√	√	√	☆	—	★	★	★	★	★	—
	SHD-60E-CT/PCIe/FAX	☆	☆	☆	☆	√	√	√	√	√	√	☆	—	★	★	★	★	★	—
	SHD-60E-CT/PCIe/EC	☆	☆	☆	☆	√	√	√	√	√	√	☆	—	★	★	★	★	★	—
	SHD-120E-CT/PCIe	☆	☆	☆	☆	√	√	√	√	√	√	☆	—	★	★	★	★	★	—
	SHD-120E-CT/PCIe/FAX	☆	☆	☆	☆	√	√	√	√	√	√	☆	—	★	★	★	★	★	—
	SHD-120E-CT/PCIe/EC	☆	☆	☆	☆	√	√	√	√	√	√	☆	—	★	★	★	★	★	—
	SHD-240E-CT/PCIe	☆	☆	☆	☆	√	√	√	√	√	√	☆	—	★	★	★	★	★	—
	SHD-240E-CT/PCIe/FAX	☆	☆	☆	☆	√	√	√	√	√	√	☆	—	★	★	★	★	★	—
	SHD-240E-CT/PCIe/EC	☆	☆	☆	☆	√	√	√	√	√	√	☆	—	★	★	★	★	★	—
	SHD-240E-CT/PCIe/VAR	☆	☆	☆	☆	√	√	√	√	√	√	☆	—	★	★	★	★	★	—
	SHD-30A-CT/cPCI/SS7	☆	☆	☆	☆	√	√	√	√	√	√	☆	—	★	★	★	★	★	—
	SHD-60A-CT/cPCI/SS7	☆	☆	☆	☆	√	√	√	√	√	√	☆	—	★	★	★	★	★	—
	SHD-120A-CT/cPCI/SS7	☆	☆	☆	☆	√	√	√	√	√	√	☆	—	★	★	★	★	★	—
	SHD-240A-CT/cPCI	☆	☆	☆	☆	√	√	√	√	—	—	√	√	★	★	★	★	★	—
	SHD-480A-CT/cPCI	☆	☆	☆	☆	√	√	√	√	—	—	√	√	★	★	★	★	★	—
	SHD-30B-CT/cPCI/SS7/FAX	☆	☆	☆	☆	√	√	√	√	√	√	☆	—	★	★	★	★	★	—
	SHD-60B-CT/cPCI/SS7/FAX	☆	☆	☆	☆	√	√	√	√	√	√	☆	—	★	★	★	★	★	—
	SHD-240S-CT/cPCI	☆	☆	☆	☆	√	√	√	√	—	—	√	√	★	★	★	★	★	—
	SHD-480S-CT/cPCI	☆	☆	☆	☆	√	√	√	√	—	—	√	√	★	★	★	★	★	—
SHN	SHN-32A-CT/PCI	☆	☆	☆	☆	√	√	√	√	—	—	☆	—	★	★	★	★	★	—
	SHN-8B-CT/PCI+	☆	☆	☆	☆	√	√	√	√	√	√	☆	—	★	★	★	★	★	—
	SHN-16B-CT/PCI+	☆	☆	☆	☆	√	√	√	√	√	√	☆	—	★	★	★	★	★	—
	SHN-32B-CT/PCI+	☆	☆	☆	☆	√	√	√	√	√	√	☆	—	★	★	★	★	★	—
	SHN-60B-CT/PCI+	☆	☆	☆	☆	√	√	√	√	√	√	☆	—	★	★	★	★	★	—
	SHN-120B-CT/PCI+	☆	☆	☆	☆	√	√	√	√	√	√	☆	—	★	★	★	★	★	—
	SHN-60B-CT/PCIe+	☆	☆	☆	☆	√	√	√	√	√	√	☆	—	★	★	★	★	★	—
	SHN-120B-CT/PCIe+	☆	☆	☆	☆	√	√	√	√	√	√	☆	—	★	★	★	★	★	—
	SHN-120B-CT/PCIe/VAR	☆	☆	☆	☆	√	√	√	√	√	√	☆	—	★	★	★	★	★	—
	SHN-480C-CT/PCIe	☆	☆	☆	☆	√	√	√	√	√	√	☆	—	★	★	★	★	★	—
ATP	SHT-2A/USB	☆	☆	☆	☆	√	√	√	√	—	—	√	√	★	★	★	★	★	—
	SHT-4A/USB	☆	☆	☆	☆	√	√	√	√	—	—	√	√	★	★	★	★	★	—
	SHT-2B/USB	☆	☆	☆	☆	√	√	√	√	—	—	√	√	★	★	★	★	★	—
	SHT-4B/USB	☆	☆	☆	☆	√	√	√	√	—	—	√	√	★	★	★	★	★	—
	SHT-8A/PCI	☆	☆	☆	☆	√	√	√	√	√	√	☆	—	★	★	★	★	★	—
	SHT-16A-CT/PCI	☆	☆	☆	☆	√	√	√	√	√	√	☆	—	★	★	★	★	★	—
	ATP-24A/PCI	☆	☆	☆	☆	√	√	√	√	√	√	☆	—	★	★	★	★	★	—
	ATP-24A/PCI+	☆	☆	☆	☆	√	√	√	√	√	√	☆	—	★	√	★	√	★	√
	ATP-24A/PCIe	☆	☆	☆	☆	√	√	√	√	√	√	☆	—	★	★	★	★	★	—
	ATP-24A/PCIe+	☆	☆	☆	☆	√	√	√	√	√	√	☆	—	★	√	★	√	★	√
	SHT-8B/PCI	☆	☆	☆	☆	√	√	√	√	√	√	☆	—	★	★	★	★	★	—
	SHT-16B-CT/PCI	☆	☆	☆	☆	√	√	√	√	√	√	☆	—	★	★	★	★	★	—
	SHT-16B-CT/PCI/MP3	☆	☆	☆	☆	√	√	√	√	√	√	☆	—	★	√	★	★	★	—

	SHT-16B-CT/cPCI	☆	☆	☆	☆	√	√	√	√	√	√	☆	—	★	★	★	★	★	—
	SHT-16B-CT/cPCI/MP3	☆	☆	☆	☆	√	√	√	√	√	√	☆	—	★	√	★	★	★	—
DTP	SHD-30A-CT/PCI/FJ	☆	☆	☆	☆	√	√	√	√	√	√	☆	—	★	★	★	★	★	—
	SHD-60A-CT/PCI/FJ	☆	☆	☆	☆	√	√	√	√	√	√	☆	—	★	★	★	★	★	—
	SHD-30B-CT/PCI/FJ	☆	☆	☆	☆	√	√	√	√	√	√	☆	—	★	★	★	√	★	√
	SHD-60B-CT/PCI/FJ	☆	☆	☆	☆	√	√	√	√	√	√	☆	—	★	★	★	√	★	√
	DTP-30C/PCI	☆	☆	☆	☆	√	√	√	√	√	√	☆	—	★	★	★	★	★	—
	DTP-30C/PCI+	☆	☆	☆	☆	√	√	√	√	√	√	☆	—	★	√	★	√	★	√
	DTP-60C/PCI	☆	☆	☆	☆	√	√	√	√	√	√	☆	—	★	★	★	★	★	—
	DTP-60C/PCI+	☆	☆	☆	☆	√	√	√	√	√	√	☆	—	★	√	★	√	★	√
	DTP-120C/PCI	☆	☆	☆	☆	√	√	√	√	√	√	☆	—	★	★	★	★	★	—
	DTP-120C/PCI+	☆	☆	☆	☆	√	√	√	√	√	√	☆	—	★	√	★	√	★	√
	DTP-30C/PCIe	☆	☆	☆	☆	√	√	√	√	√	√	☆	—	★	★	★	★	★	—
	DTP-30C/PCIe+	☆	☆	☆	☆	√	√	√	√	√	√	☆	—	★	√	★	√	★	√
	DTP-60C/PCIe	☆	☆	☆	☆	√	√	√	√	√	√	☆	—	★	★	★	★	★	—
	DTP-60C/PCIe+	☆	☆	☆	☆	√	√	√	√	√	√	☆	—	★	√	★	√	★	√
	DTP-120C/PCIe	☆	☆	☆	☆	√	√	√	√	√	√	☆	—	★	★	★	★	★	—
	DTP-120C/PCIe+	☆	☆	☆	☆	√	√	√	√	√	√	☆	—	★	√	★	√	★	√
DST	SHR-16DA-CT/PCI	☆	☆	☆	☆	√	√	√	√	√	√	☆	—	★	★	★	★	★	√
	SHR-24DA-CT/PCI	☆	☆	☆	☆	√	√	√	√	√	√	☆	—	★	★	★	★	★	√
	DST-24B/PCI	☆	☆	☆	☆	√	√	√	√	√	√	☆	—	★	★	★	★	★	—
	DST-24B/PCI(SSW)	☆	☆	☆	☆	√	√	√	√	√	√	☆	—	★	★	★	★	★	—
	DST-24B/PCI+	☆	☆	☆	☆	√	√	√	√	√	√	☆	—	★	√	★	√	★	√
	DST-24B/PCI+(SSW)	☆	☆	☆	☆	√	√	√	√	√	√	☆	—	★	√	★	√	★	√
	DST-24B/PCIe	☆	☆	☆	☆	√	√	√	√	√	√	☆	—	★	★	★	★	★	—
	DST-24B/PCIe+	☆	☆	☆	☆	√	√	√	√	√	√	☆	—	★	√	★	√	★	√
SHF	SHF-2D/PCI	☆	☆	☆	☆	√	√	√	√	√	√	☆	—	★	★	★	★	★	—
	SHF-4D/PCI	☆	☆	☆	☆	√	√	√	√	√	√	☆	—	★	★	★	★	★	—
	SHF-4D/PCIe	☆	☆	☆	☆	√	√	√	√	√	√	☆	—	★	★	★	★	★	—
IPR	SynIPRecorder	—	☆	—	☆	☆	☆	☆	☆	—	☆	—	—	—	☆	—	☆	☆	☆
HMP	SYNWAY-HMP	☆	☆	☆	☆	√	√	√	√	—	—	☆	—	★	★	★	★	★	—

Legend: COD: Coder DEC: Decoder
√: Hardware-based —: Unsupported
☆: Software-based, enabled via software algorithm by the driver ★: Software-based, enabled via the external ACM program

Note: For more information about the DTP series boards, refer to the section Voice Recording.

1.4 SynCTI Supported OS

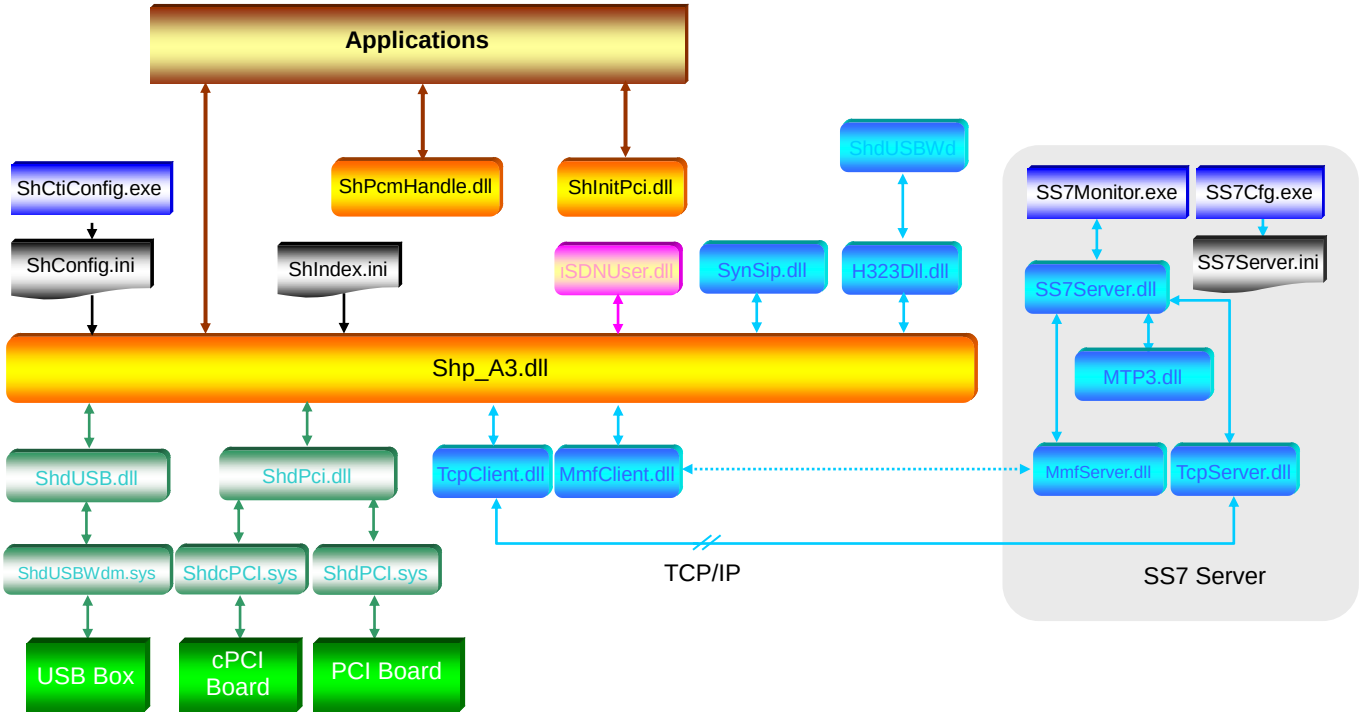
The SynCTI driver program supports two mainstream operating systems: WINDOWS and LINUX. See below for details.

- WINDOWS: Windows 5.0 and above versions all supported, either 32-bit or 64-bit.
- LINUX: includes CentOS, Debian, FC4, FC6, FC7, Gentoo, Red Hat A4 U4, Red Hat AS3, RH9.0, Suse, Tribox2.0, Ubuntu

Note: The board with Compact PCI bus must run in Windows 2000 or above.

1.5 SynCTI Driver Architecture (CTI Series)

It is the SynCTI driver architecture in Windows shown below:

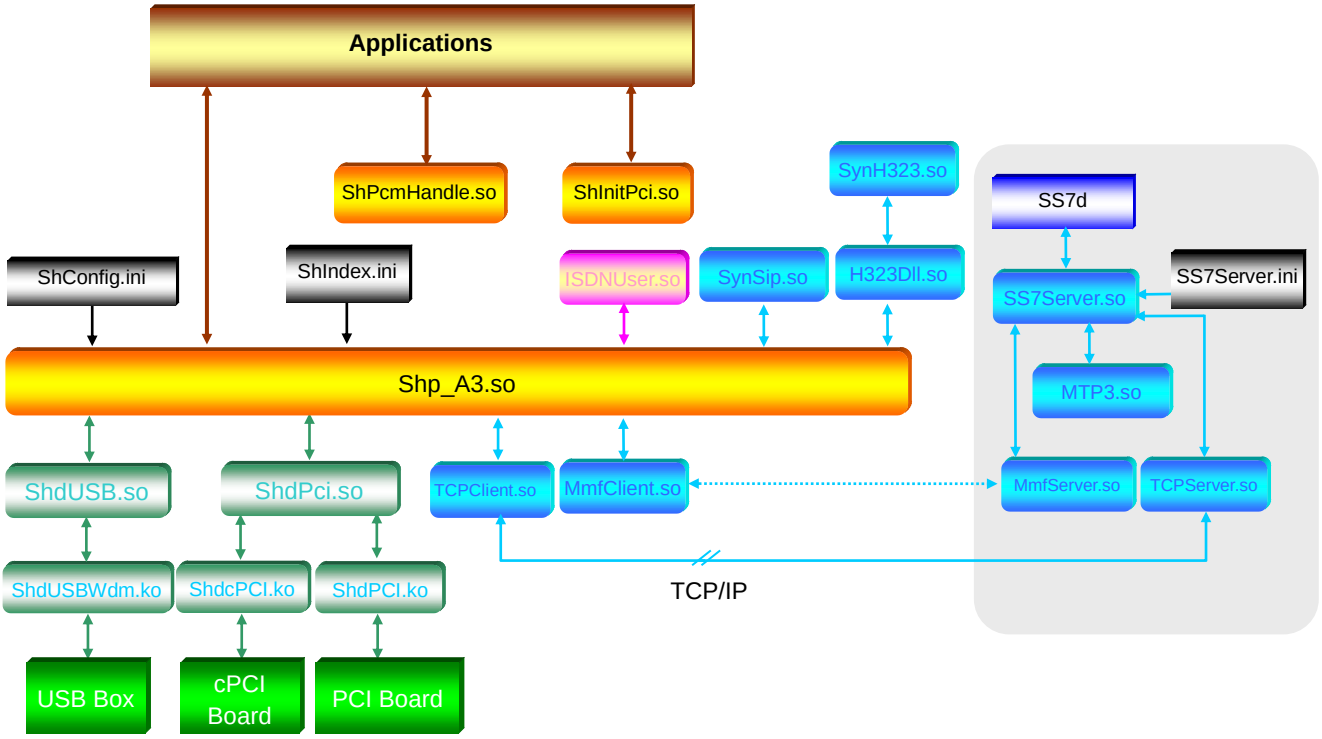


The following table describes each component in the figure above:

Components	Description
Applications	User applications
ShPcmHandle.dll	Provides API functions for the application; enables conversion between various audio formats
ShInitPci.dll	Provides interfaces for the application; helps users obtain basic information about the board
ShIndex.ini	Configuration file of the form where lists voice files used for memory playback by index
ISDNUser.dll	ISDN message processing modules. Via interface functions, analyze and process the commands and data transferred from A3 according to the ISDN protocol, and then pass the results back to A3
ShCtiConfig.exe	Automatic configuration program
ShConfig.ini	System configuration file which corresponds to ShCtiConfig.exe. Refer to ShConfigAdvanced.ini under the driver installation directory for detailed information
Shp_A3.dll	Provides API functions for the application; helps achieve all functions of the board
ShdUSB.dll ShdPci.dll	Transfer data, states and commands between the A3 layer and the SYS
ShdUSBWdm.sys ShdcPCI.sys ShdPCI.sys	Constitute the SYS layer. Used to add or remove hardware, communicate with the DSP program. These three files respectively process USB, cPCI and PCI interfaces
SS7Monitor.exe	SS7 server
SS7Cfg.exe	SS7 configuration program
SS7Server.ini	Configuration file for SS7 server
SS7Server.dll	SS7 server scheduling module. Enables transmission of SS7 signaling, route control, etc
MTP3.dll	Constitutes the MTP3 layer in SS7 server
MmfServer.dll	Stand-alone version of SS7 server. Communicates with SS7Server.dll and

	MmfClient.dll via transmission of signaling and data
MmfClient.dll	Stand-alone version of SS7 client. Communicates with MmfServer.dll and the A3 layer via transmission of signaling and data
TcpServer.dll TcpClient.dll	Network versions of SS7 server and client. Communicate with each other via data transmission according to the TCP/IP protocol
USB Box, cPCI Board, PCI Board	Voice boards respectively with USB, cPCI and PCI interfaces
SynSip.dll	SIP signaling processing component
H323Dll.dll	H.323 signaling adaptation layer
SynH323.dll	H.323 signaling processing component

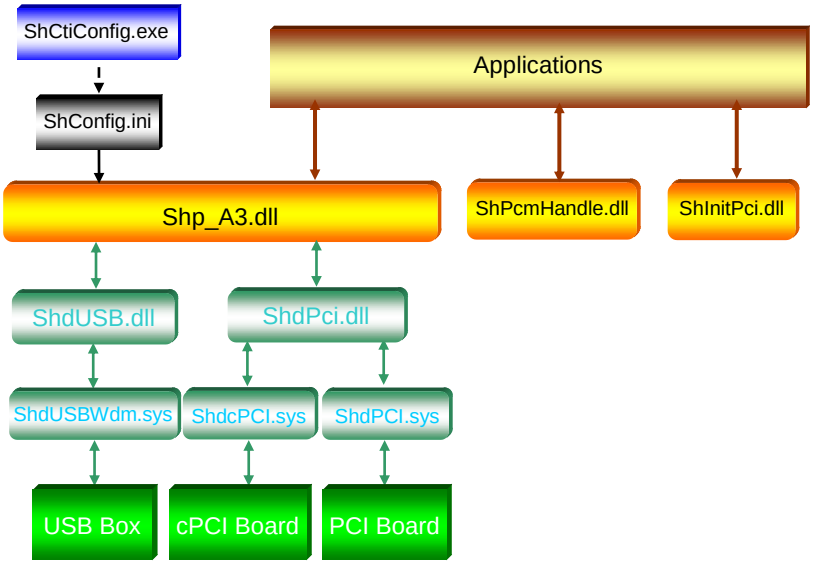
It is the SynCTI driver architecture in Linux shown below:



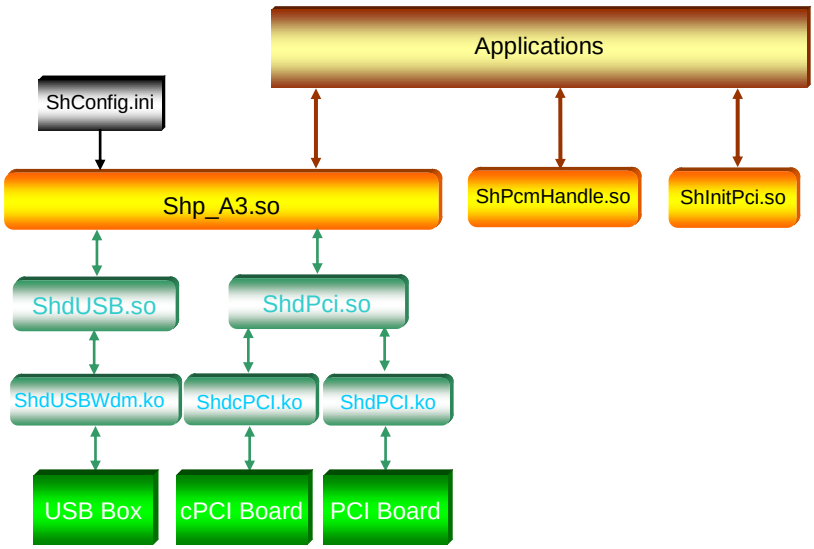
Note: Each component in the figure above functions as same as that correspondingly represented in the driver architecture in Windows.

1.6 SynCTI Driver Architecture (REC Series)

It is the SynCTI driver architecture in Windows shown below:



It is the SynCTI driver architecture in Linux shown below:



Note: Each component in the driver architecture for the REC Series functions as same as the corresponding one for the CTI Series.

1.7 Software Tool for SynCTI

1.7.1 System Configuration Tool - ShCtiConfig.exe

ShCtiConfig.exe is a software tool for visual configuration of the ShConfig.INI file. It supports configuration of common parameters for all board models from Synway. Upon the completion of driver (SynCTI) installation, what you need to do, if you don't want to edit the configuration file manually, is simply running ShCtiConfig.exe to set parameters. The configuration program can automatically read information about board model, serial number, etc. from the board firmware, calculate such basic elements as quantity of channels, provide corresponding control buttons and write the default value of each configuration item into ShConfig.INI. If you want the driver installation

integrated into this configuration program and the board configured with default settings at the same time, add a command-line parameter into the program during the startup of ShCtiConfig.exe: 1 which indicates 'default set->apply->close the configuration program' or 2 which means 'default set->confirm->close the configuration program'. Regarding how to use these parameters, see the following example.

CreateProcess("C:\\shcti\\ShCtiConfig.exe", "1", NULL, NULL, FALSE, 0, NULL, "C:\\shcti", &si, &pi);

ShCtiConfig.exe is mainly for:

- visual configuration of common parameters used by PCI boards.
- visual configuration of common parameters used by USB voice boxes.
- visual configuration of common parameters used by cPCI boards and handling of hot-swap management.

1.7.2 Tone Analyzer Tool - ShTA.exe

Failure in call and detection of remote hangup, etc. over analog trunks while using the SHT and ATP Series boards always result from the mismatch in parameters between the on-board call-processing tone detector and the PBX. While some small PBXes use different tone frequency, tone cycle from those adopted by Central Office Terminal (COT), the tone detector on the Synway board is compliant with the COT standard. Therefore, it is essential to set correct parameters for tone frequency, duty ratio of dial tone, ringback (echo) tone, busy tone, etc. ShTA.exe can be used in such case to detect and analyze the call-processing tone parameters on the PBX side to obtain accurate settings.

For detailed information about the use of ShTA.exe, refer to *Tone Analyzer Tool User Manual*, i.e. the file ShTA_UserManual_en.chm (English) or ShTA_UserManual_cn.chm (Chinese) under the SynCTI driver installation directory.

1.7.3 SS7 Server Configuration Program - SS7Cfg.exe

SS7Cfg.exe is a software tool for setting Ss7server.ini (the configuration file of the SS7 Server). For more information about the use of SS7Cfg.exe, refer to *SS7 Server Configuration Program User Manual*, i.e. the file SS7Cfg_UserManual.chm (English) or SS7Cfg_UserManual_cn.chm (Chinese) under the SynCTI driver installation directory.

1.7.4 SS7 Server - SS7monitor.exe

Ss7Monitor.exe supports the SS7 MTP3 protocol, can distribute signaling messages and monitor the system. The application must run the SS7 server first if it requires SS7 signaling.

1.7.5 MSU Decoder Tool - MsuDecode.exe

MsuDecode.exe is a software tool for decoding the messages (MSU) contained in the log file of the SS7 server. A log file with the '.msu' extension can be generated based on customer requirements to save necessary messages in binary system during the runtime of Ss7Monitor.exe. This MSU Decoder Tool can translate binary messages into the intelligible '.txt' document for direct, easy reading. For detailed information about the use of MsuDecode.exe, refer to *MSU Decoder Tool User Manual*, i.e. the file MsuDecode_UserManual.chm (English) or MsuDecode_UserManual_ch.chm (Chinese) under the SynCTI driver installation directory.

1.8 Basic Concepts

1.8.1 Channel (CTI Series)

It is an on-board physical circuit entity with the capability of completed voice processing for a call, classified as follows according to supported interfaces and signaling.

Board Series	Module / Protocol	Channel Type	Type Code
SHT	Analog trunk module	Analog trunk channel	0
	Station module	Station channel	2
	Recording module	Recording channel	3
	Microphone module	Recording channel	3
	Magnet module	Magnet channel	10
	none	Non-module channel	20
	fax channel	Fax channel	9
SHD	SS1	SS1 channel	4
	ISDN (User-side)	ISDN channel (User-side)	7
	ISDN (Network-side)	ISDN channel (Network-side)	8
	SS7-TUP protocol	TUP channel	6
	SS7-ISUP protocol	ISUP channel	11
	Fax channel	Fax channel	9
SHV		Voice-alteration channel	14
SHN	SIP	SIP channel	16
SHF	Analog trunk module	Analog trunk channel	0
	Station module	Station channel	2
	fax channel	Fax channel	9
HMP	SIP	SIP channel	16

Note: Type code is available via the function SsmGetChType.

1.8.2 Channel (REC Series)

It is an on-board physical circuit entity with the capability of completed voice processing for a call, classified as follows according to supported interfaces and signaling.

Board Series	Module / Protocol	Channel Type	Type Code
ATP	Recording module	Analog trunk recording channel	3
	Microphone module	Microphone recording channel	3
DST		Digital call recording channel	12
DTP	SS1	SS1 monitoring channel	4
	ISDN	ISDN monitoring channel	7 or 8
	SS7-TUP	TUP monitoring channel	6
	SS7-ISUP	ISUP monitoring channel	11
	DASS2	DASS2 monitoring channel	19
	DPNSS	DPNSS monitoring channel	27
IPR	IPRA_CH	Analyzer channel	26
	IPRR_CH	Recorder channel	25

Note: Type code is available via the function SsmGetChType.

1.8.3 Logical Channel Number

Each channel has two numbers in the driver: the physical number and the logical number.

Physical number (referred to as 'BCh' for short) indicates the channel number marked on the board, which is automatically allocated by the driver. Suppose N is the total number of on-board channels and M represents the total number of fax channels, the physical channel number range is:

Board Series	Physical Channel Number Range
SHT	'/FAX' unmarked in the model: 0~N-1 '/FAX' marked in the model: For voice channel, 0~N-M-1; for fax channel, N-M~N-1
SHD	0~N-1
SHV	0~N-1
SHN	0~N-1
ATP	0~N-1
DTP	CIC number: 0~N-1 Channel number: 0~N * 2 - 1
DST	0~N-1
SHF	For vioce channel: 0~N-M-; for fax channel: N-M~N-1
IPR	0~N-1

Logical number is the unified number given to each channel in the application system. The configuration item TotalAppCh is used to set the total number of channels in the system. Note that the parameter 'ch' used by most functions in this driver refers to the logical number of a channel.

Each logical channel number maps a physical number, determined by the configuration item AppCh.

Note: The channel number mentioned in this manual is the logical number except as otherwise specified.

1.8.4 Digital Trunk

A trunk is a line which links user terminal devices (e.g. user PBXes, key telephone systems, call centers, etc.) to PBX systems in the PTN (Public Telephone Network). The digital trunk serves as a digital subscriber line to the telecommunication network, usually made up of a pair of coaxial cables or twisted pair cables from the SPC exchange, one cable for transmission and the other for reception. If the trunk carries signals at 2.048Mbps (referred to as '2M cable' for short), the interface between it and the user terminal device is called E1 interface; if it carries signals at 1.544Mbps, the interface is called T1 interface.

The digital trunk uses TDM technology, consisting of multiple time slots (referred to as 'TS' for short), each of which transmits data at 64Kbps. An E1 trunk has 32 time slots while a T1 trunk has 24.

As to the E1 trunk, TS 0 is used for frame synchronization. Hence there are only 31 effective time slots, among which TS 16 usually transfers signaling and multiframe synchronization information, called D-channel, and the rest 30 time slots are used for transmitting voice data, called B-channel.

In this manual, each pair of E1/T1 lines is called a digital trunk.

1.8.4.1 Frame Synchronization over Digital Trunk

As TS 0 over the digital trunk usually carries frame synchronization information, it cannot be used as a voice time slot. Both the SHD and DTP Series boards are equipped with a frame-synchronization indicator at each port of digital trunks, tracing the frame synchronization at the link layer: Light on indicates the smooth in frame synchronization; light off or flash hints the failure.

In the use of CAS, multiframe synchronization information is also transmitted through TS 0.

When the frame synchronization status changes over the digital trunk, the application will immediately receive the

E_CHG_PcmLinkStatus event from the driver, or it can call SsmGetPcmLinkStatus at any time to obtain the current state of digital trunks.

1.8.4.2 Clock for Digital Trunk

Each digital trunk requires a clock for normal operation. See the section 'System Clock Configuration' in this chapter for more information.

1.8.4.3 Digital Trunk Configuration

Each physical digital trunk on the local end must be consistent in configuration with that from the remote PBXes to ensure good operation.

Setting of digital trunks is based on the board and carried out in the [**BoardId=x**] section, involving the following configuration items.

```
[BoardId=x]
……
PcmNumber=M          // Set the total number of on-board digital trunks
PcmSSx[0]=1          // Set Digital Trunk 0 on the board: select the signaling mode
PcmClockMode[0]=j    // Select the operating mode of the clock
PcmLinkType[0]=k     // Select the type of the physical line
…….
```

1.8.4.4 Signaling Time Slot on Digital Trunk

The signaling time slot on the digital trunk functions variously depending on the protocol type as shown in the table below.

Protocol Type	Signaling Time Slot
SS1	TS 16 is used for transmitting the ABCD signaling code.
ISDN PRI	TS 16 is used for transmitting signaling information.
SS7	TS 16 is generally used as a signaling link. You can also make other time slots serve as signaling links by modifying two configuration items UseTS16AsCircuit and Ss7SignalingTS. For more information, see 'Time Slot Allocation' in this chapter.

1.8.4.5 Digital Trunk Number

Each digital trunk has two numbers: the physical number and the logical number.

Physical PCM number refers to the on-board digital trunk number, always beginning with 0 and ending with the total number of on-board digital trunks minus 1. Logical number is given on a unified basis to all digital trunks on multiple boards in the same application system. At the time when the application has only one board in it, the physical number of a digital trunk is the same as the logical.

The mapping relationship between the physical number and the logical can be designated via configuration items TotalPcm and Pcm.

Note: The 'PCM number' mentioned in this manual indicates the logical digital trunk number except as otherwise specified.

1.8.4.6 Setting Voice CODEC on B-Channel

The configuration item DefaultVoiceFormat is used for setting the voice CODEC on the digital subscriber line.

1.8.5 User Authorization Code for Board

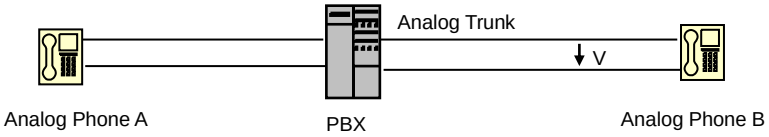
The Synway board is designed with an encrypted authorization code identification circuit so that no one except the manufacturer is able to modify the code. This feature, offering the same effect as installing a dongle in the user software, enables users to bind the application program with a particular board. By using the authorization code, your board from Synway becomes a customized, exclusive model. Below tells some merits in the use of this code to protect the user software.

- Simple operation: When Synway writes the authorization code applied for by users ahead of time into its supplied product, the user software only need check the code to see if it is valid, requiring neither algorithm for calculation nor resettings performed after each installation, unlike using the Series number for software protection which involves complex encrypted calculation as well as a sequence of settings based on the Series number at each installation.
- High reliability: Software protection by the Series number goes invalid once the encrypted algorithm leaks out. By comparison, as long as you have applied for an authorization code, you are assured in technology that no one except the manufacturer can rewrite the code, which therefore eliminates the possibility to run your software on the voice boards from various channels other than Synway or those provided for other customers by Synway, even if the code is open to the public.

The function SsmGetAccreditId or SsmGetAccreditIdEx can be used to obtain the user authorization code for the board.

1.8.6 Change in Analog Phone Line Voltage

The DC voltage on analog phone lines is -48V, provided by PBXes. Each channel on the Synway SHT/ATP Series boards has a voltage detector for examining the operating voltage on the analog trunk. The following figure is a typical model for phone connection.



If Analog Phone B performs the following operations,

（1） At the time t_1, B picks up.

（2） At t_2, B completes dialing; the PBX rings A up and sends ringback (echo) tones to B.

（3） At t_3, A picks up; both parties start a call with each other.

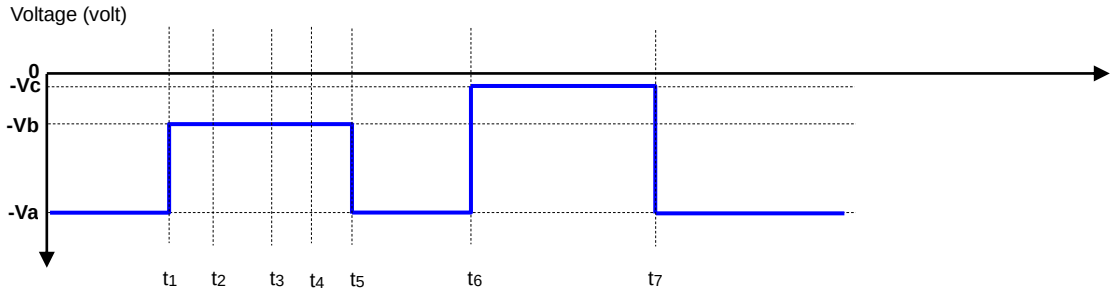
（4） At t_4, the call ends up; A hangs up and the PBX starts sending busy tones to B.

（5） At t_5, B hangs up.

（6） At t_6, B is disconnected from the PBX.

（7） At t_7, B is reconnected with the PBX.

its line voltage changes as follows provided the PBX doesn't support the polarity reversal feature.



In the figure above,

Va: the line voltage at the time when the phone is on-hook, usually being -48V.

Vb: the line voltage at the time when the phone is off-hook, usually being around -15V for the Synway board.

Vc: the line voltage at the time when the phone line is disconnected, theoretically being 0V.

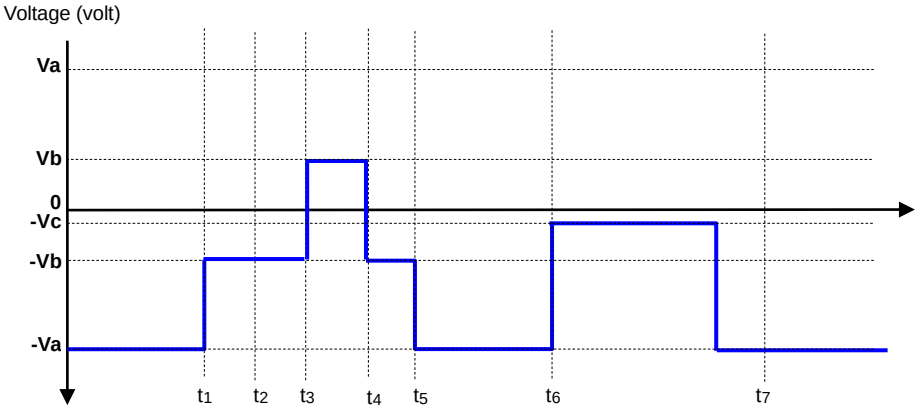
The voltage detector judges the state of lines and phones by analyzing the relationship between the actual line voltage (assumed as Vin) and Va, Vb, Vc. The judging criterion may be described as follows.

- ✧ Off-hook: If Vc<Vin<Vb and keeps for $t_{offhook}$.
- ✧ On-hook: If Vb≤Vin and keeps for t_{onhook}.
- ✧ Off-line: If Vin≤Vc and keeps for t_{off}.

The configuration items or functions in the SynCTI driver used for setting the above parameters are listed in the following table.

Parameter Name	Related Configuration Items or Functions
Vc	Configuration item: OffLineSet
t_{off}	Configuration item: OffLineDetermineTime
Vb	Configuration item: IsHangupDtrmVoltage Function: SsmSetDtrmLineVoltage
$t_{offhook}$	Configuration item: LineOncnt
t_{onhook}	Configuration item: MaxRecChFlashFilterTime

The line voltage of Analog Phone B changes as follows provided the PBX supports the polarity reversal feature.



We can see from the above figure that in case the PBX supports the polarity reversal feature, it changes the polarity of the line voltage at the calling party once the called party picks up or hangs up. In such way, the PBX passes the on-hook/off-hook information to the calling party. Over analog lines, it is the most accurate way to judge whether the called party has picked up by the polarity reversal signal of the line voltage. Remember that this feature doesn't work without the support of PBXes.

Both the Synway SHT and ATP Series boards are designed with voltage detectors. Every time when the line voltage changes and the variation range goes beyond the preset value of 5V which can be set via the function call of SsmSetEvent, the driver will send the E_CHG_LineVoltage event to the application. Also the application can

obtain the line voltage through the function SsmGetLineVoltage.

The voltage detector can:

- Monitor the line connection between the local end and the PBX.

The analog trunk recording channel on the ATP Series supports the off-line detection feature. When the line voltage gets lower than the set value of the configuration item OffLineSet and keeps for a period of time longer than the set value of the configuration item OffLineDetermineTime, the driver will conclude that the off-line fault occurs and transfer the channel state to 'off-line'.

- Judge the pickup and hangup behaviors on the line.

The recording channel on the ATP Series determines whether the phone has picked up or hung up according to the variation of the line voltage (absolute value) between Va and Vb.
In default, the analog recording channel judges the on-hook/off-hook phone state by the output value of the voltage detector. However, you can let the driver ignore the result from the voltage detector by modifying settings of the function SsmSetIgnoreLineVoltage or the configuration item IgnoreLineVoltage so as to keep the recording channel in the off-hook state.

When using the microphone recording module, as there is no feed voltage on the line, you should set the configuration item IgnoreLineVoltage to 1 and keep the recording channel always in the 'off-hook' state.

- Detects the polarity reversal signal.

The PBX provides the feed voltage, usually with negative polarity, for the analog line, and can pass some information to phones by changing the voltage polarity (e.g. change -Vb to Vb).

The polarity reversal signal for the analog phone line can be used to accurately judge the called party's pickup and hangup behavior in cooperation with the PBX. Both the ATP Series analog trunk recording channel and the SHT Series analog trunk channel support the polarity reversal detection. Every time when a channel detects the polarity reversal, the driver will send the event E_CHG_PolarRvrsCount to the application. Also the application can obtain information on polarity reversal via the function SsmGetPolarRvrsCount.

Under some circumstances, the interfering signals on the PBX or the line become so large that they will be mistaken by the driver for the polarity reversal signals. To eliminate such error, set the configuration item PolarIgnore to ignore large interferring signals.

When the analog trunk channel detects the polarity reversal signal, it is determined by the setting of the configuration item DisablePolarReverse whether to transfer the channel state or not.

1.8.7 Ringing Current on Analog Phone Line

When the PBX calls the phone through an analog line, it will send the ringing current to the phone and make the phone to ring. This current signal usually goes periodically, involving 1 sec on and 4 sec off.

Both the ATP Series analog trunk recording channel and the SHT Series analog trunk channel are designed with the ringing current detector. The configuration item RingDetectFilterPara is used to set the minimum durations of the ringing current respectively at on and off states. The configuration item RingEndDtrTime is used to set the maximum durations of the ringing current at on and off states. Once the PWL of the ringing current changes, the driver will send the E_CHG_RingFlag event to the application.

Every time when the driver detects a complete circle of ringing current, it will send the E_CHG_RingCount event to the application. When the count reaches the specified value N, the channel state will turn to 'ringing' from 'idle'. N can be set via the configuration item AlwaysToRingingOnRingCntX or ChToRingingOnRingCnt, its value in practice calculated by the following ways.

(1) If not specified in the configuration item ChToRingingOnRingCnt or set via the function SsmSetFlag which takes the parameter F_ChToRingingOnRingCnt, it is subject to the set value of the configuration item AlwaysToRingingOnRingCntX.

(2) If not designated in the configuration item AlwaysToRingingOnRingCntX, it is determined by the driver and N is 2.

Also the application can invoke the function SsmGetRingCount to obtain the count of ringing current signals.

The driver will automatically clear the count when:

the channel state transfers from 'ringing' to 'idle';

the channel state transfers from 'ringing' to 'off-hook' and keeps 'off-hook' for a period of time longer than the set value of the configuration item RecChClearRingDelayTime.

Also the application can clear the count via the function call of SsmClearRingCount.

1.8.8 Flash Signal on Analog Phone Line

The flash signal is a short temporary on-hook pulse generated by a rapid clap on the hook switch of the analog phone during a call, generally used for such operations as transferring the call to an extension with the help of the PBX. You should do the clap as swift as possible lest the PBX mistakes the flash signal for the on-hook signal.

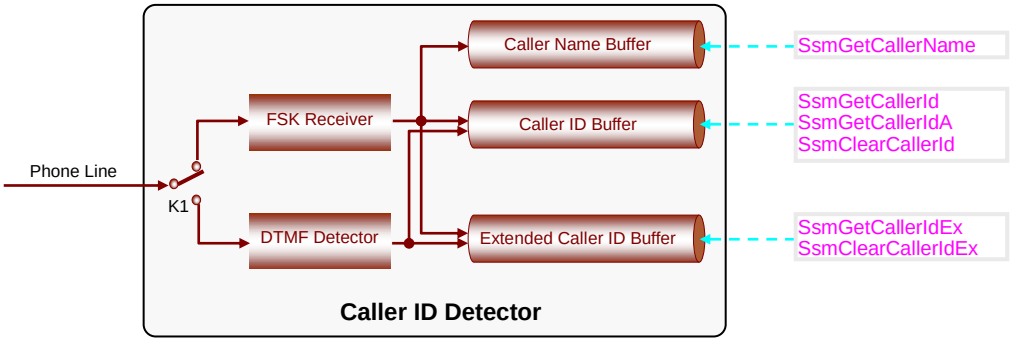
1.8.9 Caller ID on Analog Phone Line

The Caller ID service is a business provided with extra fee charged by corporations on telecommunications, which usually requires an application submitted ahead of time. There are two modes for the PBX to transmit the calling party number to the phone.

- ✧ FSK (Frequency-shift Keying) Mode: The Caller ID, i.e. the calling party number, is transmitted by the PBX somewhen between the first ring and the second of the phone.
- ✧ DTMF (Dual-tone Multi-frequency) Mode: Most PBXes transmit the Caller ID before sending the first ring to the phone. However, a few PBXes send somewhen between the first ring and the second.

In general, the PBX involved in the local area network provides the Caller ID in FSK mode while the small-sized PBX in DTMF mode. **Note that the application program should pick up the phone after the second ring to ensure a complete reception of the Caller ID. When the analog trunk channel state comes into 'ringing' can be set in the configuration file.**

Both the Synway SHT and ATP Series boards are designed with the Caller ID detector for examining the FSK or DTMF Caller ID. The operation principle is shown below:



There are 2 buffers in the driver for each channel to save the Caller ID information: Caller ID Buffer and Extended Caller ID Buffer. The latter holds the original data of the FSK Caller ID, including the calling party number, date, name of the calling party, etc. The driver processes the original information as follows:

- ✧ Separates the calling party number and saves it to Caller ID Buffer;
- ✧ Separates the name of the calling party and saves it to Caller Name Buffer

The functions, events and configuration items related to the Caller ID detector are listed in the following table.

	Name	Description
Configuration Item	CallerIdStyle	Sets the operating mode of the Caller ID detector (FSK/DTMF), i.e. set the switch k1 in the figure above
	CloseCallerIdOnReceived FSKCallerIdDtrmTime FilterInvalidCID	Set parameters for the Caller ID detector running in FSK mode
	DtmfCallerIDStyleLength DtmfCallerIDInterTimeout	Set parameters for the Caller ID detector running in DTMF mode
Function	SsmSetFlag (with the parameter F_CALLERIDTYPE)	Sets the operating mode of the Caller ID detector
	SsmGetCallerId SsmGetCallerIdEx SsmGetCallerIdA	Obtain the calling party number
	SsmGetCallerName	Obtains the name of the calling party
	SsmClearCallerId SsmClearCallerIdEx	Empty the buffer in the driver which holds the calling party number
Event	E_CHG_CIDExBuf	While the Caller ID detector is working in FSK mode, every time the driver receives a calling party number, it sends this event to the application

1.9 System Clock Configuration

Each Synway board requires a clock for normal operation. Take the following cases into consideration when setting the clock signal source for the application system based on the Synway board.

- ◆ In the case of board connection over CT Bus:
  - ✧ The clock signal source solely provided by one of the boards is called Master Clock. As a result, the board that offers the master clock is called Master Board, others called Slave Board. The signal clock source acquired by the slave board from the master board is called Slave Clock.
  - ✧ If the application system involves at least one SHD board, it is required to choose one SHD board as the master board; if no SHD boards but the DST or DTP used in the system, you should select one DST or DTP board as the master board.

- ✧ If a SHD board serves as the master board, only one digital trunk on it can supply the master clock while others must work in the Slave Clock mode.
- ✧ The clock signal from the SHD or DTP boards must keep consistent with that from PTN; otherwise it may result in noise or high signal error rate during the call.

◆ In the case of board connection not through CT Bus, the board must be set to Master Board.

◆ In the case of connection to third-party boards over CT Bus, if the third-party board is set to Master Board, all Synway boards should be set to Slave Board.

The master clock derives from two ways: Line-synchronization and Free-run. Line-synchronization is to pick up at the local end the input signal from the remote PBX as the master clock, applicable to the SHD, DTP and DST Series boards, to ensure the on-board master clock synchronous with the clock signal at the remote PBX. Free-run is to generate the clock signal by the on-board clock oscillation circuit, applicable to the SHT and ATP Series. When setting up a testing system by using the SHD boards and linking it to a real application system, you should choose 'Free-run' for the master clock of the master board.

The SHV Series boards can't serve as the master board, so they must work in the Slave Clock mode.

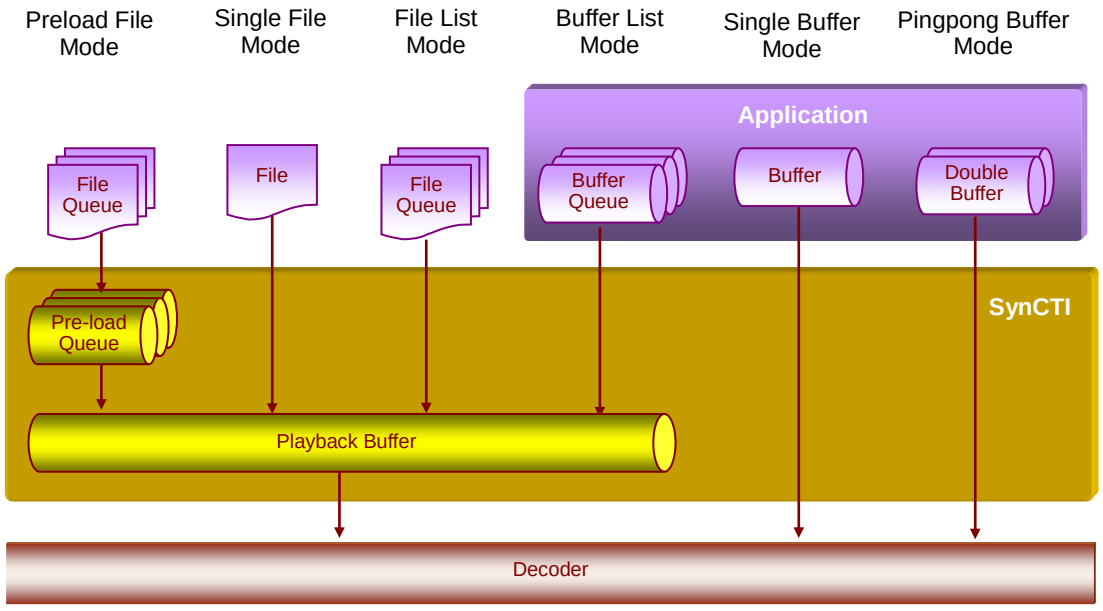
Configuration items and functions in relation to the clock signal are listed in the following table.

	Name	Description
Configuration Item	WhoSupplySysClock	Sets the clock mode for the board (Master Board/Slave Board)
	PcmClockMode	Sets the clock mode for digital trunks on the SHD and DTP Series boards
Function	SsmSetPcmClockMode	Sets the clock mode for digital trunks on the SHD and DTP Series boards. Identical in function with the configuration item PcmClockMode

1.10 Voice Processing

1.10.1 Voice Playing (CTI Series)

There are 6 modes for voice playing as shown in the following figure.



In the four voice playing modes – Preload File Mode, Single File Mode, File List Mode and Buffer List Mode, the driver will first read the source data into the internal Playback Buffer after the voice-playing operation starts. The buffer size is designated by the configuration item PlayBufSize, with a default value of 32K bytes.

1.10.1.1 Setting Volume and Direction

1.10.1.1.1CTI Series

For detailed information about the volume and the direction to play data, refer to relative content in this chapter.

- ✧ SHD Series: See the section Operation Principle of SHD Series in this chapter.
- ✧ SHT Series: See the section Operation Principle of SHT Series in this chapter.
- ✧ SHN Series: See the section Operation Principle of SHN Series in this chapter.

1.10.1.1.2REC Series

For detailed information about the volume and the direction to play data, refer to relative content in this chapter.

- ✧ ATP Series: See the section Operation Principle of ATP Series in this chapter.
- ✧ DST Series: See the section Operation Principle of DST Series in this chapter.
- ✧ DTP Series: See the section Operation Principle of DTP Series in this chapter.

1.10.1.2 Setting Termination Condition

After the start of voice playing, other than the occasion where the task stops naturally by itself or is terminated by the driver through function calls, the driver will automatically end it once the channel detects some events triggered by particular conditions, simplifying compilation of the application program. The ‘particular conditions’ mentioned here include:

- The DTMF Detector detects DTMF digits in the incoming signals.

Related functions and configuration items are listed below.

Function Name	Description	Related Configuration Item
SsmSetDtmfStopPlay	Sets whether to stop voice playing as the DTMF Detector detects DTMF digits.	DtmfStopPlayCharSet
SsmSetDTMFStopPlayCharSet	Sets the DTMF digits which stop voice playing.	
SsmGetDtmfStopPlayFlag	Obtains the flag which tells if voice playing ends as the DTMF Detector detects DTMF digits.	
SsmGetDTMFStopPlayCharSet	Obtains the DTMF digits that stop voice playing.	

Note: In case that the DTMF detector detects DTMF in the incoming signals, if the configuration item DefaultPausePlayOnRxDtmf is set to 1, the driver will intermit voice playing to ensure the accuracy of the DTMF detector, and automatically resume playing when DTMF signals totally disappear.

- The Barge-in detector detects voice activities on the line.

Function Name	Description	Related Configuration Item
SsmSetBargeinStopPlay	Sets whether to stop voice playing as the Barge-in Detector detects voice activities on the line.	DefaultBargeInStopPlay
SsmGetBargeinStopPlayFlag	Obtains the flag which tells if voice playing ends as the Barge-in Detector detects voice activities on the line.	

- Call state machine detects remote hangup.

Function Name	Description	Related Configuration Item
SsmSetHangupStopPlayFlag	Sets whether to stop voice playing as the driver state machine detects remote hangup.	HangupStopPlay

- The application program invokes the function that takes data off bus.

For particular information about characteristic features of some models in the SHD Series, refer to the section Operation Principle of SHD Series in this chapter.

1.10.1.3 Preload File Mode

The use of IVR and so forth requires the application program to play some voice segments, such as the speech prompt, date, time, number, etc. To improve the running efficiency of the application system and to reduce the times that the application visits HD, the SynCTI driver provides Preload File Mode for voice playing, whose operation principle is: Record some commonly used voice segments, e.g. '0', '1', '2', ……, '9', 'ten', 'hundred', 'thousand', 'ten thousand', to one or multiple files and define a number and an alias for each of them ahead of time. Then the driver will automatically preload those segments to the memory once the function for driver initialization SsmStartCti is invoked, which offers great facility to play such successive segments as '125'. Playing in Preload File Mode enables the driver to make seamless connection of voice segments and creates much more excellent effect than playing a number of single files in sequence.

The driver supports up to 65536 preload voice segments.

You can preload voice segments by compiling the configuration file ShIndex.ini, the detailed information on which can be found in Preload Voice Configuration File ShIndex.ini (CTI Series) in Section 3.2.

Also the application program is able to preload voice segments dynamically via function calls. See the table below for relative functions.

- Functions for loading and unloading voice segments

Function Name	Description
SsmSetMaxIdxSeg	Sets the total number of preload voice segments.
SsmGetTotalIndexSeg	Obtains the total number of preload voice segments.
SsmLoadIndexData	Loads the specified voice segments to the memory.
SsmFreeIndexData	Unloads the preload voice segments from the memory.

- Specific functions for playing preload voice segments

Function Name	Description
SsmPlayIndexList	Plays one or multiple specified preload voice segments in sequence.
SsmPlayIndexString	Plays one or multiple specified preload voice segments in sequence.
SsmStopPlayIndex	Stops playing preload voice segments.

1.10.1.4 Single File Mode

This mode is used for playing a single voice file, including the WAV file, the MP3 file and the non-header file (without file header, in need of extra designation of CODEC).

Related functions and configuration items are listed in the table below.

Function/Event Name	Description
SsmPlayFile	Starts playing a single file.
SsmStopPlayFile	Stops playing a single file.
SsmPausePlay	Suspends playing a single file.
SsmRestartPlay	Resumes file playing which is paused by the function call of SsmPausePlay.
SsmFastFwdPlay	Continues file playing after 1 sec. of fast forward.
SsmFastBwdPlay	Continues file playing after 1 sec. of fast backward.
SsmSetPlayTime	Continues file playing from the position specified by time.
SsmSetPlayPrct	Continues file playing from the position specified by percentage.
SsmGetDataBytesToPlay	Queries the amount of unplayed voice data bytes.
SsmGetDataBytesPlayed	Queries the amount of played voice data bytes.
SsmGetPlayingFileInfo	Obtains information on the voice file being played.
SsmGetPlayedTimeEx	Obtains the duration of voice playing from the first byte of the file to the current position.
E_PROC_PlayFile	An event which tells the playing progress in Single File Mode.

1.10.1.5 File List Mode

In this mode, the application first consecutively submits multiple voice files with the same CODEC. Then the driver automatically plays those files according to the submitting sequence, making seamless connection of files.

The function SsmAddToFileList is used to submit necessary files to be played to the driver. The maximum amount of submitted files is determined by the configuration item MaxPlayFileList.

Once the application invokes the function SsmPlayFileList, the driver starts a task of playing multiple files from the 1st one. Every time when a file is completely played, the driver will send the event E_PROC_PlayFileList to the application and go to the next file. After all files are played, the driver automatically closes all of them, terminates the play task and launches the E_PROC_PlayEnd to the application.

Related functions and output events are shown in the table below.

Function/Event Name	Description
SsmAddToFileList	Submits a file to be played to the driver.
SsmPlayFileList	Starts voice playing in File List Mode.
SsmClearFileList	Closes and clears all voice files submitted to the driver.
SsmStopPlayFileList	Stops voice playing in File List Mode.
E_PROC_PlayFileList	An event thrown out by the driver when a voice file is completely played

1.10.1.6 Single Buffer Mode

In this mode, the application allocates a buffer for itself, writes voice data into the buffer and then submits a play task to the driver. The driver launches an event to the application upon completing the play of all data in the buffer and ends the play task.

Related functions and output events are displayed in the table below.

Function/Event Name	Description
SsmPlayMem	Starts voice playing in Single Buffer Mode.

SsmSetStopPlayOffset	Resets the position in the buffer where the play stops.
SsmStopPlayMem	Stops voice playing in Single Buffer Mode.
SsmGetPlayOffset	Obtains the pointer to data being played in the driver during voice playing in Single Buffer Mode.
E_PROC_PlayMem	An event which tells the playing progress in Single Buffer Mode

1.10.1.7 Buffer List Mode

In this mode, the application program submits ahead of time a group of voice data blocks respectively in multiple buffers. Then the driver plays them consecutively according to the set sequence, making seamless connection of voice data blocks. Note that those blocks are managed by the application itself.

Relative functions are shown below:

Function Name	Description
SsmAddToPlayMemList	Submits a buffer to the driver.
SsmClearPlayMemList	Clears all buffers submitted to the driver.
SsmPlayMemList	Starts voice playing in Buffer List Mode.
SsmStopPlayMemList	Stops voice playing in Buffer List Mode.

1.10.1.8 Pingpong Buffer Mode

This mode uses double buffers. Its operation principle is: Before the start of voice playing, the application prepares one (or two) buffer full of voice data to be played, and then invokes the function SsmPlayMemBlock once (or consecutively twice) to submit the buffer to the driver. When the driver receives the play command, it will perform voice playing in Pingpong Buffer Mode, that is, upon a complete play of voice data in the first buffer, the driver will automatically play those in the second and meanwhile invoke the callback function from the application to pass an end-of-play message to the application. After that, the application will fill data into the next buffer and restart such process.

Compared with Single Buffer Mode, Pingpong Buffer Mode has the following characteristics.

- The minimum buffer size can be set to 192 bytes;
- Support for callback programming.

Since neither Windows nor Linux is a true real-time operating system, there is bound to be some delay in transmitting voice data from the application buffer to the on-board voice decoder. Besides, due to the indirect transmission of voice data, first from the application program (User Mode) to the device driver (Kernel Mode), then from the device driver to the board, the delay in time is not invariable but changes depending on how busy the application is. A relatively effective way to shorten the delay time is to reduce the buffer size and enhance the priority for the application to update voice data at the same time. The driver's thread priority is quite high, and in this case all related source codes in the callback function can acquire the priority as high as that of the driver. From the above we can see that the use of Pingpong Buffer Mode can minimize this kind of delay, so it is applicable to those occasions that demand high real-time capability in voice playing.

Related functions are shown below:

Function Name	Description
SsmPlayMemBlock	Submits a buffer to the driver and starts voice playing in Pingpong Buffer Mode.
SsmStopPlayMemBlock	Stops voice playing in Pingpong Buffer Mode.

1.10.1.9 Common Termination Function

The function SsmStopPlay can be used to stop any kind of play tasks.

1.10.1.10 Acquiring Information on Voice Playing

The following functions can be called to obtain relative information about voice playing.

Function Name	Description
SsmQueryPlayFormat	Queries if the channel supports the specified playing format.
SsmGetPlayType	Queries the mode of voice playing performed currently on the channel.
SsmCheckPlay	Queries various situations during voice playing.
SsmGetPlayedPercentage	Obtains the percentage of the played data to the entire.
SsmGetPlayedTime	Obtains the duration of voice playing.

1.10.2 Voice Recording

1.10.2.1 Setting Recording Signal Source

The recording signal source varies on the board model. Refer to the operation principle of corresponding board Series for details.

1.10.2.2 Setting Termination Condition

After the start of voice recording, other than the occasion where the task stops naturally by itself or is terminated by the application through function calls, the driver will automatically end it once the channel detects some events triggered by particular conditions, simplifying compilation of the application program. The 'particular conditions' mentioned here include:

(1) The DTMF Detector detects DTMF digits in the incoming signals.

Related functions and configuration items are listed below.

Function Name	Description	Related Configuration Item
SsmSetDTMFStopRecCharSet	Sets whether to stop voice recording as the DTMF Detector detects DTMF digits.	DtmfStopRecCharSet

(2) Call state machine detects remote hangup.

Related functions and configuration items are listed below.

Function Name	Description	Related Configuration Item
SsmSetHangupStopRecFlag	Sets whether to stop voice recording as the driver state machine detects remote hangup.	HangupStopRec

1.10.2.3 Setting Prerecord Feature

Prerecord is a process that the driver starts a prerecord task according to reset parameters and stores the recorded data in its internal buffers; thus, when the application invokes functions to record data into a file, it can copy the recorded data in the buffer from the driver to this file before the start of normal recording. This feature helps prevent loss of the foremost part of voice data during File Mode recording caused by delay in application response and the like.

As to the buffer for prerecord data, the size can be set via the configuration item RecordBufSize.

Related functions and configuration items are listed below.

Function Name	Description	Related Configuration Item
SsmSetPrerecord	Sets the prerecord feature.	PrerecordEnable PrerecordMode PrerecordInsertTime PrerecordCodec
SsmGetPrerecordState	Obtains the prerecord settings.	

1.10.2.4 Setting Tail-Truncation Feature

When the application stops recording data into files, the driver will automatically truncate the last section of the recorded voice data. This feature is applicable to some particular occasions, for example, in case the driver's detection of DTMF signals stops the File Mode recording and the application doesn't like those DTMF digits shown in the file.

This feature only supports File Mode recording and usually works with the settings' automatic stop of voice recording upon detection of DTMF'.

Related functions and configuration items are listed below.

Function Name	Description	Related Configuration Item
SsmSetTruncateTail	Sets the parameters for the tail-truncation feature	TruncateTailOnRecordToFile
SsmGetTruncateTailTime	Obtains the preset parameters for the tail-truncation feature.	

1.10.2.5 Setting AGC Feature

If the AGC (Automatic Gain Control) feature is enabled, the driver will automatically adjust the input signal amplitude, increasing that of small signals and decreasing that of large signals.

The function SsmSetRecAGC is used to enable or disable the AGC feature, whose parameters can be set via the following configuration items.

- AGCMAXGAIN
- AGCMINGAIN
- AGCMAXLEVEL
- AGCMINLEVEL
- AGCDOWNRATIO
- AGCUPRATIO
- AGCKEEPTIME

For DST-24B/PCI, DST-24B/PCI+, DST-24B/PCIe, DST-24B/PCIe+ boards, the parameters for AGC module can be set both uplink and downlink via the following configuration items:

- AGCMAXGAINUPLINK
- AGCMINGAINUPLINK
- AGCMAXLEVELUPLINK
- AGCMINLEVELUPLINK
- AGCDOWNRATIOUPLINK
- AGCUPRATIOUPLINK
- AGCMAXGAINDOWNLINK
- AGCMINGAINDOWNLINK
- AGCMAXLEVELDOWNLINK
- AGCMINLEVELDOWNLINK
- AGCDOWNRATIODOWNLINK
- AGCUPRATIODOWNLINK

Note:

- **The AGC operation principle for the Synway voice board is to adjust the gain of amplifiers in AGC circuits according to the voltage level of originally input signals so as to control the output signal level within an expected range. By setting AGC parameters, we can provide various AGC effects to satisfy different application environments and different users. Our AGC parameters may be divided into three categories: first, AGCMAXGAIN (AGCMAXGAINUPLINK, AGCMAXGAINDOWNLINK) and AGCMINGAIN(AGCMINGAINUPLINK, AGCMINGAINDOWNLINK), used to control the range of DSP gain increase and decrease; second, AGCMAXLEVEL(AGCMAXLEVELUPLINK, AGCMAXLEVELDOWNLINK) and AGCMINLEVEL(AGCMINLEVELUPLINK, AGCMINLEVELDOWNLINK), used to set the expected value of the AGC output voltage level; third, AGCDOWNRATIO (AGCDOWNRATIOUPLINK,AGCDOWNRATIODOWNLINK),AGCUPRATIO(AGCUPRATIOUPLINK, AGCUPRATIODOWNLINK) and AGCKEEPTIME, used to control the speed of AGC gain decrease/increase.**
- **Our default AGC settings are suitable for most application environments. If you are required to reset the parameters for AGC in particular cases, please do it under the instruction of our technicians.**
- **Because the driver will automatically disable the AGC feature when the channel state transfers to S_CALL_STANDBY, although the application can still perform recording on the channel, the data recorded afterwards are obviously not endued with the AGC feature. If the application would like to recover the AGC feature in this case, it is necessary to set the configuration item OpenRecEnAndPlayEnOnIdle.**

1.10.2.6 Acquiring Information on Voice Recording

Related functions are shown below.

Function Name	Description
SsmQueryRecFormat	Queries if the channel supports the specified recording format.
SsmGetRecType	Queries the mode of voice recording performed currently on the channel.
SsmGetRecTime	Obtains the duration of voice recording.
SsmCheckRecord	Queries various situations during voice recording.

1.10.2.7 Brief Introduction of Recording Functions

The driver will allocate in itself a recording buffer for each channel. The buffer size can be set via the configuration RecordBufSize item, with a default value of 32768 bytes.

There are 3 categories of recording functions respectively for 3 recording modes: File Mode, Buffer Mode and Pingpong Buffer Mode.

1.10.2.7.1File Mode

Related functions are shown below.

Function Name	Description
SsmRecToFile SsmRecToFileA SsmRecToFileEx	Start voice recording in File Mode.

SsmStopRecToFile	Stops voice recording in File Mode.
SsmPauseRecToFile	Suspends voice recording in File Mode.
SsmRestartRecToFile	Resumes voice recording in File Mode.
SsmChkRecToFile	Queries various situations during voice recording in File Mode.
SsmGetDataBytesToRecord	Obtains the amount of unrecorded voice data bytes.

1.10.2.7.2Buffer Mode

In this mode, the application program first calls the function SsmRecToMem to submit a buffer to the driver. Then the driver writes the recorded data into this buffer and throws out progress messages according to the set conditions.

Related functions are shown below.

Function Name	Description
SsmRecToMem	Starts voice recording in Buffer Mode.
SsmStopRecToMem	Stops voice recording in Buffer Mode.
SsmGetRecOffset	During voice recording in Buffer Mode, obtains the pointer to the position in the driver buffer where the recorded data are written.

1.10.2.7.3Pingpong Buffer Mode

This mode uses double buffers. Its operation principle is: Before the start of voice recording, the application prepares one (or two) recording buffer, and then invokes the function SsmRecordMemBlock once (or consecutively twice) to submit the buffer to the driver. After the application calls SsmRecordMemBlock for the first time, the driver will start voice recording in Pingpong Buffer Mode and write the recorded data into the buffer. When one buffer is full, the driver will automatically shift to another and meanwhile invoke the callback function from the application to pass an end-of-record message to the application. Via this callback function, the application can call SsmRecordMemBlock to submit a new buffer and do subsequent processing of the recorded data.

Compared with Buffer Mode, Pingpong Buffer Mode has the following characteristics.

- The minimum buffer size can be set to 192 bytes.
- Support for the callback operation mode.

Since neither Windows nor Linux is a true real-time operating system, there is bound to be some delay in transmitting recorded data from the board to the application buffer. Besides, as voice data need to be passed from the device driver (Kernel Mode) to the application program (User Mode), involving the switch from Kernel Mode to User Mode, the delay in time is not invariable but changes depending on how busy the application is. A relatively effective way to shorten the delay time is to reduce the buffer size and enhance the priority for the application to process voice data at the same time. The driver's thread priority is quite high, and in this case all related source codes in the callback function can acquire the priority as high as that of the driver. From the above we can see that the use of Pingpong Buffer Mode can minimize this kind of delay, so it is applicable to those occasions that require high real-time capability in voice recording.

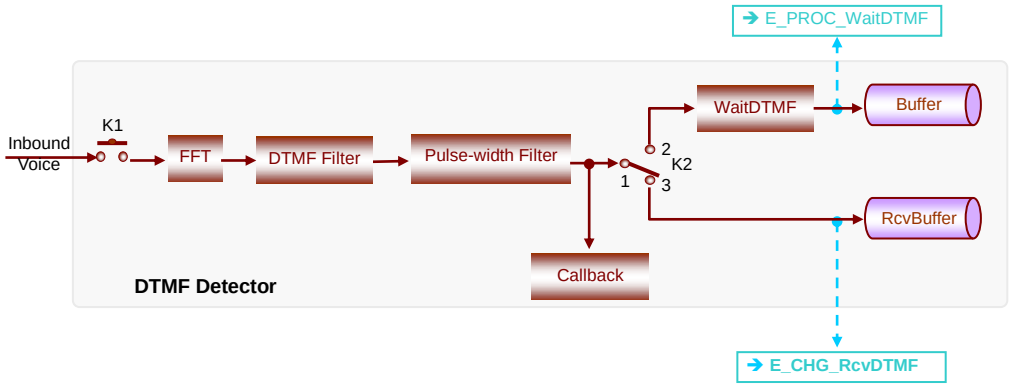
Related functions are shown below:

Function Name	Description
SsmRecordMemBlock	Submits a buffer to the driver and starts voice recording in Pingpong Buffer Mode.
SsmStopRecordMemBlock	Stops voice recording in Pingpong Buffer Mode.

1.10.3 DTMF Detector

All Synway boards provide an independent DTMF detector for each channel on them.

The following figure shows the operation principle of the DTMF detector.



The switch K1 is controlled by the configuration item AlwaysEnableRxDtmf, the driver and the application together. When AlwaysEnableRxDtmf is set to 1, K1 is switched on; when AlwaysEnableRxDtmf is set to 0, it is the driver that determines the state of K1 based on channel state transition: switch on K1 as the channel turns to the talking state; switch off K1 as the channel leaves the talking state followed by call disconnection. Also the application can invoke the function SsmEnableRxDtmf to control the state of K1.

When a station channel stays in the idle state, the DTMF detector is disabled under the automatic control of the driver, which however can be enabled via the configuration item RcvDtmfOnIdle.

FFT (Fast Fourier Transform) figured above indicates FFT signal processing provided to transform input voice signals from the time domain to the frequency domain.

The DTMF filter examines the frequency of input signals to see if there are DTMF digits and outputs its judgement by voltage level.

The pulse-width filter is used for eliminating the pulse width of DTMF signals so as to avoid the effect of interlarded DTMF signals on the examining result. The detected DTMF digits are output via the switch K2 while being passed to the callback function.

The switch K2 is automatically controlled by the driver to determine the outgoing route of DTMF signals. Usually the driver puts K2 at the 1-3 position, stores the detected DTMF digits to the buffer RcvBuffer, and meanwhile sends the event E_CHG_RcvDTMF to the application program.

RcvBuffer holds the detected DTMF, whose size can be specified via the configuration item RxDtmfBufSize. The following functions helps access RcvBuffer.

Function Name	Description
SsmClearRxDtmfBuf	Empty the buffer in the driver which stores DTMF digits, i.e. the DTMF

	detector buffer.
SsmGetDtmfStr SsmGetDtmfStrA	Obtains the DTMF digits saved in the DTMF detector buffer.
SsmGetRxDtmfLen	Obtains the number of DTMF digits saved in the DTMF detector buffer.
SsmGet1stDtmf SsmGet1stDtmfClr	Obtains the first DTMF digit received by the DTMF detector buffer.
SsmGetLastDtmf	Obtains the last DTMF digit received by the DTMF detector buffer.

When the application invokes the function SsmSetWaitDtmf to start a task of WaitDtmf, K2 will automatically switch to the 1-2 position and the detected DTMF digits be sent to the WaitDtmf component for subsequent processing. Note that none of the detected DTMF digits will enter RcvBuffer after the task of WaitDtmf is started, however, the DTMF digits which already exist in RcvBuffer will be copied to the buffer.

1.10.3.1 DTMF Filter

The DTMF filter examines the frequency of input signals to see if there are DTMF digits involved, and prevents the mistaking of voice signals for DTMF.

The corresponding relationship between DTMF digits and the frequency is shown in the following table.

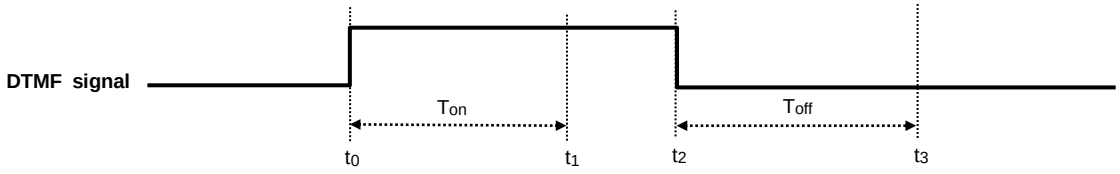
	1209Hz	1336Hz	1477Hz	1633Hz
697Hz	1	2	3	A
770Hz	4	5	6	B
852Hz	7	8	9	C
941Hz	*	0	#	D

The following configuration items and functions help ensure the DTMF filter to make accurate judgements.

Configuration Item or Function	Description
Configuration item: DualAndAllFreqEnScale Function: SsmSetFlag (with the parameter F_DualAndAllFreqEnScale)	Sets the threshold value in percentage for the rate of DTMF in-band energy to overall energy. The DTMF filter calculates the percentage of DTMF in-band energy to overall energy in real time and compares it with a set threshold value. Only if the percentage is greater than the threshold value will the DTMF filter conclude that the on-line signals are DTMF and output the judgement by voltage level (High voltage level means detection of DTMF signals while low level implies nondetection).
DtmfEnergy	Sets the absolute value of DTMF in-band energy. Only if the DTMF in-band energy is greater than the set value of this configuration item will the driver confirm that those detected are DTMF signals.
Configuration item: HighAndLowFreqEnScale Function: SsmSetFlag (with the parameter F_HighAndLowFreqEnScale)	Sets the proportion of the high-frequency DTMF energy to the low-frequency. Only if the proportion is greater than the set value of this configuration item will the driver confirm that those detected are DTMF signals.

1.10.3.2 DTMF Pulse-width Filter

The figure below shows the operation principle of the DTMF pulse-width filter.



In the figure above, T_{on} and T_{off} represent the minimum DTMF signal durations respectively at on and off states (voltage level). In practice, after the voltage level of the input DTMF signal transforms from low to high, only if the signal stays at on or off state for more than T_{on} /T_{off} will the driver conclude that it is a real DTMF digit that goes through the line, write this digit into a buffer and send the event E_CHG_RcvDTMF to the application program at the same time.

T_{on} and T_{off} are set via the configuration item ReceiveDtmfSensitive or the function SsmSetFlag (with the parameter F_RCVDTMFSENS).

The configuration item OmitABCD is used to determine on whether to discard the received a, b, c or d digit. If this item is set to RESERVED, the a, b, c or d will be sent to the subsequent processing component with other digits, otherwise be discarded.

The configuration item DtrmOnDtmfHighLevel is used to determine the time for the driver to output the DTMF digit. In the use of the high-level output setting, the driver will immediately detect the output of DTMF digits when the signal last at on state for more than T_{on} (i.e. at the time t_1 in the figure above); in the use of low-level output setting, the driver can detect the output of DTMF digits only when the signal transforms from on to off state and keeps at off state for more than T_{off} (i.e. at the time t_3 in the figure above).

1.10.3.3 WaitDtmf

WaitDtmf is a task of receiving particular DTMF digits submitted to the driver by the application. It is started when the application invokes the function SsmSetWaitDtmf. The DTMF digits detected afterwards by the DTMF detector will not enter RcvBuffer but come directly into the WaitDtmf component. Then the driver determines on whether to terminate this task according to preset conditions: it will send the event E_PROC_WaitDTMF to the application and end this WaitDtmf task once the termination condition is satisfied.

Related functions are listed below.

Function/Event Name	Description
SsmSetWaitDtmf SsmSetWaitDtmfEx SsmSetWaitDtmfExA	Start the WaitDtmf task.
SsmCancelWaitDtmf	Cancel the WaitDtmf task.
SsmChkWaitDtmf	Queries the WaitDtmf task progress.
E_PROC_WaitDTMF	Reports the WaitDtmf task progress.

1.10.3.4 Callback Programming

The Callback component is used to process the application's callback function. Provided the application has configured a callback function by calling the function SsmSetRxDtmfHandler, the driver will invoke this callback function when it detects DTMF digits.

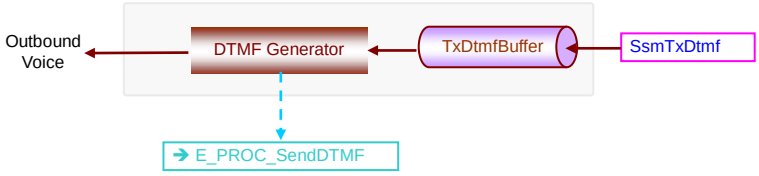
1.10.3.5 Dial Pulse Detection

The DTMF detector for the analog trunk recording channel is able to detect the dial pulse provided the configuration item EnablePulseKeyDetect is properly reset.

1.10.4 DTMF Generator (CTI Series)

The DTMF generator can produce standard DTMF digits on the line, including '0', '1', '2', '3', '4', '5', '6', '7', '8', '9', '*', '#', 'A', 'B', 'C', 'D'.

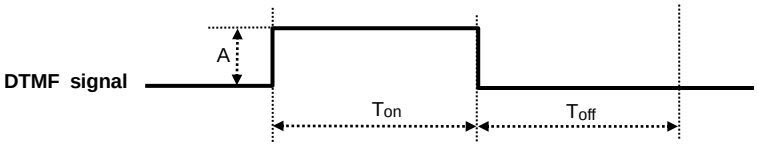
The following figure shows the operation principle of the DTMF generator.



The Synway board prepares an independent DTMF generator and a TxDtmfBuffer for each channel.

The size of TxDtmfBuffer can be designated via the configuration item TxDtmfBufSize, with a default value of 50 characters.

The DTMF generator is able to create DTMF digits which in frequency and amplitude completely meet the country criteria for DTMF dialing. Below is a typical DTMF signal produced by the DTMF generator.



As shown above, T_{on} and T_{off} are used to describe the waveform of a DTMF digit: T_{on} indicates the signal duration at on state while T_{off} represents the signal duration at off state. Both of them can be set via the configuration item TxDtmfTimePara or the function SsmSetTxDtmfPara. The function SsmGetTxDtmfPara is for obtaining the T_{on} and T_{off} settings.

The signal amplitude A can be set via the function SsmSetFlag (with the parameter F_TXDTMFAMP). By default, the value for SHT, ATP series is 0X8976; the value for SHD, DTP series is 0X3f3f, among which the 8 higher bits set the amplitude of low frequency and the 8 lower bits set the amplitude of high frequency. The conversion between Amplitude A and DB is Y=20*lg(X/16), among which Y is the DB value and X is the set value. Take the default value 0X8976 for example. According to the conversion formula, the low-frequency DB value is 18.65=20*lg(137/16) and the high-frequency DB value is 17.35=20*lg(118/16).

After the application invokes the function SsmTxDtmf, the diver will store the DTMF digits to be transmitted in TxDtmfBuffer and start a task to transmit DTMF at the same time. Every time when the DTMF generator finishes transmitting a digit, it will automatically take the next one out of TxDtmfBuffer for transmission. As TxDtmfBuffer gets empty, which indicates termination of the task, the driver will send the event E_PROC_SendDTMF to the application.

You can use the function SsmStopTxDtmf to stop transmission at any time and the function SsmChkTxDtmf to

query the task progress.

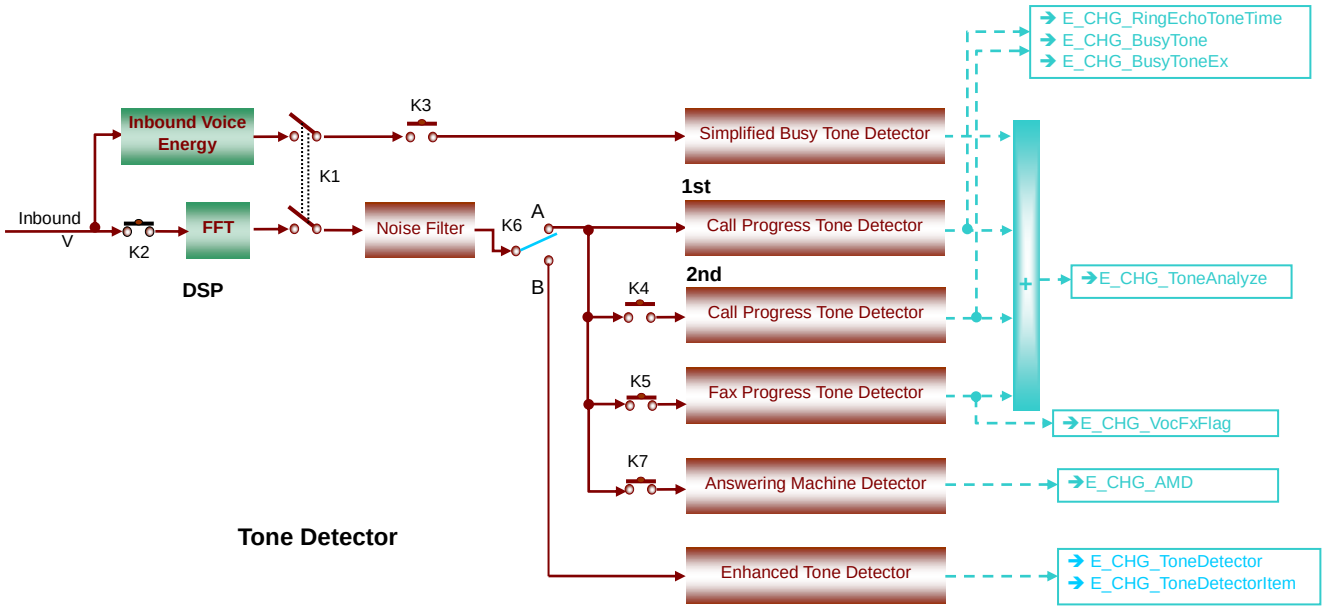
1.10.5 Tone Detector

Each channel on the Synway board has an independent tone detector which is mainly for:

- Call progress tone detection on the analog line.
- Fax tone detection.
- Other special applications, such as tone detection on the digital trunk channel in the automatic call testing system.

Tip: Skip this section if your system does not involve such applications mentioned above. Otherwise, if your system contains analog trunk channels or analog trunk recording channels, read through this section carefully as it is quite essential to your system's compatibility.

The figure below shows the operation principle of the tone detector and the event outputs concerned.



The switch K1, which can be set via the function SsmStartToneAnalyze/ SsmCloseToneAnalyze, controls the operation of the tone detector as well as the remote pickup detector. It is under an automatic control of the driver based on channel state transition and switched on after system initialization. Refer to the section on channel state machine for details.

K2 is automatically controlled by the DSP program: it is switched off when starting the on-channel FSK receiver, and switched on otherwise.

K6 determines the use of the enhanced tone detector. It can be set by the configuration item ToneDetectorMode. By default, K6 is connected with A, using the 1st and 2nd call progress tone detectors as well as the fax progress tone detector for detection. When K6 is connected with B, the enhanced tone detector is used for detection.

The tone detector is made up of the following components.

- **FFT**

FFT indicates FFT signal processing provided to transform input voice signals from the time domain to the frequency domain. The processing results are sent to the subsequent component for analysis and calculation. Also the overall energy of on-line voice signals is calculated in this component. Since FFT calculation is performed in DSP chips, no host resource needs to be used.

- **Noise Filter**

  Used for eliminating noises on the line.

- **Call Progress Tone Detector**

  It is used to examine if there are call progress tones, whether single tones or dual tones, on the line. Each tone detector has 2 call progress tone detectors independent to each other. The switch K4, which is switched off in default and can be set via the configuration item Enable2ndToneAnalyzer or the function SsmStart2ndToneAnalyzer, determines whether to use the second call progress tone detector or not.

- **Fax Progress Tone Detector**

  It is used to examine if there are single fax progress tones on the line. Each tone detector has 2 fax progress tone detectors independent to each other. This detector becomes useless when K4 is connected to the second call progress tone detector.

- **Simplified Busy Tone Detector**

  When the application program starts the FSK receiver on the analog trunk channel, the on-board DSP program will automatically switch off K2 so that the tone detector is disconnected and fails to work in a normal state. However, some particular occasions using the SHT Series boards require the application to detect busy tones while receiving FSK data to ensure an accurate judgement on the remote hangup behavior in time. The simplified busy tone detector is just designed for such requirements. The application can use it simply by switching on K1 and K3 at the time when K2 is automatically switched off by the DSP program. As this detector only accepts the energy of incoming signals output by the Barge-in detector, it is unable to conduct sufficient analysis over the signal frequency but has to examine the signal waveform to make sure if there are busy tones on the line. Therefore, we name it the simplified busy tone detector.

- **Enhanced Tone Detector**

  As a perfect substitute for call progress tone detector and fax progress tone detector, the enhanced tone detector can detect not only the kinds of tones detected by these two detectors, but also some particular tones (such as SIT signals).

  Note: So far only a part of Synway boards listed below support the enhanced tone detector. See the table for details.

Board Model	Enhanced Tone Detector Supported or Not
SHT-8A/PCI	Supported
SHT-8B/PCI	Supported
SHT-8B/PCI/FAX	Supported
SHT-8C/PCI/FAX	Supported
SHT-8C/PCI/EC	Supported
SHT-16A-CT/PCI	Supported
SHT-16B-CT/PCI	Supported
SHT-16B-CT/PCI/FAX	Supported

SHT-16B-CT/cPCI	Supported
SHT-16B-CT/cPCI/MP3	Supported
SHT-16B-CT/cPCI/FAX	Supported
SHT-16C-CT/PCI/FAX	Supported
SHT-16C-CT/PCI/EC	Supported
SHT-16D-CT/PCIe	Supported
SHT-4A/USB	Supported
SHT-2A/USB	Supported
SHT-2B/USB	Supported
SHT-4B/USB	Supported
SHF-2D/PCI	Supported
SHF-4D/PCI	Supported
SHF-4D/PCIe	Supported

- **Answering Machine Detector**

It is used to distinguish the answering machine from the human being that answers after remote pickup. K7 is used to control whether the answering machine auto-run . This function also can be sets via the configuration item EnableAMD, The default value of K7 is connected (switched on).

The following configuration items also can influence the action of answering machine detector. Details are shows as below.

AMDNoSoundAfterDialTime: Uses to judge the line silence time after dial tone overtime or not. If the line silence time is over the value of AMDNoSoundAfterDialTime, events E_CHG_AMD will be throw out.(at this condition, the value of E_CHG_AMD is 5)

AMDNoSoundTime: Uses to judge the silence time after tone or color ring detected overtime or not. If the silence time is over the value of AMDNoSoundTime, events E_CHG_AMD will be throw out.(at this condition, the value of E_CHG_AMD is 4)

AMDTimeOut: Uses to judge the whole AMD detecting process overtime or not. If this process is over the value of AMDTimeOut, events E_CHG_AMD will be throw out.(at this condition, the value of E_CHG_AMD is 3)

AMDToneCount: Uses to judge if the tone detected time is overtime or not. If this time is over the value of AMDToneCount, the event E_CHG_AMD will be thrown out (at this condition, the value of E_CHG_AMD is 1).

AMDTOn: Uses to set the lowest duration when the voice goes into the High voltage state.

AMDTOff: Uses to set the lowest duration when the voice goes into the low voltage state.

AMDTimeA: Uses to set the lowest silence duration before the phone pickup by man detected.

AMDTimeB: Uses to set the lowest duration when the phone pickup by man detected.

AMDTimeC: Uses to set the maximal duration when the phone pickup by man detected.

AMDTimeD: Uses to set the lowest silence duration after the phone pickup by man detected.

AMDSilentEnergy: Uses to set an energy value that can judge the voice is silence or not.

1.10.5.1 FFT

The FFT component in the on-board DSP chips provides FFT signal processing, transforming input voice signals from the time domain to the frequency domain and performing a FFT calculation of on-line voice signals every 16ms. The results are input to the subsequent component.

Following the FFT signal processing, the driver will send the events listed below to the application.

- E_OverallEnergy: sends the overall energy calculated by FFT to the application. The application need not respond to this event in most cases.

- E_CHG_PeakFrq: sends the calculated frequency and in-band energy of the peak signal in the spectrum to the application. The function SsmSetPeakFrqDetectBW is used to set the bandwidth of the peak frequency. The application need not respond to this event in most cases. **Note that this event is thrown out only when the peak frequency changes.**

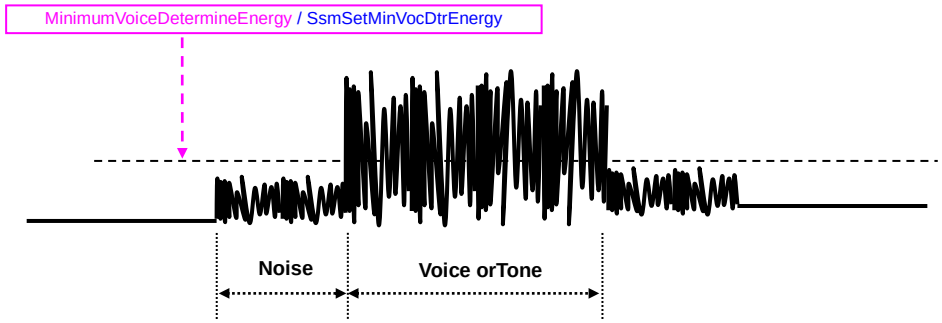
Note:

- **The FFT component quits working upon the start of the FSK receiver.**

1.10.5.2 Noise Filter

Usually there is no need to reconfigure the noise filter, for its default settings are adequate for most situations.

The following figure demonstrates the operation principle of the noise filter.

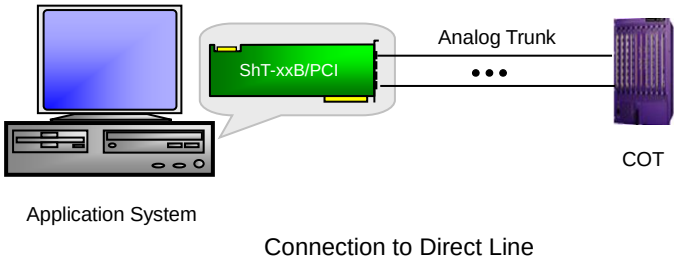


When the overall energy is less than the threshold value specified by the configuration item MinimumVoiceDetermineEnergy (or the function SsmSetMinVocDtrEnergy), the driver will identify the signal as a noise, otherwise as a true voice or tone.

1.10.5.3 Call Progress Tone Detector (SHT Series Only)

Since the analysis made by the call progress tone detector on the analog trunk channel has a direct effect on the call progress, it appears exceptionally important to properly use and configure the call progress tone. In general, application systems can be divided into two categories based on different connection methods: connection to direct line and connection to extension line.

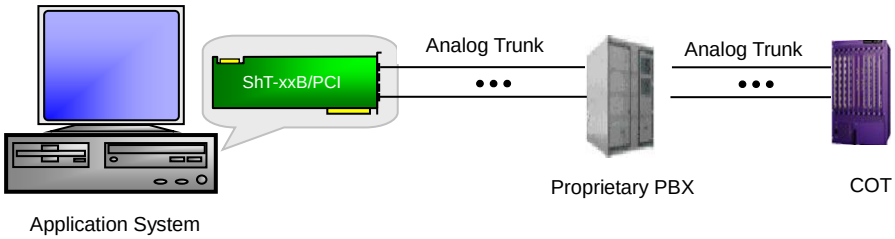
Below is the application system that connects to the direct line (COT: Central Office Terminal).



Connection to Direct Line

Such application system only requires the board driver to detect one kind of call progress tone.

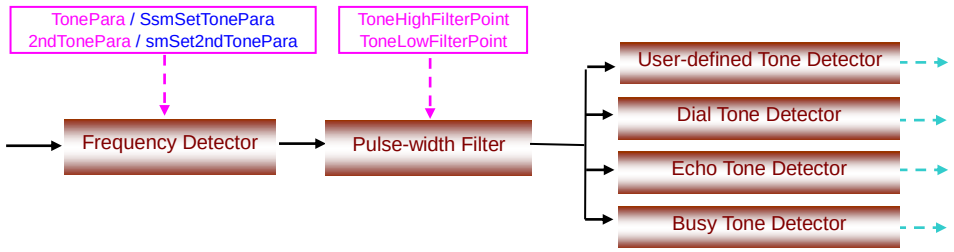
Below is the application system that connects to the extension line.



Connection to Extension Line

Since the proprietary PBX and the COT involved in the above situation probably differ in tone from each other, the board driver is required to detect two kinds of call progress tone at the same time.

There are two call progress tone detectors - the 1st and the 2nd - available for various cases. Its operation principle is shown in the figure below.



Call Progress Tone Detector

As illustrated above, the call progress tone detector consists of the frequency detector, the pulse-width filter, the dial tone detector, the echo (ringback) tone detector, the busy tone detector and the user-defined tone detector.

The user-defined tone detector is only applicable to some particular instances.

The dial tone detector, the echo (ringback) tone detector, the busy tone detector and the user-defined tone detector can work in double harness.

- **Frequency Detector**

The frequency detector examines the frequency of input signals to determine if there contain tones at the specified frequency. It calculates the percentage of in-band energy of dual tones to overall energy in real time and compares the result with a set threshold value. If the percentage is greater than the threshold value, it will conclude that the on-line signals are tones and output the judgement by voltage level (High voltage level means detection of tones while low level implies nondetection).

You can use the configuration item TonePara or the function SsmSetTonePara to set the two frequency parameters for the frequency detector in the 1st call progress tone detector, including the center frequency, the bandwidth and the threshold value for the percentage of in-band energy.

Likewise, you can use the configuration item 2ndTonePara or the function SsmSet2ndTonePara to set the two frequency parameters for the frequency detector in the 2nd call progress tone detector, including the center frequency, the bandwidth and the threshold value for the percentage of in-band energy.

- **Pulse-width Filter**

The pulse-width filter is used to filter signals by pulse width so as to eliminate the effect of interlarded tones on

detection. In case the voltage level of the signal transforms from low to high, only if the signal stays at on state for more than the time specified by the configuration item ToneHighFilterPoint will the driver confirm the tone's coming to being; In case the voltage level of the signal transforms from high to low, only if the signal stays at off state for more than the time specified by the configuration item ToneLowFilterPoint will the driver confirm the tone's disappearing.

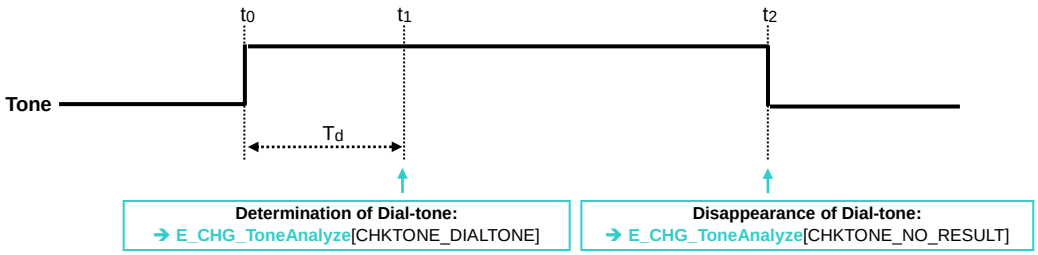
Note:

(1) The configuration parameters of the pulse-width filter also work on the 1st and 2nd call progress tone detectors as well as the fax progress tone detector.

(2) On the analog trunk channel or analog trunk recording channel, the results acquired by the dial tone detector, the echo (ringback) tone detector, the busy tone detector and the user-defined tone detector in the 1st call progress tone detector will automatically trigger corresponding events to change the state machine; however, whether the results obtained in the 2nd call progress tone detector trigger corresponding events to change the state machine or not is determined by the configuration item Check2ndToneOnAutoDial.

1.10.5.3.1Dial Tone Detector

Detection of dial tones is accomplished by the dial tone detector. The figure below shows the typical dial-tone waveform.

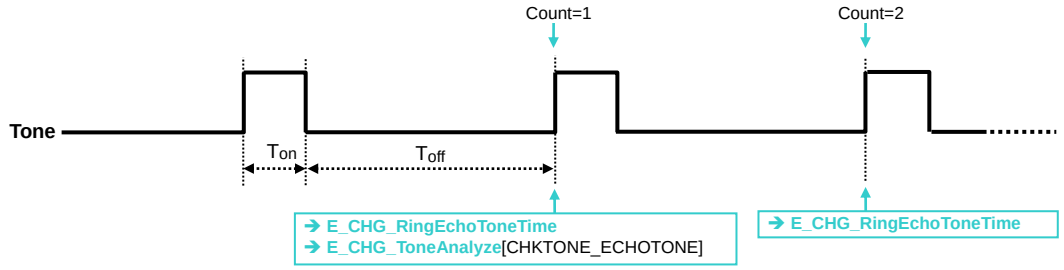


The tone detector will throw out the event E_CHG_ToneAnalyze (with the parameter CHKTONE_DIALTONE) once the tone duration at on state counted by the driver reaches the threshold value T_d which is set to judge the dial tone, and send off E_CHG_ToneAnalyze (with the parameter CHKTONE_NO_RESULT) when the dial tone disappears.

For the 1st call progress tone detector, T_d is specified by the configuration item IsDialToneDetermineTime or the function SsmSetIsDialToneDtrTime; for the 2nd (if available), T_d is designated by the configuration item 2ndIsDialToneDetermineTime or the function SsmSet2ndIsDialToneDtrTime.

1.10.5.3.2Ringback Tone Detector

Detection of ringback tones is accomplished by the ringback tone detector. The figure below shows the typical ringback-tone waveform.



Main parameters for the ringback tone includes T_{on} (duration at on state) and T_{off} (duration at off state). Once detecting a signal either at on or off state lasts for a time that matches the set value, the call progress tone detector considers the signal as a ringback tone and adds 1 to the ringback tone count. Both T_{on} and T_{off} accept the error of ±20%.

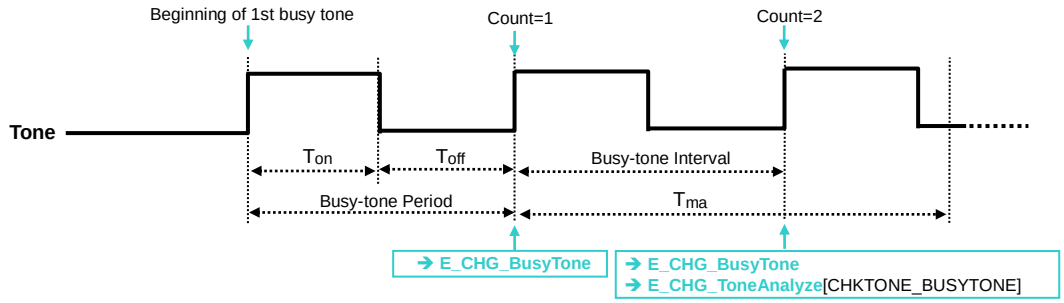
For the ringback (echo) tone detector in the 1st call progress tone detector, the configuration item RingEchoTonePara or the function SsmSetRingEchoTonePara can be used to set T_{on} and T_{off}; for that in the 2nd (if available), the configuration item 2ndRingEchoTonePara or the function SsmSet2ndRingEchoTonePara is used.

The driver will send off the event E_CHG_RingEchoToneTime every time when the count output by the ringback tone counter on a ringback tone detector in the 1st or 2nd call progress tone detector varies, but throw out the event E_CHG_ToneAnalyze (with the parameter CHKTONE_ECHOTONE) only when the count turns to 1. Also the function SsmGetRingEchoToneTime can be used to acquire the duration of ringback tones.

1.10.5.3.3Busy Tone Detector

1.10.5.3.3.1 General Features

Detection of busy tones is accomplished by the busy tone detector. The figure below shows the typical busy-tone waveform.



As shown in the figure above, a busy tone cycle falls into two parts: one corresponding to the signal duration at on state (T_{on}) and the other to the duration at off state (T_{off}). Because the duty ratio of busy tones is 50%, the value of T_{on} is equal to that of T_{off} and the driver only uses the busy tone cycle to determine if the on-line signals are busy tones. The acceptable error in the busy tone cycle is ±20%.

Once the signal duration at on or off state matches the set parameters of busy tones, the busy tone detector will consider the input signals as busy tones, add 1 to the corresponding busy tone counter (i.e. 'Count' mentioned in the above figure) and throw out the E_CHG_BusyTone event; when the count output by the busy tone counter equals a specified value, it will send off the event E_CHG_ToneAnalyze (with the parameter

CHKTONE_BUSYTONE). The results from the busy tone detector also can be obtained via the following functions.

Function Name	Description
SsmGetBusyAMDToneCount	Obtains the count of busy tone cycles output by the 1st call progress tone detector.
SsmGet2ndBusyAMDToneCount	Obtains the count of busy tone cycles output by the 2nd call progress tone detector.
SsmGetBusyToneLen	Obtains the busy tone duration output by the 1st call progress tone detector.
SsmGet2ndBusyToneLen	Obtains the busy tone duration output by the 2nd call progress tone detector.

Each busy tone detector is able to examine 4 groups of busy tones which differ in cycle at one time.

Configuring the busy tone detector in the 1st call progress tone detector:

- The configuration item BusyTonePeriod (or the functions SsmSetBusyTonePeriod, SsmSetBusyTonePeriodEx) is used to set the busy tone cycle;
- The configuration item IsBusyToneDetermineCount (or the function SsmSetIsBusyToneDtrCnt) is used to determine upon detection of which busy tone does the driver send off the E_CHG_ToneAnalyze event.

Configuring the busy tone detector in the 2nd call progress tone detector:

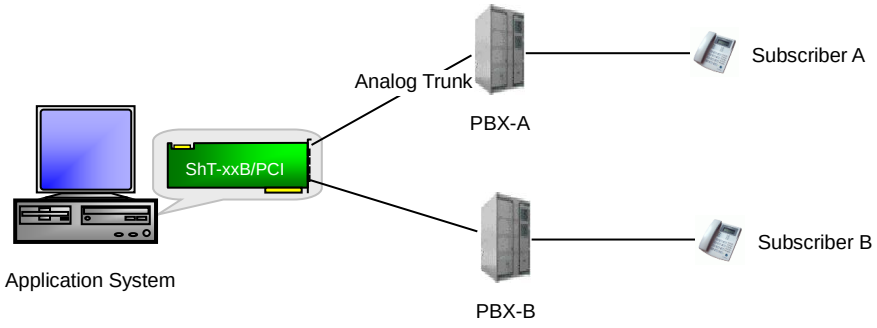
- The configuration item 2ndBusyTonePeriod (or the functions SsmSet2ndBusyTonePeriod, SsmSetBusyTonePeriodEx) is used to set the busy tone cycle;
- The configuration item 2ndIsBusyToneDetermineCount (or the function SsmSet2ndIsBusyToneDtrCnt) is used to determine upon detection of which busy tone does the driver throw out the E_CHG_ToneAnalyze event.

In practice, the driver may mistake some signals resembling busy tones occasionally involved in voice signals as busy tones. To solve such problem, the busy tone detector sets a guard time threshold value (marked as T_{max}). When the interval between two busy tone cycles gets longer than T_{max}, the busy tone counter will automatically set the count to 0 so as to avoid misdetection.

The configuration item MaxBsTnOffTime is used to set T_{max} and is applicable to busy tone detectors in both the 1st and 2nd call progress tone detectors.

1.10.5.3.3.2 Back-to-back Busy Tone Detection

The conventional method to detect busy tones may become useless on the analog trunk channel in some particular instances, such as in the application system based on the SHT Series boards shown below.



The SHT board in the application system uses two trunk channels to connect with Subscriber A and Subscriber B respectively through PBX-A and PBX-B, which enables a two-way call between A and B via CT bus. When one party (e.g. Subscriber A) hangs up and PBX-A sends busy tones to the board, Subscriber B will hang up too upon hearing the busy tone and command PBX-B to deliver busy tones to the board. At that time if the on-board circuits are not thoroughly disconnected yet, busy tones on either channel will go distorted and lose the regular waveform under the influence of echoes. As a result, the busy tone detector fails to detect busy tones accurately and the two channels stay occupied all the time.

To solve such problem, the busy tone detector is designed with the capability of back-to-back busy tone detection, which requires no extra configuration. However, the result of back-to-back busy tone detection is not contained in those output by the tone detector but supplied solely by the E_CHG_BusyToneEx event to the application program. Also it can be acquired by using the function SsmGetBusyToneEx.

Note: Only the 1st call progress detector supports back-to-back busy tone detection.

1.10.5.3.4User-defined Tone Detector

The user-defined tone detector is only applicable to some particular occasions, such as when it is required to detect some user-defined waveforms during the runtime of the dial tone detector, ringback tone detector and busy tone detector. This detector is able to detect both continuous and periodic tones.

Configuring the user-defined tone detector in the 1st call progress tone detector:

- The configuration item AppointedToneAnalyzerSwitch is used to start or stop the detection.
- The configuration item IsAppointedToneDetermineTime is used to set the minimum duration of continuous signals. This detector judges continuous signals in the same way as the dial tone detector. When the tone continues for more than the time specified by this configuration item, the driver will throw out the event E_CHG_ToneAnalyze (with the parameter CHKTONE_APPOINTEDTONE).
- The configuration items AppointedTonePara and IsAppointedToneDetermineCount are used to set the durations of periodic signals respectively at on and off states as well as the minimum count of tone cycles. This detector judges periodic signals in the same way as the ringback tone detector. When the count obtained is greater than the minimum count designated by IsAppointedToneDetermineCount, the driver will send off the event E_CHG_ToneAnalyze (with the parameter CHKTONE_APPOINTEDTONE).

Configuring the user-defined tone detector in the 2nd call progress tone detector:

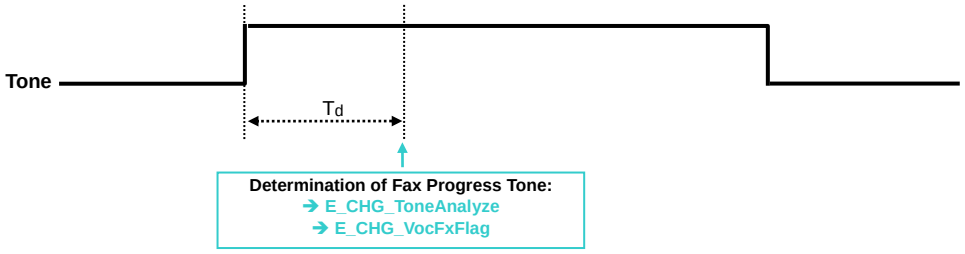
- The configuration item 2ndAppointedToneAnalyzerSwitch is used to start or stop the detection.
- The configuration item 2ndIsAppointedToneDetermineTime is used to set the minimum duration of continuous signals. This detector judges continuous signals in the same way as the dial tone detector. When the tone continues for more than the time specified by this configuration item, the driver will throw out the event E_CHG_ToneAnalyze (with the parameter CHKTONE_APPOINTEDTONE).
- The configuration items 2ndAppointedTonePara and 2ndIsAppointedToneDetermineCount are used to set the durations of periodic signals respectively at on and off states as well as the minimum count of tone cycles. This detector judges periodic signals in the same way as the ringback tone detector. When the count obtained is greater than the minimum count designated by 2ndIsAppointedToneDetermineCount, the driver will send off the event E_CHG_ToneAnalyze (with the

parameter CHKTONE_APPOINTEDTONE).

1.10.5.4 Fax Progress Tone Detector (CTI Series)

There are two fax progress tone detectors respectively called the 1st fax progress tone detector and the 2nd fax progress tone detector, which are used to detect at one time two fax progress tones at different frequency. The fax progress tone is a single continuous tone.

The figure below shows the typical fax-progress-tone waveform.



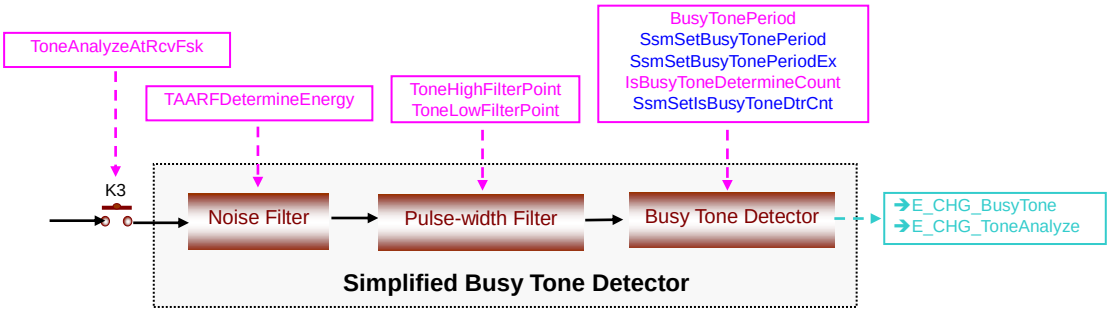
The fax progress tone detector starts counting when it detects the signal at specified frequency. When the count reaches the threshold value T_d, the driver will throw out the events E_CHG_VocFxFlag and E_CHG_ToneAnalyze (with the parameter CHKTONE_VOICEF1 or CHKTONE_VOICEF2) in order. Also the application can obtain the result via the function SsmGetVocFxFlag.

The parameters of the fax progress tone detector include the center frequency, the bandwidth, the threshold value for the percentage of in-band energy to overall energy and the minimum guard time T_d. Those of the 1st fax progress tone detector can be set via the configuration item VoiceFreqF1Para or the function SsmSetVoiceFxPara while those of the 2nd via the configuration item VoiceFreqF2Para or the function SsmSetVoiceFxPara.

Note: The detection of fax answering tones during AutoDial on the analog trunk channel or analog trunk recording channel usually indicates that the called party (fax machine) has picked up. Therefore, the result obtained by the fax progress tone detector will automatically trigger the corresponding event to change the state machine.

1.10.5.5 Simplified Busy Tone Detector

Its operation principle, relative configuration items or functions, output events are all shown in the following figure.



As illustrated above, the operation of the simplified busy tone detector is controlled by the switch K3 which can be set via the configuration item ToneAnalyzeAtRcvFsk.

The noise filter herein works in the same way as the 'Noise Filter' described in the previous section except that the threshold value for judging the noise is set via the configuration item TAARFDetermineEnergy. Usually there is no

need to reconfigure the noise filter, for its default settings are adequate for most situations.

The pulse-width filter herein runs in an identical manner as the 'Pulse-width Filter' described in the previous section, involving the same configuration items.

What makes the simplified busy tone detector different from the call progress detector is that it doesn't examine the frequency of input signals, i. e. it assumes all input signals are tones, so that it doesn't contain the frequency detector.

The algorithm used to examine the busy tone waveform is the same as that supported by the call progress tone detector to examine the regular busy tone waveform. Like the busy tone detector in the 1st call progress tone detector, this detector is able to examine 4 groups of busy tones which differ in cycle at one time and uses the same waveform parameters, i.e.

- ✧ The configuration item BusyTonePeriod (or the functions SsmSetBusyTonePeriod, SsmSetBusyTonePeriodEx) is used to set the busy tone cycle;
- ✧ The configuration item IsBusyToneDetermineCount (or the function SsmSetIsBusyToneDtrCnt) is used to determine upon detection of which busy tone does the driver send off the E_CHG_ToneAnalyze event.

The simplified busy tone detector also has the ability to prevent misdetection of busy tones, which is enabled similarly by setting the guard time T_{max} via the configuration item MaxBsTnOffTime.

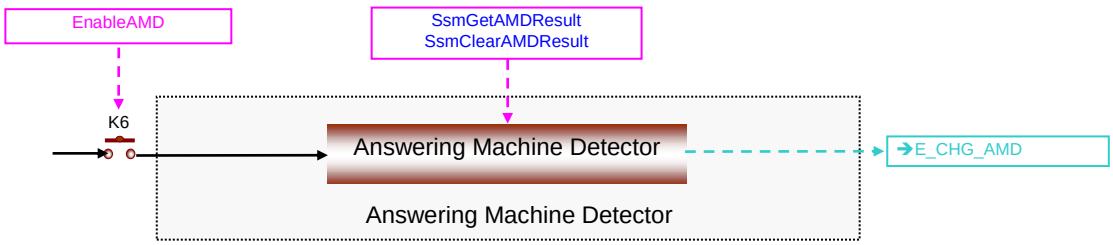
This detector outputs in an identical manner as a standard busy tone detector. Every time when a busy tone cycle is detected, the detector will add 1 in the busy tone counter (i.e. 'Count' mentioned in the above figure) and throw out the E_CHG_BusyTone event; when the count reaches a specified value, it will send off the event E_CHG_ToneAnalyze (with the parameter CHKTONE_BUSYTONE).

Note:

- **For the analog trunk channel, the E_CHG_ToneAnalyze event will be automatically triggered at the same time to change the state machine.**
- **Only the SHT Series analog trunk channel is designed with the simplified busy tone detector.**

1.10.5.6 AMD (Answering Machine Detector)

the following figure show the principle diagram, relative configuration items, functions and output event of AMD.



Answering machine detector is controlled by K6 which also can set by configuration item EnableAMD.

The AMD distinguish the answering machine from the human being through analyzing the on-line voice. This detector can widely uses in dialing system to improve the woke efficiency of seat telephone operator. Now, the

successful rate of distinguishing the answering machine from the human being is higher then 90%.

The result of AMD can be obtain through output the parameter of event E_CHG_AMD or via calling the function SsmGetAMDResult to get it return value. calling the function SsmClearAMDResult can clear the detecting result of last time.

Note:

- **AMD works normal only when the Tone Detector is enabled.**
- **If the channel is an analog trunk channel, AMD will auto-start after dialing. If the channel is a digital trunk channel, AMD will auto-start after the channel turns into the talking state.**

1.10.5.7 Enhanced Tone Detector

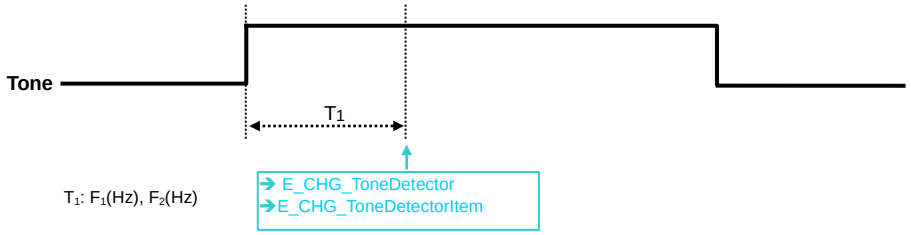
The operation principle of the enhanced tone detector is to divide tones into three categories for detection, i.e. continuous tone, periodic tone and special information tone (SIT - three continuous single tones). Set with different parameters according to various characteristics of these three kinds of tones, this detector can properly detect all of them.

All parameters used are described as follows.

- The 1st mid-frequency F_1
  - For continuous tones, when a single tone is detected, F_1 represents its frequency; when a dual tone is detected, F_1 indicates the 1st mid-frequency of the tone.
  - For periodic tones, when a single tone is detected, F_1 represents its frequency; when a dual tone is detected, F_1 indicates the 1st mid-frequency of the tone.
  - For a special information tone (SIT), F_1 represents the frequency of the 1st band.
- The 2nd mid-frequency F_2
  - For continuous tones, when a single tone is detected, F_2 is equal to 0; when a dual tone is detected, F_2 indicates the 2nd mid-frequency of the tone.
  - For periodic tones, when a single tone is detected, F_2 is equal to 0; when a dual tone is detected, F_2 indicates the 2nd mid-frequency of the tone.
  - For a special information tone (SIT), F_2 represents the frequency of the 2nd band.
- The frequency of the 3rd band F_3
  - Only valid for special information tones (SIT), F_3 represents the frequency of the 3rd band.
- Durations T_1, T_2, T_3
  - For continuous tones, T_1 indicates the duration at on state; T_2, T_3 are invalid.
  - For periodic tones, T_1 indicates the duration at on state in a period; T_2 indicates the duration at off state in a period; T_3 is invalid.
  - For a special information tone (SIT), T_1, T_2, T_3 respectively indicate the durations of three continuous single tones.
- The filtering point number at on and off states C_{on}, C_{off} (time length at each point is 16ms)
  - Set the time to judge the wave form (upwards or downwards). When the filtering point number gets larger than C_{on}, we say the wave goes into the on state; when the filtering point number is larger than C_{off}, the wave is considered getting the off state.
- The frequency error threshold F_{err}(Hz)
  - Sets the accepted error of the parameters F_1, F_2, F_3.

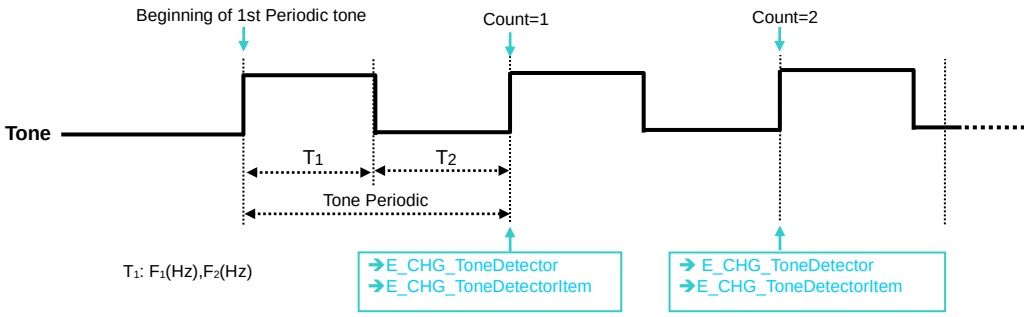
These three kinds of tones are illustrated as follows:

- Continuous Tone



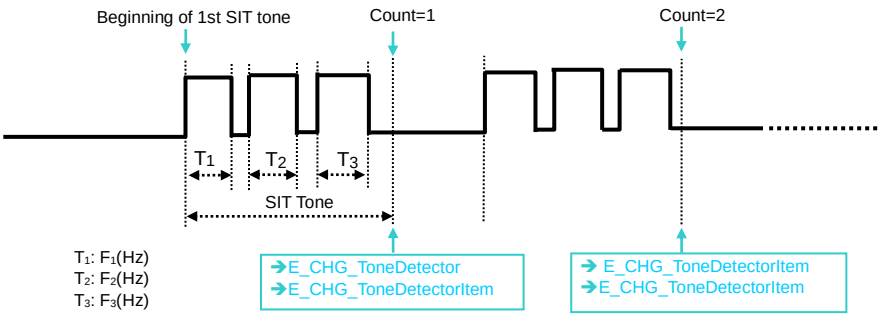
The dial tone and the fax tone F_1/F_2 belong to this kind and should be configured as continuous tones. Once the duration at on state T_1 is detected, the events E_CHG_ToneDetector and E_CHG_ToneDetectorItem will be thrown out.

- Periodic Tone



The ringback tone, the howler tone and the busy tone all belong to this kind and should be configured as periodic tones. Once a complete cycle is detected, the events E_CHG_ToneDetector and E_CHG_ToneDetectorItem will be thrown out.

- SIT (Special Information Tone)



An SIT tone consists of three continuous single tones, each of which has its own frequency. Such kind of tones should be configured as SIT tones. Once three continuous single tones are detected, the events E_CHG_ToneDetector and E_CHG_ToneDetectorItem will be thrown out.

Relative configuration items:

- ToneDetectorItem[n] describes all parameters, such as frequency, signal duration at off state, signal duration at on state, etc., for the tones to be detected.
- MaxToneDetectorItem indicates how many kinds of tones will be detected by the enhanced tone detector.
- VoiceOffDetermineTime is used to set the time of silence detection. Its value depends on the period of the ringback tone in the common mode, usually being 1.2 times the period of the ringback tone. However, it needs to be set manually in the enhanced mode.

The parameter dwParam of the event E_CHG_ToneDetector tells the kind of tones and the N which this kind of tones corresponds to in ToneDetectorItem[N]. The event E_CHG_ToneDetectorItem records the period of tones, and the 2 lower bits of dwParam indicate how many periods the tone lasts.

To be compatible with previously developed application programs, the enhanced tone detector provides the following interfaces other than the event interfaces E_CHG_ToneDetector and E_CHG_ToneDetectorItem.

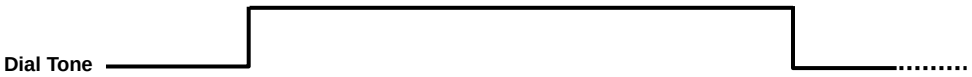
- API Interface: SsmStartToneAnalyze, SsmCloseToneAnalyze, SsmGetToneAnalyzeResult, SsmClearToneAnalyzeResult
- Event Interface: E_CHG_ToneAnalyze

1.10.6 Tone Generator (CTI Series)

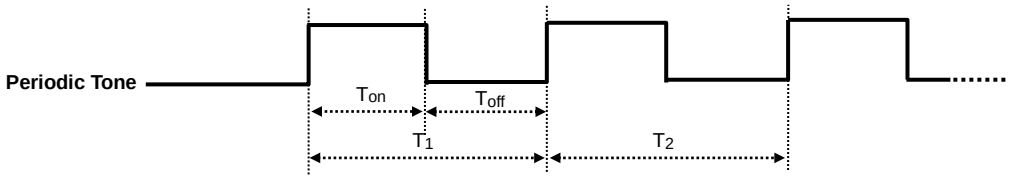
The 'tone' mentioned herein refers to the call progress tone at a constant frequency and a certain tempo. It uses various tempos to imply different on-line states, for example, the remote end hangs up, the line is busy, etc. Each channel on the Synway boards has an independent tone generator.

The parameters of the tone generator include the center frequency (either single or dual), the amplitude and the tempo. Tones can be separated by tempo into two categories: the continuous tone and the periodic tone.

The continuous tone is the tone that keeps going at on state throughout the runtime of the tone generator. The dial tone is of this kind with the waveform as shown below.



The periodic tone is the tone which undergoes several signal cycles, each of which involves a duration at on state and a duration at off state. The busy tone and the ringback tone are of this kind with the typical waveform as shown below.



In the figure above, T_1 and T_2 respectively represent the 1st and 2nd cycles while T_{on} and T_{off} indicate the signal durations respectively at on and off states. The rate of T_{on} to T_{off} is called duty ratio: the busy tone's duty ratio is 1:1 and the ringback tone's is 1:4.

1.10.6.1 Configuring Tone Generator

Configuration items or functions related to frequency and amplitude are displayed in the following table.

Function Name	Description	Related Configuration Item
SsmQueryOpSendTone	Queries if the channel supports the tone generator.	
SsmSetTxTonePara	Sets frequency and amplitude of each channel's tone generator.	DefaultSendToneFrequence DefaultSendToneVolume
SsmGetTxTonePara	Obtains frequency and amplitude of each channel's tone generator.	

1.10.6.2 Tone Generation

Functions related to the tone generator are shown in the following table.

Function Name	Description
SsmSendTone	Starts the tone generator.
SsmSendToneEx	Starts the tone generator (allowed to set T_{on} and T_{off} in parameters).
SsmStopSendTone	Stops the tone generator.
SsmChkSendTone	Obtains the type of tones being sent by the tone generator.

Note: The operations listed below are forbidden once the tone generator is started (except for 120A series boards).

- Voice playing;
- Run of the DTMF generator;
- FSK data transmission.

1.10.7 Echo Canceller (CTI Series)

The echo cancellation feature helps effectively improve the quality of voice communications and greatly enhance the accuracy of the DTMF detector and the Barge-in detector.

The operating mode of the echo canceller can be set via the function SsmSetEchoCanceller or the configuration item EnableEchoCancellor, with a default setting of 'Enabled'. **Note: Because the echo canceller once enabled will change incoming signals, if FSK data reception is required to be performed on a channel, you should disable the echo canceller and not enable it until the reception is completed to ensure the reliability in data communication.**

You can modify the configuration item LoadFskBin to optimize the performance and the effect of the echo canceller for the SHD Series 30/60/120-channel boards. However, it may cause those boards to lose the call recording and FSK data receive/transmit features due to the DSP resources being exhausted.

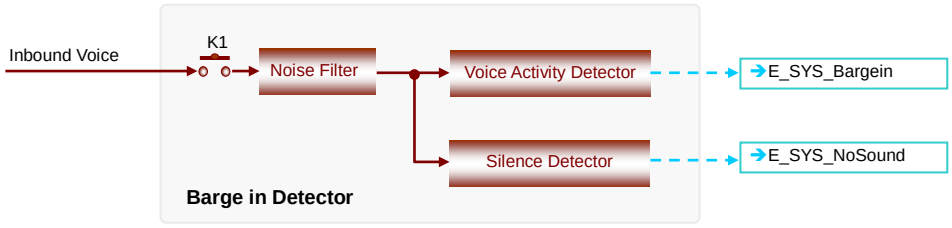
The echo canceller afforded by the Synway board in default is adequate for most application environments. If you are required to change the operating mode or reset the parameters of the echo canceller in some particular instances, please do it under the instruction of our technicians. Its parameters can be set via the function SsmSetEchoCancelDelaySize or the configuration item EchoCancelDelaySize.

1.10.8 Barge-in Detector

The Barge-in detector is used to detect voice activities on the line, often applicable to those occasions that involve the voice-controlled recording, the automatic speech recognition (ASR), etc. As long as a channel possesses the capability of echo cancellation, it will prevent the echoes of outgoing signals from the local end (e.g. voices

generated during voice playing) going to the Barge-in detector, which thereby brings about quite precise results for the Barge-in detector. To know if a channel has the capability of echo cancellation, refer to relative hardware manuals.

The Barge-in detector is composed of the noise filter, the voice activity detector and the silence detector as shown in the figure below.



The switch K1 which is under an automatic control of the driver determines the operating mode of the Barge-in detector and the driver control mechanism is demonstrated by the following table:

Channel Type	Driver Behavior	Trigger Condition
The conferencing channel to be scheduled	Switch on K1	The channel enters the conference room.
	Switch off K1	The channel leaves the conference room.
The TUP / ISUP channel The SS1 channel	Switch on K1	The channel state transfers to S_CALL_TALKING.
	Switch off K1	The channel state transfers out of S_CALL_TALKING.
The ISDN channel	Switch on K1	The channel state transfers to S_CALL_WAIT_REMOTE_PICKUP (outgoing call) or S_CALL_RINGING (incoming call).
	Switch off K1	The channel state transfers out of S_CALL_TALKING.
The analog trunk channel	Switch on K1	The channel state transfers out of S_CALL_STANDBY.
	Switch off K1	The channel state transfers to S_CALL_STANDBY.
The station channel	Switch on K1	The channel state transfers to S_CALL_PICKUPED.
	Switch off K1	The channel state transfers to S_CALL_STANDBY.
The analog trunk recording channel	Switch on K1	Detects the pickup behavior.
	Switch off K1	Detects the hangup behavior.

If you are using the SHD Series boards, you can have K1 switched on all the time via the configuration item AlwaysDetectBargeIn no matter which state the channel transfers to.

If you are using the SHN, DTP or SHV Series boards, Switch K1 can be controlled only by the configuration item AlwaysDetectBargeIn but not by the driver.

The noise filter is used to distinguish on-line noises from real voice signals. You can use the configuration item VoiceEnergyMinValue to set the noise threshold value, judging the incoming or outgoing signals as on-line noises if the signal intensity stays lower than the set value; the function SsmSetBargeInSens or the configuration item BargeInSensitive to set the sensitivity of the noise filter; and the configuration item DefaultDtmfIsSound to determine whether to consider the DTMF digits interlarded in incoming signals as voice signals.

The voice activity detector is used to detect the on-line voice activities. The driver will throw out the E_SYS_BargeIn event once voice activities are detected in incoming signals which continue for more than the set value of the function SsmGetIsBargeInDtrmTime or the configuration item BargeInDtrmTime. Also the application can obtain the result via the SsmDetectBargeIn function.

The silence detector is used to judge if the line is silent or not. The E_SYS_NoSound event is triggered when no voice signal occurs on the line (i.e. the line keeps silent) within a period of time longer than the set value of the function SsmSetNoSoundDtrmTime or the configuration item IsNoSoundDtrmTime. The detected result can be obtained via the function SsmDetectNoSound and the silent duration can be acquired via the function

SsmGetNoSoundTime.

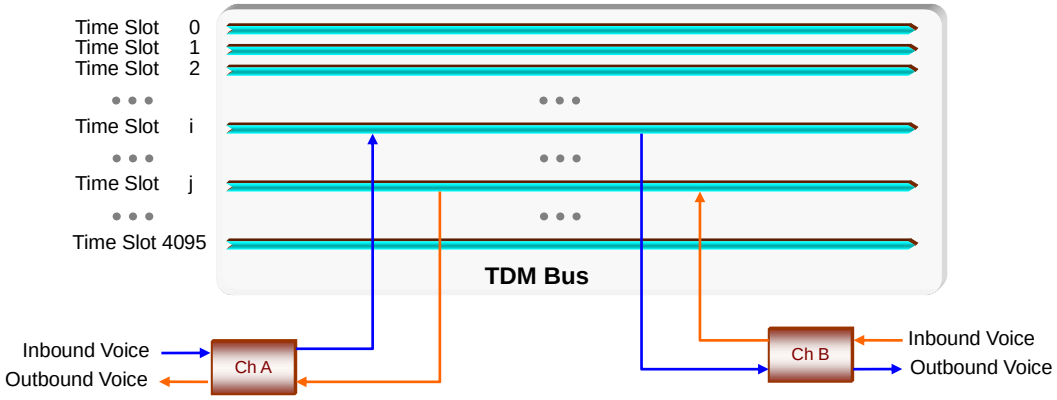
1.10.9 TDM Capability

The Synway boards support two TDM modes:

- Interchannel Exchange: The capability to exchange voice data between on-board channels. Each Synway board has such capability.
- Interboard Exchange: The capability to exchange voice data between channels respectively on different boards. All Synway boards except the SHT and ATP Series 4-channel (the '-USB' model) and 8-channel boards are equipped with such capability.

The cPCI boards use H.110 bus while the rest use H.100 bus.

The following figure demonstrates the operation principle of voice exchange by TDM.



In the figure above, the two-way connection between Ch A and Ch B is performed as follows: Put incoming voice data from Ch A to Time Slot i on TDM bus and then transmit them from Time Slot i to Ch B. In this way, Ch B is enabled to hear the incoming call from Ch A; likewise, Ch A can hear the incoming call from Ch B as long as incoming voice data from Ch B are put to Time Slot j on TDM bus and then transmitted from there to Ch A.

During system initialization, the driver will put incoming signals from each channel to a time slot on TDM bus provided the configuration item InVoiceToBus is set to 1 (default).

Related functions often used are listed in the following table.

Exchange Operation	Function Name	Description
Establishing Two-way Connection	SsmTalkWith	Establishes a two-way connection between 2 channels.
	SsmTalkWithEx	Establishes a two-way connection between 2 channels, allowed to set the volume on the 2 channels.
Tearing off Two-way Connection	SsmStopTalkWith	Tears off the two-way connection between Ch 1 and Ch 2.
Establishing One-way Connection	SsmListenTo	Establishes a one-way connection from the talker channel to the listener channel.
	SsmListenToEx	Establishes a one-way connection from the talker channel to the listener channel, allowing setting the volume on the talker channel.

	SsmLinkFrom	Establishes a one-way connection from the talker channel to the listener channel, permitting the listener channel to listen to multiple channels simultaneously.
	SsmLinkFromEx	Establishes a one-way connection from the talker channel to the listener channel, permitting the listener channel to listen to multiple channels simultaneously and allowing setting the volume on the talker channel as well.
	SsmLinkFromAllCh	Establishes a number of one-way connections from one talker channel to multiple listener channels, permitting the listener channel to listen to multiple channels simultaneously and allowing setting the volume on the talker channel as well.
Tearing off One-way Connection	SsmStopListenTo	Tears off the one-way connection established by using the function SsmListenTo or SsmListenToEx.
	SsmStopLinkFrom	Tears off the one-way connection established by using the function SsmLinkFrom or SsmLinkFromEx.
	SsmUnLinkFromAllCh	Tears off the one-way connection established by using the function SsmLinkFromAllCh.
Tearing off all Connections	SsmClearChBusLink	Tears off all bus connections on the channel and recovers the channel to the state upon the driver's initialization.

Below lists the advanced TDM functions which are used to establish or tear off the one-way connection between channels and time slots:

Exchange Operation	Function Name	Description
Channel to Time Slot	SsmLinkToBus	Changes the time slot which is set to receive the signals from the onto-bus mixer.
	SsmUnLinkToBus	Reuses the default time slot to receive the signals from the onto-bus mixer.
Time Slot to Channel	SsmLinkFromBus	Establishes the one-way connection from a time slot to the listener channel.
	SsmLinkFromBusEx	Establishes the one-way connection from a time slot to the listener channel, allowing setting the volume.
	SsmUnLinkFromBus	Tears off the one-way connection from a time slot to the listener channel

1.10.10 Distributed Teleconferencing System (CTI Series)

1.10.10.1 Basic Concepts

➢ **Distributed Conferencing Technology**

The CTI Series boards equipped with an independent mixer on each channel support teleconferencing without the need for extra conferencing boards. This technology, which we call Distributed Conferencing Technology, is Synway's patent technique. Note that the teleconferencing feature will be unsupported once the driver enables the soft-exchange over bus, and the multi-boards teleconferencing feature is not supported with those boards excluding the CT-Bus interface.

➢ **Conference Room**

A conference room consists of all channels that participate in a same conference. Each conference room has a unique number which is either specified by the application or allocated by the driver. The total number of channels

in a conference room is allowed to range from 3 to the total number of all channels in the application system, which can be set via the configuration item ConfDefaultMaxGroupMember or designated by the function SsmCreateConfGroup while creating the conference room.

The total number of conference rooms can be determined via the configuration item ConfMaxGroup.

- **Conference Room Number**

Each conference room has a unique number which is called Conference Room Number and ranges from 0 to the set value of the configuration item ConfMaxGroup minus 1.

- **Conference Channel**

The channel which joins a teleconference is called Conference Channel.

- **Conference Channel Number**

It is the logical number for each channel in the conference room, counted from 0. Note that the conference channel number is not always the same as the channel number.

- **Conference Scheduling Program**

Since the available conference mixer sources are limited, when the number of channels speaking at one time in a conference room containing a relatively large number of members exceeds the maximum accepted by the conference mixer, there is bound to be some part of the on-channel voices incapable to enter the conference mixer. The SynCTI driver contains a conference scheduling program which provides an automatic scheduling over the conference and allows only those on-channel voice signals which meet the scheduling conditions to enter the conference mixer.

The conference scheduling is based on the priority level of the conference channel. All conference mixers on the board are allocated first to the channels of high priority level, and then to those of low priority.

Provided the number of schedulable mixer sources is N, the number of organizer channels is m, the number of chairman channels is n and the number of dynamic speaker channels being speaking is k, the conference scheduling follows the rules below.

(1) First of all, allocate the conference mixer sources to m organizer channels. Then go to the next step if still some mixer sources left.
(2) If n≤N-m, all the n chairman channels can obtain mixer sources; if n>N-m, the scheduling program will arrange them by voice signal intensity (volume) in a high-to-low order and ensure the first N-m ones to acquire mixer sources. Start the allocation in the next priority level if still some mixer sources left.
(3) If k≤N-m-n, all the k dynamic speaker channels being speaking can obtain mixer sources; if k>N-m-n, the scheduling program will arrange them by voice signal intensity (volume) in a high-to-low order and ensure the first N-m-n ones to acquire mixer sources. A dynamic speaker channel, after seizing a mixer, if there is no other channel with higher priority preempting mixer sources against it, will keep occupying the mixer.

- **Conference Channel's Speaking Mode**

Each channel is given certain speaking rights when it enters a conference room. The SynCTI driver supports 6 speaking modes as described in the following table.

Speaking Mode	Description
Organizer	Can speak as well as listen to other channels. This mode is of the highest priority level. The channel in this mode is called Organizer Channel for short. It is suggested that the application program set only one organizer channel for each conference room.
Chairman	Can speak as well as listen to other channels. This mode is of the priority lower than the Organizer mode but higher than the Dynamic Speaker mode. The channel in this mode is called Chairman Channel for short.
Dynamic Speaker	Can speak as well as listen to other channels. This mode is of the priority lower than the Chairman mode. The channel in this mode is called Dynamic Speaker Channel for short.
Listener	Can listen to other channels but not speak. The channel in this mode is called Listener Channel for short, which is not involved in conference scheduling.
Background Music	Can speak, but whether it can listen to other channels is determined by the configuration item PlayVoiceIsListen. It is usually used to play background music to a conference room. **Note: Whether a channel in this mode takes part in conference scheduling and is of which priority are both determined by the configuration item BackgroundVoicePriority.** The channel in this mode is called Background Music Channel for short.
Dynamic Speaker ONLY	Can speak but not listen to other channels. The channel which serves as a dynamic speaker but cannot listen to other channels is called Dynamic Speaker ONLY Channel for short.

➢ **DTMF Clamping**

When a conference participant presses keys on the phone, the on-line DTMF tones will go into the conference mixer and be heard by other channels. The SynCTI driver supports the DTMF clamping feature which can effectively eliminate the DTMF signals generated by keypress.

The operation principle of this feature is: The incoming signals will be immediately interrupted once the on-board DTMF detector detects DTMF signals on the conference channel and resumed when all DTMF signals disappear.

The configuration item ClearInVoiceOnRxDtmf or the function SsmSetFlag (with the parameter F_ClearInVoiceOnRcvDtmf) is used to determine whether to enable the DTMF clamping feature or not.

➢ **Coming Voice Blocking**

In a normal situation, once a conference channel enters a conference room, the incoming voice signals on it are allowed to go into the conference mixer and be heard by other channels only if it meets the scheduling conditions. Invoke the function SsmSetFlag (with the parameter F_InVoiceToBus) or use the configuration item InVoiceToBus if you want to temporarily forbid this channel to speak.

1.10.10.2 Managing Conference Room

Below is a list of functions and configuration items related to the conference room.

Function Name	Category	Description
SsmCreateConfGroup	Function	Creates a conference room.
SsmFreeConfGroup	Function	Cancels a conference room.

ConfDefaultMaxGroupMember ConfDefaultMaxGroupSpeaker ConfDefaultMaxGroupSpeaking	Configuration Item	Set parameters in relation to the conference room
SsmGetConfCfgInfo SsmGetTotalConfGroup SsmGetConfGrpInfo SsmGetConfGrpCfgInfo SsmGetConfGrpId SsmValidateGrpId	Function	Obtain relative information about the conference room.
SsmGetConfGrpMmbrId SsmGetConfGrpMmbrInfo SsmGetConfChInfo	Function	Obtain relative information about members in the conference room.

1.10.10.3 Managing Conference Channel

Below is a list of functions and configuration items related to the conference channel.

Function Name	Category	Description
SsmJoinConfGroup	Function	Puts a channel into the conference room.
SsmExitConfGroup	Function	Puts a channel out of the conference room.
SsmSetListenVlmInConf	Function	Sets the voice volume in the conference room which is heard by a channel.

1.10.10.4 Playing Background Music to Conference Room

If you want to play background music and conference notice, etc. to the conference room, do it as follows:

- Choose a conference channel to play the voice. There are three ways available:
  - Set an independent background music channel for the conference room. This method is quite flexible, allows the application to determine the priority of each channel according to the actual needs. However, the shortcoming is it requires an extra channel.
  - Select a channel from those in the conference to be the organizer channel. It is the most ideal solution to have an organizer channel in the conference, but you should manage by programming to make this channel the last one to leave the conference.
  - Select a chairman channel or a dynamic speaker channel if no organizer channel available in the conference room. In this case, we suggest the chairman channel be your first choice. The shortcoming of this method is that the application needs to turn over the play task to another conference channel if the playing channel leaves the conference midway due to some particular reasons (e.g. an accidental disconnection).
- Invoke the function SsmSetPlayDest to switch on k1-1 illustrated in the operation principle of the SHT/SHD/SHN Series boards. For more information, refer to the section Operation Principle of SHT Series or Operation Principle of SHD Series or Operation Principle of SHN Series in this chapter.
- Invoke the playing functions. See the section Voice Playing in this chapter for more information.

1.10.10.5 Recording Voice in Conference Room

If you need to record the voice played in the conference room, do it as follows:

- Choose a conference channel to record the voice. Although any conference channel can be used for recording, you'd better select the one which won't leave the conference midway to avoid the handover of

the record task between channels. Below are the suggested ways to make a choice among channels.

- ✧ Make a prior choice of the background music channel.
- ✧ Select an organizer channel.
- ✧ Select a chairman channel.
- ✧ Select a dynamic speaker channel.
- ✧ Select a listener channel or a dynamic speaker channel.

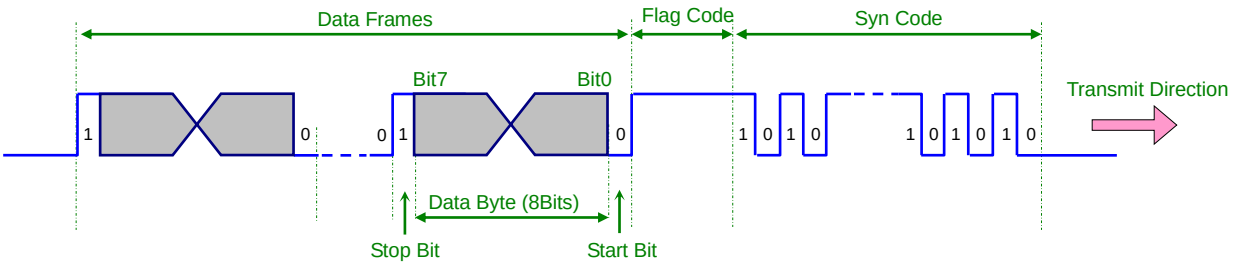
➢ Invoke the function SsmSetRecBack to switch on k6-1 and k6-2 illustrated in Operation Principle of SHD Series or Operation Principle of SHN Series if the channel is on the SHD/SHN board and required to record the voice and play the background music at the same time.

➢ Invoke the recording functions. See the section Voice Recording in this chapter for more information.

1.11 FSK Transceiver (CTI Series)

BFSK (Binary Frequency Shift Keying) is a digital communication technology using different carrier frequencies to represent the digits 0 and 1 in the binary system, often adopted to transmit the calling party number, short messages and so on through the phone line.

The BFSK data is composed of frames as illustrated below.



A complete BFSK data stream consists of three parts: the syn code, the flag code and the data. The syn code is of the 0/1 up-and-down waveform and the flag code is 1 at on state. In each data byte which is guided by the start bit 0 and followed by the stop bit 1, the digit at Bit 0 is the first to be transmitted while that at Bit 7 is the last.

Note: In some protocols the flag code is called ‘synchronization end character’, and it is together with the syn code called ‘synchronization indicator string’ which verifies the synchronization has been established.

The Synway CTI Series boards supply each channel on them with an FSK transceiver which works at the baud rate of 1200bps in the semiduplex mode. The frequency of the digit 1 herein is 1200Hz while 0 is 2200Hz.

1.11.1 FSK Transmitter

Related functions, configuration items and events are listed in the following table.

Category	Name	Description
Configuration Item	FreqBit0 FreqBit1 Baudrate MdlAmp	Set the parameters of the FSK transmitter.
Function	SsmSetFskPara	Sets the parameters of the FSK transmitter.

	SsmTransFskData	Converts data into the BFSK data stream.
	SsmStartSendFSK	Starts the FSK transmitter.
	SsmCheckSendFsk	Queries the working progress of the FSK transmitter.
	SsmStopSendFsk	Stops the FSK transmitter.
Event	E_PROC_SendFSK	The event which is sent to the application when the driver finishes transmitting all data.

1.11.2 FSK Receiver

It is ascertained that the FSK transceiver on the Synway board works at the baud rate of 1200bps, the frequency of the binary digit 1 is 1200Hz while 0 is 2200Hz. All these three values are not allowed to be modified via functions or configuration items.

FSK functions, configuration items and events are listed in the following table

Name	Description
Configuration Item: ToneAnalyzeAtRcvFsk	Determines whether the FSK receiver will analyze the tone during its runtime.
Configuration Item: FskMarkSignal	Sets the way to receive data for the FSK receiver.
Configuration Item: FskFrameMode	Sets the frame format that accepted by the FSK receiver.
Configuration Item: FskEchoCancelDelay Function: SsmSetFlag (with the parameter F_EchoCancelInFsk)	Sets whether to stop the echo canceller when the FSK receiver is working.
Function: SsmStartRcvFSK Function: SsmStartRcvFSK_II Function: SsmStopRcvFSK	Start/stop the FSK receiver.
Function: SsmGetRcvFSK Function: SsmCheckRcvFSK	Obtains the FSK data; Queries the working progress of the FSK receiver.
Function: SsmClearRcvFSKBuf	Clears the driver buffer of the FSK receiver.
Event: E_PROC_RcvFSK	The event which is sent to the application when the driver detects the end of the FSK task following starting the FSK receiver.

1.12 Fax (CTI Series)

1.12.1 Number of Fax Channels

For SHT and SHD Series boards, the following models support fax channels:

- SHT-8B/PCI/FAX
- SHT-8C/PCI/FAX
- SHT-16B-CT/PCI/FAX
- SHT-16B-CT/cPCI/FAX
- SHT-16C-CT/PCI/FAX
- SHT-16D-CT/PCIe
- SHD-30B-CT/PCI/SS7/FAX
- SHD-60B-CT/PCI/SS7/FAX
- SHD-30B-CT/cPCI/SS7/FAX
- SHD-60B-CT/cPCI/SS7/FAX
- SHD-30C-CT/PCI/FAX
- SHD-60C-CT/PCI/FAX
- SHD-30E-CT/PCI/FAX(SSW)

- ✧ SHD-60E-CT/PCI/FAX(SSW)
- ✧ SHD-120E-CT/PCI/FAX(SSW)
- ✧ SHD-240E-CT/PCI/FAX(SSW)
- ✧ SHD-30E-CT/PCIe/FAX
- ✧ SHD-60E-CT/PCIe/FAX
- ✧ SHD-120E-CT/PCIe/FAX
- ✧ SHD-240E-CT/PCIe/FAX

For SHF Series boards, the following models support fax channels:

- ✧ SHF-2D/PCI
- ✧ SHF-4D/PCI
- ✧ SHF-4D/PCIe

The table below lists the available fax channels of the above boards.

Board Model	Number of Fax Channels											
	2		4		16		24		32		64	
	N_{Fax}	N_{Voc}	N_{Fax}	N_{Voc}	N_{Fax}	N_{Voc}	N_{Fax}	N_{Voc}	N_{Fax}	N_{Voc}	N_{Fax}	N_{Voc}
SHT-8B/PCI/FAX	-	-	4	8	-	-	-	-	-	-	-	-
SHT-8C/PCI/FAX	-	-	4	8	-	-	-	-	-	-	-	-
SHT-16B-CT/PCI/FAX	-	-	4	16	-	-	-	-	-	-	-	-
SHT-16B-CT/cPCI/FAX	-	-	4	16	-	-	-	-	-	-	-	-
SHT-16C-CT/PCI/FAX	-	-	4	16	-	-	-	-	-	-	-	-
SHT-16D-CT/PCIe	-	-	4	16	-	-	-	-	-	-	-	-
SHD-30B-CT/PCI/SS7/FAX	-	-	-	-	-	-	24	30	-	-	-	-
SHD-60B-CT/PCI/SS7/FAX	-	-	-	-	16	60	-	-	-	-	-	-
SHD-30B-CT/cPCI/SS7/FAX	-	-	-	-	-	-	24	30	-	-	-	-
SHD-60B-CT/cPCI/SS7/FAX	-	-	-	-	16	60	-	-	-	-	-	-
SHD-30C-CT/PCI/FAX	-	-	-	-	-	-	24	30	-	-	-	-
SHD-60C-CT/PCI/FAX	-	-	-	-	16	60	-	-	-	-	-	-
SHD-30E-CT/PCI/FAX(SSW)	-	-	-	-	-	-	-	-	32	30	-	-
SHD-60E-CT/PCI/FAX(SSW)	-	-	-	-	-	-	-	-	-		64	60
SHD-120E-CT/PCI/FAX(SSW)	-	-	-	-	-	-	-	-	-	-	64	120
SHD-240E-CT/PCI/FAX(SSW)	-	-	-	-	-	-	-	-	-	-	64	240
SHD-30E-CT/PCIe/FAX	-	-	-	-	-	-	-	-	32	30	-	-
SHD-60E-CT/PCIe/FAX	-	-	-	-	-	-	-	-	-	-	64	60
SHD-120E-CT/PCIe/FAX	-	-	-	-	-	-	-	-	-	-	64	120
SHD-240E-CT/PCIe/FAX	-	-	-	-	-	-	-	-	-	-	64	240
SHF-2D/PCI	2	2	-	-	-	-	-	-	-	-	-	-
SHF-4D/PCI	-	-	4	4	-	-	-	-	-	-	-	-
SHF-4D/PCIe	-	-	4	4	-	-	-	-	-	-	-	-

Note: In this table, N_{Fax} is 'the total number of fax channels', N_{Voc} means 'the total number of available voice channels', and - implies 'unsupported'.

1.12.2 Supported Fax Rate

Board Model	4800bps		9600bps		12000bps		14400bps		33600bps	
	Transmit	Receive	Transmit	Receive	Transmit	Receive	Transmit	Receive	Transmit	Receive
SHT-8B/PCI/FAX	√	√	√		√		√			

SHT-8C/PCI/FAX	√	√	√	√	√	√	√			
SHT-16B-CT/PCI/FAX	√	√	√		√		√			
SHT-16B-CT/cPCI/FAX	√	√	√		√		√			
SHT-16C-CT/PCI/FAX	√	√	√	√	√	√	√			
SHT-16D-CT/PCIe	√	√	√	√	√	√	√	√		
SHD-30B-CT/PCI/SS7/FAX	√	√	√		√		√			
SHD-60B-CT/PCI/SS7/FAX	√	√	√		√		√			
SHD-30B-CT/cPCI/SS7/FAX	√	√	√		√		√			
SHD-60B-CT/cPCI/SS7/FAX	√	√	√		√		√			
SHD-30C-CT/PCI/FAX	√	√	√	√	√	√	√	√		
SHD-60C-CT/PCI/ FAX	√	√	√	√	√	√	√	√		
SHD-30E-CT/PCI/FAX(SSW)	√	√	√	√	√	√	√	√		
SHD-60E-CT/PCI/FAX(SSW)	√	√	√	√	√	√	√	√		
SHD-120E-CT/PCI/FAX(SSW)	√	√	√	√	√	√	√	√		
SHD-240E-CT/PCI/FAX(SSW)	√	√	√	√	√	√	√	√		
SHD-30E-CT/PCIe/FAX	√	√	√	√	√	√	√	√		
SHD-60E-CT/PCIe/FAX	√	√	√	√	√	√	√	√		
SHD-120E-CT/PCIe/FAX	√	√	√	√	√	√	√	√		
SHD-240E-CT/PCIe/FAX	√	√	√	√	√	√	√	√		
SHF-2D/PCI	√	√	√	√	√	√	√	√	√	√
SHF-4D/PCI	√	√	√	√	√	√	√	√	√	√
SHF-4D/PCIe	√	√	√	√	√	√	√	√	√	√

1.12.3 Supported Fax File Format

The fax channels on the Synway board support the fax protocol T.30 and T.4, and allowed transmitting MH, MR, MMR data and Tiff files. The coding format can be set via the configuration item.

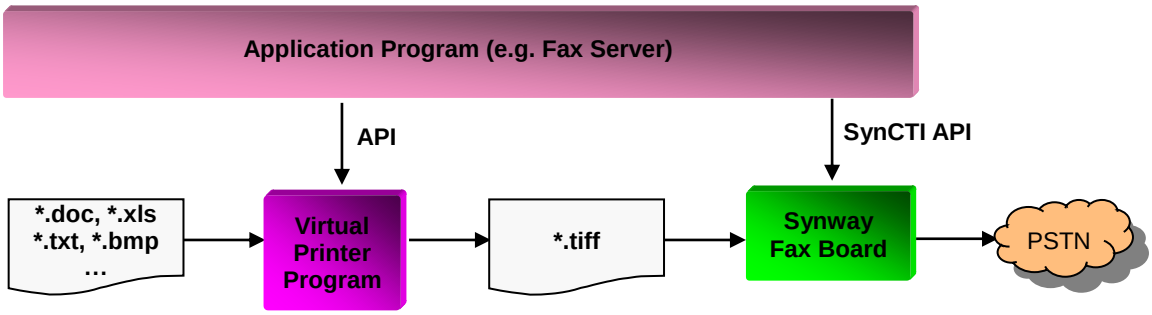
The Tiff file is the mainstream one used to store fax data, which can be widely used and is easy to read, archive and modify. And the Tiff file supported by fax channels on the Synway board has the following properties.

- ✓ Color: Black and white
- ✓ Coding Format: MH, MR, MMR
- ✓ Resolution: 204*196, 204 * 98, 200 * 200, 200 * 100
- ✓ Page Data: 1728 pixels per line
- ✓ Other properties. See the table below:

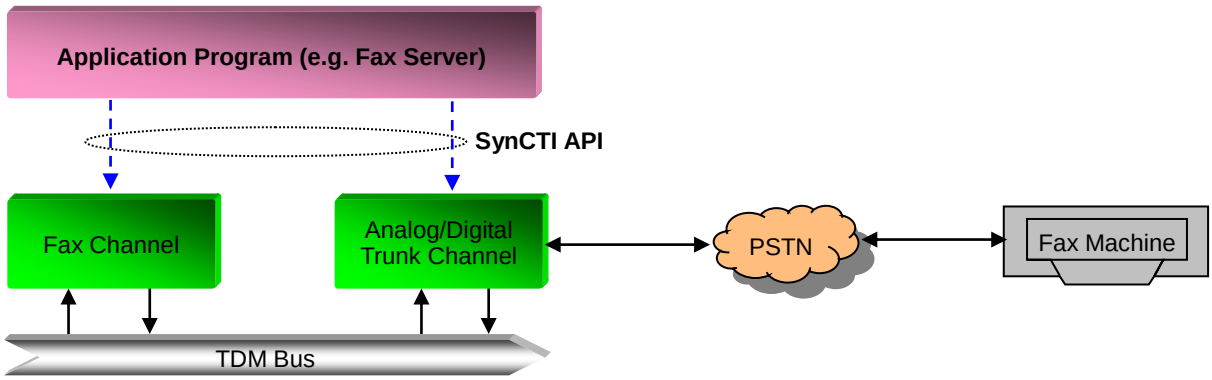
Tiff compression	T4 capability	Remarks
2	--	CCITT HM – simplified MH code
3	0	CCTIT G3 1D – MH code, without the EOL(end-of-line) flag
3	4	CCTIT G3 1D – MH code, with the EOL(end-of-line) flag
3	1	CCTIT G3 2D – MR code, without the EOL(end-of-line) flag
3	5	CCTIT G3 2D – MR code, with the EOL(end-of-line) flag
4	--	CCITT G4 – MMR code

Note: When the Tiff compression is at the value of 3, the T4 capability is only to distinguish the MH code from the MR code, but not to detect whether there is the EOL flag at the ending or not. In such situation, the self-adaption feature the driver has helps to make a further confirmation of the file's real attribute.

The operation principle for the application program to send files in any format is shown below.



As illustrated In the figure above, the virtual printer program is used to convert a file of any other format to a tiff file. The analog/digital trunk channel as well as the fax channel is indispensable for a complete fax transmission or reception as shown below.



The fax transmission (or reception) process is divided into four distinct phases.

(1) Initiate and establish a call on the analog or digital trunk channel.

(2) Establish a two-way connection between the fax channel and the analog/digital trunk channel.

(3) Transmit (or receive) fax files.

(4) Disconnect the call.

1.12.4 Setting Faxing Parameters

Related functions which involve the faxing parameters are listed in the following table.

Function Name	Description
SsmFaxSetMaxSpeed	The initial faxing rate, also the maximum rate used to send or receive fax. This rate is applicable to all channels.
SsmFaxSetID	Sets the local fax identification code, usually using the local phone number or some letters. This code will be sent to the remote fax machine during faxing.

1.12.5 Acquiring Information on Faxing

Related parameters used in the faxing process can be obtained via the following functions.

Function Name	Description & Acquired Information
SsmFaxGetSpeed	Obtains the actual fax rate, whose possible values are 24, 48, 72, 96, 120, 144, 168, 192, 216, 240, 264, 288, 312, 336, respectively representing 2400, 4800, 7200, 9600, 12000, 14400, 16800, 19200, 21600, 24000, 26400, 28800, 31200, 33600bps.
SsmFaxGetCodeMode	Obtains the real fax CODECs, whose possible values are 0, 1, 2, respectively representing MH, MR, MMR.

SsmFaxGetID	Obtains the remote fax identification code, usually after the handshake phase during faxing.
SsmFaxGetPages	Obtains the number of sent/received pages during faxing. If this function returns 0 when invoked after faxing, it indicates the current fax transmission or reception fails.
SsmFaxGetAllBytes	Obtains the total number of bytes on the fax page being sent.
SsmFaxGetSendBytes	Obtains the number of sent bytes on the current page during fax transmission. You can use this function to get such number after invoking the function SsmFaxStartSend.
SsmFaxGetRcvBytes	Obtains the number of received bytes on the current page during fax reception. You can use this function to get such number after invoking the function SsmFaxStartReceive.
SsmFaxGetChStateMsg	Obtains the information about the state of the specified channel during faxing.
SsmFaxCheckEnd	Obtains the fax channel state.

1.12.6 Fax Transmission

Functions related to fax transmission are listed in the following table.

Function Name	Description
SsmFaxStartSend	Starts transmission of a single fax file.
SsmFaxStartSendEx	Starts transmission of a single fax file, can specify the start page and the end page.
SsmFaxSendMultiFile	Starts transmission of multiple fax files.
SsmFaxSendMultiFileEx	Starts transmission of multiple fax files, can specify the start page and the end page.
SsmFaxAppendSend	Appends a single file to send. It works only if the fax channel stays in the fax-transmitting state.
SsmFaxCheckEnd	Obtains the fax channel state.
SsmFaxSetMaxSpeed	Sets the faxing rate.
SsmFaxSetID	Sets the local fax identification code which will be displayed on the LCD of the remote fax machine.
SsmFaxGetID	Obtains the remote fax identification code.
SsmFaxGetAllBytes	Obtains the total number of bytes on the fax page being sent.
SsmFaxGetSendBytes	Obtains the number of sent bytes on the current page during fax transmission.
SsmFaxGetSpeed	Obtains the current faxing rate.
SsmFaxGetPages	Obtains the number of sent/received pages during faxing.
SsmFaxStop	Stops the current faxing compulsorily.
SsmFaxGetCodeMode	Obtains the fax CODECs.

1.12.7 Fax Reception

Functions related to fax reception are listed in the following table.

Function Name	Description
SsmFaxStartReceive	Starts fax reception. The received fax data will be stored in a set fax file.
SsmFaxCheckEnd	Obtains the fax channel state.
SsmFaxSetMaxSpeed	Sets the faxing rate.
SsmFaxSetID	Sets the local fax identification code which will be displayed on the LCD of the remote fax machine.
SsmFaxGetID	Obtains the remote fax identification code.
SsmFaxGetRcvBytes	Obtains the number of received bytes on the current page during fax reception.

SsmFaxGetSpeed	Obtains the current faxing rate.
SsmFaxGetPages	Obtains the number of sent/received pages during faxing.
SsmFaxStop	Stops the current faxing compulsorily.
SsmFaxGetCodeMode	Obtains the fax CODECs.

1.13 On-board Audio Power Amplifier

Channel 0 on each ATP, DST or SHT Series board is equipped with the analog audio amplifier circuit and the speaker output jack, which allows the board to connect directly with the external speaker and enables the voice to be delivered to and played through the speaker. For detailed information about the on-board audio power amplifier, see the operation principle of the corresponding board.

1.14 Application Programming Based on Synway Board

1.14.1 Setting Event Output Mode

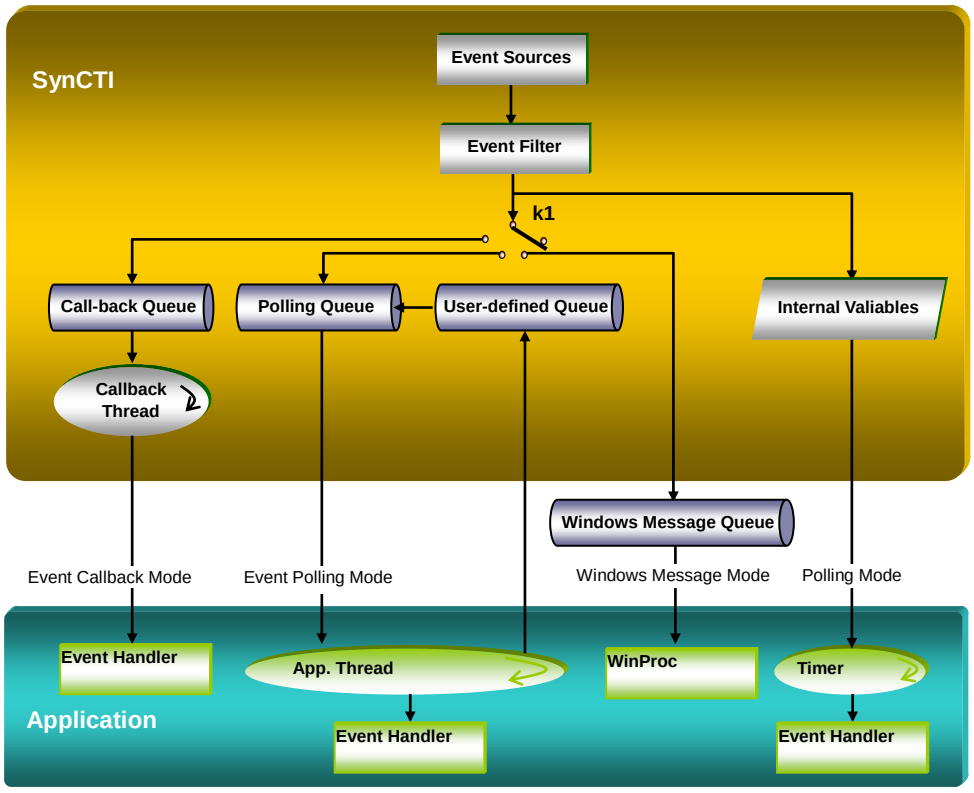
The written program based on the Synway board usually consists of three parts.

- Initialization codes, including both parts for initializing the board and the driver, used only once upon starting the application program.
- Processing codes, used to give commands to the driver and process in time the information returned via the event from the driver so as to enable particular features on demand.
- Uninstallation codes, covering the resources needed for releasing the board and the driver, used only once upon exiting the application program.

The SynCTI driver supports the following programming modes:

- ✧ Polling Mode: The application continuously calls the relative query functions supplied by the driver to obtain the information on task progress. These days this mode gradually goes out of use, for the high cost of computer resources and the low efficiency limit it to small application systems only. Therefore, this manual does not give further description on it.
- ✧ Event Polling Mode: The application program invokes the event polling functions offered by the driver. The application's caller thread is hung in case there is no event from the driver, and gets activated for event processing once some events available from the driver.
- ✧ Event Callback Mode: When the driver generates some event, it invokes the callback function which the application has been registered to process the event.
- ✧ Windows Message Mode (only applicable to Windows OS, the data structure of the output events can only be MESSAGE_INFO as well): The driver sends the events to the Windows message queue and processes them through the Windows unified message queue processing mechanism. Few applications use this mode for programming as it carries a limited number of parameters. That's why this manual does not elaborate this mode.

See how these four programming modes work through the following figure.



The switch K_1 is controlled by the function SsmSetEvent. Every time the SynCTI driver starts up, it works in the polling mode. If the application need use Event Polling Mode or Event Callback Mode, the function SsmSetEvent should be called to change the mode. Refer to the programming examples listed below for details.

1.14.1.1 Programming Example in Event Polling Mode

```
#include <windows.h>                    //include the required Windows header file
#include "shpa3api.h"                   //include the header file required by the SynCTI driver
……

void main()
{
    //initialize the SynCTI driver and the board
    if ( ! SsmStartCti (……) )
    {                                   //failed
        ……
        MessageBox( "Initialize Voice card failure" );
        return;
    }

    //obtain relative information about the board and the channel
    int nMaxCh = SsmGetMaxCh();                 //obtain the total number of channels
    for (int ch=0; ch < nMaxCh; ch ++)
    {
        int nChType = SsmGetChType (ch);        //get the channel type
        ……
    }

    //set the event output mode
    EVENT_SET_INFO EventMode;
    EventMode.DwWorkMode = EVENT_POLLING;       //use the event polling mode
    SsmSetEvent(0xffff, -1, true, &EventMode);

    //transaction processing
    MESSAGE_INFO Event;
    While(1)
    {
```

```
        memset(&MessageInfo, 0, sizeof(PMESSAGE_INFO));
        If (the application program exits) break;                    //codes for quitting the application program

        //wait for and respond to the event thrown out by the driver
        If ( SsmWaitForEvent (50, &Event) == 0 )          //wait for some event
        {
            switch(Event.EventCode)
            {
            case E_CHG_ChState：                          //the channel state changes
                ……                                        //transaction processing codes
                break;
                ……
            case E_XXXXXX：
                ……
                break;
            default:
                break;
            }
        }
    }

    SsmCloseCti();                                        //close the driver upon exiting
}
```

In need of using the extended data structure of the output event, you may write the 'transaction processing' part among the above codes following the way below.

```
    //transaction processing
    UCHAR pucBuffer[300];
    SSM_EVENT SsmEvent;
    while(1)
    {
        memset(&SsmEvent, 0, sizeof(SSM_EVENT));
        memset(pucBuffer, 0, sizeof(UCHAR)* 300);
        SsmEvent.pvBuffer = (PVOID)pucBuffer;
        SsmEvent.dwBufferLength = 300;
        if(SsmWaitForEventA(50, &SsmEvent) == 0)
        {
            switch(SsmEvent.wEventCode)
            {
            case E_CHG_ChState:
                ……                                    //do something
                break;
            case E_RCV_DSTDChannel:
                switch(SsmEvent.dwParam)
                {
                case DST_AUDIO_CHG:
                    ……                                //do something
                    break;
                case DST_MSG_CHG:
                    ……                                //do something
                    break;
                default:
                    break;
                }
            default:
                break;
            }
        }
    }
```

1.14.1.2 Programming Example in Event Callback Mode

```
#include <windows.h>          //include the required Windows header file
#include "shpa3api.h"         //include the header file required by the SynCTI driver
BOOL bExit = FALSE;           //the variable to control whether the application program exits or not
……

//the callback function supplied by the application program to process the event thrown out by the driver
```

```
int CALLBACK MyCallback (WORD wEvent, int nReference, DWORD dwParam, DWORD dwUser)
{
    switch(wEvent)
    {
    case E_CHG_ChState:
        ……                          //do something
        break;

    case E_XXXXXX：
        ……                          //do something
        bExit = TRUE;               //exit the application program
        break;

    ……

    default:
        break;


    }
    return 1;
}

//the application's main thread
void main()
{
    ……
    if ( ! SsmStartCti (……) )                    //initialize the SynCTI driver and the board
    {                                            //failed
        ……
        MessageBox( "Initialize Voice card failure" );
        return;
    }

    int nMaxCh = SsmGetMaxCh();                  //obtain the total number of channels
    for (int ch=0; ch < nMaxCh; ch ++)
    {
        int nChType = SsmGetChType (ch);         //get the channel type
        ……
    }

    //set the event output mode
    EVENT_SET_INFO EventMode;
    EventMode.DwWorkMode = EVENT_CALLBACK;       //event callback mode
    EventMode.lpHandlerParam = MyCallback;       //register the callback function
    SsmSetEvent(0xffff, -1, true, &EventMode);

    While(!bExit) ;                              //wait for the program's end

    SsmCloseCti();                               //close the driver upon exiting
}
```

In need of using the extended data structure of the output event, you may write the callback functions offered by the application program following the way below.

```
int CALLBACK MyCallback (PSSM_EVENT pEvent)
{
    switch(pEvent->wEventCode)
    {
    case E_CHG_ChState:
        ……                          //do something
        break;

    case E_XXXXXX：
        ……                          //do something
        bExit = TRUE;               //exit the application program
        break;
```

```
        case E_RCV_DSTDChannel:
            switch(pEvent->dwParam)
            {
            case DST_AUDIO_CHG:
                ……                    //do something
                break;
            case DST_MSG_CHG:
                ……                    //do something
                break;
            default:
                break;
            }
        default:
            break;
        }
        return 1;
}
```

Note: When the event callback mode is used for programming, it is necessary to add the keyword CALLBACK before the callback function declaration. Otherwise the program may crash because of resource release.

1.14.2 Data Structure of Output Event

There are two data structures optional for the SynCTI driver to send off an event: MESSAGE_INFO and SSM_EVENT. While MESSAGE_INFO contains common parameters and is applicable to the CTI and REC Series boards, SSM_EVENT as the extension of MESSAGE_INF can offer more information about events and is mainly used for the DST Series boards provided the SynCTI driver is of the version 4.7.3.0 or above. But it is not applicable to the Windows Message programming mode.

For example, if several boards are being used together and one of them is a DST Series board, only the SSM_EVENT data structure works and only the event polling mode and the event callback mode are choosesable.

Note: MESSAGE_INFO is applicable to functions SsmWaitForEvent and SsmGetEvent; SSM_EVENT is suitable for SsmWaitForEventA and SsmGetEventA.

1.14.2.1 MESSAGE_INFO

The MESSAGE_INFO structure declaration is:

```
typedef struct    _MESSAGE_INFO
{
    WORD     wEvent;              //event code
    int      nReference;          //reference value
    DWORD                         dwParam;          //output parameter
}MESSAGE_INFO, *PMESSAGE_INFO;
```

- **wEvent**

wEvent represents the event code. Generally, the events thrown out by the SynCTI driver can be classified into the following categories.

- ✧ E_CHG_**xxxx**: An internal state or a counter of the driver changes;
- ✧ E_PROC_**xxxx**: A task submitted by the application progresses;
- ✧ E_SYS_**xxxx**: The driver detects some event occurs;
- ✧ E_RCV_**xxxx**: The driver receives a message or an event from the remote PBX.

'xxxx' herein above implies the event designator.

The events generated in the SynCTI driver are separated into 2 categories: common event and uncommon event.

All event codes and corresponding behaviors are elaborated in the following table.

Event Code	Macro in shpa3api.h	Common Event	Description
0x0000	E_PROC_Recognize	√	Voice recognition ends.
0x0001	E_CHG_ISDNStatus	—	ISDN: The ISDN LAPD layer changes.
0x0002	E_RCV_Ss7Msu	—	SS7: A new message (MSU) is received from the SS7 server.
0x0003	E_CHG_Mtp3State	—	SS7: The SS7 MTP3 layer changes, usually to indicate if some DPC route is usable or not.
0x0004	Reserved		
0x0005	E_CHG_FaxPages	√	Fax Channel: The driver finishes receiving or transmitting a page of fax data.
0x0006	E_PROC_FaxEnd	√	Fax Channel: The driver finishes receiving or transmitting all fax data.
0x0007	E_CHG_PcmLinkStatus	√	The synchronization status of the digital trunk changes.
0x0008	E_CHG_LineVoltage	—	The voltage on the analog phone line changes.
0x0009	E_RCV_CAS	—	SS1 Channel: The ABCD signaling code from the remote PBX changes.
0x000A	E_RCV_R2	—	SS1 Channel: The R2 signal from the remote PBX is received.
0x000B	E_PROC_WaitDTMF	√	The task of WaitDTMF is completed and submitted via the function SsmSetWaitDtmf, SsmSetWaitDtmfEx or SsmSetWaitDtmfExA.
0x000C	E_CHG_RcvDTMF	√	DTMF Detector: A DTMF digit is received.
0x000D	E_PROC_SendDTMF	√	DTMF Generator: The task of transmitting DTMF started by the function SsmTxDtmf is completed.
0x000E	E_PROC_SendFlash	√	The task of sending the flash signal is completed.
0x000F	E_PROC_PlayEnd	√	Voice Playing: The task of playing voice ends, which can be started by one of the following functions. ✧ SsmPlayFile ✧ SsmPlayIndexString ✧ SsmPlayIndexList ✧ SsmPlayFileList ✧ SsmPlayMem ✧ SsmPlayMemList ✧ SsmPlayMemBlock
0x0010	E_PROC_PlayFile	—	Voice Playing: It indicates the file playing progress.
0x0011	E_PROC_PlayFileList	—	Voice Playing: The driver finishes playing a file in the file queue.
0x0012	E_PROC_PlayMem	—	Voice Playing: It indicates the voice playing progress in Single Buffer Mode.
0x0013	E_PROC_RecordEnd	√	Voice Recording: The task of recording voice terminates.
0x0014	E_PROC_RecordFile	—	Voice Recording: It indicates the file recording progress.
0x0015	E_PROC_RecordMem	—	Voice Recording: It indicates the memory recording progress.
0x0016	E_PROC_SendFSK	√	The FSK transmitter finishes sending all data.
0x0017	E_PROC_RcvFSK	√	The task of RcvFSK ends.
0x0018	E_CHG_ChState	√	State Machine: The channel state changes.
0x0019	E_PROC_AutoDial	√	State Machine: The task of AutoDial progresses.
0x001A	E_CHG_RemoteChBlock	—	TUP/ISUP Channel: The operation to block the remote channel is completed.
0x001B	E_CHG_RemotePCMBlock	—	TUP/ISUP Channel: The operation to block the remote PCM is completed.
0x001C	E_SYS_ActualPickup	—	Analog Trunk Channel: The pickup command has been executed.
0x001D	E_CHG_RingFlag	—	Analog Trunk Channel/Analog Trunk Recording Channel: The voltage level of the ringing current changes.
0x001E	E_CHG_RingCount	√	Analog Trunk Channel: The counter for signal cycles in the ringing current detector changes.
0x001F	E_CHG_CIDExBuf	√	Extended buffer area that stores the CallerID information in FSK and DTMF: The size of Extended Caller ID Buffer changes.
0x0020	E_CHG_RxPhoNumBuf	√	DTMF Detector: A new called party number is received.
0x0021	E_CHG_PolarRvrsCount	—	Analog Trunk Channel: A polarity reversal is detected on the line.
0x0022	E_SYS_RemotePickup	—	Analog Trunk Channel: The enhanced remote pickup

			detector detects that the called party picks up.
0x0023	E_CHG_FlashCount	√	Station Channel or Recording Channel: A flash operation is detected on the phone.
0x0024	E_CHG_HookState	√	Station Channel: A pickup or hangup behavior is detected on the phone.
0x0025	E_CHG_ToneAnalyze	√	Tone Detector: The analyzed result changes.
0x0026	E_OverallEnergy	—	Tone Detector: The overall energy on the line changes.
0x0027	E_CHG_OvrlEnrgLevel	√	Tone Detector: It indicates the overall energy. Whether this event is output or not is determined by the configuration item OvrlEnrgEventOut.
0x0028	E_CHG_BusyTone	√	Tone Detector: The call progress tone detector detects the change in number of busy tone cycles.
0x0029	E_CHG_BusyToneEx	—	Tone Detector: The busy tone is detected by back-to-back busy tone detection.
0x002A	E_CHG_VocFxFlag	√	Tone Detector: The voltage level of single tones changes, usually for detecting the fax tone.
0x002B	E_CHG_ToneValue	—	Tone Detector: The tone voltage level changes.
0x002C	E_CHG_RingEchoToneTime	—	Tone Detector: The count of the ringback tone counter changes.
0x002D	E_CHG_PeakFrq	—	Tone Detector: The peak frequency changes.
0x002E	E_SYS_BargeIn	√	Barge-in Detector: The detected result changes.
0x002F	E_SYS_NoSound	√	Tone Detector: The line keeps silent.
0x0030	E_SYS_TIMEOUT	√	Global Timer: The timer started by the function SsmStartTimer overflows.
0x0031	E_CHG_SpyState	—	DTP Series: The state of the monitoring circuit changes.
0x0032	Reserved		
0x003c	E_CHG_CICRxPhoNumBuf	—	SS7 Virtual Circuit: New called party numbers are received.
0x003d	E_CHG_CICState	—	SS7 Virtual Circuit: The circuit state changes.
0x003e	E_PROC_CICAutoDial	—	SS7 Virtual Circuit: The task of ShgAutoDial progresses.
0x003f	E_RCV_Ss7IsupUtuinf	—	SS7: The USR message is received.
0x0040	E_CHG_Mtp2Status	√	SS7 signaling link: The signaling link state changes.
0x0041	E_RCV_DSTDChannel	√	DST Series: The D-channel event.
0x0042	E_RCV_Ss7SpyMsu	—	SS7: New monitoring messages (MSU) are received from the SS7 server.
0x0043	E_CHG_ToneDetector	√	Tone Detector: The event to output the detection result in the new mode.
0x0044	E_CHG_ToneDetectorItem	—	Tone Detector: The event to count the periods of tones in the new mode.
0x0046	E_PROC_FaxDcnTag	√	Fax Channel: When the fax reception is successfully completed, judge if the remote fax machine has ever been compelled to stop.
0x0047	E_CHG_AMD	√	Tone detector, used to analyze if it is a man or an answer machine that picks up the phone.
0x0048	E_RCV_Ss7IsupCpg	—	SS7: The CPG message is received.
0x0049	E_CHG_CbChStatus	—	Large-capacity Channel Bank: To monitor any change in the connection state of a line between an on-board channel and a channel bank.
0x004a	E_RCV_SS7Mtp2Msu	√	SS7: A piece of MTP2 MSU message is received. (The configuration item AppHandleMtp2Msu should be set to 1)
0x0050	E_REFER_Status	—	SsmIpInitiateTransfer: Status after call transfer.
0x0051	E_CHG_SpyHangupInfo	—	DTP series: The monitoring circuit receives the off-hook event.
0x0052	E_CHG_CallBackRingCount	√	Analog Trunk Channel: The counter for signal cycles in the ringback current detector changes.
0x0053	E_CHG_RcvMF	—	ATP series: New MF characters are received.
0x0054	E_CHG_Pcm32LineState	√	PCM1280E: Changes in line synchronization status and signal state on PCM32 channels.
0x0055	E_RCV_SPY_CAS	—	DTP Series: The monitoring circuit receives the CAS.
0x0056	E_CHG_RingEchoToneCnt	√	Analog Trunk Channel: The count of ringback tone changes
0x0057	E_Ss7_L2ToL3_IND	√	SS7: MTP2 changes or MTP2 MSU is received. (The configuration item AppHandleMtp2Msu should be set to 2).
0x0060	E_RCV_IPR_DChannel	√	IPR Series: The D-channel event.
0x0061	E_RCV_IPR_DONGLE_ADDED	√	IPR Series: USB KEY detected.
0x0062	E_RCV_IPR_DONGLE_REMOVED	√	IPR Series: The removal of USB KEY detected.

0x0063	E_RCV_IPR_NIC_LINKED	√	IPR Series: Specified network card detected (Reserved. Not used at present).
0x0064	E_RCV_IPR_NIC_UNLINKED	√	IPR Series: Specified network card not detected.
0x0065	E_RCV_IPR_AUTH_OVERFLOW	√	IPR Series: Authorization Overflow.
0x0066	E_RCV_IPR_MEDIA_SESSION_STARTED	√	IPR Series: Session started.
0x0067	E_RCV_IPR_MEDIA_SESSION_STOPED	√	IPR Series: Session stopped. After the driver throws out this event, it will automatically stop transmitting RTP. Invoking the function SsmIPRStopSendSession at that time will fail therefore.
0x0068	E_RCV_IPR_AUX_MEDIA_SESSION_STARTED	√	IPR Series: Auxiliary session started. This event occurs during an extension-to-extension call.
0x0069	E_RCV_IPR_MEDIA_SESSION_FOWARDING	√	IPR Series: Session being forwarded.
0x006a	E_RCV_IPR_MEDIA_SESSION_FOWARD_STOPED	√	IPR Series: Session forwarding stopped.
0x006b	E_RCV_IPR_STATION_ADDED	√	IPR Series: The entrance of terminal detected.
0x006c	E_RCV_IPR_STATION_REMOVED	√	IPR Series: The exit of terminal detected.
0x006d	E_IPR_LINK_REC_SLAVER_CONNECTED	√	IPR Series: Recording Slaver connection detected.
0x006e	E_IPR_LINK_REC_SLAVER_DISCONNECTED	√	IPR Series: Recording Slaver disconnection detected.
0x006f	E_IPR_SLAVER_INIT_CB	√	IPR Series: Feedback on the recording Slaver initialization
0x0070	E_IPR_ACTIVE_SESSION_CB	√	IPR Series: Feedback on the requirement for the IPRR channel to enable Session receiving (for example, invoking SsmIPRActiveSession).
0x0071	E_IPR_DEACTIVE_SESSION_CB	√	IPR Series: Feedback on the requirement for the IPRR channel to disable Session receiving (for example, invoking SsmIPRDeActiveSession).
0x0072	E_IPR_START_REC_CB	√	IPR Series: Feedback on the requirement for the IPRR channel to start recording (for example, invoking SsmRecToFile, SsmRecToMem).
0x0073	E_IPR_STOP_REC_CB	√	IPR Series: Feedback on the requirement for the IPRR channel to stop recording (for example, invoking SsmStopRecToFile, SsmStopRecToMem).
0x0074	E_IPR_PAUSE_REC_CB	√	IPR Series: Feedback on the requirement for the IPRR channel to pause recording (for example, invoking SsmPauseRecToFile).
0x0075	E_IPR_RESTART_REC_CB	√	IPR Series: Feedback on the requirement for the IPRR channel to restart recording (for example, invoking SsmRestartRecToFile).
0x0076	E_IPR_START_SLAVER_CB	√	IPR Series: Feedback on the requirement of starting Slaver (for example, invoking SsmIPRStartRecSlaver).
0x0077	E_IPR_CLOSE_SLAVER_CB	√	IPR Series: Feedback on the requirement of closing Slaver (for example, invoking SsmIPRCloseRecSlaver).
0x0078	E_IPR_RCV_DTMF	√	IPR Series: In-band or RFC2833 DTMF detected.
0x0079	E_IPR_ACTIVE_AND_REC_CB	√	IPR Series: Feedback on the requirement for the IPRR channel to enable Session receiving and start recording (for example, invoking SsmIPRActiveAndRecToFile).
0x007a	E_IPR_DEACTIVE_AND_STOPREC_CB	√	Feedback on the requirement for the IPRR channel to disable Session receiving and stop recording (for example, invoking SsmIPRDeActiveAndStopRecToFile).
0x007b	E_RCV_IPA_DONGLE_ADDED	√	IPA Series: USB KEY detected.
0x007c	E_RCV_IPA_DONGLE_REMOVED	√	IPA Series: The removal of USB KEY detected.
0x007d	E_RCV_IPA_APPLICATION_PENDING	√	IPR Series: Once the failure or removal of the USB KEY is detected, the application will go into the pending state after waiting a period of time.
0x007e	E_RCV_IPR_AUX_MEDIA_SESSION_STOPED	√	IPR Series: Auxiliary session stopped. This event occurs during an extension-to-extension call.
0x007f	E_BOARD_ICMP_CHANGE	√	SHN B-type/C-type Series: The ICMP result changes.
0x0080	E_RCV_IsdnSpyMsu	—	ISDN: A new monitoring message (MSU) is received from the ISDN server.
0x0081	E_RCV_DecodeSs7Msu	—	SS7: New decoding messages are received from the SS7

			server.
0x0082	E_CHG_RCV_SELCALL	√	ATP, SHT Series: A Selcall Tone datum received.
0x0083	E_RCV_IsdnL2SpyMsu	—	The DTP board receives the original message of L2 in ISDN protocol. The function SsmGetIsdnL2SpyMsu can be used to obtain the content of the message.
0x0084	E_CHG_AMD_TIME	√	The event of AMD detection.
0x0085	E_REG_OPTIONS_RESPONSE	—	VoIP Board: The Option message responded.
0x0086	E_RCV_HMP_DONGLE_ADDED	√	HMP Series: USB KEY detected.
0x0087	E_RCV_HMP_DONGLE_REMOVED	√	HMP Series: The removal of USB KEY detected.
0x0088	E_REG_REQUEST	√	Registration request received by the application under the mode of registration authentication.
0x0089	E_REG_REGSTATUS	√	Change in registered user's status
0x008a	E_RCV_REFER	√	Call transfer.
0x008b	E_REG_NOTIFY_RESPONSE	—	VoIP Board: The Notify message responded.
0x008C	E_SIPSTACK_REGSTATUS	√	Registration status of the protocol stack changed.
0x008d	E_HOLDING_CALL	√	Hold event of the IP channel.
0x008e	E_SIP_RESPONSE	√	SIP message response event
0x008f	E_RCV_IPR_IGMP_MESSAGE	√	IPR series: Used to inform IGMP in ROIP recording so that the application can respond IGMP in time. This event is put out via the configuration item IGMPEventEnable.
0x0090	E_SYS_Lost_Interrupt	√	Board: Disconnected and data missing
0x0091	E_SIPSTACK_PCMSTATE	√	PCM channel state in the OPTIONS request message in SIP protocol
0x0092	E_SIP_SUBSCRIBE	√	Receive the Subscribe message
0x0093	E_RecEnergy	—	Recording energy

➢ **nReference** & **dwParam**

The physical meanings of both parameters nReference and dwParam have relation to the value of wEvent as shown in the table below.

Event Type	nReference	dwParam
E_PROC_Recognize	Logical channel number	The recognition result within the range below 1: Obtain the recognition result (Accept) 2: Do not understand (Reject) 3: Do not hear properly (Failed) 4: Hear nothing (Silence)
E_CHG_ISDNStatus	Logical digital-trunk number	Unused. The function SsmISDNGetStatus can be used to acquire the particular output parameters.
E_RCV_Ss7Msu	unused	Unused. The function SsmGetSs7Msu is used to take out messages.
E_CHG_Mtp3State	DPC (Destination Point Code) number	The message to show if the route from the local end to the specified DPC supports SS7 signaling 1: Support; 0: Unsupport.
E_CHG_FaxPages	Logical channel number	The total number of pages sent or received by the driver
E_PROC_FaxEnd	Logical channel number	The detailed task (fax transmission or reception) progress 0: Task (including the handshake phase as well as the faxing process) unfinished; 1: Task finished. The channel turns back to the 'idle' state. 2: Error in faxing process or task terminated by the application program via the function SsmFaxStop. The channel turns back to the 'idle' state. 3: All fax data sent or received. Begins to negotiate the disconnection with the remote end.
E_PROC_FaxDcnTag	Logical channel number	0: The faxing process is complete. 1: If the fax reception is terminated by receiving a DCN message, it may result from the compulsive stop of the fax machine.
E_CHG_PcmLinkStatus	Logical digital-trunk	The state of the digital trunk. See description on the

	number	parameter pwPcmLinkStatus in the function SsmGetPcmLinkStatus for details.
E_CHG_LineVoltage	Logical channel number	The voltage value (volt) on the analog phone line.
E_RCV_CAS	Logical channel number	The newly received ABCD signaling code, the 4 lower bits being valid Bit3 Bit2 Bit1 Bit0: Correspond to the ABCD code Rest bits: All set to 0
E_RCV_R2	Logical channel number	The newly received R2 signal 0: The R2 signal disappears. 1~15: The value of the forward R2 signal. 1~6: The value of the backward R2 signal.
E_PROC_WaitDTMF	Logical channel number	The reason why the task of WaitDTMF ends. 1: Time out 2: The designated end character received 3: A DTMF string of the designated length received The function SsmChkWaitDtmf can be used to acquire more information.
E_CHG_RcvDTMF	Logical channel number	The 16 higher bits represent the total number of received DTMF digits while the 16 lower bits are newly received digits, serving a purpose similar to a continuous call of the functions SsmGetRxDtmfLen and SsmGetLastDtmf.
E_PROC_SendDTMF	Logical channel number	=0: All DTMF digits in the buffer sent =1: The transmitting task terminated by the application program
E_PROC_SendFlash	Logical channel number	Unused
E_PROC_PlayEnd	Logical channel number	The reason why the task of voice playing ends 1: All voice data played out 2: Terminates upon receiving DTMF digits 3: Terminates upon detecting voice activities on the line. 4: Terminates upon detecting the remote client's hangup behavior. 5: Terminated by the application program 6: The task of file playing paused 7: Terminated by bus operation 8: The task of file playing terminated by the network fault See description on the SsmCheckPlay function for more information.
E_PROC_PlayFile	Logical channel number	One of the following parameters: ✧ Percentage of played voice data ✧ Time for playing voice data ✧ Total number of played bytes ✧ Total number of unplayed bytes It is the function SsmSetEvent that determines which one above to be output, the default value being 'time for playing voice data'.
E_PROC_PlayFileList	Logical channel number	The index value of the voice file being played in the file queue (numbered from 0)
E_PROC_PlayMem	Logical channel number	One of the following parameters: -1: The driver's playing pointer goes beyond the middle position of the buffer. -2: The driver's playing pointer goes beyond the end of the buffer and back to the top. Other values: The address offset of the driver's playing pointer in the buffer (calculated by byte)
E_PROC_RecordEnd	Logical channel number	The reason why the task of voice recording ends 1: Terminated by the application program 2: Terminates upon detecting DTMF digits 3: Terminates upon detecting the remote client's hangup behavior 4: Terminates when the recorded data reach a specified length or the recording operation lasts for a specified time. 5: The task of file recording paused 6: Writing recorded data to files failed 7: RTP timeout 8: The RTP payload format is changed in the session, and the new payload format is unsupported
E_PROC_RecordFile	Logical channel	One of the following parameters:

	number	✧ Time for recording voice data ✧ Total number of recorded bytes It is the function SsmSetEvent that determines which one above to be output, the default value being 'time for recording voice data'.
E_PROC_RecordMem	Logical channel number	One of the following parameters: -1: The driver's recording pointer goes beyond the middle position of the buffer. -2: The driver's recording pointer goes beyond the end of the buffer. Other values: The address offset of the driver's recording pointer.
E_PROC_SendFSK	Logical channel number	=0: The driver completes the transmission of all FSK data =1: The transmission task is terminated by the function SsmStopSendFsk
E_PROC_RcvFSK	Logical channel number	The reason why the FSK receiver stops 1: Time out, reception failed 2: The designated end byte received 3: Received FSK data reach the designated length 4: Data of the specified format received 5: Interrupted by the application program via SsmStopRcvFSK. The application can invoke the function SsmGetRcvFSK to acquire FSK data after getting this event.
E_CHG_ChState	Logical channel number	The 16 higher bits indicate the last channel state; The 16 lower bits represent the new channel state. See description on the function SsmGetChState for channel state values and more information.
E_PROC_AutoDial	Logical channel number	The progress value of the AutoDial task. See description on the function SsmChkAutoDial for the particular meaning of parameters.
E_CHG_RemoteChBlock	Logical channel number	The value to show the operation progress of blocking/unblocking the remote circuit. 0: The block caused by the local end successfully unblocked 1: The remote end successfully blocked 2: Waiting for the block-verified signal from the remote PBX 3: Waiting for the unblock-verified signal from the remote PBX
E_CHG_RemotePCMBlock	Logical digital-trunk number	The value to show the operation progress of blocking/unblocking the remote digital trunk. The 16 lower bits in the return value mean: 0: The block caused by the local end successfully unblocked 1: The remote end successfully blocked 2: Waiting for the block-verified signal from the remote PBX 3: Waiting for the unblock-verified signal from the remote PBX The 16 higher bits indicate the blocking mode.
E_SYS_ActualPickup	Logical channel number	0 (reserved).
E_CHG_RingFlag	Logical channel number	The change in voltage level of the ringing current. Bit31: Ringing current voltage level flag - 1: ringing current available; 0: no ringing current. Bit30~Bit0: The duration of ringing current at the last voltage level (ms)
E_CHG_RingCount	Logical channel number	The number of ringing current cycles.
E_CHG_CIDExBuf	Logical channel number	The total number of characters in Extended Caller ID Buffer. The function call of SsmGetCallerIdEx can help acquire the calling party number.
E_CHG_RxPhoNumBuf	Logical channel number	The total number of characters in Extended Callee ID Buffer. The function call of SsmGetPhoNumStr can help acquire the called party number.
E_CHG_PolarRvrsCount	Logical channel number	The value of the counter for calculating polarity reversals in the driver.

E_SYS_RemotePickup	Logical channel number	0 (reserved).
E_CHG_FlashCount	Logical channel number	The detected flash times.
E_CHG_HookState	Logical channel number	The user phone's behavior: 0: Hang up 1: Pick up
E_CHG_ToneAnalyze	Logical channel number	The result of the tone detector: The busy tone detected result of the tone detector: 16 higher bits: For a common tone detector, 16 higher bits represent the call progress tone detector number - 0: first group; 1: second group; 2: the return value of 16 lower bits is 7 or 8; 3: the return value of 16 lower bits is 4, 5 or 6. For an enhanced tone detector, the return value of 16 higher bits is always 0. 16 lower bits: Detected tone 1: Dial tone detected 2: Busy tone detected 3: Ringback tone detected 4: The line keeps silent after detecting the ringback tone 5: No sound detected 6: Voice detected, used for examining if the called party answers (picks up) or not 7: Tone at F1 frequency (which is set by the function SsmSetVoiceFxPara) detected, used for checking how the called party answers during audio dial 8: Tone at F2 frequency (which is set by the function SsmSetVoiceFxPara) detected, used for checking how the called party answers during audio dial 9: Tone type specified by users detected
E_OverallEnergy	Logical channel number	The value of overall energy detected by the tone detector
E_CHG_AMD	Logical channel number	Analysis result of an AMD event: 0: Man's pickup detected 1: Tone detected 2: Color ring or prompt tone detected 3: Time out 4: Line silent after tone or prompt tone detected 5: Line silent after dial tone detected 6: Busy tone detected
E_CHG_OvrlEnrgLevel	Logical channel number	The overall energy flag, the 16 lower bits being valid. Bit15 (tone identification): 0/1 There are tones/no tones Bit14 (function call status): 0 Reserved Bit13-0 (duration): A period of time when there are tones/no tones
E_CHG_BusyTone	Logical channel number	The busy tone detected result of the tone detector: 16 higher bits: Tone detector number - 0: first group, 1: second group 16 lower bits: Number of busy tone cycles.
E_CHG_BusyToneEx	Logical channel number	About the busy tone detected by tone detector using back-to-back busy tone detection: 0: The busy tone disappears 1: Busy tone detected
E_CHG_VocFxFlag	Logical channel number	The type of fax tones. 1: Tone at F1 frequency detected 2: Tone at F2 frequency detected
E_CHG_ToneValue	Logical channel number	Tone Detector: The value to show changes in the tone voltage level Only the 16 lower bits are in effect. Bit15(tone identification): 0/1 There are tones/no tones Bit14(function call status): 0/1 Successful/failed Bit13-0(duration): A period of time when there are tones/no tones
E_CHG_RingEchoToneTime	Logical channel number	Tone Detector: The time that a ringback tone cycle takes (ms)

E_CHG_PeakFrq	Logical channel number	Tone Detector: Peak frequency
E_SYS_BargeIn	Logical channel number	The detected result of the Barge-in detector changes: 0: The line keeps silent 1: Voice detected on the line
E_SYS_NoSound	Logical channel number	Unused.
E_SYS_TIMEOUT	Timer number	Unused.
E_CHG_SpyState	Logical SpyCic number	The 16 higher bits represent the last SpyCic state. The 16 lower bits imply the new SpyCic state. See description on the function SpyGetState for SpyCic state values and more information.
E_CHG_CICRxPhoNumBuf	Virtual circuit number	The length of the called party number. The application program need invoke the function ShgGetPhoNumStr after getting this event to acquire the called party number.
E_CHG_CICState	Virtual circuit number	The circuit state value. See description on the function ShgGetChState for details.
E_PROC_CICAutoDial	Virtual circuit number	The progress value of the ShgAutoDial task. 1: DIAL_ECHOTONE = 2, ringback tones are detected after sending the entire called party number 3: DIAL_BUSYTONE= 4, the called party is in a 'busy' state and the auto dial stops. 6: DIAL_VOICE= 7, the called party picks up and the auto dial stops. 9: DIAL_NOANSWER= 10, nobody answers and the auto dial fails. 10: DIAL_FAILURE= 11, the auto dial fails. 11: DIAL_INVALID_PHONUM = 12, the number unallocated and the auto dial stopped
E_RCV_DSTDChannel	Logical channel number	The subevent code, such as DST_AUDIO_CHG, DST_MSG_CHG which begin with 'DST_'. See *SynCTI Programmer's Manual - D-channel Event User Manual* for the meaning of each event code.
E_RCV_Ss7IsupUtuinf	Logical channel number	User-to-user information where the first byte means the data length (unsigned int) and the following are data (unsigned char[]).
E_CHG_Mtp2Status	SS7 signaling link number	The SS7 signaling link state value: 1: Out of service 2: Initial alignment 3: Aligned ready 4: Aligned not ready 5: In service 6: Processor outage
E_RCV_Ss7SpyMsu	Unused	Unused. The function SsmGetSs7SpyMsu is used to take out messages.
E_CHG_ToneDetector	Logical channel number	2 lower bits: The detected tone type 2 higher bits: Item number, i.e. the number for ToneDetectorItem For more information, refer to the configuration item ToneDetectorItem
E_CHG_ToneDetectorItem	Logical channel number	2 lower bits: Count of periods 2 higher bits: Item number, i.e. the number for ToneDetectorItem For more information, refer to the configuration item ToneDetectorItem
E_RCV_Ss7IsupCpg	Logical channel number	The size of the CPG message.
E_CHG_CbChStatus	Logical channel number	0: Line connected 1: Line disconnected
E_RCV_SS7Mtp2Msu	SS7 signaling link number	Length of the current MSU message.
E_CHG_RingEchoToneCnt	Logical channel number	Count of ringback tones.
E_Ss7_L2ToL3_IND	SS7 link number	enum { MTP2_MSU_RX_IND =1, SS7_MTP2_IN_SVC,

		SS7_MTP2_OUT_SVC, SS7_MTP2_REM_PR_OUT, SS7_MTP2_REM_PR_OK, SS7_MTP2_RXD_BSNT, SS7_MTP2_RTVD_MSG, SS7_MTP2_RTVL_COMPL, SS7_MTP2_RTVL_NOT_POS, }; The parameter dwParam being MTP2_MSU_RX_IND indicates that the MTP2 MSU message is received. Use the event structure SSM_EVENT to obtain the content of the message.
E_RCV_IPR_DChannel	Unused	The subevent code, such as DST_AUDIO_CHG, DST_MSG_CHG which begin with 'DST_'. See *SynCTI Programmer's Manual - D-channel Event User Manual* for the meaning of each event code.
E_RCV_IPR_DONGLE_ADDED	Unused	Unused.
E_RCV_IPR_DONGLE_REMOVED	Unused	Unused.
E_RCV_IPR_NIC_LINKED	Unused	Unused.
E_RCV_IPR_NIC_UNLINKED	Unused	Unused.
E_RCV_IPR_AUTH_OVERFLOW	Overflow mode: 1: Number of Sessions overflows 2: Number of Stations overflows	The 16 higher bits represents the number of Sessions. The 16 lower bits represents the number of Stations.
E_REFER_Status	Logical channel number	Status values after call transfer. See below for details. 0: Refer is in an idle state 1: Refer is in a Trying state. That is, the subscription is pending 2: In a state of 180RingBack 3: In a state of 200ok, 3XX 4: In a state of 4XX, 5XX, 6XX failure 40: Channel is in an unused state
E_CHG_SpyHangupInfo	Logical SpyCic number	The 16 higher bits represent the monitored PCM number for SpyCic that sends the disconnect message. The 16 lower bits imply the hangup information of SpyCic. To be exact: 0: Called party hangs up first 1: Calling party hangs up first See the description on the function SpyGetHangupInfo for more information.
E_CHG_CallBackRingCount	Logical channel number	The number of ringback current cycles.
E_CHG_RcvMF	Logical channel number	The 16 higher bits represent the total number of the received MF characters; the 16 lower bits represent the newly received characters.
E_CHG_Pcm32LineState	Logical channel number	Only the lower 16 bits are used to represent the line synchronization status and the signal state, among which bit0-bit7 indicate the line synchronization status and bit8-bit15 indicate the signal state. Refer to the function SsmGetPcm32LineState for details.
E_RCV_SPY_CAS	Logical channel number	The received ABCD signaling codes, The 4 lower bits are valid.
E_RCV_IPR_MEDIA_SESSION_STARTED	Logical channel number	SessionId. More Session information can be obtained from pvBuffer of SSM_EVENT. pvBuffer can be transformed to pIPR_SessionInfo.
E_RCV_IPR_MEDIA_SESSION_STOPED	Logical channel number	SessionId. More Session information can be obtained from pvBuffer of SSM_EVENT. pvBuffer can be transformed to pIPR_SessionInfo.
E_RCV_IPR_AUX_MEDIA_SESSION_STARTED	Logical channel number	SessionId. More Session information can be obtained from pvBuffer of SSM_EVENT. pvBuffer can be transformed to pIPR_SessionInfo.
E_RCV_IPR_MEDIA_SESSION_FOWARDING	Logical channel number	SessionId.
E_RCV_IPR_MEDIA_SESSIO	Logical channel	SessionId.

N_FOWARD_STOPED	number	
E_RCV_IPR_STATION_ADDED	Unused	Unused.
E_RCV_IPR_STATION_REMOVED	Unused	Unused.
E_IPR_LINK_REC_SLAVER_CONNECTED	Unused	The 16 higher bits represent SlaverId.
E_IPR_LINK_REC_SLAVER_DISCONNECTED	Unused	The 16 higher bits represent SlaverId.
E_IPR_SLAVER_INIT_CB	Unused	The 16 higher bits represent SlaverId, and the 16 lower bits represent the return value of Slaver. See below for the return values: 0-successful; 1-unknown; 2-timeout; 3-message data abnormal; 4-SessionId is active; 5-this SessionId is not found in the source busy list; 6-fail to create a new Session; 7-fail to create a file; 8-no active SessionId available; 9-SessionId is in the recording state; 10-SessionId is in the state of recording stop; 11-SessionId is not in the recording state; 12-SessionId is not in the state of recording to file; 13-SessionId is not in the state of recording pause; 14-recorded data is too short; 15-error occurs in initializing the WAV file header; 16-Slaver has been assigned with resource and restarted; 17-fail to apply for resources; 18-Slaver is not started; 19-unsupported encoding format; 20-unsupported RTP; 21-not enough resources available.
E_IPR_ACTIVE_SESSION_CB	Logical channel number	same as what dwParam represents in E_IPR_SLAVER_INIT_CB.
E_IPR_DEACTIVE_SESSION_CB	Logical channel number	same as what dwParam represents in E_IPR_SLAVER_INIT_CB.
E_IPR_START_REC_CB	Logical channel number	same as what dwParam represents in E_IPR_SLAVER_INIT_CB.
E_IPR_STOP_REC_CB	Logical channel number	same as what dwParam represents in E_IPR_SLAVER_INIT_CB.
E_IPR_PAUSE_REC_CB	Logical channel number	same as what dwParam represents in E_IPR_SLAVER_INIT_CB.
E_IPR_RESTART_REC_CB	Logical channel number	same as what dwParam represents in E_IPR_SLAVER_INIT_CB.
E_IPR_START_SLAVER_CB	Logical channel number	same as what dwParam represents in E_IPR_SLAVER_INIT_CB.
E_IPR_CLOSE_SLAVER_CB	Logical channel number	same as what dwParam represents in E_IPR_SLAVER_INIT_CB.
E_IPR_RCV_DTMF	Logical channel number	The 2 higher bytes represent the corresponding SlaverId, the 3rd byte represents the DTMF direction (0 represents primary; 1 represents secondary), and the 4th byte represents the DTMF information.
E_IPR_ACTIVE_AND_REC_CB	Logical channel number	same as what dwParam represents in E_IPR_SLAVER_INIT_CB.
E_IPR_DEACTIVE_AND_STOPREC_CB	Logical channel number	same as what dwParam represents in E_IPR_SLAVER_INIT_CB.
E_RCV_IPA_DONGLE_ADDED	Unused	Unused.
E_RCV_IPA_DONGLE_REMOVED	Unused	Unused.
E_RCV_IPA_APPLICATION_PENDING	Unused	Unused
E_RCV_IPR_AUX_MEDIA_SESSION_STOPED	Logical channel number	SessionId. More Session information can be obtained from pvBuffer of SSM_EVENT. pvBuffer can be transformed to pIPR_SessionInfo.
E_BOARD_ICMP_CHANGE	Board ID	ICMP result: 0: ICMP abnormal; 1: ICMP normal; 2: A3 cannot ping the network port of the board; 3: A3 can ping the network port of the board; 4: The shn537.bin application abnormal; 5: The shn537.bin application normal. Note: The driver will throw out this event only when the ICMP result or the shn537.bin changes. 0, 1 mean the shn573.bin initiates the ping with the use of

		SsmCheckBoardIcmp.
E_RCV_IsdnSpyMsu	Logical digital-trunk number	Length of the current MSU message.
E_RCV_DecodeSs7Msu	Unused	Unused. The function SsmGetDecodeSs7Msu is used to take out decoding messages.
E_CHG_RCV_SELCALL	Logical channel number	Received Selcall Tone data.
E_RCV_IsdnL2SpyMsu	Logical digital-trunk number	The size of the new message received by L2 in ISDN protocol.

E_CHG_AMD_TIME	Logical channel number	Bit15~Bit0: The continuous count for the on/off state. The formula to calculate the duration is: counts*16 (ms);
		Bit23~Bit16: The current status of AMD detection:
		enum
		{
		T1_WaitOff=0, // Wait at off state. Until the duration at off state keeps longer than AMDTOff will the state change to “T1_CountOff”. The count will be restarted once high level is detected during the time of T1_WaitOff.
		T1_CountOff=1, // Detect the duration at off state. The duration should be accumulated and added with the time at “T1_WaitOff”. It will not be calculated until high level is detected and keeps longer than AMDTOn. If the result is greater than AMDTimeA, the state will change to T2_CountOn; if the result is smaller than AMDTimeA, the state will go back to T1_WaitOff.
		T2_CountOn=2, // Detect the duration at on state. The duration should be accumulated and added with the time at high level during “T1_CountOff”. If the duration at on state becomes longer than AMDTimeC, the system will throw out the color ring event and return to the T1_WaitOff state. Only when low level is detected and keeps longer than AMDTOff will the duration at on state be calculated (The count will be restarted once high level is detected during the process). If the result is smaller than AMDTimeB, the state will change to T1_CountOff; if the result is larger than AMDTimeC, the color ring event will be thrown out and the state will go back to T1_CountOff; otherwise, the state will change to T3_CountOff, and meanwhile the time at on state will be cleared.
		T3_CountOff=3, // Detect the duration at off state. The duration should be accumulated and added with the time at low level during “T2_CountOn”. If the duration at off state becomes longer than AMDTimeD, the system will throw out the men’s pickup event and finish the AMD detection. Only when high level is detected and keeps longer than AMDTOn will the duration at off state be calculated (The count will be restarted once low level is detected during the process). If the result is greater than AMDTimeD, the men’s pickup event will be throw out and the AMD detection finishes; otherwise, the color ring event will be thrown out and the state will go back to T1_WaitOff.
		};
		Bit31~Bit24: The current status of low or high level:
		0: denotes the duration at off state;
		1: denotes the duration at on state.

E_REG_OPTIONS_RESPONSE	Unused	The response value of the Option message. The server address of the response message can be obtained via the parameter pvBuffer in function SSM_EVENT.
E_REG_REQUEST	Unused	SSM_EVENT pvBuffer: output registered account.
E_REG_REGSTATUS	Registration Status	Registration index number, i.e. SSM_EVENT pvBuffer: output registered account.
E_RCV_REFER	Logical channel number	Unused.
E_REG_NOTIFY _RESPONSE	Unused	The response value of the Notify message. The To Field of the response message can be obtained via the parameter

		pvBuffer in function SSM_EVENT.
E_SIPSTACK_REGSTATUS	Registration ID	Bit15~Bit0: The registration status: 2: Successful; 3: Failed; -1(0XFFFF): Cancel registration Successfully. Bit31~Bit16: The response code of the registration message. The code is -1(0XFFFF) if the remote end didn't respond the registration message.
E_HOLDING_CALL	Logical channel number	1, Call hold successful; 0, Call unhold successful.
E_SIP_SUBSCRIBE	Unused	Unused, you can get the content of the Subscribe message from pvBuffer of SSM_EVENT.
E_SIP_RESPONSE	Logical channel number	SIP message response code
E_SYS_Lost_Interrupt	Unused	Board ID number (corresponding to the ID in the configuration tool)
E_SIPSTACK_PCMSTATE	PCM channel number	PCM channel state: 0: down 1: up

1.14.2.2 SSM_EVENT

The SSM_EVENT structure declaration is:

```
typedef struct   _SSM_EVENT
{
    WORD     wEventCode;              //event code
    int      nReference;              //reference value
    DWORD    dwParam;                 //output parameter

    DWORD    dwUser;                   //user parameter
    DWORD    dwSubReason;             //subreason value
    DWORD    dwXtraInfo;              //extended information value

    PVOID    pvBuffer;        //pointer in the buffer which holds event information
    DWORD    dwBufferLength;          //size of the buffer which holds event information

    DWORD    dwDataLength;            //size of event information
    DWORD    dwEventFlag;             //event flag

    DWORD        dwReserved1;         //reserved parameter 1
    LONGLONG     llReserved2;         //reserved parameter 2
}SSM_EVENT, *PSSM_EVENT;
```

Parameters involved in the structure are described as follows.

- **wEventCode, nReference, dwParam**

These three parameters have the same meaning as wEvent, nReference and dwParam in the MESSAGE_INFO structure.

- **dwUser**

The meaning of dwUser varies on the event output mode as shown in the table below.

Event Output Mode	Meaning
Event Polling Mode	In case the event is defined by users, its value equals to the value of the subsection dwUser in the parameter pEvent when the function SsmPutUserEventA is invoked. Otherwise it is meaningless.
Event Callback Mode	Directly passes the value of dwUser in the parameter pEventSet when the function SsmSetEvent is invoked by the application program.

➢ **dwSubReason**

The parameter dwSubReason is the subreason value which gets meaningful only when the value of the parameter wEventCode equals to that of E_RCV_DSTDChannel or E_RCV_IPR_DChannel. The meaning of dwSubReason bears on the parameter dwParam carried in the event thrown out by the driver. The value of SESSIONID will be saved in the parameter dwSubReason upon the event E_IPR_ACTIVE_AND_REC_CB being thrown out. For more information, see *SynCTI Programmer's Manual - D-channel Event User Manual*.

➢ **dwXtraInfo**

The parameter dwXtraInfo is the extended information which gets meaningful only when the value of the parameter wEventCode equals to that of E_RCV_DSTDChannel, E_RCV_IPR_DChannel, E_RCV_IPR_STATION_ADDED or E_RCV_IPR_STATION_REMOVED. The meaning of dwXtraInfo bears on the parameter dwParam carried in the event thrown out by the driver. In the events E_RCV_IPR_STATION_ADDED and E_RCV_IPR_STATION_REMOVED, the 16 higher bits represent the transfer protocol, and the 16 lower bits represent StationId (The value of nStationId is 0xffff if SIP TRUNK is monitored). For more information about E_RCV_DST_DChannel and E_RCV_IPRDChannel, see *D-channel Event Manual*.

➢ **pvBuffer, dwBufferLength, dwDataLength**

The parameters pvBuffer, dwBufferLength and dwDataLength get meaningful only when the value of the parameter wEventCode equals to that of E_RCV_DSTDChannel, E_RCV_IPR_DChannel, E_RCV_IPR_MEDIA_SESSION_STARTED, E_RCV_IPR_MEDIA_SESSION_STOPED, E_RCV_IPR_AUX_MEDIA_SESSION_STARTED. Their meanings vary on the event output mode as shown in the table below.

Event Output Mode	Meaning
Event Polling Mode	pvBuffer is the pointer to the buffer which stores event information. The application should allocate a buffer space of not less than 512 bytes to pvBuffer and set the value of dwBufferLength to the actual size of the allocated buffer space (byte) while calling the function SsmWaitForEventA or SsmGetEventA. After the function SsmWaitForEventA or SsmGetEventA returns events, the content in pvBuffer means differently based on the carried parameter dwParam of the event thrown out by the driver. See *SynCTI Programmer's Manual - D-channel Event User Manual* for more information. dwDataLength is the true size of data in pvBuffer (byte).
Event Callback Mode	pvBuffer is the pointer to the buffer which stores event information. After generating the E_RCV_DSTDChannel, E_RCV_IPR_DChannel, E_RCV_IPR_MEDIA_SESSION_STARTED, E_RCV_IPR_MEDIA_SESSION_STOPED, E_RCV_IPR_AUX_MEDIA_SESSION_STARTED events, the driver will automatically allocate a buffer space for pvBuffer before invoking the callback function previously registered by the application. In this case, the parameter dwBufferLength has no meaning. The content in pvBuffer means differently based on the carried parameter dwParam of the event thrown out by the driver. In the events E_RCV_IPR_MEDIA_SESSION_STARTED, E_RCV_IPR_MEDIA_SESSION_STOPED, E_RCV_IPR_AUX_MEDIA_SESSION_STARTED, pvBuffer stores all the information of structure IPR_SessionInfo and can be compulsorily transformed into pIPR_SessionInfo to acquire information more conveniently. See *D-channel Event Manual* for more information about E_RCV_DST_DChannel and E_RCV_IPR_DChannel. dwDataLength is the true size of data in pvBuffer (byte).

➢ **dwEventFlag**

dwEventFlag is the event flag where only Bit0 and Bit2 are in effect and other bits all reserved. See the table below for what Bit0 and Bit2 mean.

Bit	Meaning
Bit0	=0: This event is created by the driver =1: This event is created by the application. To be exact, it is the self-defined event submitted to the driver by the function SsmPutUserEventA.
Bit2	=0: The event information is not truncated =1: The event information is truncated Bit2 gets meaningful only in the event polling mode. As long as one of the following two conditions is met, it will be set to 1 by the driver. ✧ The value of dwDataLength in the parameter pEventSet goes greater than 256 when the application invokes the function SsmPutUserEventA. ✧ The value of dwBufferLength in the parameter pEventSet when the application invokes the function SsmWaitForEventA is less than that of dwDataLength in the event temporarily saved to the driver before being output.

➢ **dwReserved1, llReserved2**

Both are reserved parameters.

1.14.2.3 IPR_SessionInfo

The structure of IPR_SessionInfo is:

```
//session information
typedef struct IPR_SessionInfo
{
    int nCallRef;       //The call index
    int nStationId;     //The StationId this Session corresponds to
    int nStationId2;    //Another StationId this Session corresponds to
    DWORD dwSessionId;
    IPR_Addr PrimaryAddr;     //A party's IP address and port in this session
    int nPrimaryCodec;        //A party's codec ID used in this Session
    IPR_Addr SecondaryAddr;       //Another party's IP address and port in this session
    int nSecondaryCodec;      //Another party's codec ID used in this Session
    char szFowardingIp[32];   //The IP address from where this Session forwards RTP packages, empty if no forwarding
    int nFowardingPPort;      //The port to where a party in this Session forwards RTP packages, being -1 if no forwarding
    int nFowardingSPort;      //The port to where another party in this Session forwards RTP packages , being -1 if no forwarding
}*pIPR_SessionInfo;
```

1.14.2.4 Reg_Info

The structure of Reg_Info is:

```
typedef struct tag_Reg_Info
{
   int    nRegIndex;// Index of the registered user
    int    nRegState;// Registration status: 0 failed; 1 successful
    int    nT0State;// State controller of the timer T0
```

```
    char    szTelNo[MAX_SIPIP_ADDR_LENGTH];// Registration username
    char    szClientAddress[MAX_SIPIP_ADDR_LENGTH];// IP address of the registered client
    char    szClientPort[MAX_SIPIP_ADDR_LENGTH];// Port number of the registered terminal
    char    szUserID[MAX_SIPIP_ADDR_LENGTH];// Authentication username
    int     nExpires; // Registration expiration
    int     nRegCountForAuthT0;// Counter of the timer T0
    int     nRegCountForAuthT1;// Counter of the timer T1
    char    szNonce[MAX_SIPIP_ADDR_LENGTH];
    char    szDispalyName[MAX_SIPIP_ADDR_LENGTH];
    int    nAuth;// Authentication parameter, value of 0 or 1
    char szPassword[MAX_SIPIP_ADDR_LENGTH];// User password
}Reg_Info;
```

1.14.2.5 RegResp

The structure of RegResp is:

```
typedef struct tag_RegResp
{
    int    nAuth;// Authentication parameter, value of 0 or 1
    char szPassword[MAX_SIPIP_ADDR_LENGTH];// User password
    char  szUsername[MAX_SIPIP_ADDR_LENGTH];// Authentication username (spare), the same as the username
}RegResp;
```

1.14.3 Event Filter

The application program may ask the SynCTI driver to or not to throw out an event. This feature is enabled via the function SsmSetEvent.

When the application invokes the function SsmSetEvent and uses the parameter wEvent=0xffff, whether the driver sends out an event or not is determined by the configuration of DefaultEventOutput. See description on this configuration item for details.

1.14.4 Self-defined Event

The application may generate events by itself during runtime, which are called self-defined events. To facilitate the application programming, the driver allows direct transmission of self-defined events, i.e. the application program can save its self-defined events to the event queue in the driver and process them via the function acquired through the driver supplied event.

The functions SsmPutUserEvent and SsmPutUserEventA can be used to submit the self-defined event to the driver.

1.14.5 SynCTI Programming Guide

This section highlights the keys to SynCTI driver programming respectively in MS VC/C++, VB, C++BUILDER, Delphi, PB6.5 and other compilation environments under Windows, as well as in C language under Linux.

1.14.5.1 MS VC/C++

The SynCTI driver provides the following files which are needed for programming in MS VC/C++:

- ✧ shpa3api.h: C/C++ header file
- ✧ shp_a3.lib: C/C++ import library file

1.14.5.2 VB

In Visual Basic 6.0, all functions in the dynamic-link library SHP_A3.DLL are declared by the file Shpa3api.bas from the driver. See Shpa3api.bas for details.

Actually the board operation in VB is also controlled via the call of shp_a3.dll. Therefore, the above description on programming in VC is also applicable to this section. Users are strongly recommended to first read through *1.14.5.1 MS VC/C++* and refer to the demo program compiled in C language while developing their application program based on the SynCTI driver.

In VB 6.0, a DLL function can be called only after it is declared. Regarding how to declare a DLL function, refer to the help file 'Declare statement' for VB.

Programmers can declare functions in VB by themselves referring to the function declaration in C language, if some new functions in shp_a3.dll are not declared in Shpa3api.bas. For example, below is a function declared in Shpa3api.h:

int WINAPI SsmPlayFile(int ch, LPSTR pszFileName, int nFormat, DWORD dwStartPos, DWORD dwLen);

Its declaration in Shpa3api.bas is:

Declare Function SsmPlayFile Lib 'shp_a3.dll' (ByVal ch As Long, ByVal pszFileName As String,

ByVal nFormat as Long, ByVal dwStartPos as Long,

ByVal dwLen As Long) As Long

Corresponding to 32-bit BOOL, 16-bit WORD and 32-bit DWORD in 32-bit VC environment, 32-bit Long, 16-bit Integer and 32-bit Long are used in VB environment. Note that **each parameter and its return value should contain the same number of bytes.**

Similar to programming in C language under Win9x/NT, programming for our voice boards in VB also can be performed by three steps:

- ➢ Initialization codes, including parts for initializing the driver and obtaining the total number and type of channels, written in the function Form_load.
- ➢ Processing codes, written in the timer or the function Form_Activate. For detailed information, refer to the relative demo program.
- ➢ Uninstallation codes, covering the components for uninstalling the driver and closing the board, written in the function Form_Unload.

Notes:

- Form_Activate is an endless-loop function. If it is used for writing processing codes, DoEvents must be called to ensure that other programs run normally.
- The fact that VB is an interpretive programming language requires you to take precautions that while using a DLL, the size of relative parameters must be defined before the function such as SsmGetLastErrMsg or SsmGetDtmfDStr, where each parameter takes a string and returns data following the function call in C language, being invoked in VB. For example:

  Dim ErrStr As String * 200

  SsmGetLastErrMsg(ErrStr)
- In case of programming in VB under WinNT, when the initialization and main processing codes are respectively written in the function Form_load and the timer, the codes in timer should not be invoked

before successful call of the function SsmStartCti. See below for details. Method 1: Create a global variable (with the default value of 'FALSE') and set it to 'TRUE' at the end of the function Form_load. The timer will determine whether to run the main processing codes according to the value of this variable. Method 2: Set Timer.Enable=False at the beginning of the function Form_load and Timer.Enable=True at the end. At the same time, set Timer.Enable=False in the function Form_unload before calling the function SsmCloseCti.

- As VB 6.0 doesn't support multithreading, when the event callback mode is used for programming, some of the functions (such as Right, Left, etc.) provided with VB 6.0 can't work properly as part of callback functions and the IDE of VB 6.0 can't set break points in callback functions, which bring a big trouble to debugging. Hence, we suggest using the event polling mode instead of the event callback mode for progrmaming in VB 6.0.

1.14.5.3 C++ BUILDER

The driver provides Shpa3api.h for your direct use.

Since shp_a3.lib is compiled in VC, it cannot be directly used by project files in C++BUILDER. However, you can take an import library LIB from DLL using the software tool IMPLIB.EXE provided by C++BUILDER. See the help file for IMPLIB.EXE to find how to use it. Here is a short example.

```
cd \windows\system
implib shp_a3.lib shp_a3.dll
```

What you need to do is copy the generated shp_a3.lib to the directory of your developed program.

Notes:

- **If there are two C language development systems from Borland installed in the hard disk, IMPLIB.EXE provided by C++ BUILDER must be in use, or errors may occur at runtime.**

1.14.5.4 Delphi

In Delphi, all functions in the dynamic-link library shp_a3.dll are declared by Shpa3api.pas. See Shpa3api.pas for details.

Just like in VB6.0, a DLL function can be called only after it is declared in Delphi. Regarding how to declare a DLL function, refer to the help file 'external declarations' for Delphi. Programmers can declare functions in Delphi by themselves referring to the function declaration in C language, if some new functions in shp_a3.dll are not declared in Shpa3api.pas. For example, below is a function declared in Shpa3api.h:

```
int WINAPI SsmPlayFile(int ch, LPSTR pszFileName, int nFormat, DWORD dwStartPos, DWORD dwLen);
```

Its declaration in Delphi is:

```
function SsmPlayFile (ch: Integer; pszFileName: PCHAR; nFormat:Integer; dwStartPos:
DWORD;dwLen:DWORD ) : Integer; stdcall; external 'Shp_a3.dll'。
```

Notes:

- The statement "stdcall; external 'Shp_a3.dll'"must be added to the end of the declaration.
- The char* data type used in C language should be declared as pchar data type in Delphi.

1.14.5.5 PB6.5

In Power Builder, all functions in the dynamic-link library shp_a3.dll are declared via the option 'Global External

Function' in the 'Declare' menu. Programmers can declare functions in PB6.5 by themselves referring to the function declaration in C language, if some new functions in shp_a3.dll are not declared in Shpa3api.pas. For example, below is a function declared in Shpa3api.h:

int WINAPI SsmPlayFile(int ch, LPSTR pszFileName, int nFormat, DWORD dwStartPos, DWORD dwLen);

Its declaration in Power Builder is:

Function Long SsmPlayFile (Long ch,String pszFileName,Long nFormat, UnsignedLong dwStartPos,UnsignedLong dwLen) Library "shp_a3.dll"

Note: The parameter type used in PB6.5 is similar to that in VB. Refer to the preceding section *1.14.5.2 VB*.

1.14.5.6 Other Programming Environments

Under 32-bit Win95/NT, a number of programming languages may be used in various ways to call functions in the dynamic-link library SHP_A3.DLL. As there are too many programming languages and each has its own requirements, we cannot describe all of them in this manual. Read carefully about the description on the programming language you are using.

1.14.5.7 Linux

The SynCTI driver provides the following files which are needed for programming in C/C++:

- shpa3api.h: C/C++ header file
- libshpa3.so: Shared library file

Generally, programming for our voice boards in C/C++ language under Linux are performed by three steps.

- Initialization codes, including parts for initializing the board and obtaining the total number and type of channels, used only once upon starting the application program.
- Processing codes, used to give commands to the driver and process the event from the driver. For detailed information, refer to the relative demo program.
- Uninstallation codes, covering the components for releasing the board and the driver, used only once upon exiting the application program.

1.14.5.8 Use of Character String

The use of character strings in not a few advanced languages differs from that in C language. Please follow the norms of C language when invoking functions which include character strings.

For the output character strings, like the second parameter in SsmGetDtmfStr, the application program should allocate storage space ahead of time. A lot of advanced languages support the automatic management of string's space allocation. However, when they are invoked across different language platforms, there is still a need to allocate space manually because of the unexpected demand of the other end.

In VB6.0, strings can be used as follows:

```
Dim DtmfStr As String*200
SsmGetDtmfStr(Ch, DtmfStr)
```

In VB.Net, strings can be used as follows:

```
Dim DtmfStr As New String(Chr(0), 200)
```

SsmGetDtmfStr(Ch, DtmfStr)

In C#, strings can be used as follows:

String DtmfStr = new String((Char)0, 200);

SsmGetDtmfStr(Ch, DtmfStr);

For the input character strings, like the two parameters in SsmStartCti, no displayed space is required to be allocated as our DLL wouldn't modify its content. However, don't forget to add the character '\0' at the end of strings to comply with the standard of C language.

In VB.net, strings can be used as follows:

Dim ShConfig as String

Dim ShIndex As String

ShConfig = My.Application.Info.DirectoryPath & "\ShConfig.ini" & Chr(0)

ShIndex = My.Application.Info.DirectoryPath & "\ShIndex.ini" & Chr(0)

SsmStartCti(ShConfig, ShIndex)

1.14.6 Demo Program (CTI Series)

To help developers understand the way to use some functions and features as soon as possible, Synway provides a large number of demo programs which are listed in the following table.

Name	Description	Programming Environment					
		Windows					Linux
		VB	VC	C#	PB	DELPHI	gcc
Call	Demonstrates how to process inbound calls.	√	√	—	√	√	√
Dial	Demonstrates how to enable outbound calls.	√	√	—	√	√	√
Conference Demo	Demonstrates how to enable teleconference, applicable to the station channel, analog trunk channel and digital trunk channel. See the file DemoConferenceUserManual.pdf (English) or DemoConferenceUserManual_cn.pdf (Chinese) in the Demo folder for more information.	—	√	—	—	—	—
Fax	Demonstrates how to make the Synway board serve as a simple fax server. Refer to the file DemoFAXUserManual.pdf (English) or DemoFAXUserManual_cn.pdf (Chinese) in the Demo folder for more information.	√	√	√	—	√	√
PBX	Demonstrates how to use the Synway board as a small PBX. Refer to the file DemoPbxUserManual.pdf (English) or DemoPbxUserManual_cn.pdf (Chinese) in the Demo folder for more information.	—	√	√	—	—	—
	The digital trunk board debug platform, used to debug the application program for the digital trunk board, can work as a simple PBX.	—	√	—	—	—	—
FSK	Demonstrates how to use the Synway board in sending and receiving FSK data.	—	√	—	—	—	—
Ss7Server	It is the SS7 server.	—	√	—	—	—	—
Test	Demonstrates the use of API functions.	—	√	—	—	—	√
Multi-thread VC	Demonstrates how to do programming for the Synway board by multithreading.	—	√	—	—	—	—
Event-driven Demo	Demonstrates how to use the event thrown out by the SynCTI driver for programming.	—	√	—	—	—	—

1.14.7 Demo Program (REC Series)

To help developers understand the way to use some functions and features as soon as possible, Synway provides a large number of demo programs which are listed in the following table.

Name	Description	Programming Environment			
		Windows			Linux
		VB	VC	C#	gcc
Recorder-ATP	Demonstrates how to use the ATP Series call-recording boards to monitor and record the analog phone line. Refer to the file DemoATPUserManual.pdf (English) or DemoATPUserManual_cn.pdf (Chinese) for details.	—	√	√	√
Recorder-DST	Demonstrates how to use the DST Series boards to monitor and record the digital subscriber line through high-impedance and parallel connection. Refer to the file DemoDSTUserManual.pdf (English) or DemoDSTUserManual_cn.pdf (Chinese) for details.	√	√	—	—
Recorder-DTP	Demonstrates how to use the DTP Series boards to monitor and record the digital trunk (E1).	√	√	—	—
RecPlay UseMemBlock	Demonstrates how to record in Pingpong Buffer Mode.	—	√	—	—

1.14.8 Driver-provided Timer

The driver provides two kinds of timers for the application program: Global Timer and Channel Timer.

- Global Timer

The driver has 128 system timers in it, using the hardware interrupt signal as the timed pulse, offering greater precision than timers provided by the operating system. The function SsmStartTimer is used to apply to the driver for a timer and set the parameters for it. The driver will throw out the E_SYS_TIMEOUT event every time when the timer overflows. The function SsmStopTimer is for terminating a timer.

- Channel Timer

The channel timer is applicable to the polling mode. When it is started, the driver can only save the current time to a variable but not judge if the timer overflows or not. Not until the application program calls the function ElapseTime will the driver return the timer's duration. Each channel has 10 timers allocated by the driver. They are started via the function StartTimer.

1.14.9 Driver-provided Debugging Feature

The SynCTI driver has the capability of API and event debugging, usually for the application system being developed, tested or preliminarily run. When error occurs in the API function call by the application program, or in the event message, or in the driver, the driver will output an error message to help the developer find the cause as soon as possible provided that the corresponding 'debugging information output' option is enabled.

The following configuration items are usable for debugging the application.

Configuration Item	Description
ApiLogEnable	Sets whether to output the API debugging information.
ApiLogSetEventRange	Sets the event range to limit the output events to be written into the log in the use of ApiLogEnable.
ApiLogSetChRange	Sets the channel range to limit the output events to be written into the log in the use of ApiLogEnable.

If the application uses the polling mode, it needs to continuously invoke some query functions in the timer, which leads to the output of a lot of function-call messages. The following configuration items may forbid some common query functions to output messages on function calls.

Configuration Item	Description
Mask_SsmGetNoSoundTime	Controls whether to output the message on the function call of SsmGetNoSoundTime.
Mask_SsmGetChStateKeepTime	Controls whether to output the message on the function call of SsmGetChStateKeepTime.
Mask_SsmGetPlayedTime	Controls whether to output the message on the function call of SsmGetPlayedTime.
Mask_SsmGetPlayedPercentage	Controls whether to output the message on the function call of SsmGetPlayedPercentage.
Mask_SsmGetPlayOffset	Controls whether to output the message on the function call of SsmGetPlayOffset.
Mask_SsmGetRecTime	Controls whether to output the message on the function call of SsmGetRecTime.
Mask_SsmGetRecOffset	Controls whether to output the message on the function call of SsmGetRecOffset.
Mask_SsmGetChState	Controls whether to output the message on the function call of SsmGetChState.

The driver-output API debugging information can be displayed and observed with the help of the third-party software tool Debugview.exe, which supports Windows Me, Windows 2000, Whistler Beta 1, Beta 2 and RC 1.

You can download Debugview.exe from the website http://technet.microsoft.com/en-us/sysinternals/bb896647.aspx#top free of charge and run it directly after decompressing the ZIP file, without the need of installation.

1.15 SHT Series (CTI Series)

1.15.1 Brief Introduction of SHT Series

The features supported by the SHT Series boards are shown in the following table.

Board Model	Form Factor	Max. Ports	Voice Processing Capabilities	Conferencing Capabilities	Voltage Detector	Audio Socket	FSK	TDM Bus	Max Fax Resource
SHT-2A/USB	USB	2	√	√	√	√	√	N/A	N/A
SHT-4A/USB	USB	4	√	√	√	√	√	N/A	N/A
SHT-2B/USB	USB	2	√	√	√	√	√	N/A	N/A
SHT-4B/USB	USB	4	√	√	√	√	√	N/A	N/A
SHT-8B/PCI	PCI	8	√	√	√	√	√	N/A	N/A
SHT-8B/PCI/FAX	PCI	8	√	√	√	√	√	N/A	4
SHT-8C/PCI/FAX	PCI	8	√	√	√	√	√	N/A	4
SHT-8C/PCI/EC	PCI	8	√	√	√	√	√	N/A	N/A
SHT-16C-CT/PCI/EC	PCI	16	√	√	√	√	√	H'100	N/A
SHT-16D-CT/PCIe	PCIe	16	√	√	√	√	√	H.100	N/A*
SHT-16A-CT/PCI	PCI	16	√	√	√	√	√	H.100	N/A
SHT-16B-CT/PCI	PCI	16	√	√	√	√	√	H.100	N/A
SHT-16B-CT/PCI/FAX	PCI	16	√	√	√	√	√	H.100	4
SHT-16C-CT/PCI/FAX	PCI	16	√	√	√	√	√	H.100	4
SHT-16B-CT/PCI/MP3	PCI	16	√	√	√	√	√	H.100	N/A
SHT-120A-CT/PCI	PCI	120	√	√	N/A	N/A	N/A	N/A	N/A
SHT-16B-CT/cPCI	cPCI	16	√	√	√	√	√	H.110	N/A

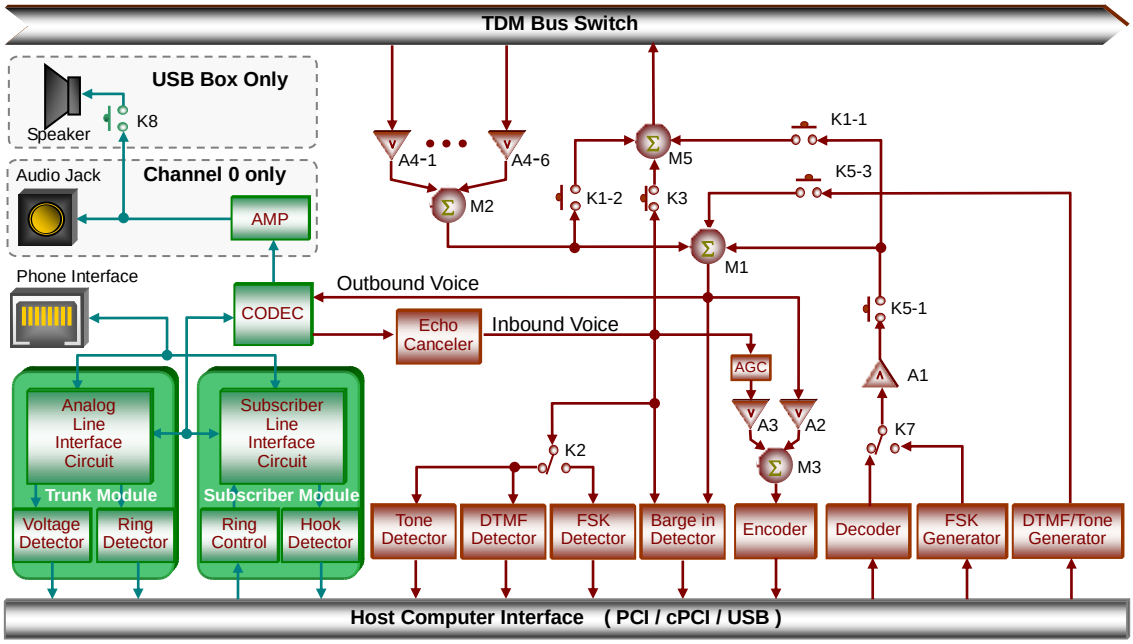
SHT-16B-CT/cPCI/MP3	cPCI	16	√	√	√	√	√	H.110	N/A
SHT-16B-CT/cPCI/FAX	cPCI	16	√	√	√	√	√	H.110	8
SHT-120A-CT/cPCI	cPCI	120	√	√	N/A	N/A	N/A	N/A	N/A

Note: The SHT-16B-CT/PCI/FAX board and the SHT-16B-CT/PCI/MP3 board differ in the hardware structure. However, they have the same board ID written in the firmware due to historical reasons. Hence it is essential to set the configuration item DSP3WORKMODE to ensure that the driver can properly distinguish these two board models.

The SHT-16D-CT/PCIe board, if inserted with an F021 module, can support 4-channel V.17 faxing.

1.15.2 Operation Principle of SHT Series

The figure below shows the operation principle of the SHT Series boards.



Description of components and symbols in the figure above:

Name	Description
M1	Outbound voice mixer.
K7	The Voice Play/Fsk Transmitter switch, controlled by the driver. When the application calls the playing function, K7 connects to the component Decoder; when it calls the function SsmStartSendFSK to start FSK data transmission, K7 connects to the component FSK Generator. For more information about the playing functions, refer to the Voice Playing section.
A1	The volume adjuster for voice playing. The volume can be set via the configuration item DefaultPlayVolume or the function SsmSetPlayVolume/SsmSetPlayGain, with the default value of 0.
K5-1	The switch that controls the voice playing operation, usually keeping off under the automatic control of the driver. Only when the application program invokes the playing function does the driver automatically switch on K5-1 and then switch off it at the end of voice playing. For more information about the playing functions, refer to the Voice Playing section.
K5-3	The switch that controls the signal output of Tone Generator and DTMF Generator, often keeping off under the automatic control of the driver. Only when the application program invokes the function SsmTxDtmf to start the DTMF generator or calls the function SsmSendTone/SsmSendToneEx to start the tone generator does the driver automatically switch on K5-3 and then switch off it at the task termination.
M2	Off-bus mixer.
A4-1 … A4-6	The volume adjuster used before the off-bus signals entering M2. Use the function SsmSetListenVlmInConf to set the volume, with the default value of 0DB. Or use the following configuration item or functions to set the volume for a channel with data to be off the bus.

	✧ SsmTalkWithEx ✧ SsmListenToEx/SsmLinkFromEx/SsmLinkFromAllCh
M3	Recording mixer The signals output from M3 enter the Encoder component and become the recording data.
A3	The volume adjuster used before incoming signals entering M3. The volume can be set via the function SsmSetRecVolume or the configuration item DefaultRecordVolume, with the default value of 0DB.
A2	The volume adjuster used before outgoing signals entering M3, switched on in default. The volume can be set via the function SsmSetRecMixer or the configuration item DefaultRecordMixerVolume, with the default value of -7.
M5	Onto-bus mixer. Mixes the playing data, the incoming signals and those output from the off-bus mixer, and puts the mixed voice onto TDM bus. Upon board initialization, the output from M5 is transported to a time slot (marked as 'ts' for short) on TDM bus. You can use the function SsmLinkToBus to designate the ts.
K3	The switch that controls whether the incoming signals enter the onto-bus mixer M5 or not, switched on in default. It can be set via the function SsmSetFlag (with the parameter F_InVoiceToBus) or the configuration item InVoiceToBus.
K1-2	The switch that controls whether the signals output from M2 go back onto TDM bus, being switched off in default. It can be set via the function SsmSetFlag (with the parameter F_MixerResToBus).
K1-1	The switch that controls whether the playing data enter the mixer M5 and go onto TDM bus or not, usually keeping off. It can be set via the function SsmSetPlayDest
K2	The switch that shifts between the FSK detector and other detectors, usually pointing to other detectors unless the FSK detector is started.

➢ **Voice Playing**

After the application invokes functions, the driver will automatically have K7 link to Decoder and switch on K5-1. The voice data processed by A1 have two places to go:

- ✧ Entering M1 and forming the final outgoing signals with other signal sources. The outgoing signals, if on Channel 0, will be sent to the on-board speaker output jack at the same time.
- ✧ Entering M5 under the control of K1-1 and forming the onto-bus signal sources with other signals.

➢ **Recording Example for Various Signal Sources**

A lot of special applications are available via the flexible use of the above switches for recording control and volume adjusters. In the following examples, ch1 means Channel 1 and ch2 is just Channel 2.

✧ Example 1: Only record the incoming signals on ch1, using the default volume.

```
SsmSetRecVolume(ch1, 0);          //start A3, set the volume gain to 0DB
SsmSetRecMixer(ch1, FALSE, 0);    //stop A2
SsmRecToFile(ch1, ……);             //start the task of file recording …
```

✧ Example 2: Record the incoming signals on ch1 (volume increased by 6DB) and the signals output from M2 (volume decreased by 3DB) at one time based on the establishement of a two-way call between ch1 and ch2.

```
SsmTalkWith(ch1, ch2);            //establish a two-way connection between ch1 and ch2 via TDM bus
SsmSetRecVolume(ch1, 2);          //start A3, set the volume gain to 2*3DB=6DB
SsmSetRecMixer(ch1, TRUE, -1);    //start A2, set the volume gain to -1*3DB=-3DB
SsmRecToFile(ch1, ……);             //start the task of file recording …
```

✧ Example 3: When ch1 joins a teleconference (assuming the conference room number is n), play background music on ch1 and record the conference through ch1. All signal sources go with normal volume, i.e. the gain is 0.

```
SsmJoinConfGroup(n, ch1, ……);      //ch1 joins the conference
SsmSetPlayDest(int ch, 1);         //switch on K1-1, put the playing data onto bus
SsmPlayFile(ch1, ……);              //start the task of voice playing on ch1, play background music
SsmSetRecVolume(ch1, 0);           //start A3, set the volume gain to 0DB
SsmSetRecMixer(ch1, TRUE, 0);      //start A2, set the volume gain to 0DB
SsmRecToFile(ch1, ……);             //start the task of file recording …
```

- **On-board Speaker Output Jack**

Channel 0 on the SHT Series boards is enabled by the analog audio amplifier circuit and the spearker output jack equipped on it to connect directly with the external speaker or headset and monitor voice signals in real time. This design is mainly for the following purposes.

- ✧ To transport incoming signals from any channel to the on-board speaker for play via TDM bus. This purpose can be achieved via the function SsmListenTo or SsmListenToEx.
- ✧ To send voice files recorded by the application straight to the on-board speaker for play via the call of playing functions. See the Voice Playing section in this chapter for details.

The AMP component in the figure above is the power amplifier for the analog audio amplifier circuit. Its gain can be set via the function SsmSetPowerAmpVlm.

Channel 0 on the USB voice box (with '/USB' marked in the model) is equipped with not only a speaker output jack but also an on-board speaker, which enables the voice playing operation. The switch K8, providing choices for the signals output from Channel 0 to enter the on-board speaker or the speaker jack, can be set via the function SsmSetLine0OutTo or the configuration item USBLine0Output, with the default setting of 'to the speaker jack'.

Note: Voices are sent onto the line as outgoing signals at the same time when transported to the on-board speaker on Channel 0.

1.15.3 Analog Trunk Channel

1.15.3.1 Generating Flash Signal on Analog Line

The flash signal is a short temporary on-hook pulse generated by a rapid clap on the hook switch of the analog phone during a call, generally used for such operations as transferring the call to an extension with the help of the PBX. You should do the clap as swift as possible lest the PBX mistakes the flash signal for the on-hook signal.

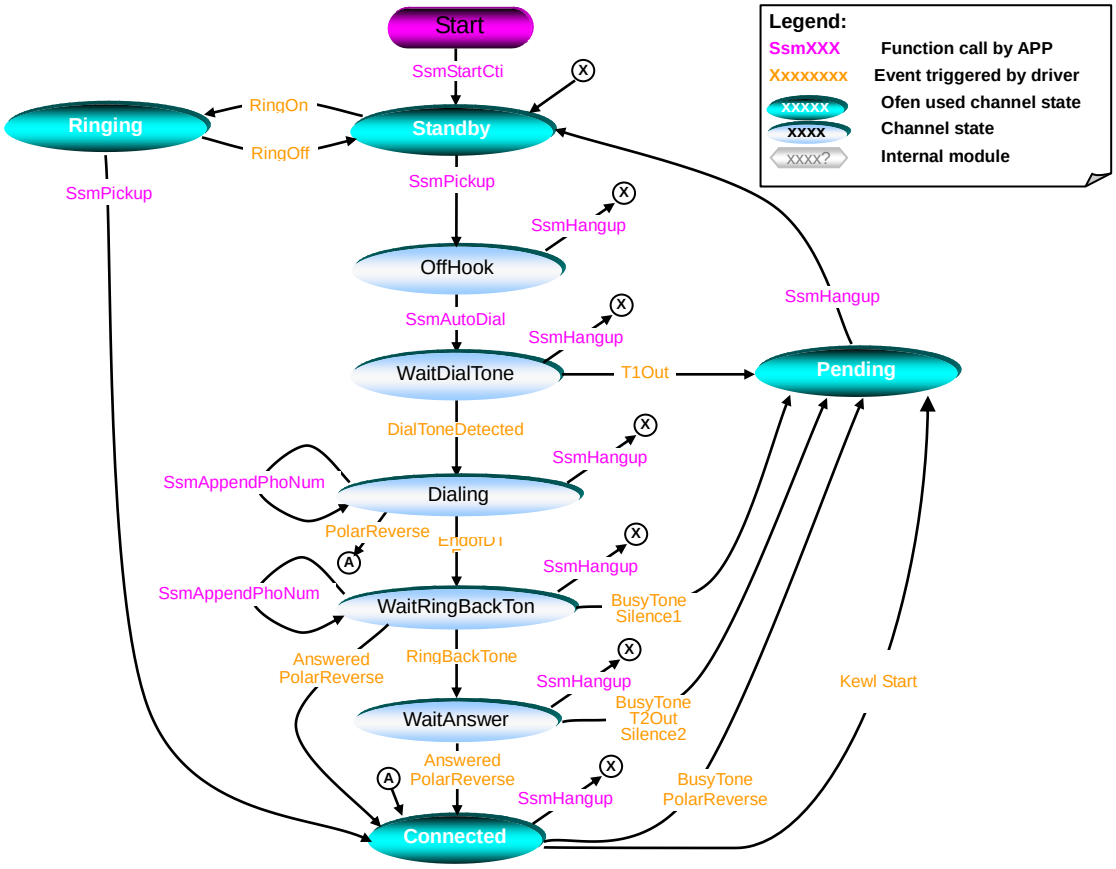
In the SynCTI driver, there are two ways available to generate a flash signal on the analog trunk.

- ➢ Invoke the function SsmTxFlash. In this way, the duration of the generated flash signal can be specified directly.
- ➢ Use the parameter – the '!' character - while invoking the function SsmTxDtmf. In this case, the duration of the generated flash signal can be set via the configuration item DefaultTxFlashTime, with the default value of 500ms.

1.15.3.2 Detecting Polarity Reversal Signal

For detailed information about the polarity reversal detection, refer to the Change in Analog Phone Line Voltage section in this chapter.

1.15.3.3 Analog Trunk Channel State Machine



Each state identifier in the above diagram is described in the following table.

State Identifier	Value	Macro Definition	Description
Standby	0	S_CALL_STANDBY	'Idle' state Before the channel state transfers to 'Idle', the driver: ➢ empties all buffers in DTMF Detector ➢ clears Caller ID Buffer ➢ closes DTMF Generator ➢ closes Tone Generator ➢ empties all buffers in the ringing current detector and then starts it ➢ empties all buffers in Tone Detector and then closes it ➢ starts (if the Caller ID detector is working in DTMF mode) or closes (if the Caller ID detector is working in FSK mode) the DTMF detector. ➢ closes the polarity reversal detector (if the polarity reversal detection is supported by hardware) ➢ sets the progress value of the AutoDial task to DIAL_STANDBY ➢ terminates the WaitDtmf task (if it has been started)
OffHook	1	S_CALL_PICKUPED	'Pickup' state Before the channel state transfers to 'Pickup',

			the driver: ➢ empties all buffers in the tone detector and then starts it ➢ empties all buffers in the ringing current detector and then closes it ➢ starts the DTMF detector if the Caller ID detector is working in FSK mode
Ringing	2	S_CALL_RINGING	'Ringing' state
Connected	3	S_CALL_TALKING	'Talking' state. Both parties can start talking when the channel is in such state. Before the channel state transfers to 'Talking', the driver: ➢ empties all buffers in the tone detector and then starts it ➢ starts the DTMF detector if the Caller ID detector is working in FSK mode ➢ resets the counter of the ringing current detector to 0
WaitDialtone	4	S_CALL_ANALOG_WAITDIALTONE	'Waiting for Dial Tone' state Before the channel state transfers to 'Waiting for Dial Tone', the driver: ➢ empties all buffers in the tone detector ➢ sets the progress value of the AutoDial task to DIAL_DIALING
Dialing	5	S_CALL_ANALOG_TXPHONUM	'Dialing' state Before the channel state transfers to 'Dialing', the driver: ➢ empties all buffers in the tone detector and then closes it ➢ closes the DTMF detector
WaitRingBackTone	6	S_CALL_ANALOG_WAITDIALRESULT	'Waiting for Ringback Tone' state Before the channel state transfers to 'Waiting for Ringback Tone', the driver: ➢ empties all buffers in the tone detector and then starts it ➢ starts the enhanced remote pickup detector (see the Enhanced Remote Pickup Detector section in this chapter for details) ➢ starts the DTMF detector ➢ starts Remote Pickup Detector
Pending	7	S_CALL_PENDING	'Pending' state
WaitAnswer	9	S_CALL_WAIT_REMOTE_PICKUP	'Waiting for Answer' state. The called party can hear the ringing tone when the channel goes into such state. If it is the station channel that makes an outbound call through the digital trunk at the local end, there should be ringback tones sent to this channel. Before the channel state transfers to 'Waiting for Answer', the driver: ➢ sets the progress value of the AutoDial task to DIAL_ECHOTONE and throw out the E_PROC_AutoDial event to the application program

All state values and corresponding macro definitions in the table above can be found in the file ShpA3Api.h.

The internal events automatically triggered by the driver itself are described as follows.

Name	Description
RingOn	This event is triggered once the driver detects ringing tones on the analog trunk. See the Ringing Current on Analog Phone Line section in this chapter for details.
RingOff	This event is triggered once the driver finds the ringing tone disappears on the analog trunk. See the Ringing Current on Analog Phone Line section in this chapter for details.
T1Out	The timer T1 overflows. T1 is set via the configuration item MaxWaitDialToneTime, with the default value of 3sec. After T1 overflows, the driver: ➢ sets the progress value of the AutoDial task to DIAL_NO_DIALTONE and sends the E_PROC_AutoDial event to the application; ➢ sets PendingReason to ANALOGOUT_NO_DIALTONE and has the channel state transfer to 'Pending'.
T2Out	The timer T2 overflows. T2 is set via the configuration item MaxWaitAutoDialAnswerTime, with the default value of 25sec. After T2 overflows, the driver: ➢ sets the progress value of the AutoDial task to DIAL_NOANSWER and sends the E_PROC_AutoDial event to the application; ➢ sets PendingReason to ANALOGOUT_NOANSWER and has the channel state transfer to 'Pending'.
Silence1	This event is triggered once the remote pickup detector outputs the E_CHG_ToneAnalyze event (with the parameter CHKTONE_NOVOICE). Refer to the Remote Pickup Detector section in this chapter for more information about the remote pickup detector. The driver: ➢ sets the progress value of the AutoDial task to DIAL_NOVOICE and sends the E_PROC_AutoDial event to the application; ➢ sets PendingReason to ANALOGOUT_NOVOICE and has the channel state transfer to 'Pending'.
Silence2	This event is triggered once the remote pickup detector outputs the E_CHG_ToneAnalyze event (with the parameter CHKTONE_ECHO_NOVOICE). Refer to the Remote Pickup Detector section in this chapter for more information about the remote pickup detector. The driver: ➢ sets the progress value of the AutoDial task to DIAL_ECHO_NOVOICE and sends the E_PROC_AutoDial event to the application; ➢ sets PendingReason to ANALOGOUT_ECHO_NOVOICE and has the channel state transfer to 'Pending'.
DialToneDetected	This event is triggered once the tone detector detects dial tones on the line. See the Tone Detector section in this chapter for details.
EndofDTMF	This event is triggered when the driver sends out all the digits in TxDtmfBuffer (after dialing).
RingBackTone	This event is triggered once the tone detector detects ringback tones on the line. See the Tone Detector section in this chapter for details.
Answered	This event is triggered once the tone detector detects the following results on the line. ✧ CHKTONE_VOICEF1: The driver sets the progress value of the AutoDial task to DIAL_VOICEF1 and sends the E_PROC_AutoDial event to the application. ✧ DIAL_VOICEF2: The driver sets the progress value of the AutoDial task to DIAL_VOICEF2 and sends the E_PROC_AutoDial event to the application. ✧ DIAL_VOICE: The driver sets the progress value of the AutoDial task to DIAL_VOICE and sends the E_PROC_AutoDial event to the application.
PolarReverse	This event is triggered when the voltage detector detects the polarity reversal signal on the line and the configuration item DisablePolarReverse is set to 0. In case a channel stays in the Dialing, WaitRingBackTone or WaitAnswer state, the driver will set the progress value of the AutoDial task to DIAL_VOICE and send the

	E_PROC_AutoDial event to the application.
BusyTone	This event is triggered once the tone detector detects busy tones on the line. Refer to the Tone Detector section in this chapter for more information. If a channel is in the Talking state at the time when busy tones are detected, the driver will: ➢ set PendingReason to ANALOGOUT_TALKING_REMOTE_HANGUPED and has the channel state transfer to 'Pending'. If a channel is in the WaitRingBackTone or WaitAnswer state at the time when busy tones are detected, the driver will: ➢ set the progress value of the AutoDial task to DIAL_BUSYTONE and sends the E_PROC_AutoDial event to the application; ➢ set PendingReason to ANALOGOUT_BUSYTONE and has the channel state transfer to 'Pending'.
Kewl Start	This event is used to detect remote hangup. When a channel is in the Connected state, if the voltage detector detects that the line voltage keeps the value of KSVoltageThreshold (whose default value is 0) for a period of time reaching KSKeepTime (whose default value is 256ms), the system will trigger this event and transfer the channel state to 'Pending'. The Kewl Start feature can be enabled by the configuration item EnableKewlStart.

The events output by the state machine are depicted in the following table.

Event Name	Description
E_CHG_ChState	The driver sends this event to the application when the channel state changes.
E_PROC_AutoDial	The driver sends this event to the application when the AutoDial task, which has been started by the function call of SsmAutoDial, progresses. Refer to the preceding content for detailed information on the driver's throwing out this event.

Note: All event declarations are listed in the file ShpA3Api.h.

1.15.3.4 Remote Pickup Detector (SHT Series Only)

The remote pickup detector applies only to the analog trunk channel, used to detect with excruciating precision if the calling party has picked up in an outgoing call.

Sometimes it is difficult to judge if the called party has picked up or not when the called party number has been completely sent in an outgoing call on the analog trunk. At present, our boards support two ways to solve such problem: judging by polarity reversal detection and by voice detection.

For detailed information about the polarity reversal signal, refer to the Change in Analog Phone Line Voltage section in this chapter.

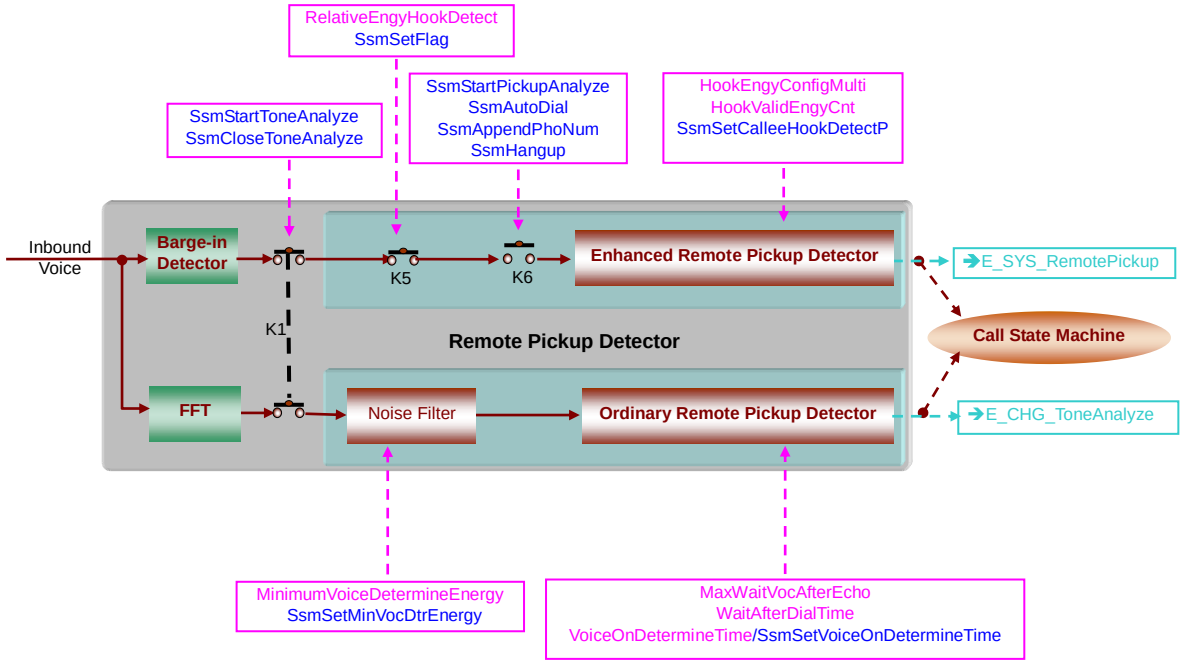
In case the remote pickup behavior is judged by voice detection, generally there are 2 different situations as stated below after dialing.

- The called party picks up the phone and speaks (e.g. say 'hello') after hearing the rings (no matter how long the phone rings before the pickup). In such situation, the calling channel detects the voice energy on the line and believes the called party has picked up as long as the voice energy and duration get greater than the preset threshold value. Note that this method goes invalid if the called party is provided with color ring service, because the PBX will play the color ring (voice signals) as the ringback tone to the calling party.

- The called party picks up the phone and keeps silent (does not speak) after hearing the rings. In such case, there will not be ringback tones as well as any other voice signals on the line to the calling party, which make the calling channel unable to decide if the called party has picked up. Hence the application

program has to judge by itself.

The SynCTI driver provides an independent remote pickup detector based on voice detection for each analog trunk channel. Its operation principle, related configuration items, functions and output events are all illustrated in the following figure.



The remote pickup detector contains two different combinations: one is of Noise Filter and Ordinary Remote Pickup Detector and the other of K5, K6 and Enhanced Remote Pickup Detector as shown in the figure above. Both are controlled by the switch K1 which can be set via the function SsmStartToneAnalyze/SsmCloseToneAnalyze. **Note: K1 controls both the tone detector and the Barge-in detector at one time.**

The ordinary remote pickup detector receives the output from the FFT component, appropriate for ordinary occasions. The event it throws out is E_CHG_ToneAnalyze.

The enhanced remote pickup detector uses a special algorithm to judge the pickup behavior on the analog phone line, offering greater precision than the ordinary remote pickup detector. The event it sends off is E_SYS_RemotePickup.

1.15.3.4.1Ordinary Remote Pickup Detector

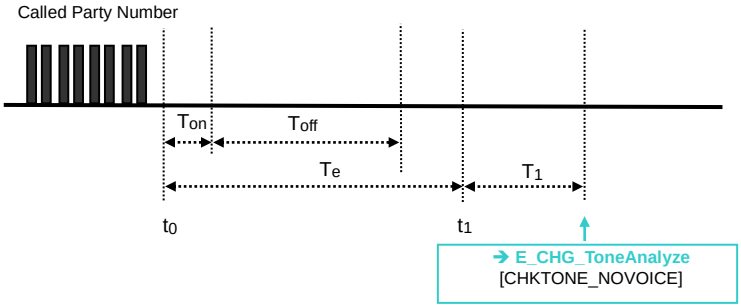
The ordinary remote pickup detector detects the voice activities on the line.

The noise filter eliminates the noise on the line. Its operation principle and parameter settings are identical to those of the noise filter mentioned in the Tone Detector section. See relative parts in the Tone Detector section for details.

When the noise filter determine that there exist voice signals on the line, the ordinary remote pickup detector won't agree until it is sure that the voice signal are not that specified by the 1st call progress tone detector.

The ordinary remote pickup detector outputs the E_CHG_ToneAnalyze event in one of the three situations stated below.

(1) The called party picks up the phone immediately when the phone rings and then keeps silent. In such case, there will not be ringback tones (Some flying ringback tones may appear on the line but are bound to be filtered out because they can't constitute an entire cycle so as not to be detected) as well as any other voice signals on the line to the calling party as shown below.



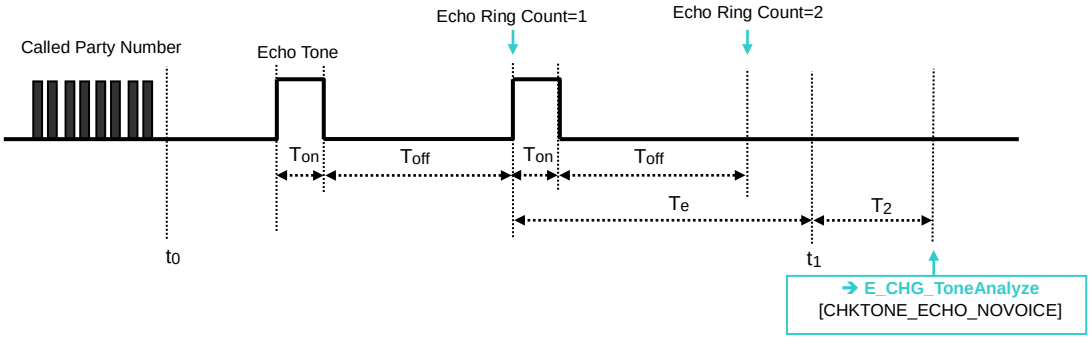
In the figure above,

T_{on}, T_{off}: Indicate the signal durations respectively at on and off set by the parameters about the ringback tone waveform.

T_e: The accepted maximum error of the ringback tone cycle, $T_e = (T_{on}+T_{off})*(1+20\%)$.

T1: The threshold value for silent duration after dialing, counted from t_1. The driver outputs E_CHG_ToneAnalyze (with the parameter CHKTONE_NOVOICE) as long as the line keeps silent longer than T_1 which is designated by the configuration item WaitAfterDialTime.

(2) The called party picks up the phone after at least one entire ring and then keeps silent. In such situation, the calling channel can detect at least one ringback tone but no voice activities as shown below.



In the figure above,

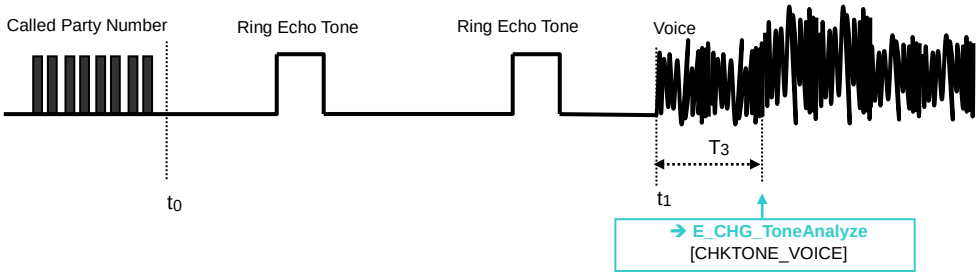
t_0: Represents the time when the dialing is completed.

T_{on}, T_{off}: Indicate the signal durations respectively at on and off set by the parameters about the ringback tone waveform.

T_e: The accepted maximum error of the ringback tone cycle, $T_e = (T_{on}+T_{off})*(1+20\%)$.

T_2: The threshold value for silent duration after ringback tones' disappearing, counted from t_1 (i.e. the time when the ringback tone has surely gone). The driver outputs E_CHG_ToneAnalyze (with the parameter CHKTONE_ECHO_NOVOICE) as long as the line keeps silent longer than T_2 which is designated by the configuration item MaxWaitVocAfterEcho.

(3) The called party picks up the phone after hearing the rings (no matter how long the phone rings before the pickup) and immediately speaks (e.g. say 'hello').



In the figure above, T_3 is the threshold value for judging voice signals, counted from t_1 when voice signals appear. The driver outputs E_CHG_ToneAnalyze (with the parameter CHKTONE_VOICE) as long as the voice signal lasts longer than T_3 which is designated by the configuration item VoiceOnDetermineTime or the function SsmSetVoiceOnDetermineTime. The function SsmGetVoiceOnDetermineTime can be used to acquire the value of T_3 saved in the driver.

1.15.3.4.2Enhanced Remote Pickup Detector

The switch K5 is switched on after the successful system initialization. It is set by the configuration item RelativeEngyHookDetect or the function SsmSetFlag (with the parameter F_RELATIVEENGYHOOKDETECT) which also can be used to obtain the state of K5.

The switch K6 can be controlled both by the application program and the driver. It is switched off after the successful system initialization.

K6 will be automatically switched on to enable the remote pickup detector as one of the following conditions is satisfied.

- The application invokes the function SsmStartPickupAnalyze.
- The driver will send the called party number digit by digit to the remote PBX in the state of S_CALL_ANALOG_TXPHONUM after the application calls the function SsmAutoDial to start an outgoing call. When the called party number has been completely transmitted, the channel state will transfer to S_CALL_ANALOG_WAITDIALRESULT after the driver automatically switches on K6.
- The application invokes the function SsmAppendPhoNum when a channel stays in the S_CALL_ANALOG_TXPHONUM or S_CALL_ANALOG_WAITDIALRESULT state.

K6 will be automatically switched off by the driver to disable the remote pickup detector as one of the following conditions is satisfied.

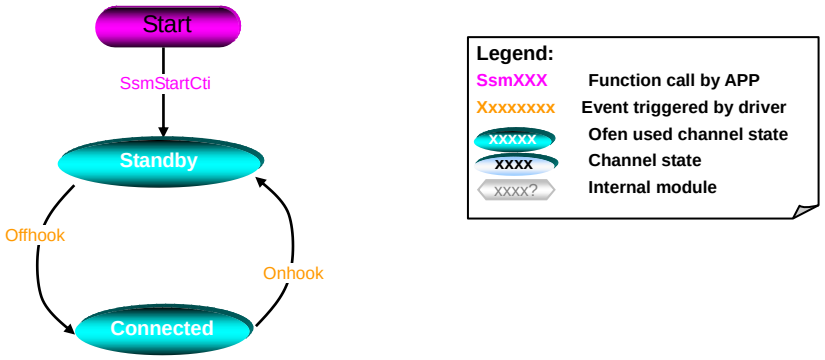
- The remote pickup detector accomplishes a detection and outputs the E_SYS_RemotePickup event.
- The application invokes the function SsmHangup.
- The state machine of the analog trunk channel receives the E_CHG_BusyTone event which is thrown out by the tone detector.

The configuration item HookEngyConfigMulti, HookValidEngyCnt or the function SsmSetCalleeHookDetectP is used to set parameters for the enhanced remote pickup detector.

When the enhanced remote pickup detector detects the called party's pickup behavior, it will send out the E_SYS_RemotePickup event to the call state machine for subsequent processing. The function SsmGetPickup can be used to obtain the result.

1.15.4 Station Channel

1.15.4.1 State Machine



Each state identifier in the above diagram is described in the following table.

State Identifier	Value	Macro Definition	Description
Standby	0	S_CALL_STANDBY	'Idle' state. When the channel state transfers to 'Idle', the driver will: ➢ empty all buffers in DTMF Detector ➢ close DTMF_Generator and terminate the unfinished WaitDtmf task. close Tone Generator
OffHook	1	S_CALL_PICKUPED	'Off-hook' state. When the channel state transfers to 'Off-hook', the driver will: ➢ start sending dial tones to the phone provided that the configuration item AutoSendDialTone is set to 1. ➢ empty all buffers in DTMF Detector and then start it.

All state values and corresponding macro definitions in the table above can be found in the file ShpA3Api.h.

Note: The driver throws out the E_CHG_ChState event to the application every time when the channel state changes.

The internal events automatically triggered the driver itself are described as follows.

Name	Description
Offhook	This event is triggered once the driver detects the pickup behavior.
Onhook	This event is triggered once the driver detects the hangup behavior.

1.15.4.2 Pickup/hangup Detection

When the phone is picked up or hung up, some dithering signals may be generated on the line. To prevent misdetection, those signals should be filtered so as not to affect the accuracy in detecting the pickup/hangup behavior. The configuration item UserOnHookFilterTime is used to set the minimum pickup signal duration. Only when the driver detects the pickup signal lasts longer than the set value of this configuration item will it confirm the pickup behavior.

The driver throws out the E_CHG_HookState event to the application every time when it confirms a pickup/hangup behavior. Also the function SsmGetHookState can be used to obtain the off-hook/on-hook state of the station channel.

1.15.4.3 Flash Signal Detection

Because the station channel serves as the user circuit on analog PBXes, it has the capability of detecting flash signals generated on the phone. Refer to the Flash Signal on Analog Phone Line section in this chapter for more information about the flash signal.

The function SsmSetLocalFlashTime or the configuration item MaxLocalFlashTime is used to set Nmax, the maximum interval for judging the flash signal. The configuration item LocalHookFilterTime is used to set Nmin, the minimum interval for judging the flash signal. The user releases the hook switch on the phone in a short period of time (assumed as n) after pressing it. If Nmin<n<Nmax, the driver affirms that the user is sending a flash signal, otherwise it regards this action as the hangup behavior.

The driver provides a flash signal counter for each station channel, adding 1 to the count and throwing out the E_CHG_FlashCount event to the application program every time when it detects a flash signal. Also the function SsmGetFlashCount can be used to acquire the count. The driver will automatically reset the counter to 0 whenever the channel state transfers to 'Standby'. You can use the function SsmClearFlashCount to clear the count as well.

1.15.4.4 Sending Ringing Signals to Phone

Each station channel on the Synway board is equipped with a ringing current generator for sending ringing signals to the telephone. As viewed from the waveform, the regular ringing signal is periodic with 1sec on and 4sec off. However, the on and off duration can be adjusted by the function SsmSetRingPeriod, and also the signal can be set to consecutive by the configuration item UserChGenerateRingMode.

Both SsmStartRing and SsmStartRingWithCIDStr can be used to send ringing signals to the telephone. SsmStartRing is for the analog and magnet phones; SsmStartRingWithCIDStr is only for the station channel but can send the calling party information besides ringing signals. Because SsmStartRingWithCIDStr only accepts the modulated FSK bit stream, so it is often used with the function fPcm_ConvertFskCID.

The driver provides a ringing signal counter for each station or magnet channel, adding 1 to the count every time when it completes a signal transmission (at on state). Also the function SsmCheckSendRing can be used to acquire the count.

The application program can invoke the function SsmStopRing any time to stop sending ringing signals to the phone

1.15.4.5 Generating Polarity Reversal Signal on Phone Line

For detailed information about the polarity reversal signal, refer to the Change in Analog Phone Line Voltage section in this chapter.

Whether the station channel on the Synway board has the capability to generate the polarity reversal signal on the analog phone line depends on if it is equipped with a B-type 1.0 series station module and if the module has the capability to generate the polarity reversal signal (Note: Some of the B-type 1.0 series station modules don't have such capability).

The polarity reversal feature needs to be enabled via the configuration item UserSendPolar.
The function SsmGetPolarState is used to get the current voltage polarity on the phone line and the function SsmSetPolarState to set the voltage polarity.

1.15.4.6 Auto-transmission of Dial Tones upon Pickup Detected

It is the configuration item AutoSendDialTone that determines if the driver, after the phone which is linked to the station channel picks up, will automatically send dial tones to the phone. As to whether the driver automatically stops sending dial tones upon detecting DTMF digits, it is set via the configuration item StopSendDialToneOnDtmf.

1.15.5 Composite Module

Synway offers a special kind of module to the SHT Series boards, that is, a module fitted with two put-in circuits respectively to an analog trunk channel and a station channel. This kind of module is called the composite module. The great advantage of this module is it enables direct connection between trunk and station which keeps call uninterrupted during power outage. To be exact, when the power of the application system is accidentally cut off, the analog trunk channel and the station channel linked to a same composite module connects directly to each other. Therefore, the station channel is still able to communicate with the telephone network via the analog trunk channel in such case.

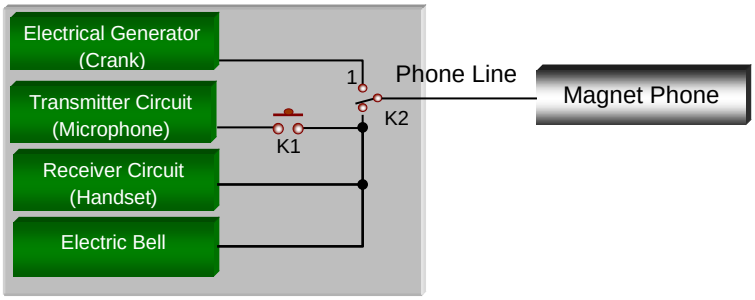
The relationship between the analog trunk channel and the station channel on the composite module is set via the function SsmSetUnimoduleState or the configuration item UnimoduleState.

1.15.6 Magnet Channel

1.15.6.1 Overview

The channel fitted with the magnet module on the SHT Series boards is called the magnet channel for short, used to connect with the magnet phone.

The internal structure of the magnet module is similar to that of a magnet phone whose operation principle is illustrated below.



As shown in the figure above, the magnet phone is composed of the transmitter circuit, the receiver circuit, the electric bell, the electrical generator and the switches K1, K2 (Although it seems all the components are connected by single lines, the actual thing is they are connected by double lines and form a loop with each other). When the crank is turned, K2 automatically switches to the point 1 so as to send the ringing current to the electric bell of the remote telephone; when K1 is pressed (equivalent to the situation that the hook switch on an ordinary telephone is released), which makes the transmitter circuit be in connection and enabled to send out voices (The main purpose to set K1 is to disconnect the dry battery at the time as no voices are transmitted).

The magnet module has the following features.

- The magnet module has only three states: ‘idle’, ‘sending rings’ and ‘ringing’, and does not support the pickup/hangup behavior. It is the same as the station module in the rhythm of sending the ringing current, that is, ring for 1sec and keep silent for 4sec. The ring can be properly detected even in the 4sec when the signal goes at off state;
- The internal ringing current generator can generate the ringing current on the line. Its operation principle

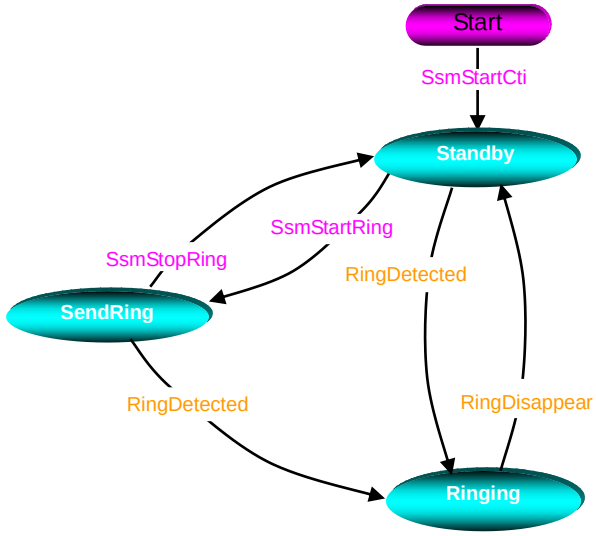
is identical to that of the ringing current generator on the station module for the SHT/B Series boards or above model. On the one hand, the magnet module is not allowed to send the ringing current if the ringing current is already available on the line; on the other hand, the magnet module will automatically stop sending the ringing current if it detects the ringing current on the line midway in transmission;

- The magnet module leaves out K1 so that the receiver circuit and the transmitter circuit are always in connection when it stays in the 'idle' or 'ringing' state.

Note:

The on-board module identification circuit cannot recognize the magnet module but mistakes it as the analog trunk module. Hence it is essential to set the configuration item IsMagnetModule to ensure that the driver can properly distinguish these two kinds of modules.

1.15.6.2 State Machine



In the figure above,

State Identifier	Value	Macro Definition	Description
Standby	0	S_CALL_STANDBY	'Idle' state. The channel goes into this state after the driver's successful initialization.
SendRing	100	S_CALL_SENDRING	'Sending rings' state. In this state, the channel is sending rings to the magnet phone and examining if there is ringing current on the line at the same time. The ringing current detection can be set via the configuration RingDetectFilterPara item.
Ringing	2	S_CALL_RINGING	'Ringing' state. The magnet phone is ringing and the driver checks if rings have disappeared in this state.

1.15.7 EM Module

E&M is a universal technology involving trunk signaling over telephone switches and PBXs. EM interface is a type of analog interface, through which information and signaling are separated. Like other kind of modules from Synway, an EM module corresponds to two channels: one is an EM Control channel and the other is an EM Voice channel.

The supported functions for the EM control channel include the API functions SsmPickup, SsmHangup and the

functions SsmGetChState, SsmGetHookState for acquiring status. SsmPickup and SsmHangup are used to send 1 and 0 signals to the remote end; SsmGetHookState is used to detect the 1 or 0 signal from the remote end; SsmGetChState is used to get the local on-hook and off-hook states which change with the local end invoking SsmPickup and SsmHangup.

The operations on the EM Voice channel are the same as those on other voice channels.

1.15.8 Non-module Channel

The SHT Series boards use the modular structure, composed of the main board and different modules. The on-board DSP chips can work properly even if there is no module installed so that the non-module channel also supports voice processing. However, since no physical interface circuits for telephone lines available on the board, voice signals can be acquired only from TDM bus. In some particular occasions when other boards with TDM bus are required to record voices, what you need do is put the voice data via TDM bus to the non-module channel and use the output from the Barge-in detector as the pre-requisites for recording control. Such feature can be enabled via the configuration item NoModuleChBusRec or the function SsmSetNoModuleChBusRec. The non-module channel has the following voice processing capabilities: recording, data exchange via TDM bus, DTMF reception and Barge-in detection.

Note: In case there is no module installed on the main board of the SHT Series, the channel type number of 20 means it is the non-module channel.

1.16 SHD Series (CTI Series)

1.16.1 Brief Introduction of SHD Series

The features supported by the SHD Series boards are shown in the following table.

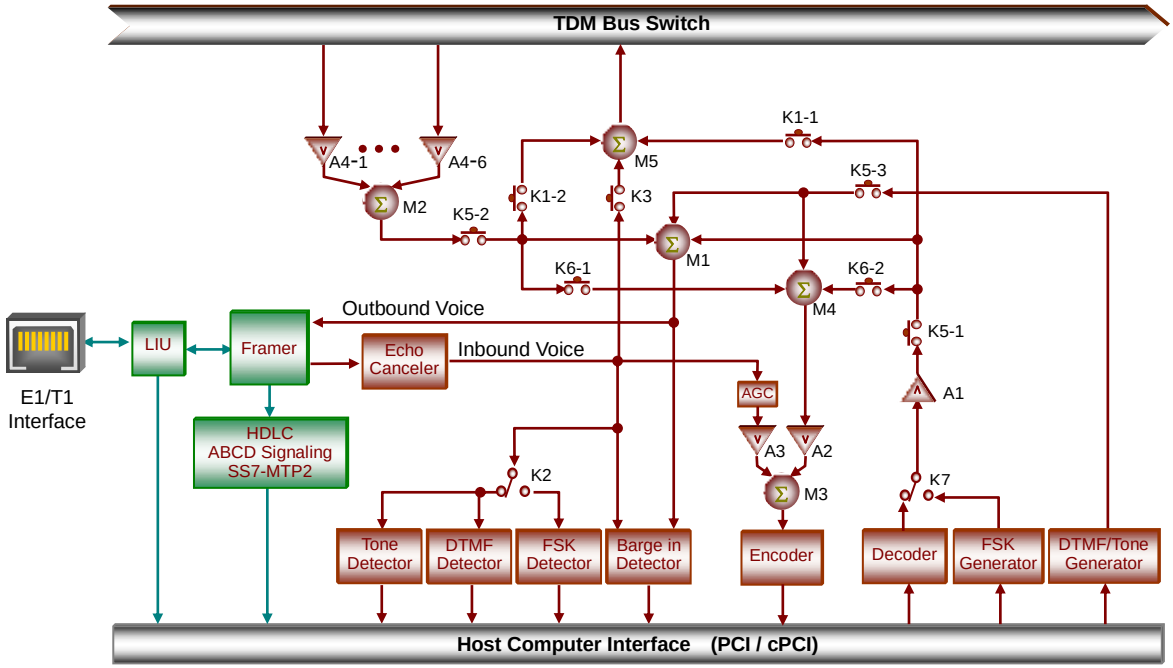
Board Model	Form Factor	Trunk Type	Max. Ports	Voice Processing Capabilities	Conferencing Capabilities	FSK	Max Fax Resource	SS1	ISDN PRI	SS7	SS7 Signaling Links
SHD-30A-CT/PCI/SS1	PCI	1 E1	30	√	√	√	N/A	√	N/A	N/A	N/A
SHD-30A-CT/PCI/ISDN	PCI	1 E1	30	√	√	√	N/A	√	√	N/A	N/A
SHD-30A-CT/PCI/SS7	PCI	1 E1	30	√	√	√	N/A	√	√	√	1
SHD-60A-CT/PCI/SS1	PCI	2 E1	60	√	√	√	N/A	√	N/A	N/A	N/A
SHD-60A-CT/PCI/ISDN	PCI	2 E1	60	√	√	√	N/A	√	√	N/A	N/A
SHD-60A-CT/PCI/SS7	PCI	2 E1	60	√	√	√	N/A	√	√	√	2
SHD-120A-CT/PCI/SS1	PCI	4 E1	120	√	√	√	N/A	√	N/A	N/A	N/A
SHD-120A-CT/PCI/ISDN	PCI	4 E1	120	√	√	√	N/A	√	√	N/A	N/A
SHD-120A-CT/PCI/SS7	PCI	4 E1	120	√	√	√	N/A	√	√	√	4
SHD-30B-CT/PCI/SS7/FAX	PCI	1 E1	30	√	√	√	24	√	√	√	1
SHD-60B-CT/PCI/SS7/FAX	PCI	2 E1	60	√	√	√	16	√	√	√	2
SHD-30C-CT/PCI	PCI	1 E1/T1	30/23	√	√	√	N/A	√(E1 Only)	√	√(E1 Only)	1
SHD-30C-CT/PCI/FAX	PCI	1 E1/T1	30/23	√	√	√	24	√(E1 Only)	√	√(E1 Only)	1
SHD-60C-CT/PCI	PCI	2 E1/T1	60/46	√	√	√	N/A	√(E1 Only)	√	√(E1 Only)	2
SHD-60C-CT/PCI/ FAX	PCI	2 E1/T1	60/46	√	√	√	16	√(E1 Only)	√	√(E1 Only)	2
SHD-30D-CT/PCI	PCI	1 E1/T1/J1	30/23	√	√	√	N/A	√	√	√	1
SHD-60D-CT/PCI	PCI	2 E1/T1/J1	60/46	√	√	√	N/A	√	√	√	2

SHD-120D-CT/PCI	PCI	4 E1/T1/J1	120/92	√	√	√	N/A	√	√	√	4
SHD-120D-CT/PCI/EC	PCI	4 E1/T1/J1	120/92	√	√	√	N/A	√	√	√	4
SHD-120D-CT/PCI/CAS	PCI	8 E1/T1	120/92	√	√	√	N/A	√	N/A	N/A	N/A
SHD-240D-CT/PCI	PCI	8 E1/T1/J1	240/184	√	√	√	N/A	√	√	√	8
SHD-240D-CT/PCI/EC	PCI	8 E1/T1/J1	240/184	√	√	√	N/A	√	√	√	8
SHD-240D-CT/PCI/CAS	PCI	8 E1/T1	240/184	√	√	√	N/A	√	N/A	N/A	N/A
SHD-30E-CT/PCI(SSW)	PCI(SSW)	1 E1/T1/J1	30/23	√	√	√	N/A	√	√	√	1
SHD-30E-CT/PCI/FAX(SSW)	PCI(SSW)	1 E1/T1/J1	30/23	√	√	√	32	√	√	√	1
SHD-30E-CT/PCI/EC(SSW)	PCI(SSW)	1 E1/T1/J1	30/23	√	√	√	N/A	√	√	√	1
SHD-60E-CT/PCI(SSW)	PCI(SSW)	2 E1/T1/J1	60/46	√	√	√	N/A	√	√	√	2
SHD-60E-CT/PCI/FAX(SSW)	PCI(SSW)	2 E1/T1/J1	60/46	√	√	√	64	√	√	√	2
SHD-60E-CT/PCI/EC(SSW)	PCI(SSW)	2 E1/T1/J1	60/46	√	√	√	N/A	√	√	√	2
SHD-120E-CT/PCI(SSW)	PCI(SSW)	4 E1/T1/J1	120/92	√	√	√	N/A	√	√	√	4
SHD-120E-CT/PCI/FAX(SSW)	PCI(SSW)	4 E1/T1/J1	120/92	√	√	√	64	√	√	√	4
SHD-120E-CT/PCI/EC(SSW)	PCI(SSW)	4 E1/T1/J1	120/82	√	√	√	N/A	√	√	√	4
SHD-240E-CT/PCI(SSW)	PCI(SSW)	8 E1/T1/J1	240/184	√	√	√	N/A	√	√	√	8
SHD-240E-CT/PCI/FAX(SSW)	PCI(SSW)	8 E1/T1/J1	240/184	√	√	√	64	√	√	√	8
SHD-240E-CT/PCI/EC(SSW)	PCI(SSW)	8 E1/T1/J1	240/184	√	√	√	N/A	√	√	√	8
SHD-30E-CT/PCIe	PCIe	1 E1/T1/J1	30/23	√	√	√	N/A	√	√	√	1
SHD-30E-CT/PCIe/FAX	PCIe	1 E1/T1/J1	30/23	√	√	√	32	√	√	√	1
SHD-30E-CT/PCIe/EC	PCIe	1 E1/T1/J1	30/23	√	√	√	N/A	√	√	√	1
SHD-60E-CT/PCIe	PCIe	2 E1/T1/J1	60/46	√	√	√	N/A	√	√	√	2
SHD-60E-CT/PCIe/FAX	PCIe	2 E1/T1/J1	60/46	√	√	√	64	√	√	√	2
SHD-60E-CT/PCIe/EC	PCIe	2 E1/T1/J1	60/46	√	√	√	N/A	√	√	√	2
SHD-120E-CT/PCIe	PCIe	4 E1/T1/J1	120/92	√	√	√	N/A	√	√	√	4
SHD-120E-CT/PCIe/FAX	PCIe	4 E1/T1/J1	120/92	√	√	√	64	√	√	√	4
SHD-120E-CT/PCIe/EC	PCIe	4 E1/T1/J1	120/82	√	√	√	N/A	√	√	√	4
SHD-240E-CT/PCIe	PCIe	8 E1/T1/J1	240/184	√	√	√	N/A	√	√	√	8
SHD-240E-CT/PCIe/FAX	PCIe	8 E1/T1/J1	240/184	√	√	√	64	√	√	√	8
SHD-240E-CT/PCIe/EC	PCIe	8 E1/T1/J1	240/184	√	√	√	N/A	√	√	√	8
SHD-240E-CT/PCIe/VAR	PCIe	8 E1/T1/J1	240/184	√	√	√	N/A	√	√	√	8
SHD-30A-CT/cPCI/SS7	cPCI	1 E1	30	√	√	√	N/A	√	√	√	1
SHD-60A-CT/cPCI/SS7	cPCI	2 E1	60	√	√	√	N/A	√	√	√	2
SHD-120A-CT/cPCI/SS7	cPCI	4 E1	120	√	√	√	N/A	√	√	√	4
SHD-240A-CT/cPCI	cPCI	8 E1	240	√	√	N/A	N/A	N/A	√	√	8
SHD-480A-CT/cPCI	cPCI	16 E1	480	√	√	N/A	N/A	N/A	√	√	16
SHD-30B-CT/cPCI/SS7/FAX	cPCI	1 E1	30	√	√	√	24	√	√	√	1
SHD-60B-CT/cPCI/SS7/FAX	cPCI	2 E1	60	√	√	√	16	√	√	√	2
SHD-240S-CT/cPCI	cPCI	8 E1	240	√	N/A	N/A	N/A	N/A	√	√	8
SHD-480S-CT/cPCI	cPCI	16 E1	480	√	N/A	N/A	N/A	N/A	√	√	16

Note: All SHD Series boards have the capability of exchanging data via TDM bus.

1.16.2 Operation Principle of SHD Series

The figure below shows the operation principle of the SHD Series boards.



Below is description on components and symbols related to the recording operation illustrated above:

Name	Description
M1	Outbound voice mixer.
K5-3 [1]	The switch that controls the signal output of Tone Generator and DTMF Generator, often being off under the automatic control of the driver. Only when the function SsmTxDtmf or SsmSendTone/SsmSendToneEx is called to start the DTMF generator or the tone generator does the driver switch on K5-3 and switch off it again after the operation concerned is complete.
M2	Off-bus mixer.
A4-1 … A4-6	The volume adjuster which is used for off-bus signals before they are sent to M2. Use the function SsmSetListenVlmInConf to set the volume, with the default value of 0DB. Or use the following configuration item or functions to set the volume for a channel with data to be off the bus. ✧ SsmTalkWithEx ✧ SsmListenToEx/SsmLinkFromEx/SsmLinkFromAllCh
K5-2 [1]	The switch that controls output signals from the off-bus mixer, automatically controlled by the driver. The driver will automatically switch off K5-2 when off-bus operation functions are called by the application program, and then switch it on again when the input to M2 is thoroughly cut off. See the TDM Capability section in this chapter for more information.
M3	Recording mixer. The signals from M3 are sent to the Encoder component for recording purpose.
A3	The volume adjuster which is used for incoming signals before they are sent to M3. The volume can be set via the function SsmSetRecVolume or the configuration item DefaultRecordVolume, with the default value of 0DB.
A2	The volume adjuster which is used for outgoing signals before they are sent to M3. By default, it is on. The volume can be set via the function SsmSetRecMixer or the configuration item DefaultRecordMixerVolume, with the default value of -7.
M4	Outbound voice mixer for recording. Mixes signals from the off-bus mixer, data being played on a channel and signals from the tone generator/DTMF generator, and then puts them onto M3.
K6-1 K6-2	Switch K6-1 controls whether to put signals form M2 into M3; Switch K6-2 controls whether to put voice data being played into M3. Both can be set via the function SsmSetRecBack.
M5	Onto-bus mixer. Mixes data being played on a channel, incoming signals and those from the off-bus mixer M2, and then puts them onto TDM bus. Upon board initialization, the output from M5 is transported to a time slot (marked as 'ts' for short) on TDM bus. You can use the function SsmLinkToBus to designate the ts.

K3	The switch that controls whether to put incoming signals into M5 or not. By default, it is on. It can be set via the function SsmSetFlag (with the parameter F_InVoiceToBus) or the configuration item InVoiceToBus.
K1-1 [1]	The switch that controls whether to put data being played on a channel first into M5 and then onto TDM bus or not, usually being off. It can be set via the function SsmSetPlayDest, used only for playing background music to a teleconference.
K1-2	The switch that controls whether to put signals from M2 back onto TDM bus. By default, it is off. It can be set via the function SsmSetFlag (with the parameter F_MixerResToBus).
	Below is the description on components and symbols concerning the playback operation
K7	The Voice Play/Fsk Transmitter switch, automatically controlled by the driver. When the application calls the playing function, K7 connects to the component Decoder; when it calls the function SsmStartSendFSK to start FSK data transmission, K7 connects to the component FSK Generator. For more information about the playing functions, refer to the Voice Playing section.
A1	The volume adjuster for voice playing. The volume can be set via the configuration item DefaultPlayVolume or the function SsmSetPlayVolume/SsmSetPlayGain, with the default value of 0.
K5-1 [1]	The switch that controls the voice playing operation, automatically controlled by the driver. Only when the application program invokes the playing function does the driver automatically switch on K5-1 and then switch it off at the end of voice playing. For more information about the playing functions, refer to the Voice Playing section.
K2	The switch that shifts between the FSK detector and other detectors, usually connected to other detectors unless the FSK detector is started.

Note[1]:

- **For some board models in SHD Series, due to the limited on-board DSP resources, only one of K5-1, K5-2, K5-3 can be switched on when K1-1 is off. As a result, in case a channel is playing voices (K5-1 is already on at that time) or starts the tone generator (K5-3 is already on at that time), and the application program invokes the function for bus operation which causes K5-2 to be off, if the configuration item LinkFromStopPlayAndTone is set to 1, the playing operation will be stopped immediately and K5-2 will be switched on; if the configuration item LinkFromStopPlayAndTone is set to 0, K5-2 will get on only after the playing operation ends up in a normal way. The bus operation functions which cause K5-2 to get off include:**

  - SsmTalkWith / SsmTalkWithEx
  - SsmListenTo / SsmListenToEx / SsmLinkFrom / SsmLinkFromEx / SsmLinkFromAllCh
  - SsmLinkFromBus / SsmLinkFromBusEx

  Involved board models are:

  - SHD-30A-CT/PCI/SS1
  - SHD-30A-CT/PCI/ISDN
  - SHD-30A-CT/PCI/SS7
  - SHD-30B-CT/PCI/SS7/FAX
  - SHD-60A-CT/PCI/SS1
  - SHD-60A-CT/PCI/ISDN
  - SHD-60A-CT/PCI/SS7
  - SHD-60B-CT/PCI/SS7/FAX
  - SHD-120A-CT/PCI/SS1
  - SHD-120A-CT/PCI/ISDN
  - SHD-120A-CT/PCI/SS7
  - SHD-30A-CT/cPCI/SS7
  - SHD-30B-CT/cPCI/SS7/FAX
  - SHD-60A-CT/cPCI/SS7
  - SHD-60B-CT/cPCI/SS7/FAX
  - SHD-120A-CT/cPCI/SS7
  - SHD-30C-CT/PCI
  - SHD-60C-CT/PCI

- SHD-30C-CT/PCI/FAX
- SHD-60C-CT/PCI/FAX

- **If K1-1 is on, K5-1 and K5-2 are allowed to be switched on at the same time but K5-3 must keep off.**

➢ **Voice Playing**

After the application invokes the playing function, the driver will automatically have K7 linked to Decoder and switch on K5-1. The voice data processed by A1 will:

- ✧ Enter M1 and form the final outgoing signals with other signal sources; or
- ✧ Enter M4 under the control of K6-2 and form the recording signal sources with other signals; or
- ✧ Enter M5 under the control of K1-1 and form the onto-bus signal sources with other signals, only for playing background music to the teleconference.

➢ **Recording Example for Various Signal Sources**

A lot of special applications are available via the flexible use of the above switches for recording control and volume adjusters. In the following examples, ch1 and ch2 respectively mean Channel 1 and Channel 2.

✧ Example 1: Only record the incoming signals on ch1 and use the default volume.

```
SsmSetRecVolume(ch1, 0);          //start A3, set the volume gain to 0DB
SsmSetRecMixer(ch1, FALSE, 0);    //stop A2
SsmRecToFile(ch1, ……);             //start the task of file recording …
```

✧ Example 2: Record the incoming signals on ch1 (volume increased by 6DB) and the signals output from M2 (volume decreased by 3DB) at one time based on the establishement of a two-way call between ch1 and ch2.

```
SsmTalkWith(ch1, ch2);            //establish a two-way connection between ch1 and ch2, switch on K5-2
SsmSetRecBack(ch1,2);             //switch on K6-1, switch off K6-2
SsmSetRecVolume(ch1, 2);          //start A3, set the volume gain to 2*3DB=6DB
SsmSetRecMixer(ch1, TRUE, -1);    //start A2, set the volume gain to -1*3DB=-3DB
SsmRecToFile(ch1, ……);             //start the task of file recording …
```

✧ Example 3: When ch1 joins a teleconference (assuming the conference room number is n) as the organizer, play background music on ch1 and record the conference (including the background music) through ch1. All signal sources go with normal volume, i.e. the gain is set to 0.

```
SsmJoinConfGroup(n, ch1, ……);     //ch1 joins the conference: switch on K5-2 and K3
SsmSetPlayDest(int ch, 1);        //switch on K1-1
SsmSetRecBack(ch1,1);             //switch on K6-1 and K6-2
SsmPlayFile(ch1, ……);              //start the task of voice playing on ch1: link K7 to Decoder, switch on K5-1
SsmSetRecVolume(ch1, 0);          //start A3, set the volume gain to 0DB
SsmSetRecMixer(ch1, TRUE, 0);     //start A2, set the volume gain to 0DB
SsmRecToFile(ch1, ……);             //start the task of file recording …
```

1.16.3 Number-receiving Rule for Incoming Call

Incoming calls are automatically handled via the channel state machine. What the application need do is preset the rule to receive numbers. The number-receiving rule means the rule followed by the local end to receive Caller ID, Callee ID and other information from the remote PBX.

As to the number-receiving rule, there are two modes available which can be set by the configuration item

DefaultRcvPhoNumLen.

- ✧ Fixed-length Mode: The local end has only one phone number of fixed length (e.g. '110'). When the driver fully receives the preset length of the called party number, the number-receiving process is considered to be over. The configuration items DefaultRcvCallerID and RcvPhoNumCfgLen are used to set particular rules in this mode.
- ✧ Prefix Mode: You can set several called party numbers for the application program at the local end, each of which begins with a prefix. The configuration items MaxPhoNumRule and Rule are used to set particular rules in this mode.

You are allowed to set independent number-receiving rules respectively for TUP channel, ISUP channel, ISDN channel and SS1 channel.

- ✧ The rule for TUP channel is set in the [TUP] section;
- ✧ The rule for ISUP channel is set in the [ISUP] section;
- ✧ The rule for ISDN channel is set in the [ISDN] section;
- ✧ The rule for SS1 channel is set in the [SS1] section.

1.16.4 SS7 Programming

1.16.4.1 OPC & DPC

Signaling Point (SP) is a node in a signaling network that originates and receives signaling messages, or transfers signaling messages from one signaling link to another, or both. CCITT suggests the use of a signaling point code (SPC) with a unique 14-bit format used at the international level for signaling message routing and identification of signaling points involved, i.e. International Signaling Point Code (ISPC). See the table below for the grouping of ISPC.

3bit	8bit	3bit
ZONE	AREA	POINT

The format of the 24-bit binary code is used for the identification of national signaling points in China (i.e. National Signaling Point Code (NSPC) Format in China). NSPC is grouped as follows.

8bit	8bit	8bit
MAIN AREA	SUBAREA	POINT

The Synway SHD Series boards support both International Signaling Point Code (ISPC) Format and National Signaling Point Code (NSPC) Format in China.

The specific signaling point code (OPC-Originating Point Code or DPC- Destination Point Code) is configured in the file Ss7Server.ini.

1.16.4.2 Time Slot Allocation

The PCM time slots on the SHD Series boards are allocated as shown in the following table.

Board Model	Sync. Time Slot	Voice Time Slot	Signaling Time Slot		
			SS1	ISDN PRI	SS7
SHD-30A-CT/PCI/SS1	0	1-15,17-31	16	N/A	N/A
SHD-30A-CT/PCI/ISDN	0	1-15,17-31	16	16	N/A
SHD-30A-CT/PCI/SS7	0	1-15,17-31	16	16	16[1]
		2-31	-	-	1[1]
SHD-60A-CT/PCI/SS1	0	1-15,17-31	16	N/A	N/A
SHD-60A-CT/PCI/ISDN	0	1-15,17-31	16	16	N/A

SHD-60A-CT/PCI/SS7	0	1-15,17-31	16	16	16[1]
		2-31	-	-	1[1]
SHD-120A-CT/PCI/SS1	0	1-15,17-31	16	N/A	N/A
SHD-120A-CT/PCI/ISDN	0	1-15,17-31	16	16	N/A
SHD-120A-CT/PCI/SS7	0	1-15,17-31	16	16	16[1]
SHD-30B-CT/PCI/SS7/FAX	0	1-15,17-31	16	16	16[1]
		2-31	-	-	1[1]
SHD-60B-CT/PCI/SS7/FAX	0	1-15,17-31	16	16	16[1]
		2-31	-	-	1[1]
SHD-30C-CT/PCI	0(E1) N/A(T1)	1-15,17-31(E1) 0-22(T1 ISDN)	16(E1)	16(E1) 23(T1)	16[1](E1)
SHD-30C-CT/PCI/FAX	0(E1) N/A(T1)	1-15,17-31(E1) 0-22(T1 ISDN)	16(E1)	16(E1) 23(T1)	16[1](E1)
SHD-60C-CT/PCI	0(E1) N/A(T1)	1-15,17-31(E1) 0-22(T1 ISDN)	16(E1)	16(E1) 23(T1)	16[1](E1)
SHD-60C-CT/PCI/ FAX	0(E1) N/A(T1)	1-15,17-31(E1) 0-22(T1 ISDN)	16(E1)	16(E1) 23(T1)	16[1](E1)
SHD-30D-CT/PCI	0	1-15,17-31	16	16	16[1]
SHD-60D-CT/PCI	0	1-15,17-31	16	16	16[1]
SHD-120D-CT/PCI	0	1-15,17-31	16	16	16[1]
SHD-120D-CT/PCI/EC	0	1-15,17-31	16	16	16[1]
SHD-120D-CT/PCI/CAS	0	1-15,17-31	16	-	-
SHD-240D-CT/PCI	0	1-15,17-31	16	16	16[1]
SHD-240D-CT/PCI/EC	0	1-15,17-31	16	16	16[1]
SHD-240D-CT/PCI/CAS	0	1-15,17-31	16	-	-
SHD-30E-CT/PCI(SSW)	0	1-15,17-31	16	16	16[1]
SHD-30E-CT/PCI/FAX(SSW)	0	1-15,17-31	16	16	16[1]
SHD-30E-CT/PCI/EC(SSW)	0	1-15,17-31	16	16	16[1]
SHD-60E-CT/PCI(SSW)	0	1-15,17-31	16	16	16[1]
SHD-60E-CT/PCI/FAX(SSW)	0	1-15,17-31	16	16	16[1]
SHD-60E-CT/PCI/EC(SSW)	0	1-15,17-31	16	16	16[1]
SHD-120E-CT/PCI(SSW)	0	1-15,17-31	16	16	16[1]
SHD-120E-CT/PCI/FAX(SSW)	0	1-15,17-31	16	16	16[1]
SHD-120E-CT/PCI/EC(SSW)	0	1-15,17-31	16	16	16[1]
SHD-240E-CT/PCI(SSW)	0	1-15,17-31	16	16	16[1]
SHD-240E-CT/PCI/FAX(SSW)	0	1-15,17-31	16	16	16[1]
SHD-240E-CT/PCI/EC(SSW)	0	1-15,17-31	16	16	16[1]
SHD-30E-CT/PCIe	0	1-15,17-31	16	16	16[1]
SHD-30E-CT/PCIe/FAX	0	1-15,17-31	16	16	16[1]
SHD-30E-CT/PCIe/EC	0	1-15,17-31	16	16	16[1]
SHD-60E-CT/PCIe	0	1-15,17-31	16	16	16[1]
SHD-60E-CT/PCIe/FAX	0	1-15,17-31	16	16	16[1]
SHD-60E-CT/PCIe/EC	0	1-15,17-31	16	16	16[1]
SHD-120E-CT/PCIe	0	1-15,17-31	16	16	16[1]
SHD-120E-CT/PCIe/FAX	0	1-15,17-31	16	16	16[1]
SHD-120E-CT/PCIe/EC	0	1-15,17-31	16	16	16[1]
SHD-240E-CT/PCIe	0	1-15,17-31	16	16	16[1]
SHD-240E-CT/PCIe/FAX	0	1-15,17-31	16	16	16[1]
SHD-240E-CT/PCIe/EC	0	1-15,17-31	16	16	16[1]
SHD-240E-CT/PCIe/VAR	0	1-15,17-31	16	16	16[1]
SHD-30A-CT/cPCI/SS7	0	1-15,17-31	16	16	16[1]

		2-31	-	-	1[1]
SHD-60A-CT/cPCI/SS7	0	1-15,17-31	16	16	16[1]
		2-31	-	-	1[1]
SHD-120A-CT/cPCI/SS7	0	1-15,17-31	16	16	16[1]
		2-31	-	-	1[1]
SHD-240A-CT/cPCI	0	1-15,17-31	N/A	16	16[1]
SHD-480A-CT/cPCI	0	1-15,17-31	N/A	16	16[1]
SHD-30B-CT/cPCI/SS7/FAX	0	1-15,17-31	16	16	16[1]
		2-31	-	-	1[1]
SHD-60B-CT/cPCI/SS7/FAX	0	1-15,17-31	16	16	16[1]
		2-31	-	-	1[1]
SHD-240S-CT/cPCI	0	1-15,17-31	N/A	16	16[1]
SHD-480S-CT/cPCI	0	1-15,17-31	N/A	16	16[1]

Note [1]: It is the configuration items UseTS16AsCircuit and Ss7SignalingTS that determine if SS7 links are available on the digital trunk and which time slot serves as the signaling link.

1.16.4.3 Setting Spare Address Codes

In TUP and ISUP calls, called party number, calling party number and other information within messages are displayed in address code format instead of ASCII format. However, the driver adopts ASCII characters to present CallerId and CalleeId, so it is essential to establish a rule for translating address codes into unified ASCII characters. As shown in the table below, the configuration item AddressSignal is used to map the spare codes to some particular characters.

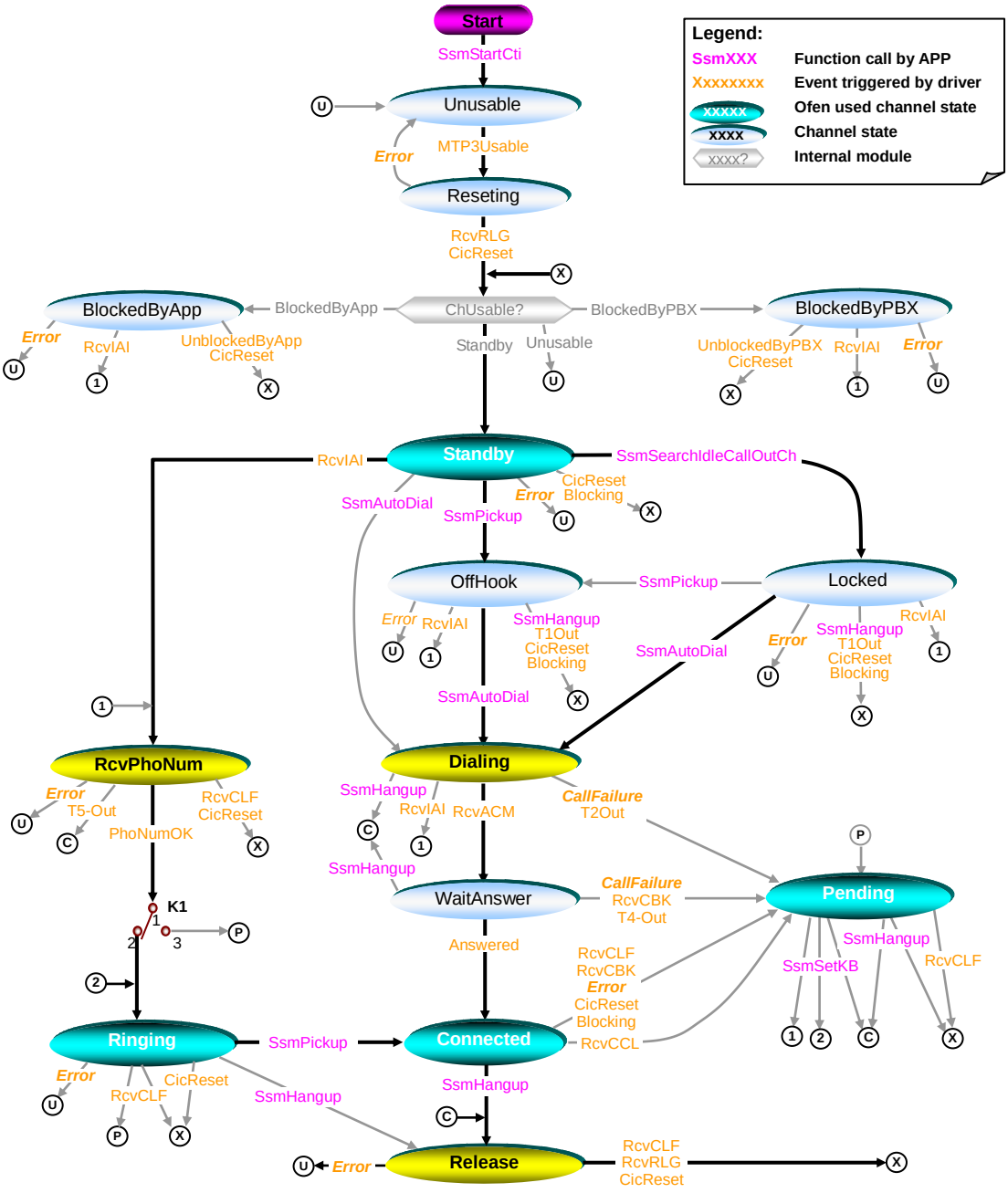
Address Code		Corresponding ASCII Character	Property
Hexadecimal	Decimal		
0	0	'0'	Fixed
1	1	'1'	Fixed
2	2	'2'	Fixed
3	3	'3'	Fixed
4	4	'4'	Fixed
5	5	'5'	Fixed
6	6	'6'	Fixed
7	7	'7'	Fixed
8	8	'8'	Fixed
9	9	'9'	Fixed
a	10	spare	Configurable via the configuration item AddressSignal
b	11	spare	Configurable via the configuration item AddressSignal
c	12	spare	Configurable via the configuration item AddressSignal
d	13	spare	Configurable via the configuration item AddressSignal
e	14	spare	Configurable via the configuration item AddressSignal
f	15	ST signal	Fixed

1.16.4.4 Configuration Instance for SS7 Server

Refer to Chapter 4 Application and Configuration Instance for SS7 Server in this manual for details.

1.16.4.5 TUP Programming and Application

1.16.4.5.1TUP Channel State Machine



The internal events automatically triggered by the driver itself are described as follows.

Event Name	Description
Mtp3Usable	The driver will send this event to all channel state machines after receiving signaling from the SS7 server so as to change the channel state from **Unusable** to **Resetting** and reset the circuit.
CicReset	This event is triggered when the driver receives the circuit reset message (GRA, GRS or RSC) following the successful reset of the circuit.

<table>
<tr><td></td><td>What the driver does: If the local end blocks the remote end or is waiting for the reply after sending the blocking message to the remote end, the driver will send the BLO message to the remote end to keep it blocked and throw out the E_CHG_RemoteChBlock event to the application program at the same time.</td></tr>
<tr><td>Blocking</td><td>This channel is blocked by the local end (Local Blocking) or the remote end (Remote Blocking). This event is triggered in the following two situations.
• Remote Blocking: The driver receives the group blocking message (MGB, SGB, HGB) or the circuit blocking message (BLO) from the remote PBX.
• Local Blocking: The application invokes the function SsmBlockLocalCh or SsmBlockLocalPCM.</td></tr>
<tr><td>UnblockedByPBX</td><td>Remote Unblocked: This event is triggered when the driver receives the MGU, HGU, SGU or UBL message from the remote PBX.</td></tr>
<tr><td>UnblockedByApp</td><td>Local Unblocked: This event is triggered after the application invokes the function SsmUnblockLocalPCM or SsmUnblockLocalCh to unblock the local end.</td></tr>
<tr><td>RcvIAI</td><td>The driver receives the incoming call request message (i.e. IAI or IAM) from the remote PBX. That whether the driver accepts this request or not depends on the channel state at that time.
<table>
<tr><th>Channel State</th><th>Description</th></tr>
<tr><td>BlockedByPBX
BlockedByApp</td><td>When the local end is blocked, the channel does not support outgoing calls and only accept incoming calls.</td></tr>
<tr><td>Dialing</td><td>The reception of the IAI/IAM message from the remote PBX during the outgoing call indicates the ‘collision’ phenomenon occurs. As stipulated in TUP, if the channel’s CIC represents the circuit controlled by the local end, the local end will abandon the IAI/IAM message and continue the outgoing call; otherwise, the local end will quit the outgoing call and process the incoming call.</td></tr>
<tr><td>OffHook
Locked</td><td>The channel is preparing to make an outgoing call but does not send the IAI or IAM message yet. The driver will reply the incoming call request message received at that time from the remote end.</td></tr>
<tr><td>Standby</td><td>The channel accepts the incoming call.</td></tr>
<tr><td>Others</td><td>The channel rejects the incoming call.</td></tr>
</table>
In case the driver accepts the incoming call, it will throw out the E_CHG_RxPhoNumBuf event to the application program after saving the called party number within the IAM message to the buffer and change the channel state to RcvPhoNum.</td></tr>
<tr><td>RcvRLG</td><td>The driver receives the RLG (Release-guard signal) message from the remote PBX.</td></tr>
<tr><td>RcvCLF</td><td>The driver receives the CLF message from the remote PBX. When this event is triggered, if the channel stays in the state:
• RcvPhoNum or Ringing:
it indicates the remote PBX (the calling party) cancels this call during the incoming call establishment.
In case the channel stays in the Ringing state and the configuration item RingToPending is set to 1, the driver will set PendingReason to PEND_RemoteHangupOnRinging and transfer the channel state to Pending; otherwise, the driver will start to disconnect this incoming call.
• S_TUP_WaitCLF (a substate of Release):
it indicates the channel receives the CLF message as expected during the call disconnection, which ends up the disconnection process.
• Pending:
No matter what causes the channel to go into the Pending state, the reception of the CLF message in this state will prompt the driver to automatically send the RLG message and start disconnecting the call.
• Connected:
For the incoming call, the reception of the CLF message in the Connected state means it is the calling party that first hangs up the phone. The driver will set PendingReason to PEND_RemoteHangupOnTalking and then transfer the channel state to Pending.</td></tr>
</table>

	However, it won't send the RLG message.
RcvACM	The driver receives the ACM (Address Complete Message) message from the remote PBX. In the outgoing call progress implementation, the driver will send the IAI/IAM message to the remote PBX when the application calls the function SsmAutoDial to start an outgoing call. Then the remote PBX will send the ACM message to the local end as it believes all necessary information including the Callee ID has been received. The driver responds to this event only when the channel stays in the **Dialing** or **AppSetCallerID** state. After receiving the ACM message, the driver will save it, set the progress value of the AutoDial task to DIAL_ECHOTONE and then transfer the channel state to **WaitAnswer**. The function **SsmGetKB** can be used to acquire the received ACM message.
Answered	This event is triggered when the driver receives the answering signal from the remote PBX. During an outgoing call, when the remote PBX is sending rings to the called party, it will send answering signals to the local end once the called party picks up the phone. The answering signal indicates one of the following messages: ✧ ANU message (answering signal, no indication) ✧ ANC (answering signal, charge) ✧ ANN (answering signal, no charge) The function SsmGetTupFlag can be used to obtain the specific content of the answering signal. This event will lead to the channel state transition from **WaitAnswer** to **Connected**. Before the channel goes into the new state, the driver will automatically ✧ start the DTMF Detector if the configuration item AlwaysEnableRxDtmf is set to 0; ✧ start the Barge-in Detector if the configuration item AlwaysDetectBargeIn is set to 0; ✧ clear the Tone Detector and start it; ✧ set the progress value of the AutoDial task to DIAL_VOICE, end up the task and throw out the event E_PROC_AutoDial to the application program.
RcvCBK	The driver receives the CBK message from the remote PBX. When this event is triggered, if the channel stays in the state • **Connected**: it indicates that the remote user first hangs up the phone in an outgoing call. The driver will set PendingReason to PEND_CalleeHangupOnTalking and transfer the channel state to Pending. • **WaitAnswer**: it indicates that the called party refuses this call and the AutoDial task ends. The driver will ✧ set the progress value of the AutoDial task to DIAL_FAILURE and the failure reason to ATDL_RcvCBK, and throw out the E_PROC_AutoDial event to the application; ✧ set PendingReason to PEND_CalleeHangupOnWaitRemotePickUp and then transfer the channel state to Pending.
RcvCCL	The driver receives the CCL message from the remote PBX, which indicates that the remote user hangs up first in an incoming call after two parties are connected. After receiving the RcvCCL event, the driver will set PendingReason to PEND_RemoteHangupOnTalking and then transfer the channel state to **Pending**. If the application invokes the function SsmSetCalleeHoldFlag to enable the 'Caller Holding' feature when the channel stays in the **Connected** state, the driver will automatically send the OPR message to the remote PBX and keep the channel in **Connected**.
T1Out	The timer T1 overflows. T1 is set to 60sec and will be automatically started when the channel goes into the Locked or OffHook state.
T2Out	The timer T2 overflows. T2 is set to 30sec and will be automatically started when the channel goes into the **Dialing** state. After T2 overflows, the driver will ✧ set the progress value of the AutoDial task to DIAL_FAILURE and the failure reason to ATDL_WaitDialAnsTimeout, and throw out the E_PROC_AutoDial event to the application;

	✧ set PendingReason to PEND_AutoDialFailed and then transfer the channel state to Pending.
T4Out	The timer T4 overflows. T4 is set by the configuration item MaxWaitAutoDialAnswerTime and will be automatically started when the channel goes into the **WaitAnswer** state. After T4 overflows, the driver will ✧ set the progress value of the AutoDial task to DIAL_NOANSWER and the failure reason to ATDL_WaitRemotePickupTimeout, and throw out the E_PROC_AutoDial event to the application; ✧ set PendingReason to PEND_AutoDialFailed and then transfer the channel state to Pending.
T5Out	The timer T5 overflows. T5 is set to 20sec and will be automatically started when the channel goes into **S_TUP_WaitPrefix** or **S_TUP_WaitSAM** (a substate of **RcvPhoNum**). After T5 overflows, the driver will automatically send the ADI (Address Incomplete) message to the remote PBX and then change the channel state to **Release**.
PhoNumOK	This event will be triggered when the driver ✧ receives all necessary information including the called party number according to the preset number-receiving rule; ✧ stops waiting for the GSM message from the remote PBX as the channel stays in **_TUP_WaitGSM** (a substate of **RcvPhoNum**) and the timer with the preset value of 5sec overflows; ✧ receives the GSM message from the remote PBX when the channel stays in **_TUP_WaitGSM** (a substate of **RcvPhoNum**). After this event is triggered, if the application enables the 'Auto-answer of Incoming Call' feature via the function SsmEnableAutoSendKB or the configuration item AutoSendACM, the driver will automatically send the ACM message to the remote PBX and transfer the channel state to **Ringing**. The configuration item DefaultACM is used to specify the message indicator field which shows the called party state in the ACM message automatically sent by the driver. The configuration item RcvPhoNumCfgLen is used to set the method for the driver to save the called party number. The function SsmSetKB is for setting the charge indicator in the backward call indicator field of the ACM message. If the 'Auto-answer of Incoming Call' feature is not enabled, the driver will set PendingReason to PEND_WaitBckStpMsg, transfer the channel state to **Suspend**, and let the application determine whether to accept this call or not.
Error	This event is triggered when the driver detects one of the following situations. ➢ The synchronization signal through TS0 on the digital trunk where this channel stays gets lost (e.g. line disconnected, clock configuration error and so on). ➢ The TCP/IP connection between the TUP stack and the SS7 server is broken. ➢ The signaling unusable message is received from the SS7 server. When this event is triggered, the driver's operation is in relation to the channel state. See below for details. ➢ In the state **Dialing**, **AppSetCallerID** or **WaitAnswer**: The driver will set the progress value of the AutoDial task to DIAL_FAILURE and the failure reason to ATDL_PcmSyncLos or ATDL_Mtp3Unusable, and throw out the E_PROC_AutoDial event to the application; ➢ In the state **Dialing**, **AppSetCallerID**, **WaitAnswer**, **Connected** or **Pending**: The driver will set PendingReason to PEND_PcmSyncLos or PEND_SsxUnusable, and then transfer the channel state to Pending.
CallFailure	If the call fails during the outgoing process, the driver triggers this event. The outgoing call fails in the following situations. ➢ The driver receives the call failure message from the remote PBX. The call failure message includes: ● The CFL message (call failed). The driver will ✧ set the result of the AutoDial task to DIAL_FAILURE and the failure reason to ATDL_RcvCFL, and throw out the E_PROC_AutoDial event to the application; ✧ transfer the channel state to Pending after setting PendingReason: if the channel

	stays in the WaitAnswer state, PendingReason is set to PEND_CalleeHangupOnWaitRemotePickUp; if the channel stays in the Dialing state, PendingReason is set to PEND_AutoDialFailed. ● The SSB, SLB, STB (Called Subscriber Busy) message. The driver will ✧ set the result of the AutoDial task to DIAL_BUSYTONE and the failure reason to ATDL_RcvSSB, ATDL_RcvSLB or ATDL_RcvSTB, and throw out the E_PROC_AutoDial event to the application; ✧ set PendingReason to PEND_AutoDialFailed, and then transfer the channel state to Pending. ● The UNN message (Callee ID unallocated). The driver will ✧ set the result of the AutoDial task to DIAL_INVALID_PHONUM and the failure reason to ATDL_RcvUNN, and throw out the E_PROC_AutoDial event to the application; ✧ set PendingReason to PEND_AutoDialFailed, and then transfer the channel state to Pending. ● The SEC, CGC, NNC, LOS, SST, ACB, DPN, EUM or ADI message (other errors in call). The driver will ✧ set the result of the AutoDial task to DIAL_FAILURE and the failure reason respectively to ATDL_RcvSEC, ATDL_RcvCGC, ATDL_RcvNNC, ATDL_RcvLOS, ATDL_RcvSST, ATDL_RcvACB, ATDL_RcvDPN, ATDL_RcvEUM, ATDL_RcvADI, and throw out the E_PROC_AutoDial event to the application; ✧ set PendingReason to PEND_AutoDialFailed, and then transfer the channel state to Pending. a) The CicReset event is triggered. b) The Blocking event is triggered. c) The Error event is triggered.

Each state identifier in the above diagram is described in the following table.

State Identifier	Description
Unusable	The channel is unusable.
Reseting	The circuit is being reset.
BlockedByPBX	Remote Blocking: When the remote PBX is stopped for maintenance, it will send the blocking message to the local end to prevent new calls. When the channel stays in such state, the driver will refuse the Outgoing Call instruction given by the application program and only accept and process incoming calls.
BlockedByApp	Local Blocking: The application program can not make an outgoing call in this state. It is often used for system maintenance. The application can block the channel via the function call of SsmBlockLocalCh or SsmBlockLocalPCM so as to inform the driver to prevent starting the outgoing call. **Note:** The channel in such state still can accept and process incoming calls.
Standby	Idle: The channel should go into this state provided it is normally reset after the successful driver initialization. The application can invoke the function SsmAutoDial to start an outgoing call in this state. The driver will automatically clear the Caller ID buffer when the channel state transfers to Idle at the end of the call, provided that the configuration item AutoClearCallerIdBufOnHangup is set to 1.
OffHook	Off-hook: The application can invoke the function SsmAutoDial to start an outgoing call in this state. The driver will start the timer T1 when the channel goes into the OffHook state. If the application does not call the function SsmAutoDial before T1 overflows, the driver will reset the channel back to the idle state.
Locked	Outgoing Call Locked: The application can invoke the function SsmAutoDial to start an outgoing call in this state. The driver will start the timer T1 when the channel goes into this state. If the application does not call the function SsmPickup or SsmAutoDial before T1 overflows, the driver will reset the channel back to the idle state.
Pending	It is the pending state.

RcvPhoNum	It is an internal state during an incoming call, including three substates: **S_TUP_WaitSAM**, **S_TUP_WaitPrefix** and **S_TUP_WaitGSM**. The driver judges the received called party number according to the preset number-receiving rule. If the number complies with the rule, the channel will transfer to the subsequent state. For more information about the number-receiving rule, refer to the section Number-receiving Rule for Incoming Call in this chapter. If the called party number involved in the IAM/IAI message from the remote PBX proves incomplete, the channel will stay in the **S_TUP_WaitSAM** substate. Every time when the driver receives the SAO or SAM message from the remote PBX, it will save the message to its internal buffer and throw out the E_CHG_RxPhoNumBuf event to the application. After the driver receives the complete called party number according to the number-receiving rule, if it is required to receive the calling party information (such as Caller ID, Calling party category, etc.) which is not yet supplied by the remote PBX, the driver will send the GRQ message to the remote PBX and transfer the channel state to **S_TUP_WaitGSM**. The configuration item ReqTypeIndicators is used to customize the Request Type Indicator field in the GRQ message sent to the remote PBX. If the called party number received by the channel does not conform to the number-receiving rule, the driver will automatically send the REL message (carrying the unallocated-number reason) and transfer the channel state to **Release**. In the **RcvPhoNum** state, the application can ✧ invoke the function SsmHangup/SsmHangupEx to refuse this call. Then the driver will send the REL message. The configuration item DefaultHangupRELInd or the function SsmSetIsupFlag (with the parameter ISUP_REL_DENY_SetToOther) can be used to set the reason why the call fails written in the REL message.
Ringing	The local end is ringing. The application program can use the function SsmPickup to accept the call and the function SsmHangup to refuse the call.
Dialing	It is an internal state during the incoming call, including the substates **S_TUP_WaitDialAnswer** and **S_TUP_WaitSetCallerID**. The function call of SsmAutoDial or SsmAutoDialEx by the application will prompt the driver to send the IAM message to the remote PBX and transfer the channel state to **S_TUP_WaitDialAnswer**. In the **S_TUP_WaitDialAnswer** substate, if the remote PBX wants to receive the calling party information which is not contained in the IAM/IAI message, it will send the GRQ message to the local end. After the driver receives the GRQ message, ➢ in case the Caller ID buffer is empty during an outgoing call on the channel while the received GRQ message shows the remote PBX is asking the local end for the calling party number, ✧ the driver will automatically send the GSM message provided the configuration item AutoSendGSM is set to 1; ✧ the driver will bring the channel into the **S_TUP_WaitSetCallerID** state to wait for the application to set the calling party number and the like, and start a timer with the preset value of 2sec at the same time, provided the configuration item AutoSendGSM is set to 0. If the timer overflows before the application invokes the function SsmSetTxCallerId, the driver will transfer the channel state back to **S_TUP_WaitDialAnswer**. ➢ otherwise, the driver will take the string in the Caller ID buffer as the calling party number and send the GSM message automatically.
WaitAnswer	The channel is waiting for the called party to pick up the phone. This state is only applicable to the outgoing call. The called party should be able to hear ringing tones in such state, and if at the local end is the station channel that makes an outgoing call through the digital trunk channel it should send the ringback tone to the station channel.
Connected	Both parties are connected and able to talk with each other. When the channel state transfers to **Connected**, the driver will

	• start the DTMF Detector if the configuration item AlwaysEnableRxDtmf is set to 0; • start the Barge-in Detector if the configuration item AlwaysDetectBargeIn is set to 0; • clear the Tone Detector and start it.
Release	It is the disconnection state, including the substates **S_TUP_WaitCLF** and **S_TUP_WaitRLG**.

All state values and corresponding macro definitions can be found in the file ShpA3Api.h.

Each state identifier in the state machine is declared in the header file ShpA3Api.h as follows.

State Identifier	Value in ShpA3Api.h	Macro Definition in ShpA3Api.h
Standby	0	S_CALL_STANDBY
OffHook	1	S_CALL_PICKUPED
Ringing	2	S_CALL_RINGING
Connected	3	S_CALL_TALKING
Pending	7	S_CALL_PENDING
WaitAnswer	9	S_CALL_WAIT_REMOTE_PICKUP
BlockedByPBX	10	S_CALL_RemoteBlock
Unusable	11	S_CALL_UNAVAILABLE
Locked	12	S_CALL_LOCKED
BlockedByApp	20	S_CALL_LocalBlock
Reseting	70	S_TUP_WaitPcmReset
RcvPhoNum	71	S_TUP_WaitSAM
	72	S_TUP_WaitGSM
	74	S_TUP_WaitPrefix
Dialing	75	S_TUP_WaitDialAnswer
	77	S_TUP_WaitSetCallerID
Release	73	S_TUP_WaitCLF
	76	S_TUP_WaitRLG

All state values and corresponding macro definitions in the table above can be found in the file ShpA3Api.h.

In the above state machine, ChUsable is the driver's internal component to judge the channel's usability. It will output values as shown in the table below.

Output Value	Description
Unusable	The channel is unusable if the Error event keeps effective.
BlockedByApp	The event Blocking keeps effective and the channel is blocked by the application program.
BlockedByPBX	The Blocking event keeps effective and the channel is blocked by the remote PBX.
Standby	The Error event stays invalid.

Note:

At this stage, the driver also performs the following operations.

- ✧ Clear the tone detector and automatically close it;
- ✧ Automatically stop the WaitDtmf task if it has been started by the application;
- ✧ Automatically stop the task of DTMF transmission if it has been started by the application;
- ✧ Automatically clear the buffer of the DTMF detector, and close the DTMF detector if the configuration item AlwaysEnableRxDtmf is set to 0;
- ✧ Automatically close the Barge-in detector if the configuration item AlwaysDetectBargeIn is set to 0;
- ✧ Automatically clear the Caller ID buffer if the configuration item AutoClearCallerIdBufOnHangup is set to 1.

1.16.4.5.2Channel Blocked and Unblocked

1.16.4.5.2.1 Basic Concepts

(1) Channel Blocked

After a channel is blocked, it is forbidden to make outgoing calls but still able to accept incoming calls in a normal way.

(2) Local Blocking

The local application program invokes functions to block a stated local channel.

(3) Remote Blocking

The remote PBX sends specific circuit/circuit group blocking messages to block a local channel/PCM. The blocking messages sent by the remote PBX are automatically processed by the driver.

(4) Remote Blocked

The local application sends particular circuit/circuit group blocking messages to block the remote PBX.

1.16.4.5.2.2 Usage

This feature is only applicable to the digital trunk channel. When the application program is required to stop the system for maintenance, the 4 steps below must be followed to ensure nomal exit.

(1) Block all local channels to stop new outgoing calls.

(2) Block all remote channels to stop new outgoing calls.

(3) Wait for the current call progress to end up.

(4) Exit the program.

When a channel is blocked, the driver will not process new outgoing calls any more (i.e. all function calls related to the auto outgoing call progress will fail).

1.16.4.5.2.3 Blocking Ways

(1) A channel can be blocked by receiving blocking messages from the remote PBX. In TUP, the channel may be blocked as it receives the circuit blocking message (BLO) or the circuit group blocking message (SGB, HGB or MGB) from the remote PBX; or

(2) The application program can invoke the function SsmBlockLocalCh() to block the local channel and/or the function SsmBlockLocalPCM() to block the local PCM.

1.16.4.5.2.4 Function List

The functions used for blocking channels are listed in the following table.

Blocking Mode		Function Name	Output Event
Local Blocked	Channel Blocked	SsmBlockLocalCh	
		SsmUnblockLocalCh	
		SsmQueryLocalChBlockState	
	Circuit Group Blocked	SsmBlockLocalPCM	
		SsmUnblockLocalPCM	
		SsmQueryLocalPCMBlockState	
Remote Blocked	Channel Blocked	SsmQueryOpBlockRemoteCh	
		SsmBlockRemoteCh	Throw out the E_CHG_RemoteChBlock event upon reception of the acknowledgement message (BLA or UBA) from the remote end.
		SsmUnblockRemoteCh	

		SsmGetRemoteChBlockStatus	Throw out the E_CHG_RemotePCMBlock event upon reception of the group acknowledgement message (MBA/HBA/SBA or MUA/HUA/SUA) from the remote end.
	Circuit Group Blocked	SsmBlockRemotePCM	
		SsmUnblockRemotePCM	
		SsmGetRemotePCMBlockStatus	

Note: When the local driver is to block the remote end, the Range field in the circuit group blocking message it sends to the remote PBX can be set via the configuration item SendGRMRange.

1.16.4.5.3Circuit Resetting

The TUP state machine will automatically send the circuit group reset message to the remote PBX and reset all TUP channels when

- ✧ the SS7 server detects that the service supplied by the MTP3 level becomes usable again, or
- ✧ the synchronization signal of the digital trunk is resumed.

The Range field in the circuit group reset message can be set via the configuration item SendGRMRange.

1.16.4.5.4CALLER HOLDING Feature

During an incoming call that both parties have joined, if the calling party first hangs up the phone, the remote PBX will send the CCL message to the local end. Upon reception of the CCL message, the driver will send back the OPR message to the remote PBX provided the application invokes the function SsmSetCalleeHoldFlag to enable the 'Caller Holding' feature.

1.16.4.5.5TUP Configuration

The table below shows all relative configuration items for TUP protocol.

Configuration Section	Configuration Item	Description
[BoardId=x]	***PcmNumber***	set the total number of on-board digital trunks as well as the signaling type, the clock operating mode and the line type for each digital trunk.
	PcmSSx	
	PcmClockMode	
	PcmLinkType	
	UseTS16AsCircuit ***Ss7SignalingTS***	determine whether some of the digital trunks serve as SS7 signaling links and specify the time slot number to support the signaling links.
[PcmInfo]	***TotalPcm***	set the total number of all digital trunks involved in the PC and build the mapping relationship between the logical number and the physical number of each digital trunk.
	Pcm	
[SS7]	***AutoHandleTup***	sets whether to use the call state machine supplied by the driver.
	Ss7ServerIP	set the SS7 server address and the IP address of the local PC.
	SecondServerIP	
	LocalIP	
	MaxWaitAutoDialAnswerTime	Outgoing Call: set relative parameters.
	CalloutCallerId	

[TUP]	SetSTSignal	
	AutoSendGSM	
	ConnectReqMsg	
	AddressSignal	
	CalloutIAM_MsgPntr	Outgoing Call: customize the IAM message to be sent to the remote PBX.
	CalloutIAM_CAT	
	CallingIndicatorBit	
	ReqTypeIndicators	
	OriginalCalleeAddrInd	
	CallerAddrInd	
	DefaultACM	Incoming Call: set relative parameters.
	AutoSendACM	
	ReqTypeIndicators	
	RingToPending	
	RcvPhoNumCfgLen	Incoming Call: set the number-receiving rule.
	DefaultRcvPhoNumLen	
	DefaultRcvCallerID	
	MaxPhoNumRule	
	Rule	
	HangupRingSendCBK	Incoming Call: set the message type used for refusing the incoming call.

Note: Those written in the ***XXXX*** font are essentially configured items while those in XXXX are optionally configured ones.

1.16.4.6 ISUP Programming and Application

1.16.4.6.1ISUP Configuration

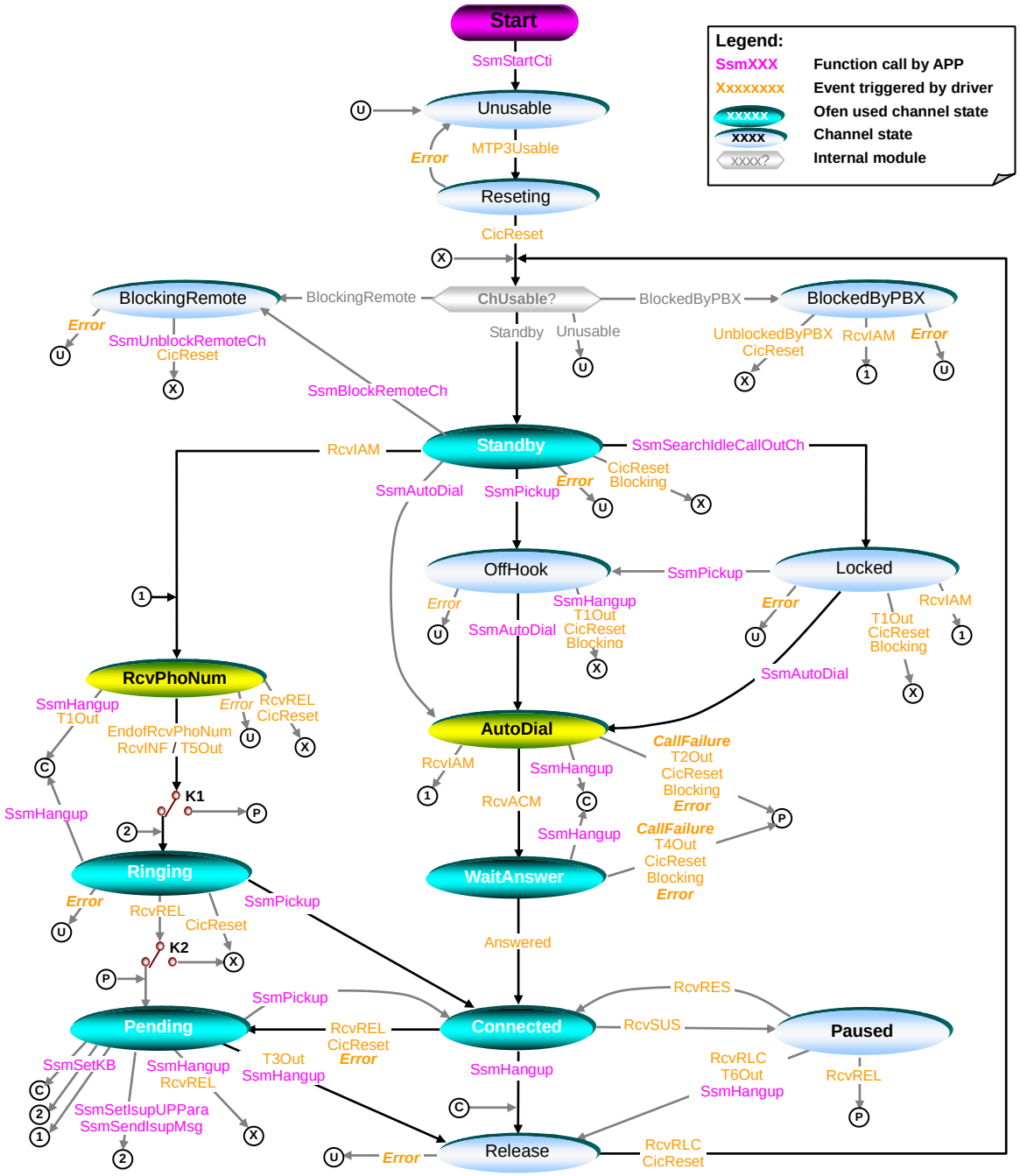
The table below lists all relative configuration items for the use of ISUP.

Configuration Section	Configuration Item	Description
[**BoardId=x**]	***PcmNumber***	set the total number of on-board digital trunks as well as the signaling type, the clock operating mode and the line type for each digital trunk.
	PcmSSx	
	PcmClockMode	
	PcmLinkType	
	UseTS16AsCircuit ***Ss7SignalingTS***	determine whether some of the digital trunks serve as SS7 signaling links and specify the time slot number to support the signaling links.
[PcmInfo]	***TotalPcm***	set the total number of all digital trunks involved in the PC and build the mapping relationship between the logical number and the physical number of each digital trunk.
	Pcm	
[SS7]	***TotalIsupPcm***	set the digital trunks which uses ISUP in the local PC.
	IsupPcm	
	AutoHandleIsup	sets whether to use the call state machine supplied by the driver.
	Ss7ServerIP	set the SS7 server address and the IP address of the local PC.
	SecondServerIP	
	LocalIP	
[ISUP]	MaxWaitAutoDialAnswerTime	Outgoing Call: set relative parameters.
	CalloutCallerId	
	SetSTSignal	

	DefaultNatureOfConnectionInd	Outgoing Call: customize the IAM message to be sent to the remote PBX.
	DefaultIAM_ForwardCallInd	
	DefaultIAM_CAT	
	DefaultIAM_TransmissionMediumRequirment	
	DefaultIAM_CalleeParam	
	DefaultIAM_CallerParam	
	DefaultIAM_OriginalCalleeParam	
	DefaultIAM_RedirectingNumber	
	bSubscriberSI，SubscriberSI	
	bOptionalFCI，OptionalFCI	
	Usr2UsrInfo	
	AutoSendINF	Outgoing Call: sets the mode to reply to the INF message.
	RingToPending	Incoming Call: set relative parameters.
	AutoSendACM	
	DefaultCalledPickupMsg	
	DefaultBackwardCallInd	
	DefaultHangupRELInd	
	DefaultCauseInd	
	SaveRGNTo1stPhoNumStr	
	DefaultRcvPhoNumLen	Incoming Call: set the number-receiving rule.
	DefaultRcvCallerID	
	MaxPhoNumRule	
	Rule	
	AddressSignal	sets the parameters of the ISUP state machine.

Note: Those written in the ***XXXX*** font are essentially configured items while those in XXXX are optionally configured ones.

1.16.4.6.2ISUP Channel State Machine



The internal events automatically triggered by the driver itself are described as follows.

Event Name	Description
Mtp3Usable	The driver will send this event to all channel state machines after receiving signaling from the SS7 server so as to change the channel state from **Unusable** to **Reseting** and reset the circuit.
CicReset	This event is triggered when the driver receives the circuit reset message (GRA, GRS or RSC). After this event is triggered, in case the channel stays in the **AutoDial** or **WaitAnswer** state

<table>
<tr><td></td><td>and the outgoing call fails, the driver will throw out the E_PROC_AutoDial event to the application. The function SsmGetAutoDialFailureReason can be used to acquire the specific reason why the AutoDial task fails.</td></tr>
<tr><td>Blocking</td><td>This channel is blocked by the local end (Local Blocking) or the remote end (Remote Blocking).
Remote Blocking: The driver receives the group blocking message (CGB) or the circuit blocking message (BLO) from the remote PBX.</td></tr>
<tr><td>UnblockedByPBX</td><td>Local Unblocked: This event is triggered when the driver receives the CGU, UBL messages from the remote PBX.</td></tr>
<tr><td>RcvIAM</td><td>The driver receives the incoming call request message (IAM) from the remote PBX.
The driver will automatically save the newly received IAM message. The function SsmGetIsupUPPara (with the parameter C_ISUP_IAM) can be used to acquire this message, and SsmGetRedirectionInfReason, SsmGetKA can be used respectively to obtain the redirection reason value and the ‘Calling Party Category’ parameter in the IAM message.
That whether the driver accepts this request or not depends on the channel state at that time.
<table>
<tr><th>Channel State</th><th>Description</th></tr>
<tr><td>BlockedByPBX</td><td>When the local end is blocked, although the channel does not support outgoing calls, it still can accept incoming calls. Therefore, the driver will respond to the IAM message in such case.</td></tr>
<tr><td>AutoDial</td><td>The reception of the IAI message from the remote PBX during the outgoing call progress implementation indicates the ‘collision’ phenomenon occurs. As stipulated in ISUP, if the channel’s CIC represents the circuit controlled by the local end, the local end will abandon the IAI message and continue the outgoing call; otherwise, the local end will quit the outgoing call and process the incoming call, which indicates this outgoing call fails. At that time the driver will throw out the E_PROC_AutoDial event to the application. The function SsmGetAutoDialFailureReason can be used to acquire the specific reason why the AutoDial task fails.</td></tr>
<tr><td>OffHook Locked</td><td>The channel is preparing to make an outgoing call but does not send the IAM message yet. The driver will respond to the incoming call request message IAM received at that time from the remote PBX.</td></tr>
<tr><td>Standby</td><td>The channel accepts the incoming call.</td></tr>
<tr><td>Others</td><td>The channel rejects this IAM message.</td></tr>
</table>
In case the driver accepts the incoming call, it will throw out the E_CHG_RxPhoNumBuf event to the application program after saving the called party number within the IAM message to the buffer and change the channel state to RcvPhoNum.</td></tr>
<tr><td>EnfofRcvPhoNum</td><td>The driver receives all called party numbers according to the preset number-receiving rule. For more information about the number-receiving rule, refer to the section Number-receiving Rule for Incoming Call in this chapter.</td></tr>
<tr><td>RcvINF</td><td>The driver receives the INF message which contains relative information about the calling party from the remote PBX.</td></tr>
<tr><td>RcvREL</td><td>The driver receives the disconnection message (REL) from the remote PBX. The function SsmGetReleaseReason can be used to obtain the specific reason why the call gets disconnected.
When this event is triggered, if the channel stays in the state
• RcvPhoNum:
it indicates the remote PBX (the calling party) cancels this call during the incoming call establishment.
• Ringing:
it indicates during the incoming call establishment, the remote PBX cancels this call when the local end is in the Ringing state. The application can determine whether to</td></tr>
</table>

	interfere in the channel state transition via the control over the switch K2 which is set by the configuration item RingToPending. If the configuration item RingToPending is set to 1, the driver will set PendingReason to PEND_RemoteHangupOnRinging and transfer the channel state to **Pending** for the further processing by the application program; otherwise, the driver will set out to disconnect this incoming call. • **Connected**: it indicates the calling party has hung up the phone. The driver will set PendingReason to PEND_RemoteHangupOnConnected and then transfer the channel state to **Pending**.
RcvRLC	The driver receives the RLC message from the remote PBX which means the disconnection phase ends up.
RcvACM	The driver receives the ACM (Address Complete Message) message from the remote PBX. In the outgoing call progress implementation, the driver will send the IAM message to the remote PBX when the application calls the function SsmAutoDial to start an outgoing call. Then the remote PBX will send the ACM message to the local end as it believes all necessary information including the Callee ID has been received in order to show the idle state of the called party. After receiving the ACM message via the function SsmGetIsupUPPara (with the parameter C_ISUP_ACM), the driver will save it to the internal buffer, set the progress value of the AutoDial task to DIAL_ECHOTONE and then transfer the channel state to **WaitAnswer**. The function SsmGetKB can be used to acquire the 8 lower bits of the backward call indicator field (2 bytes) in this ACM message. The remote PBX may send the CPG (Call Progress) message following the ACM message. The function SsmGetCpg can be used to get the CPG message.
Answered	During an outgoing call, after the remote PBX finishes receiving the called party number, it will send ringing signals to the called party if the called party is in the idle state; or it will send the ANM (Answer) message to the local end if the called party picks up the phone. After receiving the ANM message, the driver will set the progress value of the AutoDial task to DIAL_VOICE, ending up the AutoDial task, and throw out the E_PROC_AutoDial event to the application.
RcvSUS	The driver receives the SUS (Suspend) message from the remote PBX.
RcvRES	The driver receives the REC (Recover) message from the remote PBX.
CallFailure	If the driver receives the REL message from the remote PBX during an outgoing call, which means the call fails, it will send the E_PROC_AutoDial event to the application. The function SsmGetAutoDialFailureReason can help to acquire the specific reason why the AutoDial task fails.
T1Out	The timer T1 overflows. T1 is set to 60sec and will be automatically started when the channel goes into the **Locked** or **OffHook** state.
T2Out	The timer T2 overflows. T2 is set to 25sec and will be automatically started when the channel goes into **S_ISUP_WaitDialAnswer** (a substate of **AutoDial**). After T2 overflows, which indicates this call fails, the driver will throw out the E_PROC_AutoDial event to the application. The function SsmGetAutoDialFailureReason can help to acquire the specific reason why the AutoDial task fails.
T3Out	The timer T3 overflows. T3 is set to 60sec and will be automatically started when the channel goes into the **Pending** state.
T4Out	The timer T4 overflows. T4 is set by the configuration item MaxWaitAutoDialAnswerTime and will be automatically started when the channel goes into the **WaitAnswer** state. After T4 overflows, the driver will throw out the E_PROC_AutoDial event to the application. The function SsmGetAutoDialFailureReason can help to acquire the specific reason why the AutoDial task fails.
T5Out	The timer T5 overflows. T5 is set to 10sec and will be automatically started when the channel goes into **S_ISUP_WaitINF** (a substate of **RcvPhoNum**). After T5 overflows, the driver will stop waiting for the INF message from the remote PBX.
T6Out	The timer T6 overflows. T6 is set to 180sec and will be automatically started when the channel goes into the **Suspend** state.

Error	This event is triggered when the driver detects one of the following situations. ✧ The synchronization signal through TS0 on the digital trunk where this channel stays gets lost (e.g. line disconnected, clock configuration error and so on). ✧ The TCP/IP connection between the ISUP stack and the SS7 server is broken. ✧ The signaling unusable message is received from the SS7 server. If the channels stays in the **AutoDial** or **WaitAnswer** state when this event is triggered, the outgoing call fails. The driver will throw out the E_PROC_AutoDial event to the application in such case. The function SsmGetAutoDialFailureReason can help to acquire the specific reason why the AutoDial task fails.

Each state identifier in the above diagram is described in the following table.

State Identifier	Description
Unusable	The channel is unusable.
Reseting	The circuit is being reset.
BlockingRemote	The remote PBX is blocked. In the **BlockingRemote** state, the outgoing call can be started not by the remote PBX but by the local end.
BlockedByPBX	The channel is blocked by the remote PBX (Remote Blocking). When the remote PBX is stopped for maintenance, it will send the blocking message to the local end to prevent new calls. When the channel stays in such state, the driver will refuse the Outgoing Call instruction given by the application program and only accept and process incoming calls.
Standby	'Idle' state The channel should go into this state provided it is normally reset after the successful driver initialization. The application can invoke the function SsmAutoDial to start an outgoing call in this state. The driver will automatically clear the Caller ID buffer when the channel state transfers to Idle at the end of the call, provided the configuration item AutoClearCallerIdBufOnHangup is set to 1.
Locked	'Outgoing Call Locked' state The application can invoke the function SsmAutoDial to start an outgoing call in this state. The driver will start the timer T1 when the channel goes into this state. If the application does not call the function SsmPickup or SsmAutoDial before T1 overflows, the driver will reset the channel back to the idle state. The set value of T1 is 60sec.
OffHook	'Off-hook' state The application can invoke the function SsmAutoDial to start an outgoing call in this state. The driver will start the timer T1 when the channel goes into the OffHook state. If the application does not call the function SsmAutoDial before T1 overflows, the driver will reset the channel back to the idle state.
AutoDial	It is an internal state during an outgoing call, including the substates S_ISUP_WaitDialAnswer and S_ISUP_WaitSetCallerID. The function call of SsmAutoDial or SsmAutoDialEx by the application will prompt the driver to send the IAM message to the remote PBX and transfer the channel state to **S_ISUP_WaitDialAnswer**. In the **S_ISUP_WaitDialAnswer** substate, if the remote PBX wants to receive the calling party information which is not contained in the IAM message, it will send the INR message to the local end. After the driver receives the INR message, ✧ if the Caller ID buffer is not empty during an outgoing call, the driver will put the calling party information into the INF message and send the INF message back to the remote PBX as well as keep the channel in the current state. ✧ if the Caller ID buffer is empty during an outgoing call and the configuration item AutoSendINF is set to 1, the driver will take the calling party information set by the configuration item CalloutCallerId or the function SsmSetTxCallerId as the parameter, automatically reply to the INF message and keep the channel in the current state; ✧ if the Caller ID buffer for the outgoing call within the driver is empty and the

	configuration item AutoSendINF is set to 0, the driver will transfer the channel state to **S_ISUP_WaitSetCallerID** and start the timer T3 at the same time. In the **S_ISUP_WaitSetCallerID** substate, if the application invokes the function SsmSetTxCallerId, the driver will put the preset calling party information into the INF message and then send the INF message back to the remote PBX. If the application does not call the function SsmSetTxCallerId before T3 overflows, the driver will take the set value of the configuration item CalloutCallerId as the parameter, automatically send the INF message to the remote end and transfer the channel state to **S_ISUP_WaitDialAnswer**.
WaitAnswer	The channel is waiting for the called party to pick up the phone. This state is only applicable to the outgoing call. The called party should be able to hear ringing tones in such state, and if at the local end is the station channel that makes an outgoing call through the digital trunk channel it should send the ringback tone to the station channel.
Pending	'Pending' state. The function SsmGetPendingReason can help get the specific reason why the call becomes pending. ➢ If PendingReason is set to PEND_WaitBckStpMsg, it indicates the driver has completed the incoming call progress implementation and is waiting for the application to accept or refuse the incoming call. The application program can ✧ invoke the function SsmSetKB to accept or refuse the incoming call. ◆ If KB=1/6/7, the driver will send the ACM message to the remote PBX to accept the incoming call. If the application has invoked the function SsmSetIsupUPPara (with the parameter C_ISUP_ACM) to submit a self-built ACM message to the driver before the function call of SsmSetKB, this ACM message will be sent to the remote PBX; otherwise, the driver will use the ACM message which is automatically set up. After the ACM message is transmitted, the channel will transfer to the **S_CALL_RINGING** state; ◆ If KB=2/3/4/5/9, the driver will send the REL message to the remote PBX to refuse the incoming call. The reason value carried in the REL message is Called Subscriber Busy/Address Incomplete/Call Refused/No Answer/User Absent. The channel sets out to disconnect the call and goes into the **Release** state; ◆ If KB=8, the driver will send the INR message to the remote PBX and transfer the channel state back to **RcvPhoNum**. ✧ invoke in order the function SsmSetIsupUPPara (with the parameter C_ISUP_ACM) and the function SsmSendIsupMsg to accept the incoming call. Then the channel will go into the **S_CALL_RINGING** state. ✧ invoke the function SsmHangup or SsmHangupEx to refuse the coming call. The driver sets out to disconnect the call and transfers the channel state to **Release**. See description on relative functions for more information. ✧ invoke the function SsmPickup to accept the incoming call. The driver will send the CON message to the remote PBX and then transfer the channel state to **Connected**. **Note: if the driver receives the REL message from the remote PBX in this state, it will automatically send the RLC message and then transfer the channel state to Standby.** ➢ No matter what causes the channel to go into the **Pending** state, the application can command the driver to send the REL message containing the number redirection information to the remote PBX and prompt the channel to transfer to the **Release** state via the function call of SsmSetIsupFlag (with the parameter ISUP_PhoNumREL). ➢ If PendingReason is set to: ✧ PEND_RemoteHangupOnTalking (the calling party is detected to first hang up the phone during the connected incoming call); ✧ PEND_CalleeHangupOnTalking (the called party is detected to first hang up the phone during the connected outgoing call); ✧ PEND_RemoteHangupOnRinging (the calling party cancels the incoming call when the local end is ringing);

	the application program can call the function SsmHangup or SsmHangupEx to hang up the phone.
RcvPhoNum	It is an internal state during an incoming call progress, including the substates **S_ISUP_Suspend** and **S_ISUP_WaitINF**. The driver judges the received called party number according to the preset number-receiving rule. If the number complies with the rule, the channel will transfer to the subsequent state. For more information about the number-receiving rule, refer to the section Number-receiving Rule for Incoming Call in this chapter. ➢ In the incoming call, the switch K1 controls the 'Auto-answer of Incoming Call' feature. When the channel receives the complete called party number and other relative information, ✧ if the 'Auto-answer of Incoming Call' feature is enabled, the driver will automatically send the ACM message to the remote PBX and transfer the channel state to **Ringing**. The configuration item DefaultBackwardCallInd is used to set the backward call indicator in the ACM message sent automatically by the driver while the function SsmSetKB is used to set the charge indicator contained in the backward call indicator field in the ACM message. ✧ if the 'Auto-answer of Incoming Call' feature is not enabled, the driver will set PendingReason to PEND_WaitBckStpMsg, transfer the channel state to **Suspend**, and let the application determine whether to accept this call or not. The switch K1 can be set via the function SsmEnableAutoSendKB or the configuration item AutoSendACM, being switched off by default. ➢ Every time when the driver receives the SAM message from the remote PBX, it will save the message to its internal buffer and throw out the E_CHG_RxPhoNumBuf event to the application. ➢ If the called party number received by the channel does not conform to the number-receiving rule, the driver will automatically send the REL message (carrying the unallocated-number reason) and transfer the channel state to **Release**. ➢ Also the application can invoke the function SsmHangup/SsmHangupEx to refuse this call. In default, the driver will send the REL message with the reason value after that. You can use the configuration item DefaultHangupRELInd to set the reason value telling why the call fails in the REL message, or the parameter nType=ISUP_REL_DENY_SetToOther(=100) in the function SsmSetIsupFlag to change the reason value.
Ringing	'Ringing' state The application program can ✧ invoke the function SsmPickup or SsmPickupANX to accept the call. The two configuration items DefaultCalledPickupMsg and DefaultBackwardCallInd are used respectively to select the type (ANM or CON) of the message to be sent by the driver to the remote PBX and to set the backward call indicator in the ACM/COM message. ✧ invoke the function SsmHangup or SsmHangupEx to refuse the call If the driver receives the REL message (i.e. the RcvRLC event in the state machine above) from the remote PBX when the channel is in the **Ringing** state, it indicates the remote PBX cancels this call. At that time, the switch K2 which is set via the configuration item RingToPending can be used to control the channel state transition. If the configuration item RingToPending is set to 1, the driver will set PendingReason to PEND_RemoteHangupOnRinging and transfer the channel state to **Pending** for the further processing by the application program; otherwise, the driver will set out to disconnect this incoming call at once.
Connected	'Talking' state. Both parties are connected and able to talk with each other. When the channel state transfers to **Connected**, the driver will ✧ start the DTMF Detector if the configuration item AlwaysEnableRxDtmf is set to 0; ✧ start the Barge-in Detector if the configuration item AlwaysDetectBargeIn is set to 0; ✧ clear the Tone Detector and start it. The application program can ✧ invoke any function related to voice processing.

	✧ invoke the function SsmHangup or SsmHangupEx to terminate this call at any time. If the driver receives the REL message (i.e. the RcvRLC event in the state machine above) from the remote PBX when the channel is in the **Connected** state, it indicates the remote party of the call has hung up the phone.
Paused	'Suspend' state
Release	'Disconnection' state When the channel is in the **Release** state, the driver will wait for the remote PBX to send the RLC (Release Complete) message.

Each state identifier in the table above is declared in the file **ShpA3Api.h** as follows.

State Identifier	Declaration in ShpA3Api.h	
	Value	Macro
Standby	0	S_CALL_STANDBY
OffHook	1	S_CALL_PICKUPED
Ringing	2	S_CALL_RINGING
Connected	3	S_CALL_CONNECTED
Pending	7	S_CALL_PENDING
WaitAnswer	9	S_CALL_WAIT_REMOTE_PICKUP
Unusable	11	S_CALL_UNAVAILABLE
Locked	12	S_CALL_LOCKED
Release	121	S_ISUP_WaitRLC
Reseting	122	S_ISUP_WaitReset
BlockingRemote	123	S_ISUP_LocallyBlocked
BlockedByPBX	124	S_ISUP_RemotelyBlocked
Paused	129	S_ISUP_Suspend
RcvPhoNum	120	S_ISUP_WaitSAM
	126	S_ISUP_WaitINF
AutoDial	125	S_ISUP_WaitDialAnswer
	127	S_ISUP_WaitSetCallerID

In the above state machine, ChUsable is the driver's internal component to judge the channel's usability. It will output values as shown in the table below.

Output Value	Description
Unusable	The channel is unusable if the Error event keeps effective.
BlockedByPBX	The channel is blocked by the remote PBX.
BlockingRemote	The channel keeps blocking the remote PBX.
Standby	The Error event stays invalid.

Note: At this stage, the driver also performs the following operations.

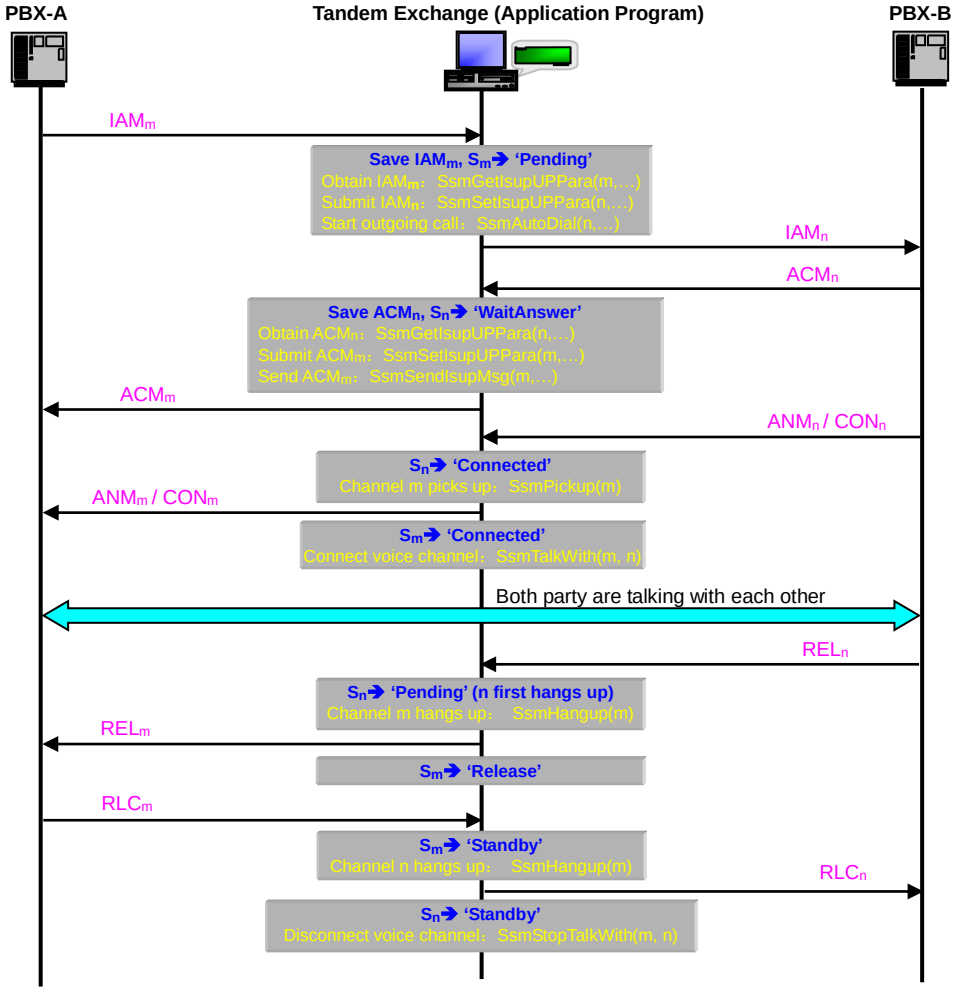
- Clear the tone detector and automatically close it;
- Automatically terminate the WaitDtmf task if it has been started by the application;
- Automatically terminate the task of DTMF transmission if it has been started by the application;
- Automatically clear the buffer of the DTMF detector, and close the DTMF detector if the configuration item **AlwaysEnableRxDtmf** is set to 0;
- Automatically close the Barge-in detector if the configuration item **AlwaysDetectBargeIn** is set to 0;
- Automatically clear the Caller ID buffer if the configuration item **AutoClearCallerIdBufOnHangup** is set to 1.

1.16.4.6.3ISUP Channel Serving as Tandem Exchange

Generally, the SHD Series boards based on ISUP protocol work as the terminating office. However, if necessary,

you can configure the ISUP channel to serve as the tandem exchange following the way below.

Assume that the circuit number on PBX-A that corresponds to Channel m in the application is m and the 'Auto-answer of Incoming Call' feature is disabled for Channel m, the circuit number on PBX-B that Channel n corresponds to is n. The following figure illustrates how the ISUP channel works as the tandem exchange to establish a call.



In the figure above,

Blue Words describe the driver's internal operation;

S_m, S_n respectively indicate the state of Channel m and that of Channel n;

XXX_m, XXX_n respectively represent the ISUP message transmitted between the application and PBX-A, and that between the application and PBX-B;

Yellow Words describe the application's behavior.

1.16.4.6.4SynCTI-provided ISUP State Machine Unused

In some case, the application is required to handle MSU messages in ISUP protocol by itself. Such feature can be enabled via the configuration item AutoHandleIsup. Upon reception of MSU messages from the remote end, the driver will deliver them to the application directly so that the ISUP state machine is not used. For more information, refer to the Advanced SS7 Programming section.

1.16.4.7 Advanced SS7 Programming

1.16.4.7.1MTP3 Service

The application program is allowed to process any protocol at the 4th level in SS7 via the direct use of the MTP3 interface provided by the SynCTI driver. In case the following configuration items are set to 0:

- ✧ AutoHandleTup (for TUP)
- ✧ AutoHandleIsup (for ISUP)
- ✧ AutoHandleSccp (for SCCP)

every time when the driver receives a message from the SS7 server, it will save the message not to its internal state machine but to a buffer, and at the same time throw out the E_RCV_Ss7Msu event. When receiving E_RCV_Ss7Msu, the application program can invoke the function SsmGetSs7Msu to take out this message for subsequent processing. The function SsmSendSs7Msu can be used to send a message.

Every time when the route from the SS7 server to a DPC varies in state (usable or unusable), relative information will be sent to the SynCTI driver, which prompts the driver to send the E_CHG_Mtp3State event to the application. In an application system involving only one DPC, the function SsmGetMtp3State can be used to get information of the route to this DPC; in the system connecting with multiple DPCs, the function SsmGetMtp3StateEx can help acquire the usability of the route to the specified DPC. The function SsmGetMtp2Status is used to obtain the state of the specified 64Kbps signaling link.

1.16.4.7.2Client Software Programming Interface on SS7 Server

The client software programming interface on the SS7 server enables those devices which are not compliant with the SS7 protocol to be capable of handling SS7 signaling messages in cooperation with the SS7 server based on the SHD series boards, via the connection by TCP/IP. The interface is provided in the form of dynamic link libraries, consisting of the TCP/IP Communications component and the SS7 Signaling Processing component, facilitating user-end development. For more information, refer to the manual *Client Software Programming Interface on SS7 Server*.

1.16.4.7.3Virtual Circuit Programming Interface on SS7 Server

The virtual circuit is the circuit which does not occupy any voice resource in the SS7 circuit switching, generally used for the prepaid system.

The SynCTI driver can configure the virtual circuit that is not bound to some physical digital trunk by setting the parameter i in the configuration item Pcm[k]=i to -1, and is able to bind the virtual digital trunk and the destination signaling point code via the [ISUPRouter] section in the configuration file ss7server.ini in the SS7 server. Fore more information about the API functions related to the virtual circuit, refer to the document *SynCTI Programmer's Manual – Programming for Virtual Circuit on SS7 server. doc*.

1.16.4.7.4SCCP Programming Interface

The SynCTI driver provides the advanced SCCP programming interface. See the description in relative documents for details. If the application is required to support SCCP by itself, it needs to set the configuration item AutoHandleSccp to 0 and use the MTP3 service offered by the SynCTI driver. See the MTP3 Service section in this chapter for more information.

1.16.5 ISDN Programming

1.16.5.1 Basic Concepts

1.16.5.1.1Network-side Mode and User-side Mode

Because the protocols for connection used at the two ends of a digital trunk may differ from each other, there are two distinguishing definitions ‘user-side’ and ‘network-side’ used in the ISDN protocol. The User Side indicates the user terminal, i.e. the end to connect with the ISDN PBX; the Network Side is the other end to connect with the ISDN user terminal. In this manual, we call the digital trunk working in the User-side mode as the user-side digital trunk, and that in the Network-side mode as the network-side digital trunk.

1.16.5.1.2TEI Value

TEI, i.e. Terminal Equipment Identifier, is used to identify the service access point in the point-to-point data link connection. The subfield of TEI is allowed to stipulate 128 TEI values, thereby offering the range of 0~127, among which 0~63 are fixed TEI values (selected by the device), indicating the TEI used by L2 to receive/transmit the I or UI frame is unchangeable; 64~126 are dynamic TEI values, indicating the TEI used by L2 to receive/transmit the I or UI frame is changeable (selected by the network); 127 is only applicable to the broadcast link.

The configuration items UserTEIValue and NetTEIValue are used to set the TEI values respectively for the user side and the network side.

1.16.5.1.3Representation of Channel Identification Information

What tells the information about the channel used in an ISDN call is named the channel identification information element. It is used to identify channels under the control of the signaling program and can be represented by the following 2 ways:

- ✧ Time Slot Diagram: The channel identification information element is represented by 32 Bit (4 bytes), corresponding to 32 time slots. If bit=1, it indicates this time slot is being used; if bit=0, it indicates this time slot is unused. In practice, usually only one bit is set to 1 with the rest all set to 0.
- ✧ Number: The channel identification information element is represented by 8 Bits (1 byte), the highest Bit7 with the fixed value of 1 and the other 7 bits (i.e. Bit6~Bit0) indicating the corresponding time slot numbers.

This element contains up to 34 octets, of which the 4th bit in the 3rd octet determines its representation way. The configuration items NetChIdentify and UserChIdentify are used to set the representation way respectively for the network side and the user side.

1.16.5.1.4 Terminating Office Mode vs. Tandem Exchange Mode

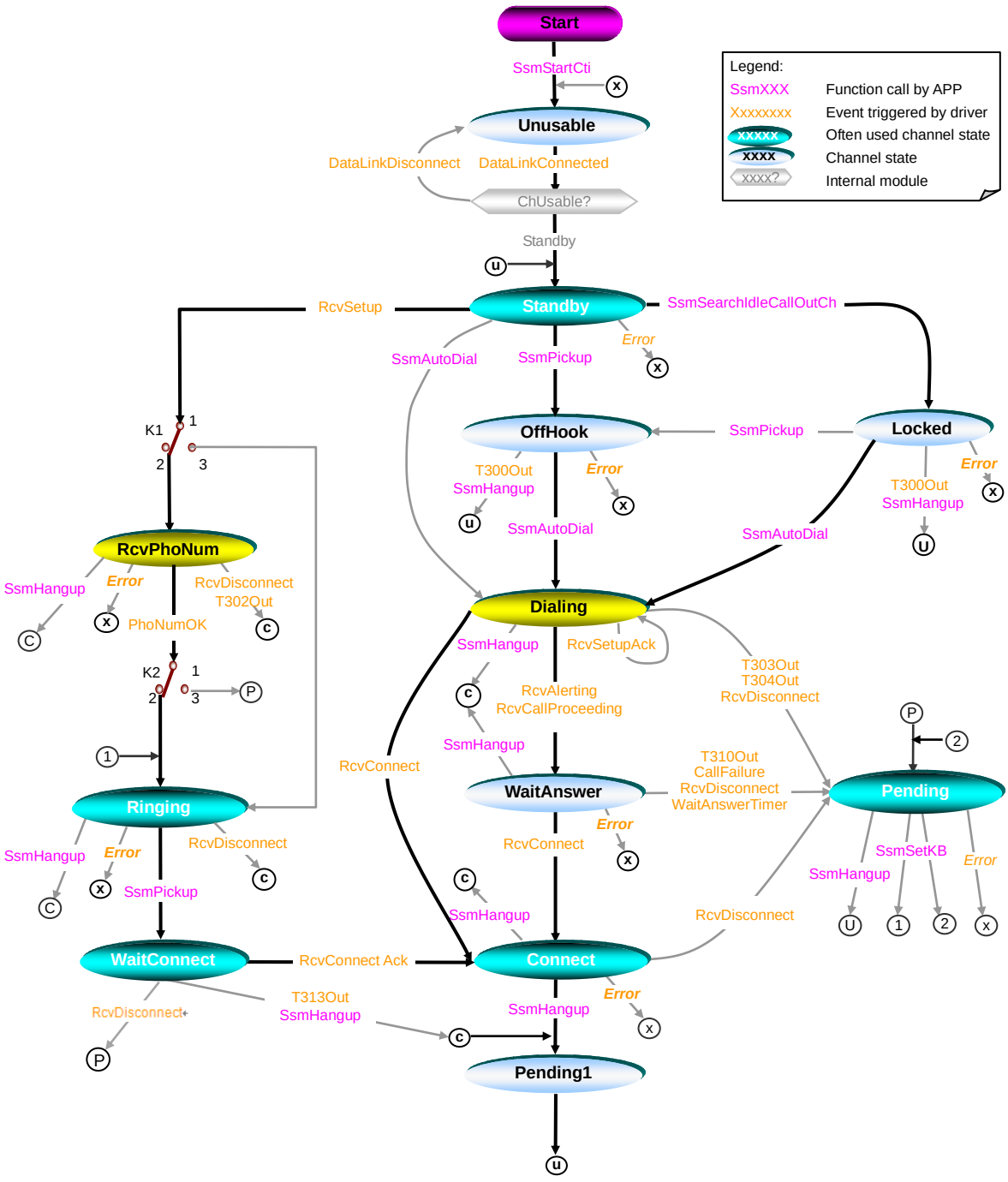
The telephone network is separated into two categories: Local Telephone Network and Long-distance Telephone Network. The local telephone network is the network covering the area with the same number, composed of the terminating office, the tandem exchange and the transmission links; the long-distance telephone network is the network covering the areas with different numbers in communication, composed of the long-distance exchange and the transmission links. At present, the telephone exchange is the core of the telephone network, using the digital SPC switching technology. That is, each call is encoded by the 64Kbit/s digital signal and occupies a time slot in the primary group, connects to different users via data switching between time slots under the control of signaling. The telephone exchange can be classified depending on the size of the service area into the first-level centre, the second-level centre, the third-level centre, the fourth-level centre and the fifth-level centre, i.e. C1, C2, C3, C4 and C5, among which C1, C2, C3, C4 are long-distance tandem exchanges while C5 is the terminating office. With the digitization of the telephone network, C1 and C2 are integrated into one level, i.e. DC1; C3 and C4 are integrated into one level, i.e. DC2. As a result, the China's telephone network has evolved from the 5-level network to the 3-level network, forming a network connection between a number of first-level centres.

The ISDN state machine supplied by the SynCTI driver is allowed to work not only in the Terminating Office mode but also in the Tandem Exchange mode. It can be set via the configuration item UserIsReceivePhoNum (user-side) or NetIsReceivePhoNum (network-side). During the incoming call processing, if the driver works in the Terminating Office mode, it will receives the called party number and other information according to the preset number-receiving rule; if the driver works in the Tandem Exchange mode, it will accept any incoming number regardless of the number-receiving rule. For more information about the number-receiving rule, refer to the section Number-receiving Rule for Incoming Call in this chapter.

1.16.5.1.5 ISDN Number and ISDN Address

The ISDN number is different form the ISDN address according to ITU-T specifications. The ISDN number is the number in relation to the ISDN network and the ISDN numbering scheme. It contains enough information for the network to choose a correct route for the call. An ISDN address includes the ISDN number as well as the appended necessary and/or optional addressing information which is not for the ISDN network to select the call route but for users to allocate the call to appropriate terminals.

1.16.5.2 ISDN Channel State Machine



In the diagram above, the internal events automatically triggered by the driver itself are described as follows.

Event Name	Description
DataLinkConnected	This event is triggered when the data link level is successfully connected with the remote PBX. The driver will throw out the E_CHG_ISDNStatus event to the application in such case.
DataLinkDisconnected	This event is triggered when the data link level is disconnected from the remote PBX. The driver will throw out the E_CHG_ISDNStatus event to the application in such case.
RcvSetup	The driver receives the SETUP (Call Setup Request) message from the remote PBX. The driver will decide whether to accept this call or not according to the current channel state: ✧ In the **Standby** state: to accept the call. The driver will save the called party number and other information in the SETUP message to its internal buffer and throw out the E_CHG_RxPhoNumBuf event to the application program. If the

	driver works in the Tandem Exchange mode, the number-receiving rule for incoming calls will be ignored. Therefore, any number can be accepted by the driver. In such case, the driver will directly transfer the channel state to **Ringing**. The switch K1 is used to control if the call is progressed in the Tandem Exchange mode or not. Its state can be set by the configuration item UserIsReceivePhoNum (user-side) or NetIsReceivePhoNum (network-side). For more information, refer to the section Terminating Office Mode vs. Tandem Exchange Mode in this chapter. ✧ In other states: to discard the SETUP message.
PhoNumOK	The driver receives all numbers (including both the calling party number and the called party number) according to the preset number-receiving rule.
RcvDisconnect	The driver receives the DISCONNECT message from the remote PBX. When this event is triggered, if the channel stays in the state ✧ **RcvPhoNum, WaitAnswer or Ringing**: it indicates the remote PBX (the calling party) cancels this call during the incoming call establishment; ✧ **Connect**: for the incoming call, the reception of the DISCONNECT message in the **Connect** state indicates the remote PBX first hangs up the phone. The driver will send the RELEASE message to the remote PBX and transfer the channel state to **Pending** after setting PendingReason to PEND_ ISDN_CALLOVER; ✧ **Dialing**: it indicates the outgoing call fails. The specific reason why the call fails can be acquired via the function SsmGetAutoDialFailureReason.
RcvConnectAck	The driver receives the CONNECT ACK message from the remote PBX. If the local end receives the CONNECT ACK message after sending the CONNECT message to the remote PBX, it indicates the call has been connected.
RcvConnect	The driver receives the CONNECT message from the remote PBX. If the channel stays in the **WaitAnswer** state at that time, the driver will automatically reply to the remote PBX by sending the CONNECT ACK message.
RcvAlerting	The driver receives the ALERTING message from the remote PBX. At that time the called party has gone into the ringing state.
RcvCallProceeding	The driver receives the CALL PROCEEDING message from the remote PBX. At that time the called party has gone into the ringing state.
RcvSetupAck	The driver receives the SETUP ACK message from the remote PBX. The event has no effect on the state transition.
RcvRelease	The driver receives the RELEASE message from the remote PBX. It will automatically respond to the remote PBX by sending the RELEASE COMPLETE message and set out to disconnect the call.
RcvReleaseComplete	The driver receives the RELEASE COMPLETE message from the remote PBX. If the local end receives the RELEASE COMPLETE message after sending the RELEASE message to the remote PBX, it indicates the call has been thoroughly disconnected.
T300Out	The timer T300 overflows. T300 is set to 60sec, the time limit on the operation after finding an idle channel and performing the pickup behavior on it to ensure the full use of the channel. The time can be set via the configuration item UserDialTime (user-side) or NetDialTime (network-side).
T302Out	In an incoming call, when the driver receives the SETUP message from the remote PBX, it will check the called party number in the message according to the preset number-receiving rule. If the number is incomplete, the driver will transfer the channel state to S_ISDN_IN_RCVING_NO (a substate of **RcvPhoNum**) and start the timer T302. If T302 overflows, this call fails and the driver will automatically send the RELEASE message to the remote PBX and change the channel state to **Standby**. T302 can be set via the configuration item UserT302 (user-side) or NetT302 (network-side).
T313Out	In an incoming call, when the channel goes into the **Ringing** state, if the application calls the function SsmPickup to pick up the phone, the driver will automatically send the CONNECT message to the remote PBX, transfer the channel state to **WaitConnect**, and start the timer T313 at the same time. If T313 overflows, this incoming call fails and the driver will send the DISCONNECT message to the remote PBX.

	T313 can be set via the configuration item UserT313, only for user-side.
T303Out	In an outgoing call, when the local end sends out the SETUP message, the driver will start the timer T303. If the driver does not yet receive the ALERTING, CALL PROCEEDING, SETUP ACK or CONNECT message from the remote PBX before T303 overflows, it will send the SETUP message once again and restart T303. If T303 overflows, this outgoing call fails and the driver will send the RELE COMP message to the remote PBX and change the channel state to **Pending**. T303 can be set via the configuration item UserT303 (user-side) or NetT303 (network-side).
T304Out	In an outgoing call, if the remote PBX can not obtain all necessary information (such as the called party number) from the SETUP message which has been sent by the local end, it will transmit the SETUP ACK message to the local end. Then the local end will wait for the application to append the called party number and start the timer T304 at the same time after receiving the SETUP ACK message. In this case, the application can invoke the function SsmAppendPhoNum to append the called party number. The driver will send the INFO message to the remote PBX and restart T304 when receiving the new number. If the driver does not yet receive the CALL PROCEEDING, ALERTING, CONNECT or DISCONNECT message from the remote PBX before T304 overflows, this outgoing call fails and the driver will send the DISCONNECT message to the remote PBX and transfer the channel state to **Pending**. T304 can be set via the configuration item UserT304 (user-side) or NetT304 (network-side).
T310Out	In an outgoing call, after the local end sends out the SETUP message and receive the CALL PROCEEDING message from the remote PBX, the driver will start the timer T310. If the driver does not yet receive the ALERTING or CONNECT message from the remote PBX before T310 overflows, this outgoing call fails and the driver will send the DISCONNECT message to the remote PBX and transfer the channel state to **Pending**. T310 can be set via the configuration item UserWaitAfterCallProceeding (user-side) or NetWaitAfterCallProceeding (network-side).
Error	This event is triggered when the driver detects one of the following situations. ✧ The synchronization signal through TS0 on the digital trunk where this channel stays gets lost (e.g. line disconnected, clock configuration error and so on). ✧ The driver receives error messages from the remote PBX. ✧ The data link level can not be connected with the remote PBX. What the driver does: ✧ If the channel stays in **Dialing** or **WaitAnswer** state: the outgoing call fails. The driver will set the progress value of the AutoDial task to DIAL_FAILURE, the reason value to ATDL_PcmSyncLos, and throw out the E_PROC_AutoDial event to the application. ✧ If the channel stays in other states: the driver will transfer the channel state to **Unusable**.
WaitAnswerTimer	In an outgoing call, after the local end sends out the SETUP message and receives the ALERTING message from the remote PBX, the driver will start the timer WaitAnswerTimer. If the driver does not yet receive the CONNECT message from the remote PBX before WaitAnswerTimer overflows, it will transfer the channel state to **Pending**. Note that the call is not released at this time and you are required to invoke the function SsmHangup to release the call. WaitAnswerTimer can be set via the configuration item MaxWaitAutoDialAnswerTime.

Each state identifier in the above diagram is described in the following table.

State Identifier	Description
Unusable	'Unusable' state. ✧ As long as one of the two events DatalinkDisconnect, Error keeps valid, the channel gets unusable and outputs the 'Unusable' state. ✧ In other situations, the channel stays idle and outputs the 'Standby' state.
Standby	'Idle' state

	The channel should go into this state provided the ISDN data link level is properly connected to the remote PBX after the successful driver initialization. The application can invoke the function SsmAutoDial to start an outgoing call in this state. The driver will automatically clear the Caller ID buffer when the channel state transfers to Standby at the end of the call, provided the configuration item AutoClearCallerIdBufOnHangup is set to 1.
OffHook	'Off-hook' state The driver will start the timer T300 when the channel goes into the **OffHook** state. If the application does not call the function SsmAutoDial before T300 overflows, the driver will reset the channel back to the idle state.
Locked	'Outgoing Call Locked' state When the application calls the function SsmSearchIdleCallOutCh, the driver will temporarily lock the channel specified by its return value and start the timer T300 at the same time. If the application does not call the function SsmPickup or SsmAutoDial before T300 overflows, the driver will reset the channel back to the idle state.
Dialing	It is an internal state during the outgoing call, including the substates **S_ISDN_OUT_WATI_NET_RESPONSE** and **S_ISDN_OUT_PLS_APPEND_NO**. After the application invokes the function SsmAutoDial, the driver will: ✧ send the SETUP message to the remote PBX; ✧ start the timer T303 at the same time; ✧ transfer the channel state to **S_ISDN_OUT_WATI_NET_RESPONSE**, waiting for the responding signals from the remote PBX; ✧ throw out the E_PROC_AutoDial event to the application. The reception of the SETUP ACK message from the remote PBX in the **S_ISDN_OUT_WATI_NET_RESPONSE** substate means the remote party wants the local end to provide more information about numbers. In such case, the driver will transfer the channel state to **S_ISDN_OUT_PLS_APPEND_NO** and start the timer T304 at the same time.
WaitAnswer	The channel is waiting for the called party to pick up the phone. The driver will start the timer T310 when the channel goes into this state. The function SsmGetFlag (with the parameter F_ISDNNet_WaitRemotePickup) can be used to acquire the specific reason why the channel goes into this state.
WaitConnect	The channel is waiting for the remote PBX to respond by sending back the CONNECT ACK message. The driver will start the timer T313 when the channel goes into this state.
Connect	'Talking' state. Both parties are connected and able to talk with each other. When the channel state transfers to **Connected**, the driver will ✧ start the DTMF Detector if the configuration item AlwaysEnableRxDtmf is set to 0; ✧ start the Barge-in Detector if the configuration item AlwaysDetectBargeIn is set to 0; ✧ clear the Tone Detector and start it. The driver's internally triggered events: ✧ RcvDisconnect: indicates the remote party first hangs up the call. This event is triggered when the driver sets the hangup reason to ISDN_CALLOVER and transfers the channel state to 'Pending'; ✧ Error: The channel will go into the 'Unusable' state if it receives one of the Error events in a call.
Ringing	'Ringing' state The channel has received all called party numbers and other relative information.
Pending	'Pending' state. The channel is waiting for further commands from the application in this state.
Pending[1]	'Pending' state, which can be ignored by the application. Without further commands, the driver program will automatically jump to the 'Standby' state.
RcvPhoNum	It is an internal state during the incoming call, including the substates **S_ISDN_IN_CHK_CALL_IN** and **S_ISDN_IN_RCVING_NO**. If the SETUP message received from the remote PBX contains all necessary information for the local end, the driver will trigger the internal event PhoNumOK; otherwise, it will send the SETUP ACK message to the remote PBX and transfer the channel state to

	S_ISDN_IN_CHK_CALL_IN, waiting for further number information from the remote PBX. If receiving the INFO message from the remote PBX in the **S_ISDN_IN_CHK_CALL_IN** substate, the driver will transfer the channel state to **S_ISDN_IN_RCVING_NO**. If the received called party number does not comply with the number-receiving rule, the driver will automatically send the RELEASE message to the remote PBX and transfer the channel state to **S_CALL_STANDBY**. The configuration items DefaultRcvPhoNumLen, DefaultRcvCallerID and RcvPhoNumCfgLen or DefaultRcvPhoNumLen, MaxPhoNumRule and Rule can be used together to set the number-receiving rule. For more information about the number-receiving rule, refer to the section Number-receiving Rule for Incoming Call in this chapter. Which state the channel will go into after completing the number reception depends on the switch K2. It is K2 that determines the ALERTING message is sent by the driver itself or the application for responding. The state of K2 can be set via the configuration item UserSideAutoSendAck (user-side)/NetSideAutoSendAck (network side) or the function SsmEnableAutoSendKB. If the setting of 'driver's automatic response' is chosen, the driver will transfer the channel state to SsmEnableAutoSendKB after sending the ALERTING message to the remote PBX; otherwise, it will start a timer whose time length is specified via the configuration item UserSideDefaultAckTimer (user-side) or NetSideDefaultAckTimer (network-side), set PendingReason to PEND_WaitBckStpMsg and then transfer the channel state to **Pending**, asking the application to decide the acceptance of this call. If the timer overflows, the driver will automatically send the reply message to the remote PBX according to settings of the configuration item UserSideDefaultAck (user-side) or NetSideDefaultAck (network-side).

Each state identifier in the table above is declared in the file ShpA3Api.h as follows.

State Identifier	Value in ShpA3Api.h	Macro Definition in ShpA3Api.h
Unusable	11	S_CALL_UNAVAILABLE
Standby	0	S_CALL_STANDBY
OffHook	1	S_CALL_PICKUPED
Dialing	81	S_ISDN_OUT_WAIT_NET_RESPONSE
	82	S_ISDN_OUT_PLS_APPEND_NO
WaitAnswer	9	S_CALL_WAIT_REMOTE_PICKUP
Connect	3	S_CALL_TALKING
WaitConnect	85	S_ISDN_IN_WAIT_TALK
Ringing	2	S_CALL_RINGING
Pending	7	S_CALL_PENDING
Locked	12	S_CALL_LOCKED
RcvPhoNum	83	S_ISDN_IN_CHK_CALL_IN
	84	S_ISDN_IN_RCVING_NO

In the above state machine, ChUsable is the driver's internal component to judge the channel's usability. It will output values as shown in the table below.

Output Value	Description
Unusable	The channel is unusable if one of the events Error, DataLinkDisconnect, Circuit Reset keeps valid.
Standby	The channel goes into the idle state provided the Error event becomes invalid, the DataLinkConnect event keeps valid and the circuit restart is successful.

Note: At this stage, the driver also performs the following operations.

- Clear the tone detector and automatically close it;
- Automatically terminate the WaitDtmf task if it has been started by the application;
- Automatically terminate the task of DTMF transmission if it has been started by the application;
- Automatically clear the buffer of the DTMF detector, and close the DTMF detector if the configuration item AlwaysEnableRxDtmf is set to 0;

- Automatically close the Barge-in detector if the configuration item AlwaysDetectBargeIn is set to 0;
- Automatically clear the Caller ID buffer if the configuration item AutoClearCallerIdBufOnHangup is set to 1.

The output events of the state machine are described in the following table.

Event Name	Description
E_CHG_ChState	The driver sends out this event to the application when the channel state changes.
E_PROC_AutoDial	The driver throws out this event to the application when there is any progress in the AutoDial task which is started by the function call of SsmAutoDial with the application.
E_CHG_RxPhoNumBuf	Every time when the driver receives the called party information from the remote PBX in an incoming call, it will throw out this event to the application.
E_CHG_ISDNStatus	This event is triggered and sent to the application when the driver receives the 'ISDN data link level usable' message.

Note: Event declarations can be found in the file ShpA3Api.h.

1.16.5.3 ISDN Configuration

The configuration items concerning ISDN calls are listed in the table below.

Configuration Section	Configuration Item	
[BoardId=x]	***PcmNumber***	set the total number of on-board digital trunks as well as the signaling type, the clock operating mode and the line type for each digital trunk.
	PcmSSx	
	PcmClockMode	
	PcmLinkType	
[PcmInfo]	***TotalPcm***	set the total number of all digital trunks involved in the PC and build the mapping relationship between the logical number and the physical number of each digital trunk.
	Pcm	
[ISDN]	***AutoHandleIsdn***	sets whether to use the call state machine supplied by the driver.
	UseISDNMode	sets the ISDN mode: User-side mode or Network-side mode.
	TotalUserLinker	set the total number of all digital trunks working in the ISDN user-side mode involved in the PC and build the mapping relationship between the logical number and the physical number of each digital trunk.
	UserPcmLink	
	UserTEIValue	set the main parameters for the ISDN user side.
	UserRestarTime	
	UserEstablishTime	
	UserT302	
	UserCrcMode	
	UserStatusReason	
	UserDialTime	Outgoing Call in ISDN User-side Mode: set relative parameters.
	UserWaitAfterCallProceeding	
	UserTxCallingPartyNum	
	NetTxCallingPartyNum	
	UserChIdentify	Outgoing Call in ISDN User-side Mode: customize the SETUP message to be sent to the remote PBX.
	UserCalledNoSet	
	UserCallingNoSet	
	UserNumIsFull	
	UserChPreference	
	UserHighLayerCompatible	
	UserLowLayerCompatible	
	CalloutCallerId	
	UserSideAutoSendAck	Incoming Call in ISDN User-side Mode: set relative

	UserSideDefaultAckTimer	parameters.
	UserIsReceivePhoNum	
	UserVoiceFormat	sets the hardware-based CODEC in the ISDN user-side mode.
	TotalNetLinker	set the total number of all digital trunks working in the ISDN network-side mode involved in the PC and build the mapping relationship between the logical number and the physical number of each digital trunk.
	NetPcmLink	
	NetTEIValue	set the main parameters for the ISDN network side.
	NetRestarTime	
	NetEstablishTime	
	NetT302	
	NetCrcMode	
	NetStatusReason	
	NetDialTime	Outgoing Call in ISDN Network-side Mode: set relative parameters.
	NetWaitAfterCallProceeding	
	NetChIdentify	Outgoing Call in ISDN Network-side Mode: customize the SETUP message to be sent to the remote PBX.
	NetCalledNoSet	
	NetCallingNoSet	
	NetNumIsFull	
	NetChPreference	
	NetHighLayerCompatible	
	NetLowLayerCompatible	
	CalloutCallerId	
	NetSideAutoSendAck	Incoming Call in ISDN Network-side Mode: set relative parameters.
	NetSideDefaultAckTimer	
	NetIsReceivePhoNum	
	DefaultRcvCallerID	
	NetVoiceFormat	sets the hardware-based CODEC in the ISDN Network-side mode.
	DefaultRcvPhoNumLen	Incoming Call in ISDN User-side and Network-side Modes: set the number-receiving rule.
	MaxPhoNumRule	
	Rule	
	UserStatusReason NetStatusReason	Advanced items for the ISDN user side and network side.
[DebugView]	IsdnLogEnable	set the log output in the ISDN user side and network side.
	DecodeIsdnMsg	
	IsdnDebugLog	
	LogCreatePeriod	
	LogMaxKeep	
	LogMaxPeriod	
	LogFilePath	

Note: Those written in the ***XXXX*** font are essentially configured items while those in XXXX are optionally configured ones.

1.16.5.4 Advanced ISDN Programming Interface

The SynCTI driver also provides the advanced programming interface to output the original messages at the HDLC level. The ISDN messages are processed by the application program itself. See the table below for the configuration items and functions related to the advanced ISDN programming interface.

	Name	Description
Configuration Item	AutoHandleIsdn	sets whether to use the advanced ISDN programming interface or not
Function	SsmSendIsdnMSU	sends a message

	SsmCheckIsdnMSU	queries if there are some messages available.
	SsmGetIsdnMSU	takes out an ISDN message.

1.16.6 SS1 Programming

1.16.6.1 Choice of Signaling Protocols

SynCTI supports SS1 and the Line side protocol, which can be set via the configuration item ProtocolType. To configure the SS1 protocol, you can use the configuration item mfcr2_Protocol to select countries.

1.16.6.2 Setting Call Progressing Parameters

The configuration item TxCas_CD is used to set the value of the C/D code element in the ABCD signaling code sent by the local end to the remote end. The configuration item RxCASFilterTime is used to set the minimum duration of the ABCD signaling send out by the remote PBX so as to eliminate the signal dithering on the line, appropriate for the E1/T1 line of low quality.

The minimum duration of the R2 signal can be set via the configuration item RxR2FilterTime or the function SsmSetFlag (with the parameter F_RXR2FILTERTIME).

When the application program exits, the configuration item IsBlockSS1In is used to decide whether to send the blocking command to the remote PBX, asking it not to start any new call towards the local end.

1.16.6.3 Setting Operation Mode for Channels in State Machine

The configuration item EnableAutoCall helps decide if to use the call state machine provided by the SynCTI driver; the function SsmSetAutoCallDirection or the configuration item AutoCallInTimeSlot is used to set the call direction for a channel. For example, a channel can be set to process incoming calls only or outgoing calls only as required.

1.16.6.4 China SS1 State Machine

1.16.6.4.1Number Hold-up Feature

The ‘Called Number Hold-up’ feature is often used by some key sectors for the purpose of eliminating the effect of the calling party’s misdialing on the application system. For example, a 110 alarm system set up on the Synway board will process the call only if the dialed number is ‘110’. That is, it will reject the call in detection of any other number, such as 1101.

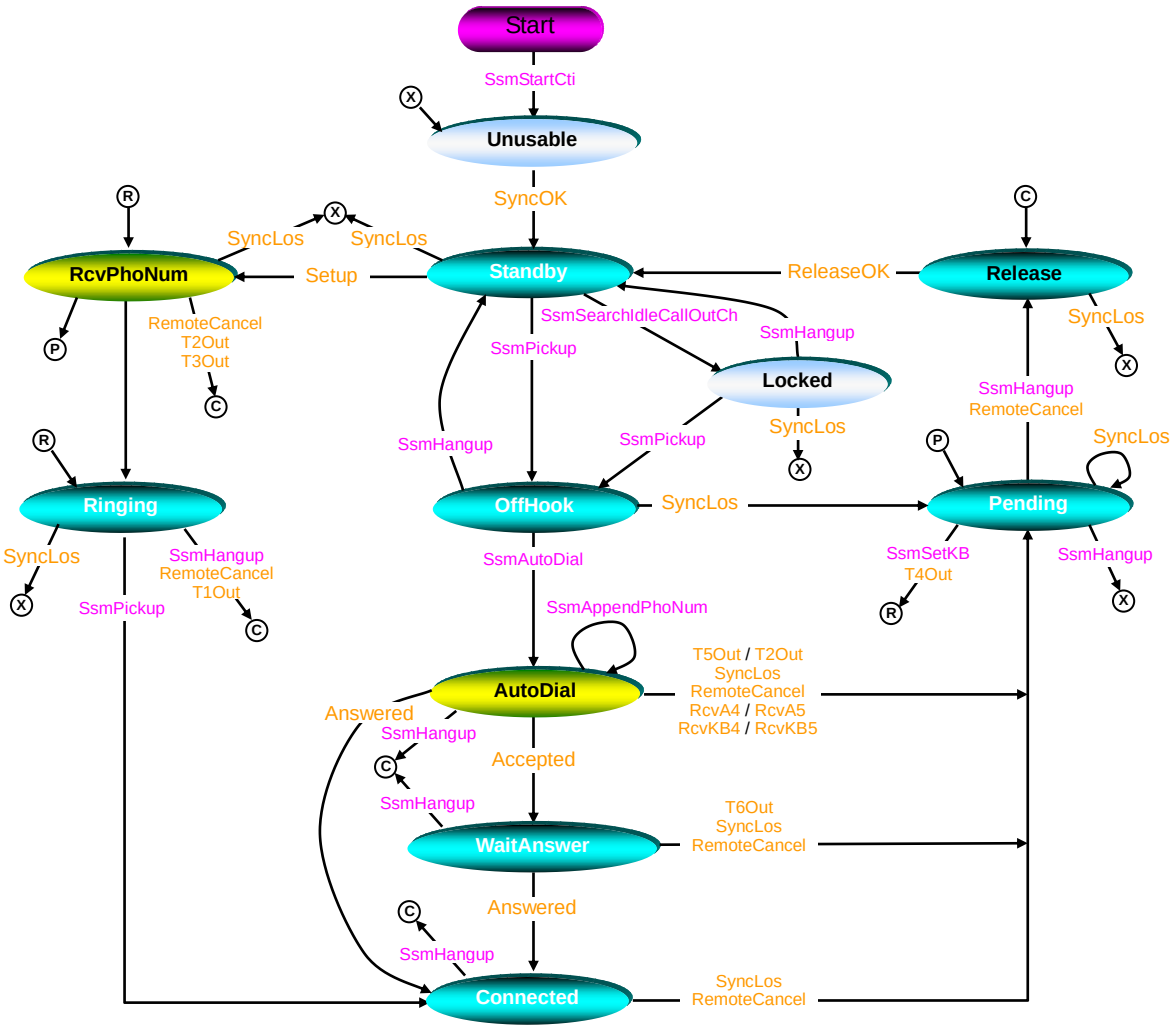
The ‘Called Number Hold-up’ feature as well as the way it works is controlled by the configuration item PhoNumHoldup or the function SsmSetFlag (with the parameter F_RCVPHONUMHOLDUP). Relative parameters can be set via the configuration items A1ToA3pWaitTime and A3pTime. For example, you can configure the 110 alarm system to hold up the called number in the form of 110x as follows.

```
PhoNumHoldup=1
A1ToA3pWaitTime=1000
A3pTime=150
```

1.16.6.4.2Connection to Dialogic SS1 Channel

While connecting the SS1 channel on the Synway board with the Dialogic SS1 channel to establish a call, you need to set the configuration item ToRingingDelayTime if the Synway board is required to serve as the incoming end.

1.16.6.4.3China SS1 State Machine



In the diagram above, the internal events automatically triggered by the driver itself are described as follows.

Event Name	Description
RemoteCancel	The remote PBX cancels the call.
SyncLos	The synchronization signal through TS0 on the digital trunk gets lost.
SyncOK	Detects the synchronization signal through TS0 on the digital trunk.
Setup	Receives the incoming call request from the remote PBX.
T1Out	The timer T1 overflows. T1 is started when the channel state transfers to ‘Ringing’, with the set value of 75sec.

T2Out	The timer T2 overflows. The SS1 channel is required to wait for the response signal from the remote PBX during the call process. T2 is used to set the maximum waiting time. The T2 overflow indicates this interactive process fails. T2 can be set via the configuration item MaxWaitMfcTime.
T3Out	The timer T3 overflows. In an incoming call, after the local end receives the complete called party number, the SS1 channel will send the A3 signal to the remote PBX and start T3 at the same time, waiting for the KD signal. If T3 overflows before the remote PBX sends out the KD signal, the incoming call progress fails. T3 can be set via the configuration item MaxWaitKDTime.
T4Out	The timer T4 overflows. In an incoming call, after the local end receives the complete called party number, the SS1 channel will go into the 'Pending' state and start T4 at the same time provided the configuration item is set to 0, which indicates the application wants to decide by itself whether to respond to this call. If T4 overflows before the application invokes the function SsmSetKB to reply, the driver will refuse this call. T4 can be set via the configuration item MaxWaitSetKBTime.
T5Out	The timer T5 overflows. After the application program calls the function SsmAutoDial to start an outgoing call, the driver will send the use request signal to the remote PBX and wait for the acknowledge signal from it. If T5 overflows before the reception of the acknowledge signal from the remote PBX, this call fails. T5 can be set via the configuration item MaxWaitOccupyAckTime.
T6Out	The timer T6 overflows. After the application invokes the function SsmAutoDial to start an outgoing call, the channel will start T6 provided the called party is idle. If T6 overflows before the called party picks up the phone, this outgoing call fails. T6 can be set via the configuration item MaxWaitAutoDialAnswerTime.
RcvA4	Receives the A4 signal (key congestion, the encounter with an unallocated number before the call reaches the called party) from the remote PBX in the outgoing call progress, indicating this call fails.
RcvA5	Receives the A5 signal (unallocated-number signal) in the outgoing call progress, indicating this call fails.
RcvKB4	Receives the KB signal of 4 (user's busy or key congestion) from the remote PBX in the outgoing call progress, indicating this call fails.
RcvKB5	Receives the KB signal of 5 (the called party number unallocated) from the remote PBX in the outgoing call progress, indicating this call fails.
Accepted	The remote PBX sends the KB signal of 1 or 6 to the local end after receiving the complete called party number in the outgoing call progress, indicating the called party is in the idle state.
Answered	The called party accepts the call in the outgoing call progress.
ReleaseOK	Finishes the disconnection from the remote PBX.

Each state identifier in the above diagram is described in the following table.

State Identifier	Description
Standby	'Idle' state The channel should go into this state provided it is normally reset after the successful driver initialization. The application can invoke the function SsmAutoDial to start an outgoing call in this state. The driver will automatically clear the Caller ID buffer when the channel state transfers to Idle at the end of the call, provided the configuration item AutoClearCallerIdBufOnHangup is set to 1
OffHook	'Off-hook' state If the SyncLos event is triggered in this state, the driver will ✧ set the progress value of the AutoDial task to DIAL_FAILURE and the failure reason to

	ATDL_PcmSyncLos, and throw out the E_PROC_AutoDial event to the application; ✧ set the pending reason to PEND_PcmSyncLos.
Ringing	The local end is ringing. The application program can ✧ invoke the function SsmGetCallerId to acquire the calling party information. ✧ call the function SsmPickup to accept the call or the function SsmHangup to refuse the call
Connected	Both parties are connected and able to talk with each other. When the channel state transfers to Talking, the driver will ✧ start the DTMF Detector; ✧ start the Barge-in Detector; ✧ clear the Tone Detector and start it. How the driver responds to the following events: ➢ The RemoteCancel Event: indicates the remote end first hangs up the phone in the call. The driver will set the pending reason to PEND_RemoteHangupOnTalking and then transfer the channel state to Pending; ➢ The SyncLos Event: ✧ The driver will set the progress value of the AutoDial task to DIAL_FAILURE and the failure reason to ATDL_PcmSyncLos, and throw out the E_PROC_AutoDial event to the application; ✧ set the pending reason to PEND_PcmSyncLos.
Pending	Waiting for further commands from the application. How the driver responds to the following events: ➢ The T4Out Event: The driver will automatically set the KB signal to 4 (key congestion) and resume the call progress. ➢ The application invokes the function SsmSetKB: The driver will set the pending reason to PEND_WaitBckStpMsg and resume the call progress for the channel. ➢ The SyncLos Event: ✧ The driver will set the progress value of the AutoDial task to DIAL_FAILURE and the failure reason to ATDL_PcmSyncLos, and throw out the E_PROC_AutoDial event to the application; ✧ set the pending reason to PEND_PcmSyncLos. ➢ The application invokes the function SsmHangup: If the pending reason is set to PEND_PcmSyncLos, the channel state will transfer to 'Unusable'; otherwise, the channel will go into the 'Release' state.
WaitAnswer	The channel is waiting for the called party to pick up the phone. The called party should be able to hear ringing tones in such state, and if at the local end is the station channel that makes an outgoing call through the digital trunk channel, it should send the ringback tone to the station channel. How the driver responds to the following events: ➢ The T6Out Event: ✧ The driver will set the progress value of the AutoDial task to DIAL_NOANSWER and the failure reason to ATDL_WaitRemotePickupTimeout, and throw out the E_PROC_AutoDial event to the application; ✧ set the pending reason to SS1OUT_NOANSWER. ➢ The SyncLos Event: ✧ The driver will set the progress value of the AutoDial task to DIAL_FAILURE and the failure reason to ATDL_PcmSyncLos, and throw out the E_PROC_AutoDial event to the application; ✧ set the pending reason to PEND_PcmSyncLos. ➢ The RemoteCancel Event: ✧ The driver will set the progress value of the AutoDial task to DIAL_FAILURE and the failure reason to ATDL_SS1RcvCAS_HANGUP, and throw out the E_PROC_AutoDial event to the application;

	✧ set the pending reason to SS1OUT_DIALING_BWD_HANGUP.
Unusable	The channel is unusable.
Locked	'Outgoing Call Locked' state
Release	'Disconnection' state
RcvPhoNum	It is an internal state during the incoming call progress implementation, including several substates. The driver judges the received called party number according to the preset number-receiving rule. If the number complies with the rule, the channel will transfer to the subsequent state. For more information about the number-receiving rule, refer to the section Number-receiving Rule for Incoming Call in this chapter. ➢ In an incoming call, when the channel receives the complete called party number and other necessary information, it is required to send the KB signal to the remote PBX. ✧ If the 'Auto-answer of Incoming Call' feature is enabled, the driver will send the value of the configuration item MfcKB as the KB signal to the remote PBX and transfer the channel state to Ringing; ✧ If the 'Auto-answer of Incoming Call' feature is not enabled, the driver will set PendingReason to PEND_WaitBckStpMsg, transfer the channel state to Pending, and let the application decide whether to accept this call or not. The 'Auto-answer of Incoming Call' feature can be set via the function SsmEnableAutoSendKB or the configuration item AutoSendKB, with the default value of 'Enabled'. ➢ Every time when the driver receives a calling party number, it will throw out the E_CHG_RxPhoNumBuf event.
AutoDial	It is an internal state during the outgoing call progress implementation, including several substates. **The parameters which can be set by the application for the outgoing call include:** ➢ The KA signal (Calling Party's Category) that should be provided by the local end for the remote PBX, configurable via the function SsmSetKA or the configuration item MfcKA; ➢ The KD signal (Originating Service Type) that should be provided by the local end for the remote PBX, configurable via the function SsmSetKD or the configuration item MfcKD; ➢ If the remote PBX asks the local end for the calling party number, the driver will automatically send the local end number saved in the buffer to the remote PBX after receiving the A6 signal and sending the KA signal. The local end number can be set via the function SsmSetTxCallerId or the configuration item TxCallerId. **In case the outgoing call fails, the driver will process as follows according to the internal event type.** ➢ The SyncLos Event: ✧ The driver will set the progress value of the AutoDial task to DIAL_FAILURE and the failure reason to ATDL_PcmSyncLos, and throw out the E_PROC_AutoDial event to the application; ✧ set the pending reason to PEND_PcmSyncLos. ➢ The T5Out Event: ✧ The driver will set the progress value of the AutoDial task to DIAL_NO_DIALTONE and the failure reason to ATDL_SS1WaitOccupyAckTimeout, and throw out the E_PROC_AutoDial event to the application; ✧ set the pending reason to SS1OUT_NOBWDACK. ➢ The RemoteCancel Event: ✧ The driver will set the progress value of the AutoDial task to DIAL_FAILURE and the failure reason to ATDL_SS1RcvCAS_HANGUP, and throw out the E_PROC_AutoDial event to the application; ✧ set the pending reason to SS1OUT_DIALING_BWD_HANGUP. ➢ The RcvA4 Event: ✧ The driver will set the progress value of the AutoDial task to DIAL_BUSYTONE and the failure reason to ATDL_SS1RcvA4, and throw out the E_PROC_AutoDial event to the application; ✧ set the pending reason to SS1OUT_BWD_A4. ➢ The RcvKB4 Event:

	✧ The driver will set the progress value of the AutoDial task to DIAL_BUSYTONE and the failure reason to ATDL_SS1RcvKB4, and throw out the E_PROC_AutoDial event to the application; ✧ set the pending reason to SS1OUT_BWD_KB4. ➢ The RcvA5 Event: ✧ The driver will set the progress value of the AutoDial task to DIAL_INVALID_PHONUM and the failure reason to ATDL_SS1RcvA5, and throw out the E_PROC_AutoDial event to the application; ✧ set the pending reason to SS1OUT_BWD_A5. ➢ The RcvKB5 Event: ✧ The driver will set the progress value of the AutoDial task to DIAL_INVALID_PHONUM and the failure reason to ATDL_SS1RcvKB5, and throw out the E_PROC_AutoDial event to the application; ✧ set the pending reason to SS1OUT_BWD_KB5. ➢ The T2Out Event: The driver will set the progress value of the AutoDial task to DIAL_FAILURE and the failure reason to one of the following values: ◆ATDL_SS1WaitAxTimeout ◆ATDL_SS1WaitAxStopTimeout ◆ATDL_SS1WaitAxTimeoutOnTxCallerId ◆ATDL_SS1WaitAxStopTimeoutOnTxCallerId ◆ATDL_SS1WaitKBTimeout ◆ATDL_SS1WaitKBStopTimeout respectively set the corresponding pending reason to: ◆SS1OUT_TIMEOUT_BWD_A ◆SS1OUT_TIMEOUT_BWD_A_STO ◆SS1OUT_TIMEOUT_CALLERID_BWD_A1 ◆SS1OUT_TIMEOUT_CALLERID_BWD_A1_STOP ◆SS1OUT_TIMEOUT_BWD_KB ◆SS1OUT_TIMEOUT_BWD_KB_STOP； and throw out the E_PROC_AutoDial event to the application;

Each state identifier and its corresponding value are listed in the table below.

State Identifier	Value	Corresponding Macro Definition in ShpA3Api.h
Standby	0	S_CALL_STANDBY
OffHook	1	S_CALL_PICKUPED
Ringing	2	S_CALL_RINGING
Connected	3	S_CALL_TALKING
Pending	7	S_CALL_PENDING
WaitAnswer	9	S_CALL_WAIT_REMOTE_PICKUP
Unusable	11	S_CALL_UNAVAILABLE
Locked	12	S_CALL_LOCKED
Release	36	S_CALL_SS1WaitIdleCAS
RcvPhoNum	—	S_CALL_Ss1InWaitPhoNum S_CALL_Ss1InWaitFwdStop S_CALL_Ss1InWaitCallerID S_CALL_Ss1InWaitKD S_CALL_Ss1InWaitKDStop S_CALL_SS1WaitIdleCAS S_CALL_SS1PhoNumHoldup S_CALL_Ss1InWaitStopSendA3p
AutoDial	—	S_CALL_Ss1OutWaitBwdAck S_CALL_Ss1OutTxPhoNum

		S_CALL_Ss1OutWaitAppendPhoNum S_CALL_Ss1OutTxCallerID S_CALL_Ss1OutWaitKB S_CALL_Ss1OutDetectA3p

The output events of the state machine are shown in the table below.

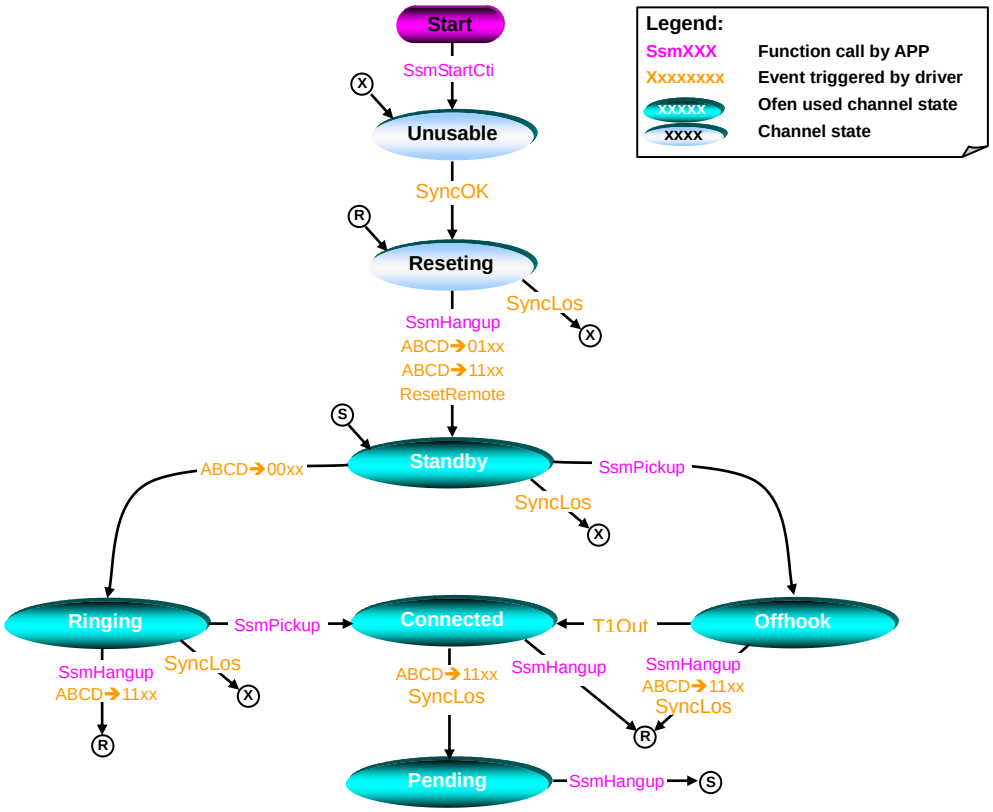
Event Name	Description
E_CHG_ChState	When the channel state changes, the driver will throw out this event to the application.
E_CHG_RxPhoNumBuf	Every time when the driver receives the called party information from the remote PBX, it will throw out this event to the application.
E_RCV_R2	When the application invokes the function SsmGetR2 to acquire the R2 message, the driver will throw out this event to the application if it receives a new R2 signal value from the remote end.
E_RCV_CAS	When the application program calls the function SsmGetCAS to acquire the current value of the ABCD signaling transmitted on the specified channel, the driver will throw out this event to the application if it receives the current CAS value.
E_CHG_CIDExBuf	The driver throws out this event when it detects the Extended Caller ID Buffer varies. The application can call the function SsmGetCallerIdEx to obtain the content in the buffer after receiving this event.

1.16.6.4.4Debugging Information Output during Outgoing Call

The configuration item MfcR2ToRxCallerIdBuf is used to control whether to save the R2 signal sent by the remote PBX during the call into the Extended Caller ID Buffer to facilitate the observation on the entire call process. And the function SsmGetCallerIdEx is used to take out strings form this buffer.

The configuration item Ss1LogEnable helps control whether to output the received/sent ABCD signaling codes and R2 signals to the log file Ss1Output.log.

1.16.6.5 Line Side State Machine



In the diagram above, the internal events automatically triggered by the driver itself are described as follows.

Event Name	Description
SyncLos	The synchronization signal through TS0 on the digital trunk gets lost.
SyncOK	Detects the synchronization signal through TS0 on the digital trunk.
ABCD→00xx	The ABCD signaling code of the remote PBX changes to 00xx.
ABCD→01xx	The ABCD signaling code of the remote PBX changes to 01xx.
ABCD→10xx	The ABCD signaling code of the remote PBX changes to 10xx.
ResetRemote	The driver has reset the circuit for the remote PBX, i.e. has sent the ABCD signaling code indicating the idle state to the remote PBX. The specific ABCD signaling code that indicates the idle state is determined by the configuration item Ss1SendIdleState.
T1Out	The timer T1 overflows. T1 is started when the channel state transfers to 'Off-hook', and can be set via the configuration item LSWaitPickup.

Each state identifier in the above diagram is described in the following table.

State Identifier	Description
Standby	'Idle' state The channel should go into this state provided it is normally reset after the successful driver initialization. If the application invokes the function SsmPickup in this state, the driver will send ABCD=11xx to the remote PBX.
OffHook	'Off-hook' state The application can send the called party number and other information to the remote PBX. How to send such information to the remote PBX depends on the PBX model. Query PBX

	manufacturers for details. T1 is started when the channel goes into the 'Off-hook' state.
Ringing	The local end is ringing. In this state, the application program can invoke the function SsmPickup to accept the call or the function SsmHangup to refuse the call.
Connected	'Talking' state. Both parties are connected and able to talk with each other. The driver will automatically start the DTMF detector when the channel state transfers to 'Connected'.
Pending	'Pending' state. Note that the driver won't set the pending reason. Hence there is no sense for the application to invoke the function SsmGetPendingReason.
Unusable	The channel is unusable.
Reseting	The circuit is being reset.

Each state identifier and its corresponding value are listed in the table below.

State Identifier	Value	Corresponding Macro Definition in ShpA3Api.h
Standby	0	S_CALL_STANDBY
OffHook	1	S_CALL_PICKUPED
Ringing	2	S_CALL_RINGING
Connected	3	S_CALL_TALKING
Pending	7	S_CALL_PENDING
Unusable	11	S_CALL_UNAVAILABLE
Reseting	36	S_CALL_SS1WaitIdleCAS

The output events of the state machine are shown in the table below.

Event Name	Description
E_CHG_ChState	When the channel state transfers, the driver will throw out this event to the application.

1.16.6.6 Advanced SS1 Programming Interface

The state machine provided by the SynCTI driver is enough for most application instances. If the application likes to perform the entire call process by itself, the low-level API functions can be used. The driver-provided state machine can be disabled via the configuration item EnableAutoCall or the function SsmSetAutoCallDirection after the system start-up. The driver can enable the mutual control between the SS1 stack and MFC by transmitting/receiving CAS and R2 after it disables the SS1 state machine.

The events related to the SS1 progress control include:

- E_RCV_CAS
- E_RCV_R2

The advanced programming interface functions related to the China SS1 progress control contain:

- SsmSendCAS
- SsmGetSendingCAS
- SsmGetCAS
- SsmSetRxR2Mode
- SsmGetR2
- SsmSendR2
- SsmSendR2Ex
- SsmStopSendR2
- SsmGetSendingR2

- SsmEnableRxDtmf

Note:

- **If the application program wants to receive DTMF digits after using the low-level progress control functions to finish the signaling progress, it must call the function SsmEnableRxDtmf to start the DTMF receiver and should call this function again to close the DTMF receiver at the end of the call.**
- **If the application is required to have the SS7 signaling cover for SS1 in a smooth way, it is not the functions in this section but the auto call progress functions provided by the system that should be used to control the call progress.**

1.16.6.7 SS1 Configuration

Below are the relative configuration items in the use of SS1.

Configuration Section	Configuration Item	
[BoardId=x]	***PcmNumber***	set the total number of on-board digital trunks as well as the signaling type, the clock operating mode and the line type for each digital trunk.
	PcmSSx	
	PcmClockMode	
	PcmLinkType	
	EnableAutoCall	auto SS1 progress control
	AutoCallInTimeSlot	sets the input and output direction of the SS1 channel to the digital trunk
[PcmInfo]	***TotalPcm***	set the total number of all digital trunks involved in the PC and build the mapping relationship between the logical number and the physical number of each digital trunk.
	Pcm	
[SS1Config]	TxCas_CD	set the parameters for SS1 CAS and R2 progress.
	RxCASFilterTime	
	MaxWaitMfcTime	
	IsBlockSS1In	
	tonesgroupA	
	tonesgroupB	
	tonesendofinfo	
	tonesanswer	
	tonesrepeatrequest	
	PcmSyncMask	
	ProtocolType	select the Lineside protocol type and set conditions for the channel state transition.
	LSWaitPickup	
	Ss1SendIdleState	
	mfcr2_Protocol	selects the country for the MFC-R2 protocol and sets the corresponding parameter.
	MaxWaitAutoDialAnswerTime	SS1 Outgoing Call: set relative parameters
	MaxWaitOccupyAckTime	
	MaxWaitKBTime	
	RepeatPhoNumOn1stR2bwdIsA5	
	MfcKD	SS1 Incoming Call: set relative parameters
	MfcKA	
	TxCallerId	
	MaxWaitSetKBTime	
	MaxWaitKDTime	
	ToRingingDelayTime	
	A1ToA3pWaitTime	
	MfcKB	
	AutoSendKB	

	PhoNumHoldup	
	A3pTime	
	DefaultRcvPhoNumLen	
	DefaultRcvCallerID	
	MaxPhoNumRule	
	Rule	
	MfcR2ToRxCallerIdBuf	set the log output for SS1 CAS and R2.
	IsBlockSS1In	
[DebugView]	Ss1LogEnable	set the log output for SS1 CAS and R2.
	Ss1LogCreateMode	
	LogCreatePeriod	
	LogMaxKeep	
	LogMaxPeriod	
	LogFilePath	

Note: Those written in the ***XXXX*** font are essentially configured items while those in XXXX are optionally configured ones.

1.16.7 Connection to Channel Bank

All SHD Series boards can connect directly to the channel bank to set up the large-capacity station system. After they are connected to the channel bank, the channel on them will be set to the station channel which has all features owned by the station channel on the SHT Series boards.

The configuration item UsageMode helps decide whether the SHD Series boards are used to start a call or connect to the channel bank.

The configuration item CBProtocolType is used to set the signaling protocol used in connecting the board to the channel bank.

The configuration item CBChannelType is used to set the channel type after the board connects to the channel bank.

After the SHD Series boards are connected to the channel bank, the outward features of the channel on them are identical to those of the corresponding channel on the SHT Series boards. Therefore, both the SHD and SHT channels in this case have the same configuration items and are configured in the same way. See description on the SHT boards for details.

1.17 SHV Series (CTI Series)

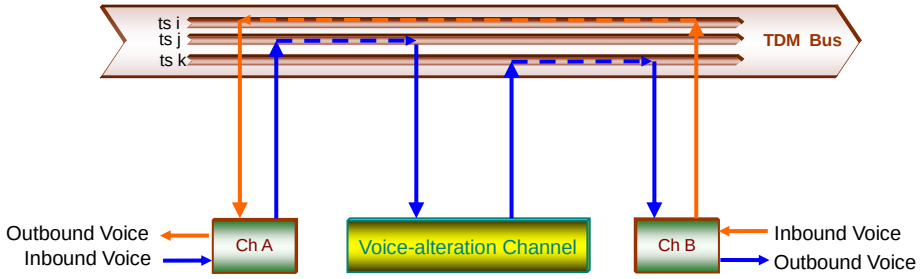
1.17.1 Brief Introduction of SHV Series

The features supported by the SHV Series boards are shown in the following table.

Board Model	Form Factor	Max. Ports	TDM Bus
SHV-120A-CT/PCI	PCI	120	H.100
SHV-240A-CT/PCI	PCI	240	H.100
SHV-240A-CT/cPCI	cPCI	240	H.110

1.17.2 Operation Principle of SHV Series

The figure below shows the typical application of the SHV Series boards.



In the figure above, the two-way connection is established between ChA and ChB via TDM bus. The incoming signals on ChB arrive at ChA through TS i and form the outgoing signals on ChA; the incoming signals on ChA enter the voice-alteration channel through TS j for processing, and then reach ChB via TS k and form the outgoing signal on ChB. In this way, the voice on ChB heard by ChA is the original voice while that on ChA heard by ChB is the altered voice.

1.17.3 List of Voice-alteration Functions

The SHV Series are the resource board and must be used with the SHD or SHT Series boards. Here is a list of functions related to voice alteration.

Function Name	Description
SsmGetMaxVCh	Obtains the total number of voice-alteration channels in the application system.
SsmGetMaxFreeVCh	Obtains the total number of idle voice-alteration channels
SsmBindVCh	Binds the voice channel to the voice-alteration channel.
SsmUnbindVCh	Unbinds the voice channel from the voice-alteration channel.
SsmSetVoiceEffect	Sets the voice alteration effect.
SsmGetVoiceEffect	Obtains the voice alteration effect
SsmSetVoiceEffectEx	Sets the voice alteration parameters for 240E VAR boards.
SsmGetVoiceEffectEx	Obtains the voice alteration parameters for 240E VAR boards.

1.18 SHN Series (CTI Series)

1.18.1 Brief Introduction of SHN Series

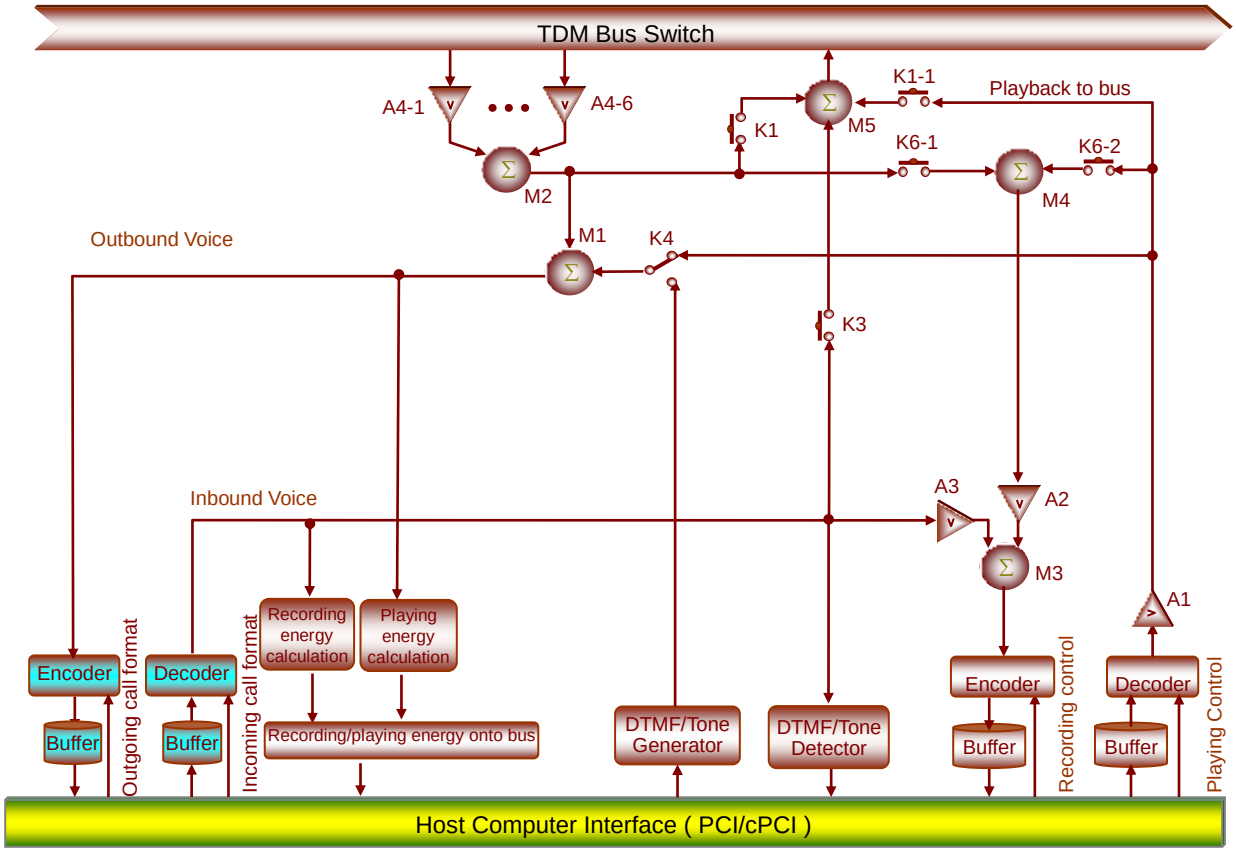
The features supported by the SHN Series boards are shown in the following table.

Board Model	Form Factor	Max. Ports	Voice Processing Capabilities	Conferencing Capabilities	Voltage Detector	TDM Bus	Max Fax Resource	SIP	H.323
SHN-32A-CT/PCI	PCI	32	√	√	√	√	N/A	√	N/A
SHN-8B-CT/PCI+	PCI	8	√	√	√	√	N/A	√	N/A
SHN-16B-CT/PCI+	PCI	16	√	√	√	√	N/A	√	N/A
SHN-32B-CT/PCI+	PCI	32	√	√	√	√	N/A	√	N/A
SHN-60B-CT/PCI+	PCI	60	√	√	√	√	N/A	√	N/A
SHN-120B-CT/PCI+	PCI	120	√	√	√	√	N/A	√	N/A
SHN-60B-CT/PCIe+	PCIe	60	√	√	√	√	N/A	√	N/A
SHN-120B-CT/PCIe+	PCIe	120	√	√	√	√	N/A	√	N/A
SHN-120B-CT/PCIe/VAR	PCIe	120	√	√	√	√	N/A	√	N/A

SHN-480C-CT/PCIe	PCIe	480	√	√	√	√	N/A	√	N/A

1.18.2 Operation Principle of SHN-32A-CT/PCI

The figure below shows the operation principle of the SHN-32A-CT/PCI board.



Below is the description on symbols in the figure above which are related to the recording operation:

Name	Description
M1	Outbound voice mixer
M2	Off-bus mixer
M3	Recording mixer The signals from M3 is sent to the Encoder component for recording purpose.
M4	Outbound voice mixer for recording Mixes signals from the off-bus mixer, data being played on a channel and signals from the tone generator/DTMF generator, and then puts them into M3.
M5	Onto-bus mixer Mixes data being played on a channel, incoming signals and those from the off-bus mixer M2, and then puts them onto TDM bus. Upon board initialization, the output of M5 is transported to a time slot (marked as 'ts' for short) on TDM bus. You can use the function SsmLinkToBus to designate the ts.
A1	The volume adjuster for voice playing. The volume can be set via the configuration item DefaultPlayVolume or the function SsmSetPlayVolume/SsmSetPlayGain, with the default value of 0.
A2	The volume adjuster which is used for outgoing signals before they are sent to M3. By default, it is on. The volume can be set via the function SsmSetRecMixer or the configuration item DefaultRecordMixerVolume, with the default value of -7.
A3	The volume adjuster which is used for incoming signals before they are sent to M3. The volume can be set via the function SsmSetRecVolume or the configuration item DefaultRecordVolume, with the default value of 0.
A4-1 … A4-6	The volume adjuster which is used for off-bus signals before they are sent to M2. The volume can be set via the function SsmSetListenVlmInConf, with the default value of 0DB.
K1-1	The switch that controls whether to put data being played on a channel first into M5 and then onto TDM bus or not, usually being off. It can be set via the function SsmSetPlayDest, used only for playing background music to a teleconference.

K1-2	The switch that controls whether to put signals from M2 back onto TDM bus. By default, it is off. It can be set via the function SsmSetFlag (with the parameter F_MixerResToBus).
K3	The switch that controls whether to put incoming signals into M5 or not. By default, it is on. It can be set via the function SsmSetFlag (with the parameter F_InVoiceToBus) or the configuration item InVoiceToBus.
K4	DTMF signals or tones cannot be sent at the same time when voices are played outbound.
K6-1 K6-2	Switch K6-1 controls whether to put signals form M2 into M3; Switch K6-2 controls whether to put voice data being played into M3. Both can be set via the function SsmSetRecBack.

➢ **Voice Playing**

After the application invokes the playing function, the decoded voice data after being processed by A1 will:

- ✧ Enter M1 and form the final outgoing signals with other signal sources; or
- ✧ Enter M4 under the control of K6-2 and form the recording signal sources with other signals; or
- ✧ Enter M5 under the control of K1-1 and form the onto-bus signal sources with other signals, only for playing background music to the teleconference.

➢ **Recording Example for Various Signal Sources**

A lot of special applications are available via the flexible use of the above switches for recording control and volume adjusters. In the following examples, ch1 and ch2 respectively mean Channel 1 and Channel 2.

✧ Example 1: Only record the incoming signals on ch1 and use the default volume.

```
SsmSetRecVolume(ch1, 0);          //start A3, set the volume gain to 0DB
SsmSetRecMixer(ch1, FALSE, 0);    //stop A2
SsmRecToFile(ch1, ……);             //start the task of file recording …
```

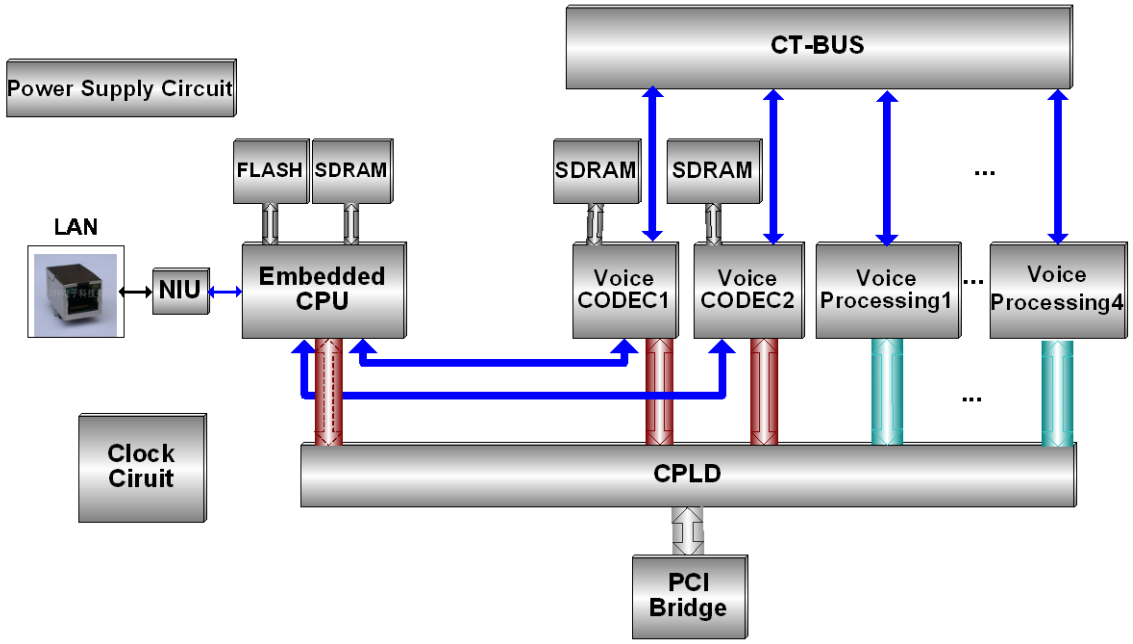
✧ Example 2: Record the incoming signals on ch1 (volume increased by 6DB) and the signals output from M2 (volume decreased by 3DB) at one time based on the establishement of a two-way call between ch1 and ch2.

```
SsmTalkWith(ch1, ch2);            //establish a two-way connection between ch1 and ch2
SsmSetRecBack(ch1,2);             //switch on K6-1, switch off K6-2
SsmSetRecVolume(ch1, 2);          //start A3, set the volume gain to 2*3DB=6DB
SsmSetRecMixer(ch1, TRUE, -1);    //start A2, set the volume gain to -1*3DB=-3DB
SsmRecToFile(ch1, ……);             //start the task of file recording …
```

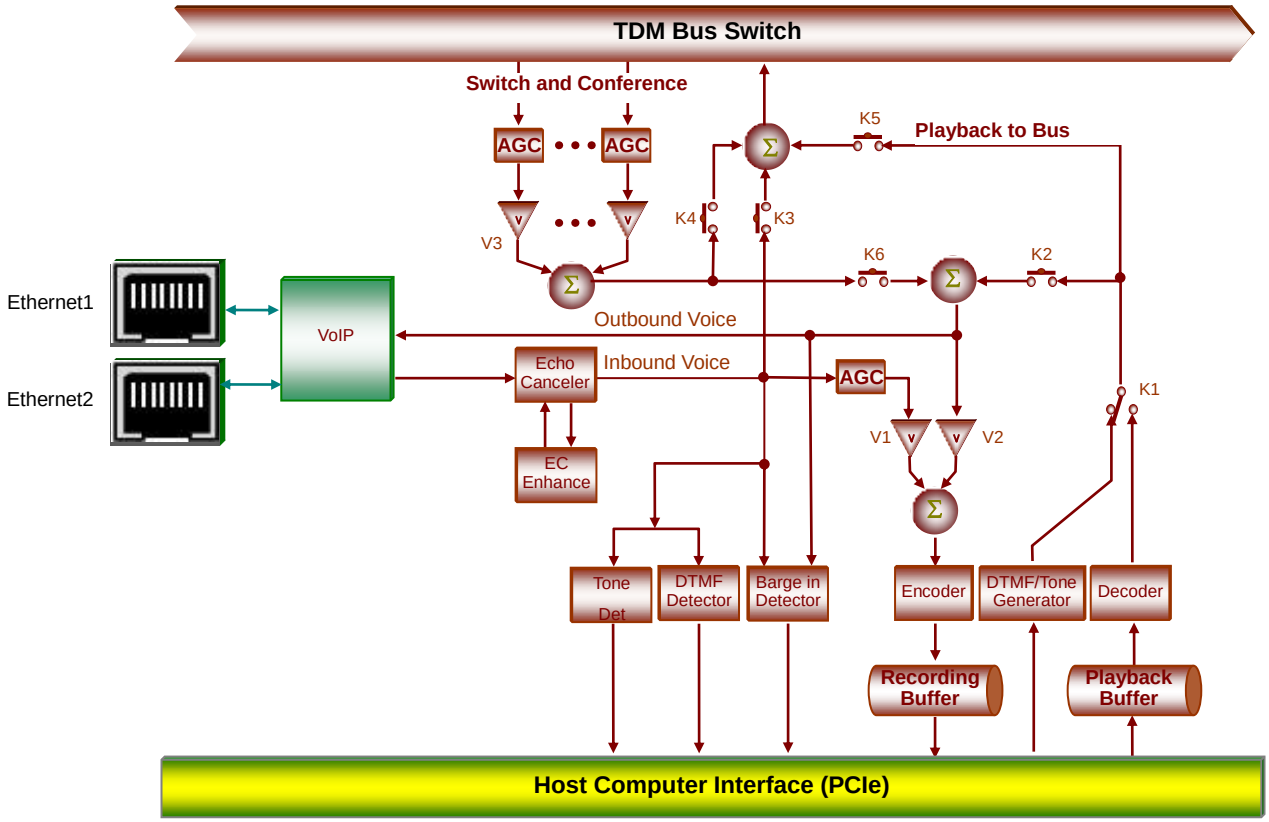
✧ Example 3: When ch1 joins a teleconference (assuming the conference room number is n) as the organizer, play background music on ch1 and record the conference (including the background music) through ch1. All signal sources go with normal volume, i.e. the gain is set to 0.

```
SsmJoinConfGroup(n, ch1, ……);     //ch1 joins the conference: switch on K3
SsmSetPlayDest(int ch, 1);        //switch on K1-1
SsmSetRecBack(ch1,1);             //switch on K6-1 and K6-2
SsmPlayFile(ch1, ……);              //start the task of voice playing on ch1: switch on K4 to link left and right ends
SsmSetRecVolume(ch1, 0);          //start A3, set the volume gain to 0DB
SsmSetRecMixer(ch1, TRUE, 0);     //start A2, set the volume gain to 0DB
SsmRecToFile(ch1, ……);             // start the task of file recording …
```

1.18.3 Operation Principle of SHN B Series



1.18.4 Operation Principle of SHN C Series



1.18.5 SIP Channel Programming

1.18.5.1 Basic Concepts

- **SIP Message**

There are two types of SIP message:

- Request: sent from Client to Server
- Response: sent from Server to Client

The table below lists all requests supported by the SIP channel on the Synway SHN Series boards.

Method	Description
INVITE	to establish a session between participants; or to modify an existing session (re-INVITE)
ACK	to confirm the reception of the final response to INVITE
BYE	to terminate a session
CANCEL	to terminate a call before it has been answered
OPTIONS	to allow a UA to query another UA or a proxy server as to its capabilities
REGISTER	to add, remove, and query bindings between address-of-record and one or more contact addresses
INFO	to carry application level information along the SIP signaling path, including out-of-band DTMF digits for the SHN Series boards
MESSAGE	to send instant messages
SUBSCRIBE	to request current state and state updates from a remote node
NOTIFY	to inform subscribers of changes in state to which the subscriber has a subscription

➢ **Message Format**

An SIP message consists of:

- a start-line
- one or more header fields
- an empty line indicating the end of the header fields
- an optional message-body

The start-line, each message-header line, and the empty line must be terminated by a carriage-return line-feed sequence (CRLF). Note that the empty line must be present even if the message-body is not.

➢ **SDP**

SDP is a session description protocol for multimedia sessions (RFC2327). It specifies the way to encode necessary information about the session. The SDP message is carried by the SIP message in a way that is analogous to a document attachment being carried by an email message, or a web page being carried in an HTTP message.

An SDP package usually contains both session information and media information.

Session information includes the following:

- The owner and identifier of the session
- Time(s) the session is active

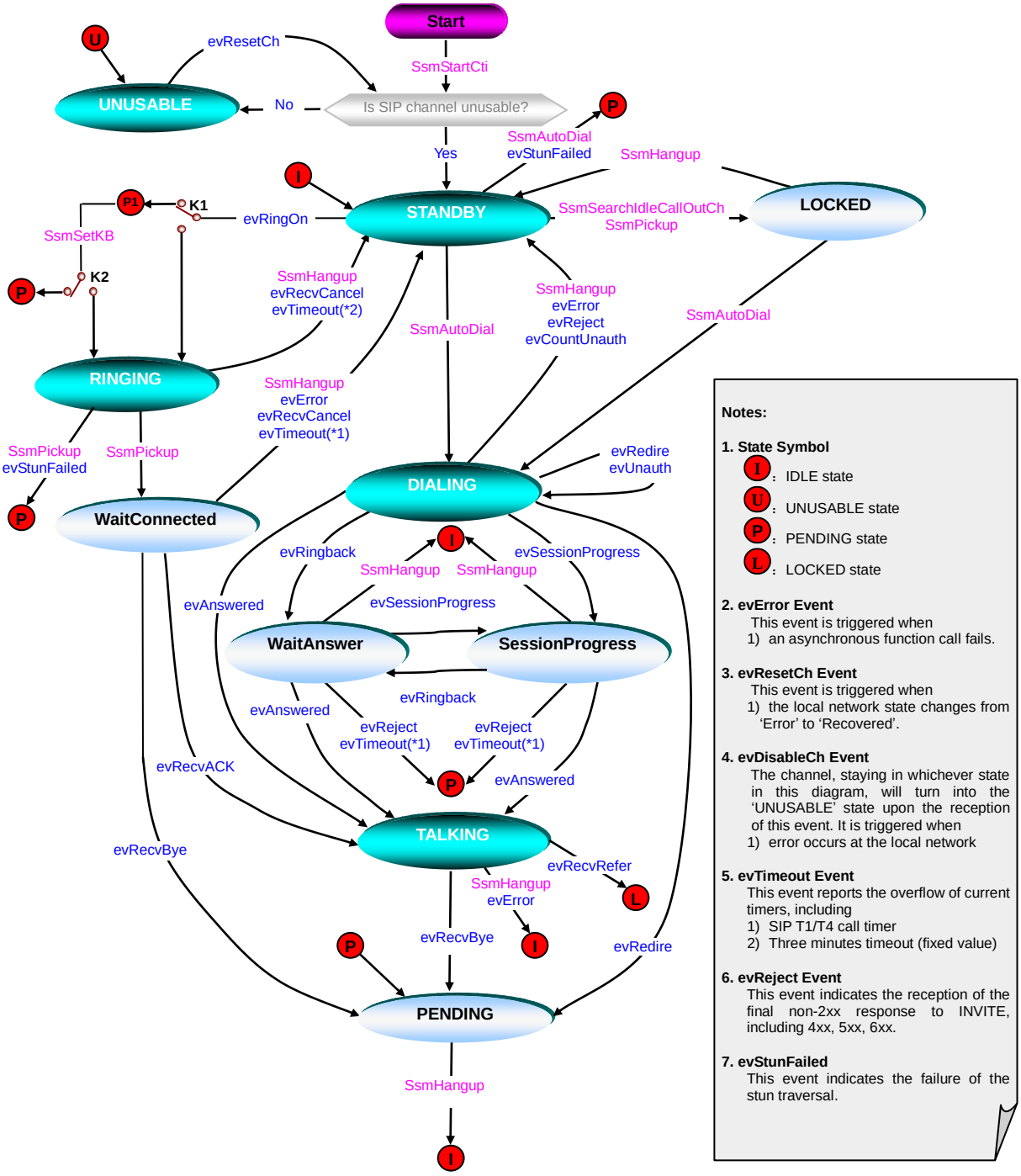
As resources necessary to participate in a session may be limited, some additional information may also be desirable:

- Information about the bandwidth to be used by the session
- Contact information for the person responsible for the session (E-Mail address, phone number, etc)

Media information includes the following:

- The type of media (video, audio, etc)
- The transport protocol (RTP/UDP/IP, H.320, etc)
- The format of the media (H.261 video, MPEG video, etc)
- Multicast address for media and transport port for media (IP multicast session)
- Remote address for media and Transport port for contact address (IP unicast session)

1.18.5.2 SIP Channel State Machine



Each state identifier in the above diagram is described in the following table.

State Identifier	Description
Unusable	'Unusable' state ✧ The channel, staying in whichever state in the above diagram, will turn into the 'UNUSABLE' state upon the reception of the evDisableCh event.
Standby	'Idle' state The channel will go into this state just after the successful board initialization. ✧ When the function SsmAutoDial is invoked successfully to start a call, the channel will turn into the 'DIALING' state. If the function call of SsmAutoDial is failed due to the stun traversal failure, the channel will turn into the 'Pending' state.

	✧ When the function SsmSearchIdleCallOutCh is invoked to find an idle channel, it will turn into the 'Locked' state. ✧ When the function SsmPickup is invoked, an idle channel will be locked and turn into the 'Locked' state. ✧ When an idle channel is available upon the reception of the event evRingOn (indicating the channel receives an incoming call), the application determines the channel state transition via the switch K1 which is set by the configuration item UseSipKB. If the configuration item UseSipKB is set to 1, the channel will turn into the state 'Pending 1' (waiting for APP to set the backward set-up message); if the configuration item UseSipKB is set to 0, the channel will turn into the 'Ringing' state. For a channel in the 'Pending 1' state, when the function SsmSetKB is invoked to accept or reject the incoming call, the application determines the channel state transition via the switch K2 which is set by the parameter btSigKB of the function SsmSetKB. If btSigKB=1, the channel will turn into the 'Ringing' state; if btSigKB=2, the channel will turn into the 'Pending' state. ✧ When no idle channels are available upon the reception of the event evRingOn (indicating the channel receives an incoming call), A3 will automatically send the response code 403 and set the reason. ✧ When the event evDisableCh (indicating network error or usable channel unavailable) is received, the channel will turn into the 'Unusable' state.
DIALING	'Waiting for Answer' state The channel will keep in this state until it receives responses from the remote end. ✧ In this state, if the evRingback event (indicating the reception of the response code 180 from the remote end) is received, the channel will turn into the 'WaitAnswer' state. ✧ In this state, if the evSessionProgress event (indicating the reception of the 183 response from the remote end) is received, the channel will turn into the 'SessionProgress' state. ✧ If the evRedire event (indicating the reception of the response code 3xx from the remote end) is received, SynSIP will call again on the same channel and automatically send an ACK message to the remote. ✧ If the evRedire event (indicating the reception of the response code 3xx from the remote end) is received, but the 3xx message is incomplete or the SynSIP failed to analyze the 3xx message, the channel will turn into the 'Pending' state. ✧ If the evUnauth event (indicating the reception of the 401/407 response from the remote end) is received, SynSIP will automatically send a call request with authentication information to the remote. ✧ If the evAnswered event (indicating the reception of the 2xx response from the remote end) is received, the channel will turn into the 'Talking' state and SynSIP will automatically send an ACK message to the remote at the same time. ✧ If the evReject event (the call is refused by the remote end), the evError event (the dial function is asynchronous), the evTimeout event (the timer overflows) and the evCountUnauth event (unauthorized) are received, the channel will turn into the 'Standby' state and set the reason as well. At this time, SynSIP will determine whether to send an ACK message to the remote according to the current events (evErrAnsw and evCountUnauth). ✧ If the function SsmHangup is invoked to hang up the phone in this state, the channel will turn into the 'Standby' state and SynSIP will automatically send a CANCEL message to the remote at the same time. ✧ SynSIP will start timers when the channel goes into the 'Dialing' state. If the application doesn't change in state before the timer overflows, the driver will reset the channel back to the idle state. ✧ If the event evDisableCh (indicating network error or usable channel unavailable) is received, the channel will turn into the 'Unusable' state.
Locked	'Outgoing Call Locked' state ✧ When the function SsmSearchIdleCallOutCh is invoked by the application, the driver will temporarily lock the channel specified by the function's return value

	✧ If the event evDisableCh (indicating network error or usable channel unavailable) is received, the channel will turn into the 'Unusable' state.
WaitAnswer	'Ringback' state ✧ In this state, if the evSessionProgress event (indicating the reception of the 183 response from the remote end) is received, the channel will turn into the 'SessionProgress' state. ✧ In this state, if the evAnswered event (indicating the reception of the 2xx response from the remote end) is received, the channel will turn into the 'Talking' state. ✧ If the evReject event (the call is refused by the remote end) and the evTimeout event (the timer overflows) are received, the channel will turn into the 'Pending' state and set the reason as well. At this time, SynSIP will determine whether to send an ACK message to the remote according to the current error reason. ✧ If the function SsmHangup is invoked to hang up the phone in this state, the channel will turn into the 'Standby' state and SynSIP will automatically send a CANCEL message to the remote at the same time. ✧ If the event evDisableCh (indicating network error or usable channel unavailable) is received, the channel will turn into the 'Unusable' state.
SessionProgress	'Session Progress' state ✧ In this state, if the evRingback event (indicating the reception of the 180 response from the remote end) is received, the channel will turn into the 'WaitAnswer' state. ✧ In this state, if the evAnswered event (indicating the reception of the 2xx response from the remote end) is received, the channel will turn into the 'Talking' state. ✧ If the evReject event (the call is refused by the remote end) and the evTimeout event (the timer overflows) are received, the channel will turn into the 'Pending' state and set the reason code as well. At this time, SynSIP will determine whether to send an ACK message to the remote according to the current error reason. ✧ If the function SsmHangup is invoked to hang up the phone in this state, the channel will turn into the 'Standby' state and SynSIP will automatically send a CANCEL message to the remote at the same time. ✧ If the event evDisableCh (indicating network error or usable channel unavailable) is received, the channel will turn into the 'Unusable' state.
Ringing	'Called Party Ringing' state When an idle channel is available upon the reception of an incoming call, SynSIP will automatically react to the 180 response to lock the idle channel and turn it into this state. ✧ In this state, if users are determined to accept the current call request and invoke the function SsmPickup successfully, the channel will turn into the 'WaitConnected' state and SynSIP will automatically send the 200 OK response to the remote at the same time. If the function call of SsmPickup is failed due to the stun traversal failure, the channel will turn into the 'Pending' state. ✧ If users choose to refuse the call and invoke the function SsmHangup, the channel will turn into the 'Standby' state and SynSIP will automatically generate a 403 Forbidden response and send it to the remote at the same time. ✧ If the evRecvCancel event (indicating the call is cancelled by the calling party) is received, the channel will turn into the 'Standby' state and SynSIP will automatically generate a 200 OK response to the CANCEL message. ✧ If the evTimeout event (indicating the timer overflows) is received, the channel will turn into the 'Standby' state and SynSIP will automatically generate a 400 Bad Request response and send it to the remote at the same time. ✧ If the event evDisableCh (indicating network error or usable channel unavailable) is received, the channel will turn into the 'Unusable' state.
WaitConnected	Waiting for connection' state ✧ If the function SsmHangup is invoked to hang up the phone in this state, the channel will turn into the 'Standby' state and SynSIP will automatically send a BYE message to the calling party at the same time. ✧ If the evRecvCancel event (indicating the call is cancelled by the calling party) is received, the channel will turn into the 'Standby' state and SynSIP will automatically

	generate a 200 OK response to the CANCEL message. ✧ If the evRecvAck event (indicating the connection of the calling party) is received, the channel will turn into the 'Talking' state. ✧ If the evRecvBye event is received, the channel will turn into the 'Pending' state and SynSIP will generate a 200 OK response to the BYE message ✧ If the evError event and the evTimeout event (indicating the timer overflows) are received, the channel will turn into the 'Standby' state. ✧ If the event evDisableCh (indicating network error or usable channel unavailable) is received, the channel will turn into the 'Unusable' state.
Talking	'Talking' state All voice channels are connected in this state. ✧ In this state, both calling and called parties can invoke the function SsmHangup to hang up the phone. Then SynSIP will generate a BYE message and send it to the remote and the channel will turn into the 'Standby' state at the same time. ✧ If the evRecvBye event (indicating the phone is hung up by the remote end) is received, the channel will turn into the 'Pending' state. ✧ If the event evDisableCh (indicating network error or usable channel unavailable) is received, the channel will turn into the 'Unusable' state. ✧ If the evRecvRefer event (indicating the reception of the 'refer' message) is received, the channel will turn into the 'Locked' state.
Pending	'Pending' state ✧ When the 'Call Terminated' state as specified by the SIP protocol is received during the call state transition, the call cannot be connected any longer and users must release all resources allocated throughout the call process and reset the circuit. Only via the call of SsmHangup can the channel go back to the 'Standby' state. ✧ If the event evDisableCh (indicating network error or usable channel unavailable) is received, the channel will turn into the 'Unusable' state.

Each state identifier in the above diagram is described in the header file ShpA3Api.h as follows.

State Identifier	Value in ShpA3Api.h	Macro Definition in ShpA3Api.h
Unusable	134	S_CALL_VOIP_CHANNEL_UNUSABLE
Standby	0	S_CALL_STANDBY
Dialing	132	S_CALL_VOIP_DIALING
Locked	12	S_CALL_LOCKED
WaitAnswer	9	S_CALL_WAIT_REMOTE_PICKUP
Ringing	2	S_CALL_RINGING
WaitConnect	133	S_CALL_VOIP_WAIT_CONNECTED
Talking	3	S_CALL_TALKING
Pending	7	S_CALL_PENDING

In the above state machine, the driver's internal component for judging the channel's usability has two values to output as shown in the table below.

Output Value	Description
Unusable	The channel is unusable if one of the events evDisableCh, DataLinkDisconnect, Circuit Reset keeps valid.
Standby	The channel goes into the idle state provided the evDisableCh event becomes invalid, the DataLinkConnect event keeps valid and the circuit restart is successful.

Note: At this stage, the driver also performs the following operations.

- Clear and close the tone detector;
- Automatically terminate the WaitDtmf task if it has been started by the application;
- Automatically terminate the task of DTMF transmission if it has been started by the application;
- Automatically clear the buffer of the DTMF detector, and close the DTMF detector if the configuration item AlwaysEnableRxDtmf is set to 0;
- Automatically close the Barge-in detector if the configuration item AlwaysDetectBargeIn is set to 0;

- Automatically clear the Caller ID buffer if the configuration item AutoClearCallerIdBufOnHangup is set to 1;
- Reset all timers in SynSIP.

The output events of the state machine are described in the following table.

Event Name	Description
E_CHG_ChState	The driver sends out this event to the application when the channel state changes.
E_PROC_AutoDial	The driver throws out this event to the application when there is any progress in the AutoDial task which is started by the function call of SsmAutoDial with the application.

Note: Event declarations can be found in the file ShpA3Api.h.

1.18.5.3 SIP Channel Configuration

Configuration items for using the SHN-32A-CT/PCI board with SIP protocol are listed below.

Configuration Section	Configuration Item	
[BoardId=x]	***ProtocolType***	Set the IP communication protocol for the VoIP board or set the VoIP board to be a VoIP resource board; Set the DTMF reception mode for the board.
	RecvDtmfType	
	DisplayName	Set operation parameters for an SIP session. They are in sequence: ✧ Displayed name ✧ DTMF transmission mod ✧ Adopted voice CODECs
	SendDtmfType	
	AudioCodecList	
	UserName	Set the parameters of the Register request. They are in sequence: ✧ Register a user name upon the start of registration, or set the part before @ in SIP Uri as the user name. ✧ Whether to enable the registration ✧ Address of the registered server ✧ Password for registration (if necessary) ✧ Alias of the SIP server ✧ Register a valid expiration time, with the default value of 3600 and the minimum value of 300.
	Register	
	Domain	
	RegPassword	
	RegRealm	
	RegExpires	
[SIP]	***LocalSipIp***	Set the intercepted address and port for the SIP protocol.
	LocalSipPort	
	RTPRange	RTP port range
	LogFile	Set the advanced board configuration items. They are in sequence: ✧ Log file ✧ Log level ✧ Size of the thread pool to capture SIP events ✧ Port heartbeat interval (for NAT traversal)
	LogLevel	
	EventThreadNum	
	HeartInterval	
	SipTransportProtocol	Set whether SIP is used over TCP or UDP
	RetransmitLost200OK	Sets if it is necessary for the board to wait for the ACK message after sending the 200OK message to establish a call.

Configuration items for using the SHN B and C Series boards with SIP protocol are listed below.

Configuration Section	Configuration Item	
[BoardId=x]	***ProtocolType***	Set the IP communication protocol and DTMF reception mode used by the board.
	RecvDtmfType	
	BoardIP	Set network card information for the SHN B and SHN C series. They are in sequence: ✧ IP address of the network card (voice channels)
	Submask	
	Gateway	

	DNS	✧ Subnet mask ✧ Gateway address DNS address
	RTPRange	Set operation parameters for an SIP session. They are in sequence: ✧ RTP port range ✧ Displayed name ✧ DTMF transmission mode Adopted voice CODECs
	DisplayName	
	SendDtmfType	
	AudioCodecList	
	UserName	Set the parameters of the Register request. They are in sequence: ✧ Register a user name upon the start of registration, or set the part before @ in SIP Uri as the user name. ✧ Whether to enable the registration ✧ Address of the registered server ✧ Password for registration (if necessary) ✧ Alias of the SIP server Register a valid expiration time, with the default value of 3600 and the minimum value of 300.
	Register	
	Domain	
	RegPassword	
	RegRealm	
	RegExpires	
[SIP]	***LocalSipIp***	Set the intercepted address and port for the SIP protocol.
	LocalSipPort	
	RTPRange	RTP port range
	LogFile	Set the advanced board configuration items. They are in sequence: ✧ Log file ✧ Log level ✧ Size of the thread pool to capture SIP events Port heartbeat interval (for NAT traversal)
	LogLevel	
	EventThreadNum	
	HeartInterval	
	SipTransportProtocol	Set whether SIP is used over TCP or UDP
	RetransmitLost200OK	Sets if it is necessary for the board to wait for the ACK message after sending the 200OK message to establish a call.

Note: Those written in the ***XXXX*** font are essentially configured items while those in XXXX are optionally configured ones.

1.18.6 Special Parameters and Structure for VoIP Resource Boards

1.18.6.1 RTP Transmit and Receive Modes for VoIP Resource Boards

```
#define IPM_SENDRECV 0x0000 // Receive and transmit
#define IPM_SENDONLY 0x0001 // Transmit only
#define IPM_RECVONLY 0x0002 // Receive only
```

1.18.6.2 Special Structure for VoIP Resource Boards

```
struct MediaParam
{
    int mode;
    char localIP[50];
    int localPort;
    char remoteIP[50];
    int remotePort;
    int sendCodecType;
```

```
    int    dtmfpayload;
};
```

mode: RTP transmit and receive modes, having the following three values.

IPM_SENDRECV (Receive and transmit)

IPM_SENDONLY (Transmit only)

IPM_RECVONLY (Receive only)

localIP: The local address intercepted in RTP.

localPort: The local port intercepted in RTP.

remoteIP: The remote address sent in RTP.

remotePort: The remote port sent in RTP.

SendCodecType: The codec for RTP load. 0: 711U, 8: 711A, 18: G729, 3: GSM, 4: G723_1, 9: G722, 96: AMR, 98: ILBC.

dtmfpayload: DTMF digits sent in RFC2833.

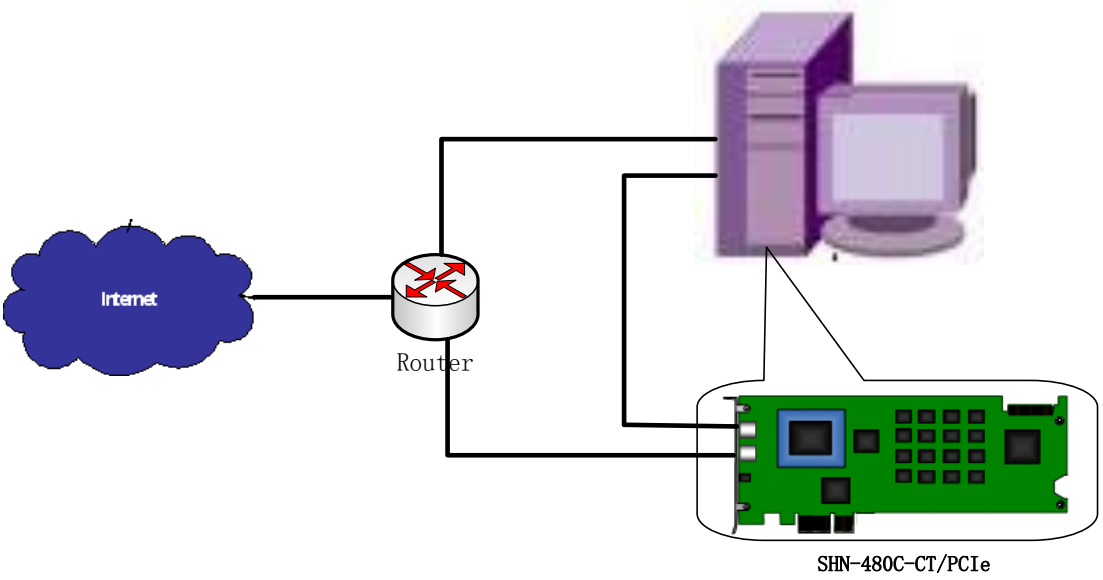
1.18.7 Special Configuration Item for SHN C Series Board

Configuration Section	Configuration Item	
[**BoardId=x**]	*OctMac0*	Sets the MAC address of on-board Ethernet 1
	OctMac1	Sets the MAC address of on-board Ethernet 2
[DHCP]	*DhcpServer*	Sets the IP address of DHCP server
	ProcessorCtrlMac	Sets the MAC address of the on-PC Ethernet port for control messages
	FilterMacRange	Sets the MAC address whose IP can be allocated by the DHCP server
	BootFileName	Sets the name of the firmware to be loaded by the DHCP server
	LogLevel	Sets the log for DHCP and TFTP. None means to disable the log feature
	DHCPRange	Sets the range of the DHCP IP address

1.18.8 Environment and Application of SHN C-type Board

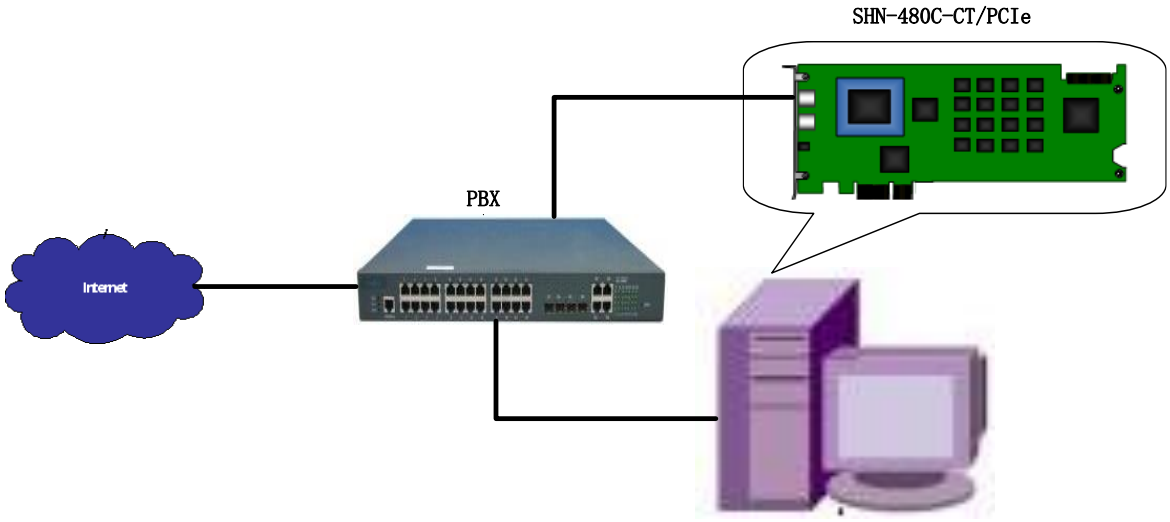
SHN-480C-CT/PCIe has two Kilomega Ethernet ports and provides multiple connection methods.

Connection Method 1 (recommended):



Note: This method uses the two Ethernet ports on the board, Ethernet 1 connecting to the PC while Ethernet 2 connecting to the Router. Meanwhile, it requires more than one network interface on the PC. As the firmware chips are loaded through Ethernet1 at the board startup, the board will fail to start without connecting Ethernet1 to the computer's network interface. That is, you shall ensure a proper connection between the Ethernet1 and the computer's network interface, configured with a statically IPv4 address which had better be not in the same network segment with other network interfaces. Otherwise, the load of the firmware may be failed.

Connection Method 2:



Note: This method only uses Ethernet1 on the board. As the firmware is loaded through Ethernet1 at the board startup, the board will fail to start without connecting Ethernet1 with the network cable. That is, you shall ensure a proper connection of both Ethernet1 and a network interface on the PC to the PBX. Here we suggest you to use a PBX instead of a router, as the router may allocate incorrect IP addresses and cause the start failure of the board.

1.18.9 Operation Principle of SHN C-type Board

1. The on-board chip loads the firmware via the communication between Ethernet1 and PC.

To ensure the normal operation of the on-board chip, it is required to load the firmware. The firmware will be loaded instantly, that is, they're loaded through Ethernet1 (fixed) at the board startup. The loaded firmware will be stored in the computer after the driver installation, so the on-board chip needs to load them through network. Note that the firmware must be reloaded upon the board reset or power off. So you shall ensure a proper connection of both Ethernet1 and PC.

OCT, the on-board chip, will be reset at the board startup and start two treads automatically: TFTP server and DHCP client. By default, the TFTP server starts working from IP address 192.168.0.201 and the DHCP client will send DHCP Discover to the connecting network every 1s. Meanwhile, the driver will start two programs OpenDHCPServer and OpenTFTPServerMT to load the firmware via the interaction between OCT's Ethernet1 and the computer's network interface.

Method A Load the firmware through two driver programs OpenDHCPServer and OpenTFTPServerMT (there is no other DHCP server existing in the network).

Step 1: Upon receiving a DHCP Discover request from the DHCP client, OpenDHCPServer will reply a DHCP Offer message to inform the IP address (such as 192.168.10.250) located to OCT, the Server IP address and the name of the firmware to be loaded (Boot file name: oct2200.img). The configuration Item ***DHCPRange*** is used to designate the IP range for OpenDHCPServer (e.g. ***DHCPRange***=192.168.10.250-192.168.10.254). No IP address within this range should be occupied, otherwise the loading of firmware will fail due to IP conflict. The configuration item ***DhcpServer*** is used to designate the Server IP address of OpenDHCPServer (e.g.***DhcpServer***=192.168.10.10). The configuration item ***BootFileName*** is used to designate the name of the firmware obtained by OCT (e.g. ***BootFileName***= oct2200.img).

Step 2: After the DHCP client receives the Offer message, OCT will stop the DHCP client, modify the IP address of Ethernet1, and at the same time examine if there are Server IP or Boot file name existing in the Offer message. Now there are two situations as stated below:

1) If both Server IP and Boot file name exist in the Offer message, the TFTP server will be stopped and a TFTP read request will be sent from Ethernet1 to Server IP. And then go to Step 3.

2) If either Server IP or Boot file name doesn't exist in the Offer message (e.g. the configuration item ***BootFileName*** has been removed), OCT will modify the default IP address of the TFTP server (192.168.0.201) to the allocated one (192.168.10.250). Then the board will start the TFTP client's thread to send the write request to the address 192.168.10.250 and go to Step 4.

Step 3: OpenTFTPServerMT will send the firmware designated by ***BootFileNam*** after it detects the TFTP read request message and will not stop sending until the firmware is loaded successfully.

Step 4: After the TFTP server receives the write request message, it will stop the DHCP client and start to receive the firmware package sent from the TFTP client of the board, and will not stop receiving until the firmware is loaded successfully. In this situation, it is not necessary to designate the firmware in the configuration item ***BootFileName*** as the driver program already knows which firmware should be sent to OCT.

Method B Load the firmware by the driver program's start of the TFTP client (There is other DHCP server existing in the network, such as a router with the DHCP feature).

This method is similar to **2)** of **Step 2** in **Method A**. You shall make sure that the IP address assigned by the

network device with DHCP feature is in the same network segment as the PC. Otherwise, the loading of firmware will fail. This method is also applicable to such situation that the IP address of the PC and the default IP of the TFTP server (192.168.0.201) are in the same network segment.

Note:

- **Method A** is the default method to load the firmware. Only if the loading of firmware by **Method A** fails, or the configuration item ***BootFileName*** is removed or never configured, will **Method B** be used.
- Make sure the ports 67, 68 and 69 will never be forbidden by firewall as they are necessary for IP allocation by DHCP and data transmission via TFTP; otherwise, the loading of firmware will probably fail. Therefore, an easy solution is just to close the firewall.

2. SIP signaling and RTP transmission

In spite of the connection method, the SIP signaling will be only carried through the computer's network interface. However, the transmission way of RTP varies according to the connection method. If Connection Method 1 is used, the RTP will be carried through Ethernet2 and the host control OCT message will be carried through Ethernet1. If Connection Method 2 is used, both RTP and the control message are carried through Ethernet1.

1.19 DST Series (REC Series)

1.19.1 Brief Introduction of DST Series

When a digital PBX, such as Siemens HIPath, Alcatel 4200 and Alcatel 4400 Series, is connected to a digital telephone, what goes through the line is not the analog signal but the pure digital signal. The physical line used to connect the digital PBX and the digital telephone is called the digital subscriber line. It is made of a D-channel and two B-channels, each channel supporting the rate of 64kbps. The D-channel is used to transmit call signaling messages, and the two B-channels can work simultaneously transporting voice data and others, like fax data, etc.

The DST Series boards are the proprietary call-recording board used for recording the voice data on the B-channel and the signaling data on the D-channel in the digital subscriber line. They have the high-impedance circuits for the input lines, equip the instantly-programmable chips in the signal recording circuit, and can enable a same board to support different PBX and digital phone models via simple settings with the application program.

So far, the DST Series boards are classified into two types : A and B.

They commonly have the following voice processing abilities:

- High-impedance recording of voice data on the B-channel, neither to cause any interruption on both parties in a call nor to be detected by them;
- Compression and recording of voice data by using the specified voice CODEC;
- Automatic Gain Control (AGC) support in data recording;
- DTMF detection;
- Detection of voice activities;
- Detection of single or dual tones at a specific frequency;
- Real-time playing of recorded voice signals;
- TDM capability;
- Playback of recorded voice data.

Besides these mentioned above, the DST series B-type boards also support the following new features.

- Detection of line faults, including line break and errors in voltage level, signal-to-noise ratio and frame synchronization;
- Flexible switch between voice channels B1 and B2;
- Board models with the suffix + support GSM, G.729A and MP3 encoding in hardware.

The D-channel signaling messages can be used to acquire the relative call information. The D-channel on the DST Series boards has the signaling processing capabilities that include the detection of the following operations by the digital phone as well as the changes in channel state.

- Off-hook, On-hook;
- Dial keys;
- Function keys;
- Voice channel open/closed;
- LED;
- Content displayed on LCD.

The board models in the DST Series are shown in the table below.

Board Model	Form Factor	Max. Ports per Board	Module Type	Max. Modules per Board
SHR-16DA-CT/PCI	PCI / PCI-X	16	MOD_16DA	4
SHR-24DA-CT/PCI	PCI / PCI-X	24	MOD_24DA	3
DST-24B/PCI	PCI	24	MOD_24DB	3 (including a module for board fixing)
DST-24B/PCI(SSW)	PCI	24	MOD_24DB	3 (including a module for board fixing)
DST-24B/PCI+	PCI	24	MOD_24DB	3 (including a module for board fixing)
DST-24B/PCI+(SSW)	PCI	24	MOD_24DB	3 (including a module for board fixing)
DST-24B/PCIe	PCIe	24	MOD_24DB	3 (including a module for board fixing)
DST-24B/PCIe+	PCIe	24	MOD_24DB	3 (including a module for board fixing)

The module types supported by the DST Series boards are listed in the table below.

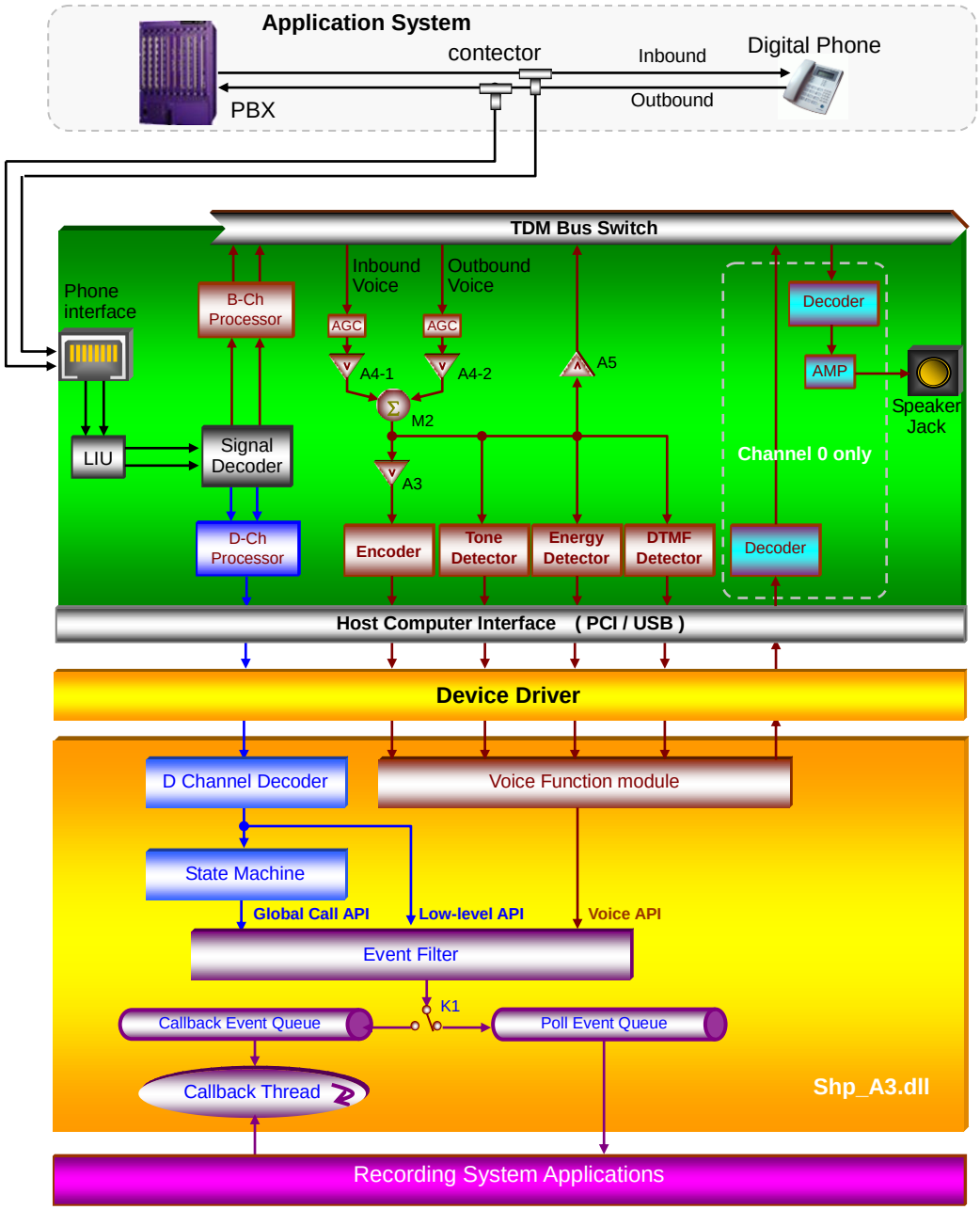
Module Type	Number of channels in the case of 2-line input	Number of channels in the case of 4-line input
MOD_16DA	4	2
MOD_24DA	8	4
MOD_24DB	8	4

1.19.2 DST Series Supported PBX Models

For the PBX models supported by the DST Series boards, refer to the document *DST Boards Supported PBX Models* supplied separately by Synway.

1.19.3 Operation Principle of DST Series

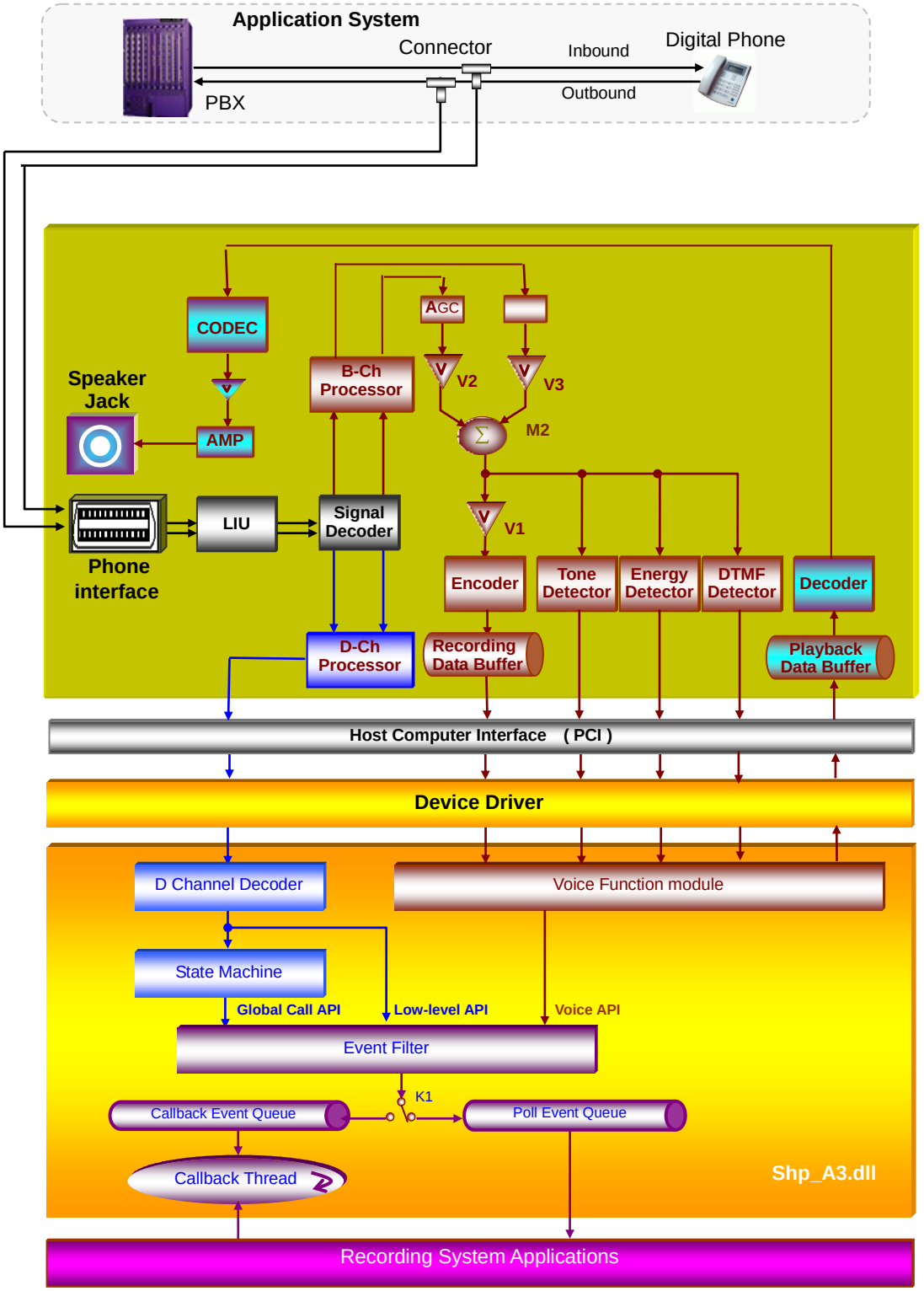
The figure below shows the operation principle of the DST Series A-type boards.



Components and symbols in the figure above are described in the following table.

Name	Description
M2	Off-bus mixer, used to mix outgoing signals with incoming ones.
A4-1	The volume adjuster for incoming signals (by reference to the digital phone). The volume can be set via the function DTRSetMixerVolume, with the default value of 0DB.
A4-2	The volume adjuster for outgoing signals (by reference to the digital phone). The volume can be set via the function DTRSetMixerVolume, with the default value of 0DB.
A3	Recording volume adjuster. The volume can be set via the configuration item DefaultRecordVolume or the function SsmSetRecVolume.
A5	Used to put the output signals from the mixer onto bus for recording and monitoring purpose. Usually there is no need to operate A5.

The figure below shows the operation principle of the DST Series B-type boards.



Components and symbols in the figure above are described in the following table.

Name	Description
M2	Off-bus mixer, used to mix outgoing signals with incoming ones.
V-2	The volume adjuster for incoming signals (by reference to the digital phone). The volume can be set via the function DTRSetMixerVolume, with the default value of 0DB.
V-3	The volume adjuster for outgoing signals (by reference to the digital phone). The volume can be set via the function DTRSetMixerVolume, with the default value of 0DB.
V1	Recording volume adjuster. The volume can be set via the configuration item DefaultRecordVolume

	or the function SsmSetRecVolume.

➢ **Recording Example for Various Signal Sources**

A lot of special applications are available via the flexible use of the above switches for recording control and volume adjusters.

✧ Example 1: Record both the incoming and the outgoing signals, using the normal volume.

```
DTRSetMixerVolume(ch, 0, 0, 0);     //set the volume gain of A4-1 and that of A4-2 to 0DB
SsmSetRecVolume(ch, 0);             //start A3, set the volume gain to 0DB
SsmRecToFile(ch, ……);               //start the task of file recording …
```

✧ Example 2: Only record the incoming signals, using the default volume.

```
DTRSetMixerVolume(ch, 0, 3, -7);    //set the volume gain of A4-1 to 3*3DB=9DB and that of A4-2 to -∞
SsmSetRecVolume(ch, -3);            //start A3, set the volume gain to -3*3DB=-9DB
SsmRecToFile(ch, ……);               //start the task of file recording …
```

➢ **Data Flow Direction in Voice Playing**

Only Channel 0 supports the voice playing operation. When the application program invokes the playing functions for Channel 0, the driver will automatically put the voices to be played onto bus and then get them off bus to arrive at the on-board audio power amplifier.

➢ **On-board Speaker Output Jack**

Channel 0 on each DST board is enabled by the analog audio amplifier circuit and the spearker output jack equipped on it to connect voice signals directly with the external speaker or headset. This design is mainly for the following purposes.

- ✧ To transport incoming signals from any channel on this board (or transport voice signals from channels on other board via CT-BUS) to the on-board speaker for play. This purpose can be achieved via the function SsmListenTo or SsmListenToEx.
- ✧ To send voice data recorded by the application straight to the on-board speaker for play via the call of playing functions for Channel 0. See the Voice Playing section in this chapter for details.

The AMP component in the figure above is the power amplifier for the analog audio amplifier circuit. Its gain can be set via the function SsmSetPowerAmpVlm.

1.19.4 DST Board Configuration

The configuration items PBXType and PhoneType are used respectively to set the PBX and the digital phone models; the configuration item DefaultVoiceFormat is used to set the CODEC supported by the digital subscriber line.

The configuration item DEventUpdates is used to filtrate repeated events.

Besides, some new configuration items are added for B-type digital station tap boards. DstRecRawData is used to set the board working mode; SetVoxChSelectEnable and VoxChSelect are used to select voice channels; SetAnalogCtrlEnable, AnalogCtrl, SetFilterSwitchEnable, FilterSwitch, SetVoltageReferenceEnable and VoltageReference are advanced configuration items which don't need to be reconfigured in most situations (If it is necessary to reconfigure them, please do under the instruction from our technicians).

1.19.5 Voice Operation on DST Series

The DST Series boards support the following voice operations.

- **Recording**

  Support voice recording in File Mode and Buffer Mode. The supported CODECs include A-Law, μ-Law, IMA ADPCM, G.729A, etc. See the Voice Recording section in this chapter for more information.

- **Playing**

  Channel 0 on each DST board supports the voice playing operation, including the play in Single File Mode and Single Buffer Mode. The supported CODECs include A-Law, μ-Law, IMA ADPCM, G.729A, etc. See the Voice Playing section in this chapter for more information.

- **Real-time Monitoring**

  The on-board speaker output jack can help each DST board to monitor the input voice signals on any channel in real time. The application can invoke the function SsmListenTo, SsmListenToEx, SsmLinkFrom or SsmLinkFromEx to start the real-time monitoring, and the function SsmStopListenTo or SsmStopLinkFrom to stop it. The function SsmSetPowerAmpVlm can be used to adjust the volume.

- **Volume Adjusting for Input Voice Signals**

  The function DTRSetMixerVolume can be used to set the volume for the voice signals from the PBX to the phone or from the phone to the PBX.

- **DTMF Detector**

  See the DTMF Detector section in this chapter for more information.

- **Tone Detector**

  See the Tone Detector section in this chapter for more information.

- **Barge-in Detector**

  In case the D-channel signaling messages are difficult to be analyzed, you can use the detected result of the Barge-in detector (voice activities detected/silence detected) as the condition to start or stop the recording and other operations. See the Barge-in Detector section in this chapter for more information.

1.19.6 Programming Guide for DST Series

The signaling protocol used for the D-channel has not been unified and is usually decided by the PBX or telephone manufacturers. As a result, different manufacturers, even one manufacturer, may produce various product Series which are quite distinct.

The driver picks up the original signaling messages on the D-channel and passes them to the application by throwing out D-channel events. Then according to these D-channel events, the application program completes the signaling analysis over the call process.

1.19.6.1 D-channel Signaling Messages

Among the D-channel signaling messages, those sent from the PBX to the phone are called PBX messages, and the corresponding events are PBX events; those sent from the phone to the PBX are called phone messages, and the corresponding events are phone events. Common PBX events include:

- Ring events;
- Audio events;
- Display events;
- LED (Light) events.

Common phone events mainly cover:

- ✧ Hook events;
- ✧ Button events.

Besides, the phone doesn't need to report some particular operations on it to the PBX, so it won't send any message to the PBX in such case. Thus even if the driver acquires all original signaling messages on the D-channel, it is probably not able to get any information about those operations on the phone, which include:

- ✧ Users operate the volume adjuster on the phone;
- ✧ The phone displays on the LCD some of the information that is independent of the PBX, such as settings of some function buttons;
- ✧ The phone changes the voice channel state, which is unnecessary to be reported to the PBX;
- ✧ The LED on the phone varies in the state, which is unnecessary to be reported to the PBX.

For detailed information about the D-channel signaling messages, refer to *DST Event Manual*.

The application should set up an exclusive state machine for a particular PBX/phone model by using some or all of the D-channel signaling messages as the condition to trigger the state transition, then properly record the relative information during the call process and the relevant voice operations, like the start and stop of recording, according to the changes in the channel state.

Those events which may affect the call state machine but are unavailable in the D-channel signaling messages, such as busy tones, ringback tones on the B-channel, can be obtained via other voice processing features the driver provides. See the Voice Processing section in this chapter for more information.

1.20 ATP Series (REC Series)

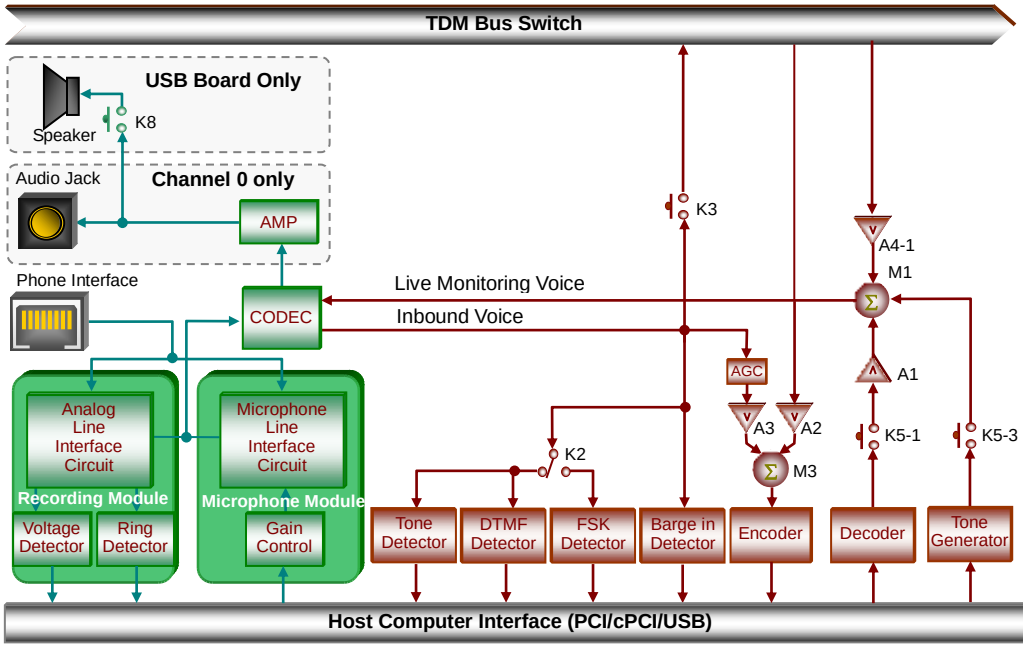
1.20.1 Brief Introduction of ATP Series

The features supported by the ATP Series boards are shown in the following table.

Board Model	Form Factor	Max. Ports	Voltage Detector	CallerID Detector	DTMF Detector	Tone Detector	Energy Detector	TDM Bus	Audio Socket
SHT-2A/USB	USB	2	√	√	√	√	√	N/A	√
SHT-2B/USB	USB	2	√	√	√	√	√	N/A	√
SHT-4A/USB	USB	4	√	√	√	√	√	N/A	√
SHT-4B/USB	USB	4	√	√	√	√	√	N/A	√
SHT-8A/PCI	PCI	8	√	√	√	√	√	N/A	√
SHT-8B/PCI	PCI	8	√	√	√	√	√	N/A	√
SHT-16A-CT/PCI	PCI	16	√	√	√	√	√	H.100	√
SHT-16B-CT/PCI	PCI	16	√	√	√	√	√	H.100	√
SHT-16B-CT/PCI/MP3	PCI	16	√	√	√	√	√	H.100	√
ATP-24A/PCI	PCI	24	√	√	√	√	√	N/A	√
ATP-24A/PCI+	PCI	24	√	√	√	√	√	N/A	√
ATP-24A/PCIe	PCIe	24	√	√	√	√	√	N/A	√
ATP-24A/PCIe+	PCIe	24	√	√	√	√	√	N/A	√
SHT-16B-CT/cPCI	cPCI	16	√	√	√	√	√	H.110	√
SHT-16B-CT/cPCI/MP3	cPCI	16	√	√	√	√	√	H.110	√

1.20.2 Operation Principle of ATP Series

The figure below shows the operation principle of the ATP Series boards.



Symbols related to the recording operation in the figure above are described in the following table.

Name	Description
M1	Real-time monitoring mixer. Either the off-bus voices or the voices played on channel 0 are allowed to enter this mixer at a time.
A4-1	The volume adjuster for off-bus signals
A1	The volume adjuster for voice playing on Channel 0. The volume can be set via the function SsmSetPlayVolume/SsmSetPlayGain or the configuration item DefaultPlayVolume, with the default value of 0.
K5-1	The switch that controls the voice playing operation, usually keeping off under the automatic control of the driver. Only when the application program invokes the playing function on Channel 0 does the driver automatically switch on K5-1 and then switch off it at the end of voice playing. For more information about the playing functions, refer to the Voice Playing section.
K5-3	The control switch of tone generator which is automatically controlled by the driver. It is turned off by default. Only when the driver invokes the function SsmSendTone or SsmSendToneEx to start the tone generator will K5-3 be turned on. After the tone generation task is finished, K5-3 will be turned off automatically.
M3	Recording mixer The signals from M3 are sent to the Encoder component for recording purpose.
A3	The volume adjuster which is used for incoming signals before they are sent to the recording mixer. The volume can be set via the configuration item DefaultRecordVolume or the function SsmSetRecVolume, with the default value of 0. Note: A3 will become invalid once the AGC feature is enabled.
A2	The volume adjuster which is used for bus signals before they are sent to M3, switched on in default. The volume can be set via the function SsmSetRecMixer or the configuration item DefaultRecordMixerVolume, with the default value of -7. This volume adjuster is only applicable to a few particular occasions. It only supports the boards with bus in ATP Series boards. Boards like ATP-24A/PCI are unsupported as they don't include bus.
K3	The switch that controls whether to put the incoming signals into the onto-bus mixer or not, switched on in default. It can be set via the function SsmSetFlag (with the parameter F_InVoiceToBus) or the configuration item InVoiceToBus.
K2	The switch that shifts between the FSK detector and other detectors, usually connected to other

	detectors unless the FSK detector is started.

➢ **Channel Type and Modules**

The ATP Series boards (excluding ATP-24A series boards) accept two kinds of modules installed on them. They are Analog Trunk Recording Module and Microphone Recording Module. The driver does not distinguish these two kinds of modules and regards their corresponding channel types both as the analog recording channel. The configuration item MicGain is used to set the gain of the input signals to the analog trunk recording channel. For Analog Trunk Recording Module, as the voice signals come from the analog phone line, the gain should be set to the normal value; for Microphone Recording Module, the gain should be set to 20DB.

➢ **Voice Playing**

Only Channel 0 supports the voice playing operation. When the application calls playing functions for Channel 0, voice data will be sent to the on-board audio power amplifier.

➢ **Voice Recording**

The recording operation based on the ATP Series boards is quite easy to be performed. The only one thing required is to control the gain of A3.

✧ Example 1: Record the incoming signals on Ch1, using the default volume.

```
SsmSetRecVolume(ch1, 0);          //start A3, set the volume gain to 0DB
SsmRecToFile(ch1, ……);            //start the task of file recording …
```

➢ **On-board Speaker Output Jack**

Channel 0 on each ATP board is enabled by the analog audio amplifier circuit and the speaker output jack equipped on it to connect directly with the external speaker or headset and monitor voice signals in real time. This design is mainly for the following purposes.

- ✧ To transport incoming signals from any channel to the on-board speaker for play via TDM bus. This purpose can be achieved via the function SsmListenTo or SsmListenToEx.
- ✧ To send voice files recorded by the application straight to the on-board speaker for play via the call of playing functions. See the Voice Playing section in this chapter for details.

The AMP component in the figure above is the power amplifier for the analog audio amplifier circuit. Its gain can be set via the function SsmSetPowerAmpVlm.

Channel 0 on the USB recording box (with '/USB' marked in the model) is equipped with not only a speaker output jack but also an on-board speaker, which enables the voice playing operation. The switch K8, providing choices for the signals output from Channel 0 to enter the on-board speaker or the speaker jack, can be set via the function SsmSetLine0OutTo or the configuration item USBLine0Output, with the default setting of 'to the speaker jack'.

Note: Although there is no TDM bus available on the 8-channel ATP boards, the driver provides the soft-switch feature which also allows the transmission of voice data from one board to another (it is usually the board linked with an external speaker) for real-time monitoring. Don't forget to set the configuration item BusPlayListen in such case.

➢ **Special Application of Non-module Channel**

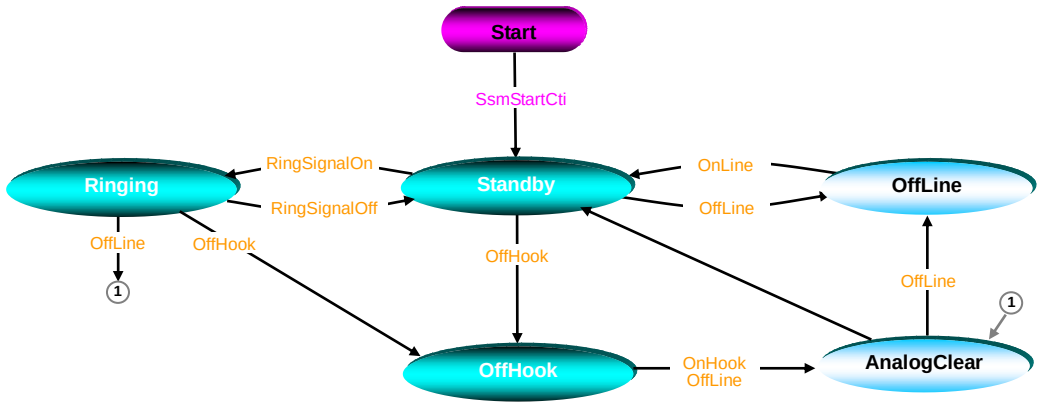
A channel on the ATP Series boards equipped with no modules can be used as the recording channel. In such case, the Barge-in detector and the recording encoder still work in a normal way, and the off-bus operation goes well as usual, which allow the recording of voice data from TDM bus via function calls. The Barge-in detector is

used for voice control purpose. This special application can be set via the configuration item NoModuleChBusRec or the function SsmSetNoModuleChBusRec.

Note:

The volume adjuster A3 should be switched off via the function SsmSetRecVolume or the configuration item DefaultRecordVolume before the start of voice recording.

1.20.3 Analog Trunk Recording Channel State Machine



Each state identifier in the above diagram is described in the following table.

State Identifier	Value	Macro Definition	Description
Standby	0	S_CALL_STANDBY	'Idle' state. The channel goes into this state after the successful driver initialization.
OffHook	1	S_CALL_PICKUPED	'Off-hook' state. When the channel state transfers to 'Off-hook', the driver will: ✧ clear the tone detector and then start it; ✧ start the DTMF detector; ✧ start the prerecord task if this feature has been enabled.
Ringing	2	S_CALL_RINGING	'Ringing' state, indicating there comes a call. In this state, the application can invoke the function SsmGetCallerId or SsmGetCallerIdEx to obtain the calling party number. See the Caller ID on Analog Phone Line section in this chapter for more information.
OffLine	8	S_CALL_OFFLINE	'Off-line' state
AnalogClear	10	S_CALL_ANALOG_CLEAR	It is the internal 'disconnection' state, only lasting for 8ms, mainly used for the driver to do the following things: ✧ clear the tone detector and then close it; ✧ clear the Caller ID buffer within the driver provided the configuration item AutoClearCallerIdBufOnHangup is set to 1; ✧ start the Caller ID detector and close the DTMF detector if the Caller ID is received in the FSK mode; otherwise, start the DTMF detector; ✧ immediately stop the prerecord operation if it is still going on; ✧ clear the DTMF buffer within the driver; ✧ start the ringing current detector.

In the diagram above, the internal events automatically triggered by the driver itself are described as follows.

Event	Description
RingSignalOn	The driver detects the ringing signal. See the Ringing Current on Analog Phone Line

	section in this chapter for more information.
RingSignalOff	The ringing signal disappears.
OffHook	The driver detects the pickup behavior on the monitored phone. Refer to the Change in Analog Phone Line Voltage section in this chapter for more information.
OnHook	The driver detects the hangup behavior on the monitored phone.
OffLine	The connection between the monitored phone and the PBX or that between the high-impedance parallel lines and the on-board ports is broken. See the Change in Analog Phone Line Voltage section in this chapter for more information.
OnLine	The connection is recovered.

1.20.4 Detection of Off-hook/On-hook State

For detailed information about the detection of the pickup and hangup behaviors on the recording channel, see the Change in Analog Phone Line Voltage section in this chapter.

1.21 DTP Series (REC Series)

1.21.1 Brief Introduction of DTP Series

The features supported by the DTP Series boards are shown in the following table.

Board Model	Form Factor	Trunk Type	Max. Ports	Hi-Z Line Interface Circuit	Voice Processing Capabilities	TDM Bus	DSS1	ISDN PRI	SS7	SS7 Signaling Links
SHD-30A-CT/PCI/FJ	PCI	E1	30	√	√	H.100	√	√	√	1
SHD-60A-CT/PCI/FJ	PCI	E1	60	√	√	H.100	√	√	√	2
SHD-30B-CT/PCI/FJ	PCI	E1/T1	30/23	√	√	H.100	√	√	√	1
SHD-60B-CT/PCI/FJ	PCI	E1/T1	60/46	√	√	H.100	√	√	√	2
DTP-30C/PCI	PCI	E1/T1	30/23	√	√	H.100	√	√	√	1
DTP-30C/PCI+	PCI	E1/T1	30/23	√	√	H.100	√	√	√	1
DTP-60C/PCI	PCI	E1/T1	60/46	√	√	H.100	√	√	√	2
DTP-60C/PCI+	PCI	E1/T1	60/46	√	√	H.100	√	√	√	2
DTP-120C/PCI	PCI	E1/T1	120/92	√	√	H.100	√	√	√	4
DTP-120C/PCI+	PCI	E1/T1	120/92	√	√	H.100	√	√	√	4
DTP-30C/PCIe	PCIe	E1/T1	30/23	√	√	H.100	√	√	√	1
DTP-30C/PCIe+	PCIe	E1/T1	30/23	√	√	H.100	√	√	√	1
DTP-60C/PCIe	PCIe	E1/T1	60/46	√	√	H.100	√	√	√	2
DTP-60C/PCIe+	PCIe	E1/T1	60/46	√	√	H.100	√	√	√	2
DTP-120C/PCIe	PCIe	E1/T1	120/92	√	√	H.100	√	√	√	4
DTP-120C/PCIe+	PCIe	E1/T1	120/92	√	√	H.100	√	√	√	4

The voice processing capabilities mentioned above include:

- ✓ DTMF Detector
- ✓ Tone Detector
- ✓ Barge-in Detector
- ✓ Voice Recording
- ✓ Voice Playing
- ✓ TDM Capability

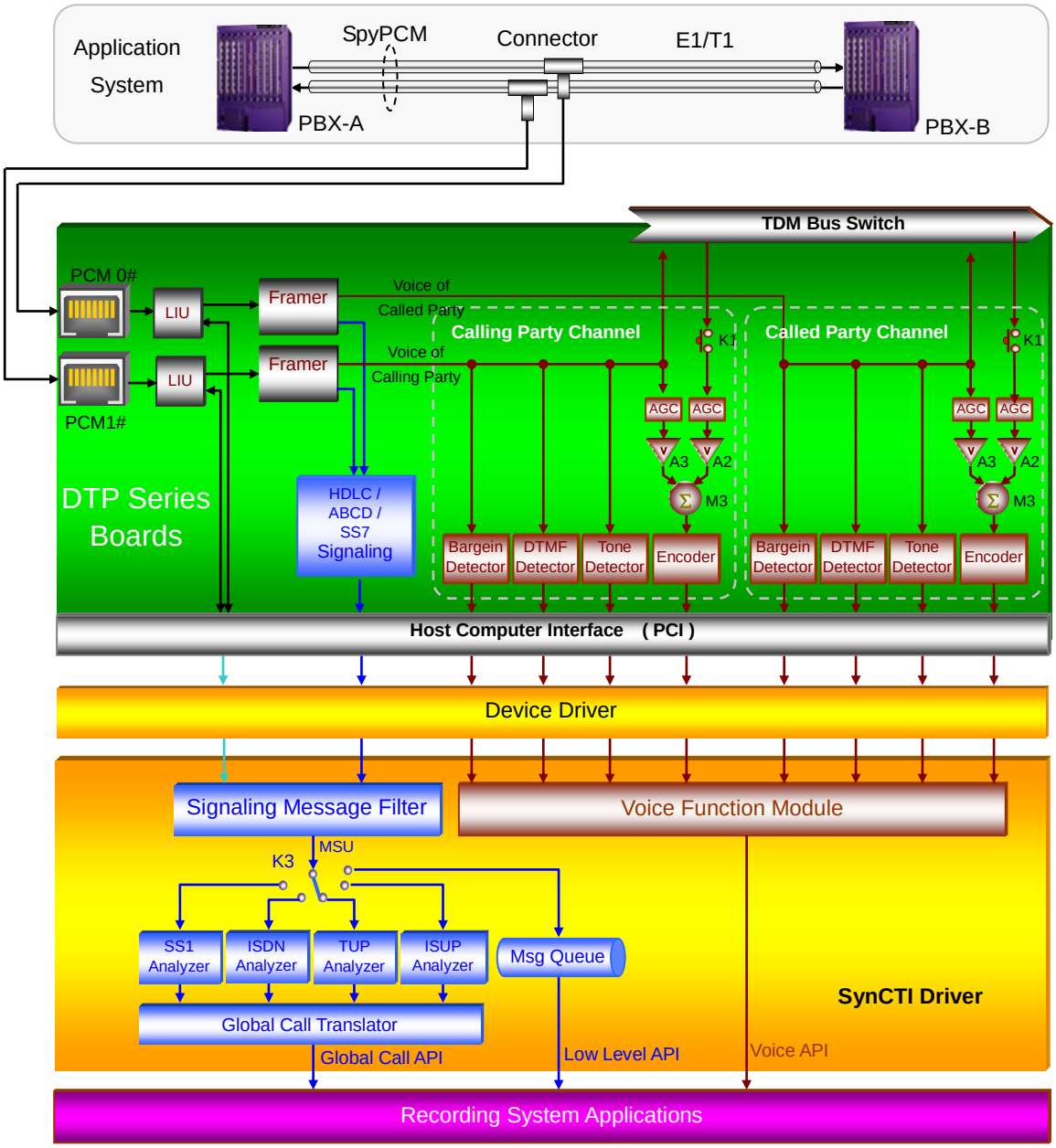
The line interface unit (LIU) on the DTP Series boards is enabled by the high-impedance circuit equipped on it to connect directly with the monitored E1/T1 lines, without the need of extra devices for high-impedance parallel connection.

In monitoring one E1/T1 link, the onto-PCM and off-PCM signals must be recorded and processed at the same time. Therefore, two E1/T1 LIU which only need to process input signals are essential for this process.

The information acquired through monitoring can be separated into two parts: signaling messages and voice channel data.

1.21.2 Operation Principle of DTP Series

The figure below shows the operation principle of the DTP Series boards.



Those symbols related to the board in the figure above are described in the following table.

Name	Description
M3	Recording mixer. The signals from M3 are sent to the Encoder component for recording purpose.
A3	The volume adjuster which is used for incoming signals before they are sent to M3. The volume can be set via the function SsmSetRecVolume or the configuration item DefaultRecordVolume, with the

	default value of 0.
A2	The volume adjuster which is used for outgoing signals before they are sent to M3, switched off in default. The volume can be set via the function SsmSetRecMixer or the configuration item DefaultRecordMixerVolume, with the default value of -7.
K3	The signaling/protocol switch. It can be set via the configuration item PcmSSx. **Note that so far the driver is not yet able to pass original signaling messages directly to the application program through Msg Queue.**

The driver is capable of analyzing SS1, ISDN, SS7-TUP and SS7-ISUP signaling with the help of its internal call state machine. The application can control and visit this state machine by using the unified API functions (i.e. Global Call API in the figure above). This section focuses on the content concerning the Global Call API functions.

In a SS7 ISUP or TUP call, the call state machine provided by the driver can save the received original messages to its internal buffer while processing the ISUP or TUP message, provided the configuration item bOpenSpySS7Msu is set to 1. And the application may call the function SsmGetSs7SpyMsu to acquire the original ISUP or TUP message from this buffer.

The operation on voice data is also quite easy. The DTP Series boards provide independent voice channels respectively for incoming signals and outgoing signals. Each voice channel is designed with the independent DTMF detector, tone detector, Barge-in detector, decoder and recording mixer.

1.21.3 Concerned Terms

- Physical PCM

It indicates the on-board hardware circuit which has the capability of data processing for digital trunks. Unlike that on the SHD Series boards, the PCM on the DTP boards only processes input signals, incapable of outputting signals. That is, the PCM on the DTP Series works in a simplex mode.

See the Digital Trunk section in this chapter for detailed information about the physical PCM.

- PCM Line to Be Monitored (referred to as ‘SpyPCM’ for short in this manual)

SpyPCM is the E1/T1 line parallelly connected through high-impedance circuits as shown in the figure above.

Because the E1/T1 line works in the full-duplex mode (one for transmission, one for reception), to record and process the onto-SpyPCM and off-SpyPCM signals at the same time, the board requires two sets of PCM circuits (i.e. the physical PCM) independently for processing the onto-SpyPCM signals and the off-SpyPCM signals. These two physical digital trunks are logically combined with necessary circuits and well able to support the recording and processing of voice signals and signaling messages on SpyPCM.

The application is required to provide the SpyPCM number (i.e. the logical SpyPCM number) while calling relative functions. The logical SpyPCM number is numbered from 0. The binding relationship between this number and the two physical PCM is determined by the configuration items TotalSpyPcm and SpyPcm.

When one of the two physical PCM bound to SpyPcm changes in status, the function SpyGetLinkStatus can be used to obtain the SpyPcm state.

- Circuit in SpyPCM (referred to as ‘SpyCic’ for short in this manual)

SpyCic indicates a specific time slot (circuit) in the SpyPCM. For SS7 signaling, SpyCic is just the CIC field in the TUP or ISUP message; for ISDN PRI and SS1 signaling, SpyCic is the time slot number in PCM.

Each SpyCic has two numbers: the physical number and the logical number. The physical SpyCic number means the SpyCic number in one PCM which corresponds to the time slot number in the same PCM as shown below.

Physical SpyCic Number	0~14	15~29
Corresponding Time Slot Number in PCM	1~15	17~31

The logical SpyCic number is the unified number given to each SpyCic in all PCM involved in the whole application system, beginning with 0. All SpyCic functions provided by the driver take the logical SpyCic number as the input parameter. The mapping relationship between the logical SpyCic number and the physical SpyCic number is determined by the configuration items TotalAppSpyCIC and AppSpyCIC in the [AppSpyCICTable] section in the configuration file. The function SpyGetMaxCic can be used to acquire the total number of SpyCic in the application system.

Because one SpyPCM is bound to two physical PCM, there is a one-to-one correspondence between the logical SpyCic number and the physical SpyCic number. Logically, one SpyCic is made up of two channels in the driver.

- Calling Party Channel & Called Party Channel

The 2 channels bound to SpyCic may serve as either the calling party or the called party in a call. In the SynCTI-provided state machine for signaling analysis, the channel processing the voice signals from the calling party to the called party is named the calling party channel, and that processing the signals from the called party to the calling party is named the called party channel. What you should pay attention is the calling party channel or the called party channel is changeable, depending mainly on the call direction. For example, a channel serves as the calling party in a call, so this time it is the calling party channel; however, it may serve as the called party in the next call, then it becomes the called party channel.

The functions SpyGetCallInCh and SpyGetCallOutCh are used respectively to obtain the calling party channel number and the called party channel number for the 2 channels bound to SpyCic in this call. The function SpyChToCic can be used to query the logical SpyCic number according to the channel number.

The SynCTI-provided state machine for signaling analysis is based on SpyCic. However, the application can invoke relative functions for voice processing on any bound channel to acquire DTMF digits and other information in the onto-PCM and off-PCM signals.

1.21.4 Voice Processing on DTP Series Boards

1.21.4.1 DTMF Detection

When the call based on a SpyCic is established, the calling party channel number and the called party channel number in this call can be obtained respectively via the functions SpyGetCallInCh and SpyGetCallOutCh. And those DTMF detection functions having these two numbers contained in their parameters can be invoked to acquire DTMF signals from both parties. For more information about the DTMF detector, refer to the DTMF Detector section in this chapter.

1.21.4.2 Voice Recording

The voice recording feature support both the file mode and the buffer mode. Relative functions and events are listed below.

Function/Event Name	Description

SpyRecToFile	A function based on the circuit number, used to start the file recording task
SpyStopRecToFile	A function based on the circuit number, used to stop the file recording task
E_PROC_RecordFile	The event thrown out by the driver at the end of the file recording task
SpyRecToMem	A function based on the circuit number, used to start the buffer recording task
SpyStopRecToMem	A function based on the circuit number, used to stop the buffer recording task
E_PROC_RecordMem	The event thrown out by the driver during the buffer recording task

Relative configuration items include:

Configuration Item	Description
DspCoder	Sets the coder loaded by an on-board DSP which gives more formats for recording (so far support of GSM and G.729A).

There are many ways available to perform voice recording on the DTP Series boards. Assuming that a SpyCic is made up of a calling party channel (referred to as CallingPartyCh) and a called party channel (referred to as CalledPartyCh), the calling party's voice (using the default volume) and the called party's voice (using the default volume) on this SpyCic should be recorded in a mixed way after both parties come into the call. The processing codes for this purpose are written as follows.

```
SpyRecToFile(SpyCic, 2, "Voc.wav", 7, 0, 0xffffffff, 0xffffffff,    0);
```

If the voice recording process should be controlled more precisely, you can use the functions based on the channel. For example, to achieve the above purpose, but to increase the volume of the calling party's voice by 3DB and decrease that of the called party's voice by 3DB, the processing codes are as follows.

```
int CalledPartyCh = SpyGetCallInCh(SpyCic);          //obtain the calling party channel number
int CallingPartyCh= SpyGetCallOutCh (SpyCic);        //obtain the called party channel number
SsmListenTo(CallingPartyCh, CalledPartyCh);          //switch on the switch K1 on CalledPartyCh
SsmSetRecVolume(CalledPartyCh, 1);                    //start A3, set the volume gain to 1*3DB=3DB
SsmSetRecMixer(CalledPartyCh, TRUE, -1);              //start A2, set the volume gain to -1*3DB=-3DB
SsmRecToFile(CalledPartyCh, "Voc.wav",…);            //start the task of file recording …
```

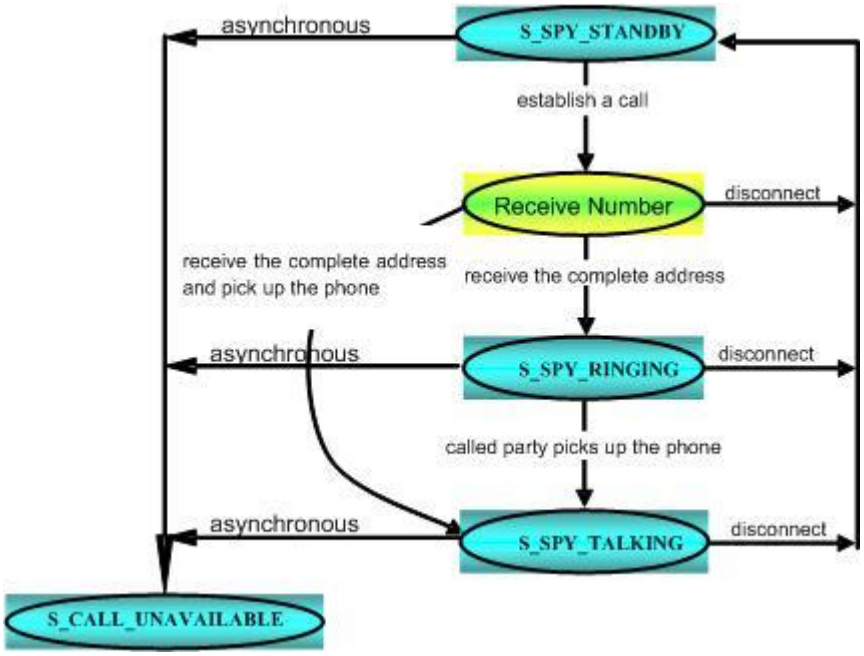
People who use G.729A or GSM for recording should pay attention that as GSM and G.729A recordings are enabled by extra DSPs, there are limits to use of these two formats. See below for details:

1. One board can use only one of the two formats for recording. That is, it is not allowed to use different formats for different channels. (Use of other formats like A-Law, μ-Law, etc. is not confined to such limit.)
2. You may perform independent-recording or mixed-recording of calling or called channel, but cannot do it of both at the same time.
3. Because the available ways for bus exchange have been locked, when you use the precise volume control mode, parameters of the function SsmListenTo must be calling and called channels for a same CIC. Whereas, for those bus functions with volume parameters, such as SsmListenToEx, the volume parameters will be ignored.

1.21.5 Driver-provided State Machine

1.21.5.1 Universal State Machine Model for Digital Trunk Recording Channel

Below is the SynCTI-provided state machine commonly used by the DTP Series boards.



Every time when the SpyCic state changes, the driver will throw out the E_CHG_SpyState event to the application. The function SpyGetState can be used to obtain the current SpyCic state.

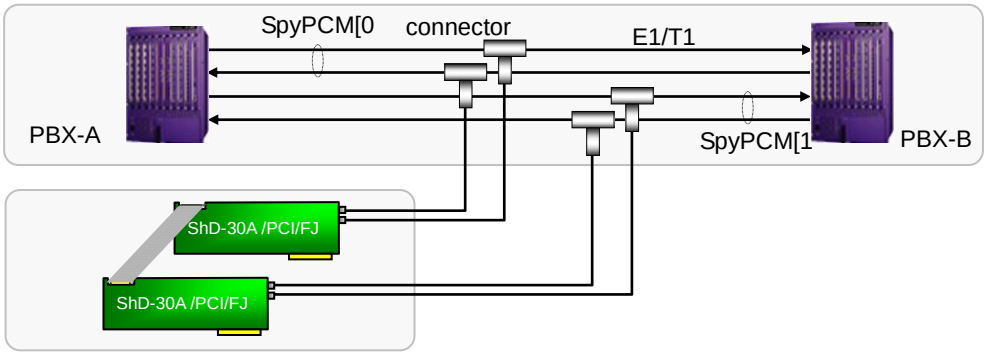
1.21.5.2 Acquisition of Call Information

The functions and events used to obtain the information about the call state machine include:

Function/Event Name	Description
E_CHG_SpyState	This event is thrown out by the driver when the state machine for SS1, ISDN, SS7 TUP or SS7 ISUP changes.
E_CHG_SpyHangupInfo	This event is thrown out by the driver when a disconnect message is detected on the line in ISDN.
SpyGetState	Obtains the SpyCic state.
SpyGetCallerId	Obtains the calling party number in this call.
SpyGetCalleeId	Obtains the called party number in this call.
SpyGetCallInCh	Obtains the calling party channel number in this call.
SpyGetCallOutCh	Obtains the called party channel number in this call.
SpyGetHangupInfo	Obtains the hangup information about this call.

1.21.6 Application and Configuration Instance for DTP Series

1.21.6.1 System Using SS1/ISDN PRI



In the figure above, SpyPCM[0] uses the SS1 protocol while SpyPCM[1] uses the ISDN protocol. This system involves two SHD-30A-CT/PCI/FJ boards. Relative configuration items in the configuration file ShConfig.ini could be set as follows.

[BoardId=0]

.....

PcmNumber=2

PcmSSx[0]=1 //SS1
PcmSSx[1]=1 //SS1
PcmClockMode[0]=2 //slave clock
PcmClockMode[1]=2 //slave clock

[BoardId=1]

.....

PcmNumber=2

PcmSSx[0]=0 //ISDN
PcmSSx[1]=0 //ISDN
PcmClockMode[0]=0 //master clock, line synchronization
PcmClockMode[1]=2 //slave clock

[PcmInfo]

TotalPcm=4
Pcm[0]=0,0
Pcm[1]=0,1
Pcm[2]=1,0
Pcm[3]=1,1

[SpyPcm]

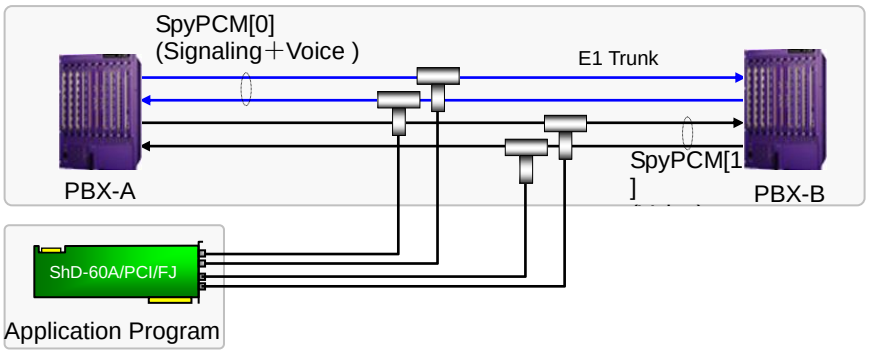
TotalSpyPcm=2
SpyPcm[0]=Pcm[0],Pcm[1]
SpyPcm[1]=Pcm[2],Pcm[3]

[AppSpyCICTable]

TotalAppSpyCIC=60 //totally 60 SpyCIC (voice channel)
AppSpyCIC[0]=SpyPcm[0],0..29 //SpyCIC[0] corresponds to Channel 0 and Channel 1, and the like…
AppSpyCIC[30]=SpyPcm[1],0..29

1.21.6.2 System Using SS7

As to the system used to monitor SS7 links, except the above configuration items necessarily set for the system using SS1 or ISDN, the section **[SS7Spy]** is also required. See below for the typical application.



In the figure above, there are 2 digital trunks between PBX-A and PBX-B. TS16 of SpyPCM[0] serves as the signaling link, using the SS7 TUP protocol. TS16 of SpyPCM[1] is unused while the rest all used as voice channels. Assume SpyPCM[0] is numbered as 4 in the CIC field of the TUP message and SpyPCM[1] is numbered as 5. Taking the TUP protocol for example, you can follow the way below to set the configuration file in the application program.

[BoardId=0]

.....

PcmNumber=4

PcmSSx[0]=7 //SS7
PcmSSx[1]=7 //SS7
PcmSSx[2]=7 //SS7
PcmSSx[3]=7 //SS7
PcmClockMode[0]=0 //master clock, line synchronization
PcmClockMode[1]=2 //slave clock
PcmClockMode[2]=2 //slave clock
PcmClockMode[3]=2 //slave clock

[PcmInfo]

TotalPcm=4
Pcm[0]=0,0
Pcm[1]=0,1
Pcm[2]=0,2
Pcm[3]=0,3

[SpyPcm]

TotalSpyPcm=2
SpyPcm[0]=Pcm[0],Pcm[1]
SpyPcm[1]=Pcm[2],Pcm[3]

[AppSpyCICTable]

TotalAppSpyCIC=60 //totally 60 SpyCIC (voice channel)
AppSpyCIC[0]=SpyPcm[0],0..29 //SpyCIC[0] corresponds to Channel 0 and Channel 1, and the like…
AppSpyCIC[30]=SpyPcm[1],0..29

[SS7Spy]

TotalSpyLinkSet=1 //totally 1 signaling link set
SpyLinkSet[0]=SpyPcm[0] //set the physical position of the signaling link in the link set
SpyLinkSet[0].SpyCICPcm[4]=SpyPcm[0]
SpyLinkSet[0].SpyCICPcm[5]=SpyPcm[1]
SpySpCodeLen=24 //Set the encoding standard for signaling points: 24bit is the National Signaling Point Code (NSPC) format, 14bit is the International Signaling Point Code (ISPC) format.

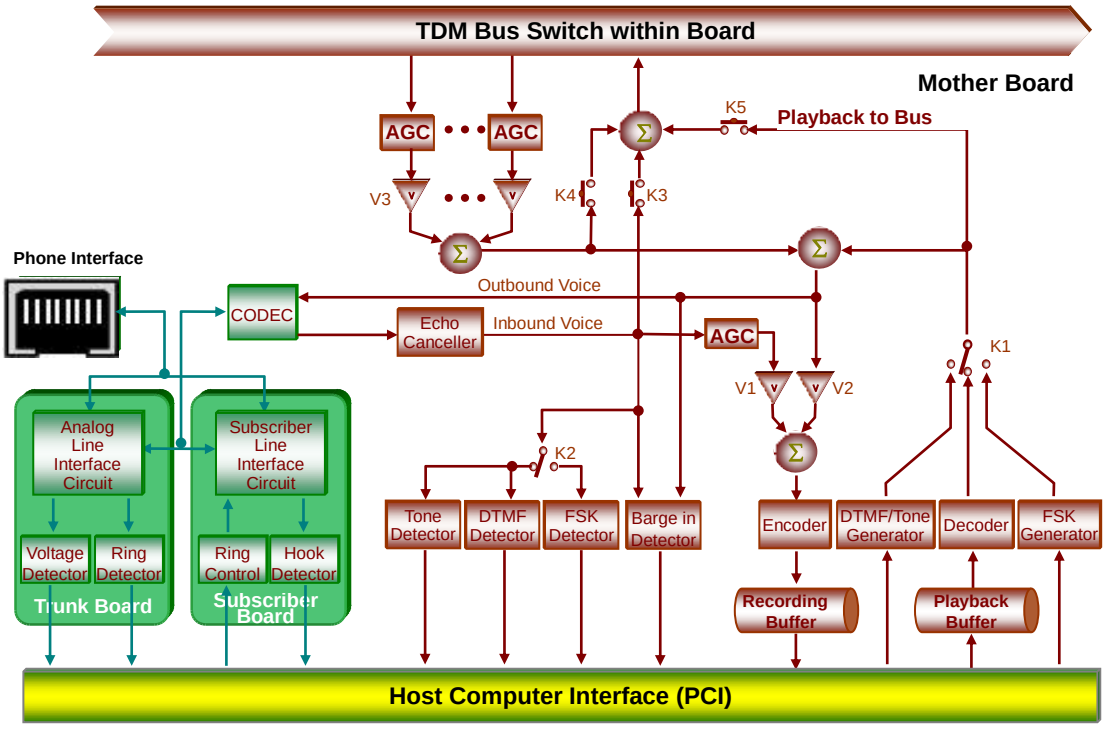
1.22 SHF Series (CTI Series)

1.22.1 Brief Introduction of SHF Series

The features supported by the SHF Series boards are shown in the following table.

Board Model	Form Factor	Max. Ports	Voice Processing Capabilities	Conferencing Capabilities	Voltage Detector	Audio Socket	FSK	TDM Bus	Max Fax Resource
SHF-2D/PCI	PCI	2	√	√	√	√	√	N/A	2
SHF-4D/PCI	PCI	4	√	√	√	√	√	N/A	4
SHF-4D/PCIe	PCIe	4	√	√	√	√	√	N/A	4

1.22.2 Operation Principle of SHF Series



The figure below shows the operation principle of the SHF Series boards.

Modules and symbols in the figure above are described in the following table.

Name	Description
M1	Outbound voice mixer
M2	Off-bus mixer
M3	Recording mixer The signals from M3 are sent to the Encoder component for recording purpose.
M5	Onto-bus mixer
K1	The Voice Play/Fsk Transmitter switch, controlled by the driver. When the application calls the playing function, K1 connects to the component Decoder; when it calls the function SsmStartSendFSK to start FSK data transmission, K1 connects to the component FSK Generator. For more information about the playing functions, refer to the Voice Playing section.
K2	The switch that shifts between the FSK detector and other detectors, usually pointing to other detectors unless the FSK detector is started.
K3	The switch that controls whether the incoming signals enter the onto-bus mixer M5 or not, switched on in default. It can be set via the function SsmSetFlag (with the parameter F_InVoiceToBus) or the

	configuration item InVoiceToBus.
K4	The switch that controls whether the signals output from M2 go back onto TDM bus, being switched off in default. It can be set via the function SsmSetFlag (with the parameter F_MixerResToBus).
K5	The switch that controls whether the playing data enter the mixer M5 and go onto TDM bus or not, usually keeping off. It can be set via the function SsmSetPlayDest, used only for playing background music to the teleconference.
V1	The volume adjuster used before incoming signals entering M3. The volume can be set via the function SsmSetRecVolume or the configuration item DefaultRecordVolume, with the default value of 0DB.
V2	The volume adjuster used before outgoing signals entering M3. The volume can be set via the function SsmSetRecMixer or the configuration item DefaultRecordMixerVolume, with the default value of -7(off).
V3	The volume adjuster used before the off-bus signals entering M2. The volume can be set via the function SsmSetListenVlmInConf, with the default value of 0DB.

■ **Voice Playing**

After the application invokes playing functions, the driver will automatically have K1 link to Decoder. The decoded voice data has two places to go:

- ✧ Entering M1 and forming the final outgoing signals with other signal sources.
- ✧ Entering M5 under the control of K5 and forming the onto-bus signal sources with other signals, only for playing background music to the teleconference.

■ **Recording Example for Various Signal Sources**

A lot of special applications are available via the flexible use of the above switches for recording control and volume adjusters. In the following examples, ch1 means Channel 1 and ch2 is just Channel 2.

✧ Example 1: Only record the incoming signals on ch1, using the default volume.

```
SsmSetRecVolume(ch1, 0);          //start V1, set the volume gain to 0DB
SsmSetRecMixer(ch1, FALSE, 0);    //stop V2
SsmRecToFile(ch1, ……);             //start the task of file recording …
```

✧ Example 2: Record the incoming signals on ch1 (volume increased by 6DB) and the signals output from M2 (volume decreased by 3DB) at one time based on the establishement of a two-way call between ch1 and ch2.

```
SsmTalkWith(ch1, ch2);            //establish a two-way connection between ch1 and ch2 via TDM bus
SsmSetRecVolume(ch1, 2);          //start V1, set the volume gain to 2*3DB=6DB
SsmSetRecMixer(ch1, TRUE, -1);    //start V2, set the volume gain to -1*3DB=-3DB
SsmRecToFile(ch1, ……);             //start the task of file recording …
```

✧ Example 3: When ch1 joins a teleconference (assuming the conference room number is n), play background music on ch1 and record the conference through ch1. All signal sources go with normal volume, i.e. the gain is 0.

```
SsmJoinConfGroup(n, ch1, ……);     //ch1 joins the conference
SsmSetPlayDest(int ch, 1);        //switch off K5, put the playing data onto bus
SsmPlayFile(ch1, ……);              //start the task of voice playing on ch1, play background music
SsmSetRecVolume(ch1, 0);          //start V1, set the volume gain to 0DB
SsmSetRecMixer(ch1, TRUE, 0);     //start V2, set the volume gain to 0DB
SsmRecToFile(ch1, ……);                                  //start the task of file recording …
```

1.23 IPR Series (REC Series)

1.23.1 Brief Introduction of IPR Series

The IPR series products are the proprietary call-recording software used for recording the voice data and signaling messages through the VoIP links. They use port mirroring to record the data over corresponding links on the IP network and can enable a same board to support different PBX models and IP terminals via simple settings with the application program.

The IPR series products include SynIPAanalyzer and SynIPRecorderSynIPAnalyzer has the following processing capabilities:

- Uses port mirroring to record the voice data on the network, neither causing any interruption on both parties of a call nor being detected by them
- Detects DTMF signaling messages
- Supports multiple PBX and phone models, as well as simultaneous monitoring to multiple ports
- Forwards the recorded RTP packages
- Supports terminal and call managements, assigning them independent IDs

SynIPRecorder has the following processing capabilities:

- Uses specified CODECs to compress and record voice data
- Sets the recording volume via functions
- Detects the RFC2833 DTMF codes
- The module is designed with two modes Master and Slave, which facilitates distributed deployment and expansion
- Supports the receiving of RTP data in different encoding formats
- Supports multiple recording methods, including recording of incoming calls, recording of outgoing calls, mix recording of incoming and outgoing calls

The signaling messages over D-channels can be used to acquire relative call information. The D-channel's signaling processing capabilities include the detection of such IP phone's operations and state changes as follows.

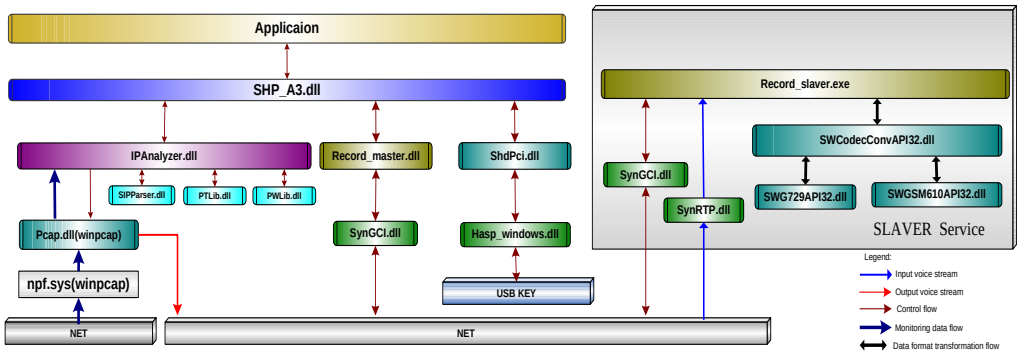
- Off-hook, On-hook
- Dial keys
- Function keys
- Voice Channel open/close
- LED
- Content displayed on LCD

The notes for IPR Series are shown in the following table.

Type	Installation	Operation	Upgrade File
USB-KEY	-	-	Independent EXE file.
NTP-480A/PCIe	Reboot after install	1. For operating systems with UAC, the NTP board should run under administrator rights. 2. To avoid conflict, the other devices in the same network of the NTP board should not use the IP address 192.168.1.7 or 192.168.1.6.	DAT file. Use the configuration tool ShCtiConfig.exe to upgrade.

1.23.2 System Model of IPR Series

The figure below shows the system model of the IPR series products:



The modules illustrated in the figure above are described in the following table:

Module Name	Description
Application	User application program
SHP_A3.DLL	Provides API functions for applications to achieve all functions of the board
IPAnalyzer.dll	Core module for signaling parsing, sole of SynIPAnalyzer
SIPParser.dll	Parser module for SIP messages
PTLib.dll, PWLib.dll	Parser module for H.323 messages
pcap.dll, npf.sys	Data packet capturing module
record_master.dll	Master module for SynIPRecorder, responsible for signaling passing, receiving and parsing
SynGCI.dll	Communication module, responsible for data communication at the bottom layer
ShdPci.dll	DLL that is responsible for interaction with the hardware layer
hasp_windows.dll	DLL that is responsible for interaction with the USB KEY
USB KEY	For authorization management
record_slaver.exe	Slaver module for SynIPRecorder, responsible for RTP receiving, decoding, encoding, mixing and recording
SynRTP.dll	RTP receiving module, used to achieve the Jitter function
SwCodecConvAPI32.dll	Master module for voice data CODEC
SWG729API32.dll	Module for G.729 CODEC
SWGSM610API32.dll	Module for GSM CODEC

1.23.3 SynIPAnalyzer Concerned Terms

- Port Mirroring

  It is a method used on a network switch to send a copy of network packets seen on one or more than one switch port (or VLAN) to another one or more than one switch port.

- Station

  It is a terminal on VoIP line, that is, a VoIP phone either in software or in hardware.

- Call

  It means the entire process of a call, which starts with the dialing or sending call messages with terminals and ends with the hang-up or sending hang-up messages with terminals. A Station may make more than one call. For example, a mobile phone can hold several calls at the same time.

- Session

  It means the establishment of a session. To be exact, it is the establishment of a voice stream. Although there may be several Sessions in a call, they appear one after another, not existing concurrently. For example, when a call to your mobile phone is connected, a Session is established; when a party of the call presses the

HOLD key, the Session is stopped; then when you cancel the HOLD command to re-enter the call, it is another new Session that is established.

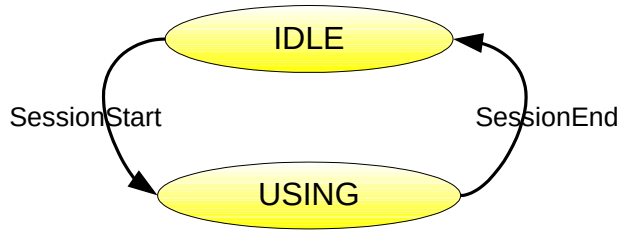
- Channel

  Channels on SynIPAnalyzer correspond to Sessions. When a new Session is established, the system will automatically search for an idle channel and bind it to the Session. Meanwhile, the channel goes into the USING state. When the Session is stopped, the system will automatically unbind the channel and the channel will go back into the IDLE state. See the channel state machine of SynIPAnalyzer for details.

- Authorization Strategy

  The authorized number of channels equals to that of monitored Stations in SynIPAnalyzer. That is, if you have purchased a license for 30, it means 30 channels are available and can be bound to 30 available Sessions. When all the 30 channels have been occupied, the new detected Session will send the event E_RCV_IPR_AUTH_OVERFLOW instead of E_RCV_IPR_MEDIA_SESSION_STARTED to represent authorization overflow. At that time, the number of current active Stations cannot be reflected from the number of channels, but you can obtain it via the function SsmIPRGetStationCount. Vice verse, when the number of current active Stations is more than 30, the event E_RCV_IPR_AUTH_OVERFLOW will be sent for reminding.

1.23.4 Channel State Machine of SynIPAnalyzer



Each state identifier in the above diagram is described in the following table:

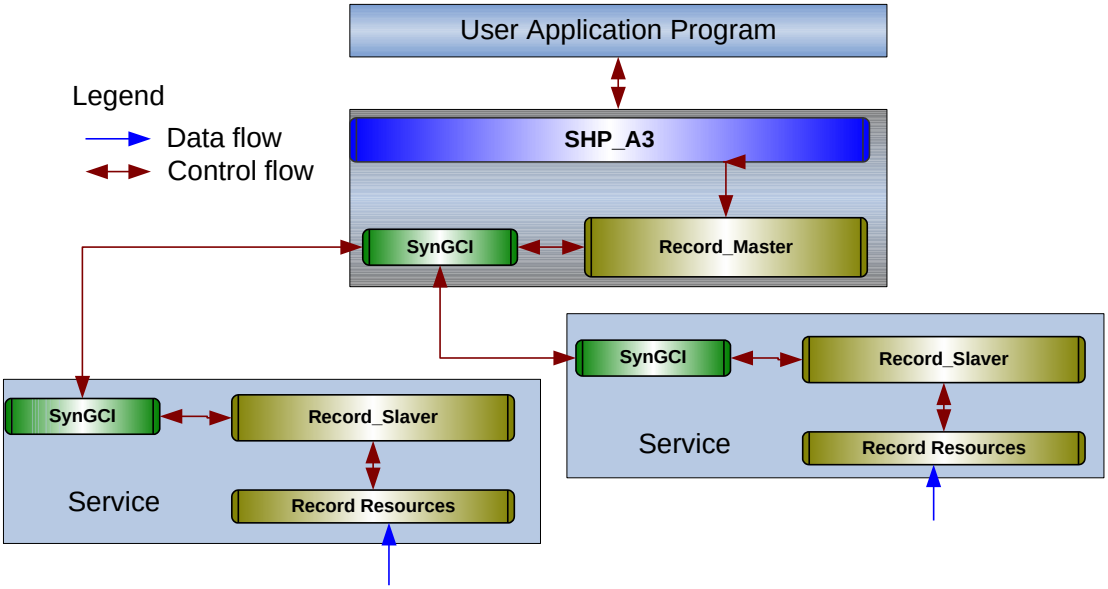
State Identifier	Description
IDLE	Channel 'IDLE' state ✧ When the channel is not bound to a Session, it is in the 'IDLE' state
USING	'USING' state ✧ When the channel is bound to a Session, it is in the 'USING' state

1.23.5 SynIPRecorder Concerned Terms

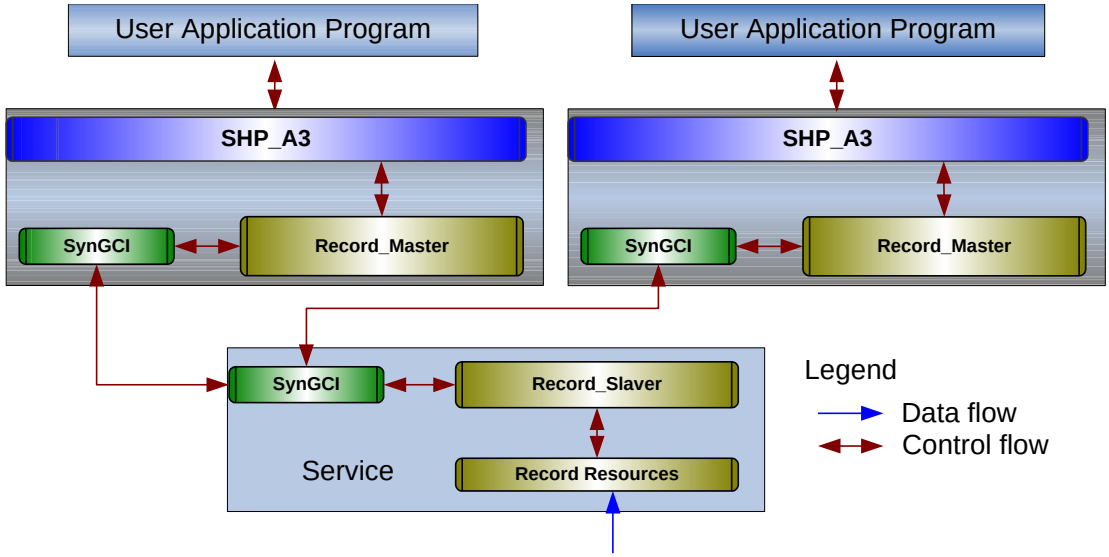
- Distributed Recording

  SynIPRecorder, whose Master end and Slaver end can work in different machines and establish communications through the TCP protocol, can adopt two distributed architectures. One is that the Master connects with multiple Record Slavers and takes a unified management over them, making recording channel expansion easily. The other is that multiple Masters connect to one Slaver, facilitating recording file management. See below for how they work exactly.

In this figure one Master connects with multiple Slavers.



In the following figure multiple Masters connect to one Slaver.



- Record Master

  Master is responsible for the dispatch and the channel authorization management over all of the Slavers. It provides the application with API functions via SHP_A3.DLL to control the Slaver.

- Record Slaver

  Running as Service on the machine, each Slaver in the initial state only has the capability of communicating with a specific Master. Only when the function SsmIPRStartRecSlaver is called in the Master and it is assigned with certain record resources and threads does it really go into work. At this time, the Master may call the functions SsmIPRActiveSession and SsmRecToFile to start a Slaver for the reception and recording of corresponding RTP packets.

- Record Resources

  How many Sessions can be held in a Slaver at the same time is determined by the number of Record

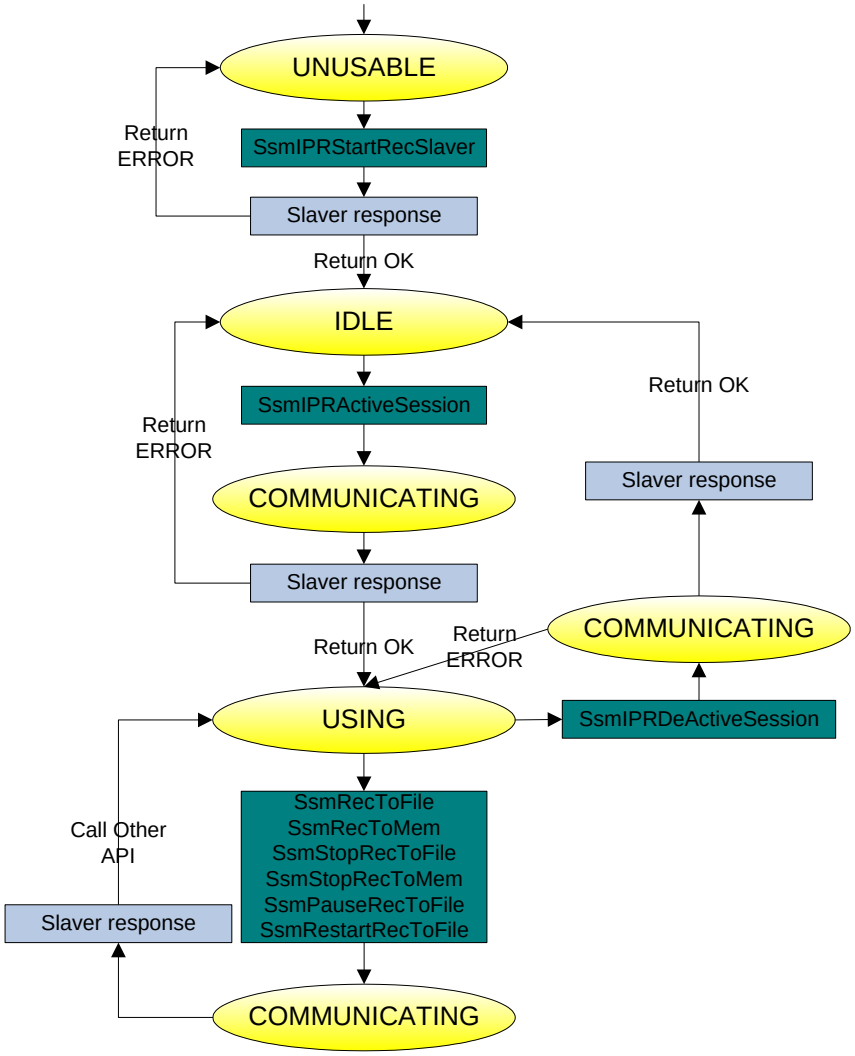
Resources. It is designated by the Master from the very beginning. Each Resource can be regarded as an independent recording unit.

- Session

  This concept to SynIPRecorder is the same as to SynIPAnalyzer. Actually, the session in SynIPRecorder is just the Session forwarded from SynIPAnalyzer.

Authorization Strategy: SynIPRecorder authorizes by number of channels and take a unified management in the Master. If you have purchased a license for 30, it means a Master, but not the slavers connected to the master, can record up to 30 channels concurrently. The Master has the right to decide how to distribute these 30 channels on the Slavers connected to it. For example, a master that connects with two Slavers - Slaver A and Slaver B can decide to distribute 10 Record Resources to Slaver A and another 20 Record Resources to Slaver B.

1.23.6 Channel State Machine of SynIPRecorder



State Identifier	Description
UNUSABLE	Channel 'UNUSABLE' state ✧ When not connecting with Slavers or not establishing record resources in the Slaver, a channel is unavailable.
IDLE	'IDLE' state ✧ A channel in this state is available for recording.

COMMUNICATING	'COMMUNICATING' state ✧ When the Master sends a request to the Slaver and is waiting for the response, a channel is in the COMMUNICATING state
USING	'USING' state ✧ A channel is occupied by a Session and cannot deal with any other Session.

1.23.7 Voice Operation on IPR Series

The IPR Series products support the following voice operations.

- **Recording**

The supported CODECs for recording include A-Law, μ-Law, IMA ADPCM, G729, GSM, MP3, etc.

- **RTP data reception**

The supported RTP CODECs include A-Law, μ-Law, G.729, G.722, G723, GSM, etc.

- **Recording Volume Adjust for Voice Signals**

The function SsmIPRSetRecVolume can be used to set the volume for the voice signals either from the PBX to the phone or from the phone to the PBX.

1.23.8 Programming Guide for IPR Series

Without unified standard, the signaling protocols used for the D-channel are usually decided by the PBX or telephone manufacturers. As a result, products from different manufacturers, even products in different series from a same manufacturer, are rather distinct.

The driver picks up the original signaling messages on the D-channel and passes them to the application by throwing out D-channel events. Then according to these D-channel events, the application program completes the signaling analysis over the call process.

It is recommended to use the EVENT_CALLBACKA or EVENT_POLLINGA programming mode. Please refer to the IPR Recording demo for details.

1.23.8.1 Original Signaling Messages on D-channel

Among the original signaling messages on D-channel, those sent from the PBX to the phone are called PBX messages, and the corresponding events are PBX events; those sent from the phone to the PBX are called phone messages, and the corresponding events are phone events. Common PBX events include:

- ✧ Ring events
- ✧ Audio events
- ✧ LCD (Display) events
- ✧ LED (Light) events

Common phone events include:

- ✧ Hook events
- ✧ Button events

In addition, the phone doesn't need to report some particular operations on it to the PBX, so it won't send any message to the PBX in such case. Thus even if the driver acquires all original signaling messages on the D-channel, it is probably not able to get any information about those operations on the phone, which include:

- ✧ Users press the volume adjuster on the phone;

- ✧ The phone displays on the LCD some of the information that is independent of the PBX, such as settings of some function buttons;
- ✧ The phone changes the voice channel state, which is unnecessary to be reported to the PBX;
- ✧ The LED on the phone varies in the state, which is unnecessary to be reported to the PBX.

For detailed information about the D-channel signaling messages, refer to *D-channel Event Manual*.

The application should set up an exclusive call state machine for a particular PBX or phone model by using some or all of the original signaling messages on the D-channel as the condition to trigger the state transition, then properly record the relative information during the call process and the relevant voice operations, like the start and stop of recording, according to the changes in the channel state.

Those events which may affect the call state machine but are unavailable from the D-channel original signaling messages, such as busy tones, ringback tones on voice channels, can be obtained via other voice processing features the driver provides.

1.23.9 Supported Protocols and Parameters for IPR Series

1.23.9.1 Universal Macro Definition and Structure

```
#define IP_TCP                          6   // TCP protocol
#define IP_UDP                          17 // UDP protocol
#define IPR_MAX_NON_STATION_LIST 25     // upper limit of none-station list
#define IPR_MAX_ADDTIONAL_PTL_PORT_LIST 20     // maximum number of ports monitored simultaneously in a protocol
#define IPR_MAX_H323_PTL_PORT_LIST     IPR_MAX_ADDTIONAL_PTL_PORT_LIST     // maximum number of ports monitored simultaneously in H323
#define IPR_MAX_SIP_PTL_PORT_LIST     IPR_MAX_ADDTIONAL_PTL_PORT_LIST  // maximum number of ports monitored simultaneously in SIP
typedef struct
{
    UCHAR               Protocol;       // transfer type (i.e., MT_TCP or MT_UDP)
    USHORT              Port;           // port number
}IPR_MONITOR_ITEM;
```

1.23.9.2 Supported Protocol

Protocol Type	Protocol ID	Macro Definition
SIP	0	PTL_SIP
CISCO SCCP	1	PTL_CISCO_SKINNY
AVAYA H323	2	PTL_AVAYA_H323
SHORTEL MGCP	3	PTL_SHORTEL_MGCP
H323	4	PTL_H323
PANASONIC MGCP	5	PTL_PANASONIC_MGCP
TOSHIBA MEGACO	6	PTL_TOSHIBA_MEGACO
SIEMENS H323	7	PTL_SIEMENS_H323

ALCATEL	8	PTL_ALCATEL
MITEL	9	PTL_MITEL
LG Nortel	10	PTL_ LG_ NORTEL
Samsung	11	PTL_SAMSUNG
Tadicom MGCP	12	PTL_TADICOM_MGCP
Zenitel	13	PTL_ZENITEL
Nortel UNIStim	14	PTL_NORTEL_UNISTIM

1.23.9.3 Configuration Structure

1.23.9.3.1 Configuration Structure for SIP

```
typedef struct
{
    IPR_MONITOR_ITEM Transport;    // main monitoring port and transfer type in SIP
    ULONG ProxyIPAddress;              // SIP Proxy/ALG/PBX IP Address
    USHORT NonStationListCount;  // amount of non-Station IP
    ULONG NonStationList [IPR_MAX_NON_STATION_LIST];// address of non-Station IP
    USHORT TransportAdditionalCount;  // number of additional monitoring ports in SIP
    IPR_MONITOR_ITEMTransport_Additional[IPR_MAX_SIP_PTL_PORT_LIST];  // additional monitoring port in SIP
    DWORD dwSpecial;  // auxiliary processing for some special PBXes. Currently, it can only be set to 1 to support the SIP/AASP protocol.
    BOOL bMixCSProtocol;  // necessary when both UDP and TCP protocols are used in a call.
} IPR_SIP_CFGS;
```

1.23.9.3.2 Configuration Structure for CISCO SCCP

```
typedef struct
{
    IPR_MONITOR_ITEM SCCP;          // monitoring port and transferring type in SCCP
}IPR_SCCP_CFGS;
```

1.23.9.3.3 Configuration Structure for AVAYA H323

```
typedef struct
{
    IPR_MONITOR_ITEM H225CS;       // monitoring port and transfer type in AVAYA H323
    IPR_MONITOR_ITEM H225RAS;    //registration management port and transfer type in AVAYA H323
    USHORT NonStationListCount;    //count of none-Station IP
    ULONG NonStationList [IPR_MAX_NON_STATION_LIST];    //address of none-Station IP
    USHORT H225CSAdditionalCount;    //count of additional monitoring port in AVAYA H323
```

IPR_MONITOR_ITEM H225CS_Additional[IPR_MAX_H323_PTL_PORT_LIST]; //additional monitoring port in AVAYA H323

USHORT H225RASAdditionalCount; //count of additional monitoring port in AVAYA H323

IPR_MONITOR_ITEM H225RAS_Additional[IPR_MAX_H323_PTL_PORT_LIST]; //additional registration management port in AVAYA H323

}IPR_H323_CFGS;

1.23.9.3.4Configuration Structure for SHORTEL MGCP

typedef struct

{

IPR_MONITOR_ITEM CallAgent; //agent port and transfer type in Shortel MGCP

IPR_MONITOR_ITEM Gateway; //gateway port and transfer type in Shortel MGCP

}IPR_SHORTEL_MGCP_CFGS;

1.23.9.3.5Configuration Structure for H323

typedef struct

{

IPR_MONITOR_ITEM H225CS; //monitoring port and transfer type in H323

IPR_MONITOR_ITEM H225RAS; //registration management port and transfer type in H323

USHORT NonStationListCount; //count of none-Station IP

ULONG NonStationList [IPR_MAX_NON_STATION_LIST]; //address of none-Station IP

USHORT H225CSAdditionalCount; //count of additional monitoring port in H323

IPR_MONITOR_ITEM H225CS_Additional[IPR_MAX_H323_PTL_PORT_LIST]; //additional monitoring port in H323

USHORT H225RASAdditionalCount; //count of additional registration management port in H323

IPR_MONITOR_ITEM H225RAS_Additional[IPR_MAX_H323_PTL_PORT_LIST]; //additional registration management port in H323

}IPR_H323_CFGS;

1.23.9.3.6Configuration Structure for PANASONIC MGCP

typedef struct

{

IPR_MONITOR_ITEM CallAgent; //agent port and transfer type in Panasonic MGCP

IPR_MONITOR_ITEM Gateway; //gateway port and transfer type in Panasonic MGCP

}IPR_PANASONIC_MGCP_CFGS;

1.23.9.3.7 Configuration Structure for TOSHIBA MEGACO

```
typedef struct
{
    IPR_MONITOR_ITEM Megaco_H248; //port and transfer type in Toshiba MEGACO
}IPR_TOSHIBA_MEGACO_CFGS;
```

1.23.9.3.8 Configuration Structure for Siemens H323

```
typedef struct
{
    IPR_MONITOR_ITEM Proprietary;    // Siemens H323 Proprietary Parameters
    IPR_MONITOR_ITEM H225CS;         // Siemens H323 H225CS Parameters
} IPR_SIEMENS_H323_CFGS;
```

1.23.9.3.9 Configuration Structure for Alcatel

```
typedef struct
{
    IPR_MONITOR_ITEM Alcatel;        // Alcatel Parameters
}IPR_ALCATEL_CFGS;
```

1.23.9.3.10 Configuration Structure for Mitel

```
typedef struct
{
    IPR_MONITOR_ITEM Mitel;          // Mitel Parameters
}IPR_MITEL_CFGS;
```

1.23.9.3.11 Configuration Structure for LG Nortel

```
typedef struct
{
IPR_MONITOR_ITEM LTPS;               // LG Nortel parameters of "Line Terminal Proxy Server"
    IPR_MONITOR_ITEM Phone;              // LG Nortel parameters of IP Phone
    USHORT NonStationListCount;
    ULONG NonStationList[IPR_MAX_NON_STATION_LIST];
}IPR_LG_NORTEL_CFGS;
```

1.23.9.3.12 Configuration Structure for Samsung

```
typedef struct
{
    IPR_MONITOR_ITEM Samsung;    // Samsung Parameters
}IPR_SAMSUNG_CFGS;
```

1.23.9.3.13 Configuration Structure for Tadicom MGCP

```
typedef struct
{
    IPR_MONITOR_ITEM CallAgent; // Tadicom MGCP parameters of Call Agent side
    IPR_MONITOR_ITEM Gateway; // Tadicom MGCP parameters of Agent side
}IPR_TADICOM_MGCP_CFGS;
```

1.23.9.3.14 Configuration Structure for Zenitel

```
typedef struct
{
    IPR_MONITOR_ITEM Zenitel;   // Zenitel Parameters
}IPR_ZENITEL_CFGS;
```

1.23.9.3.15 Configuration Structure for Nortel UNIStim

```
typedef struct
{
    IPR_MONITOR_ITEM NortelUNIStim;      // Nortel UNIStim Parameters
}IPR_NORTEL_UNISTIM_CFGS;
```

1.23.9.4 Programming Example for VoIP Recording System based on SynIPAnalyzer and SynIPRecorder

```
        // 1. Board Initialization and Callback Function Registration
    // Step 1: Start the board and register the callback function
    EVENT_SET_INFO EventMode;
    EventMode.dwWorkMode = EVENT_CALLBACKA;          // In event callback mode
    EventMode.lpHandlerParam = EventCallback;              // Callback function
    EventMode.dwUser = (DWORD)this;
    int nIsSsmStartCtiOK = SsmStartCtiEx("ShConfig.ini", "ShIndex.ini", true, &EventMode);
    if(nIsSsmStartCtiOK != 0)
    {
```

```
            SsmGetLastErrMsg(szErrMsg);
            AfxMessageBox(szErrMsg, MB_OK, 0);
            return FALSE;
        }
        // Step 2: Obtain the total number of boards
        if(SsmGetMaxUsableBoard() != SsmGetMaxCfgBoard())
        {
            SsmGetLastErrMsg(szErrMsg);
            AfxMessageBox(szErrMsg, MB_OK, 0);
        }
        // Step 3: Obtain the total number of channels
        MaxLine=SsmGetMaxCh();
        nIPRChNum = 0;
        nUsedIPRChNum = 0;
        nIPAChNum = 0;
        bUpdateSlaverDisplay = FALSE;
        // Step 4: Obtain the channel type for each channel
        for(int i = 0; i < MaxLine; i++)
        {
            if(SsmGetChType(i) == IPRR_CH)
            {
                nIPRChNum++;
            }
            else if(SsmGetChType(i) == IPRA_CH)
            {
                nIPAChNum++;
            }
        }

    // 2. Callback Function Implementation
int EventCallback(PSSM_EVENT pEvent)
{
    BOOL bFind = FALSE;
    int i, nResult, nPtlType, nStationId;
    char szEvtName[100], szIPP[50], szIPS[50], szTemp[100];
    pIPR_SessionInfo pSessionInfo;
    PIPR_CISCO_SCCP_CALL_INFO pSCCPInfo;
    PIPR_CALL_INFO pCallInfo;
    SYSTEMTIME time;

    nResult = 0;
    pIPRecorderDlg->bErrorOccurred = FALSE;
    switch(pEvent->wEventCode)
    {
```

```
    case E_CHG_ChState:
        // Channel state in SynIPAnalyzer or SynIPRecorder changes
        UpdateChannelState();
        break;
    case E_RCV_IPR_DChannel:
        // SynIPAnalyzer receives the D-channel events for IPR series
        // Update relative information according to the D-channel events
        if(pEvent->dwParam == DST_CISCO_SCCP_CALL_INFO)
        {
            // CISCO's CALL_INFO event, updating the CALL information
            nPtlType = pEvent->dwXtraInfo >> 16;
            nStationId = pEvent->dwXtraInfo & 0xffff;
            pSCCPInfo = (PIPR_CISCO_SCCP_CALL_INFO)pEvent->pvBuffer;
            // Update the CALL information
            //...
        }
        else if(pEvent->dwParam == DST_CALL_SUSPENDED || pEvent->dwParam ==
DST_CALL_RELEASED)
        {
            // release the call and update call information
            nPtlType = pEvent->dwXtraInfo >> 16;
            nStationId = pEvent->dwXtraInfo & 0xffff;
            pCallInfo = (PIPR_CALL_INFO)pEvent->pvBuffer;
        }
        break;
    case E_RCV_IPR_STATION_ADDED:
        // SynIPAnalyzer detects the entering of a new Station
        break;
    case E_RCV_IPR_STATION_REMOVED:
        // SynIPAnalyzer detects the exit of a Station
        break;
    case E_RCV_IPR_AUTH_OVERFLOW:
        // SynIPAnalyzer or SynIPRecorder authorization overflow
        break;
    case E_RCV_IPR_MEDIA_SESSION_STARTED:
        // SynIPAnalyzer detects the establishment of a new Session
        // Step 1: Confirm there is a slaver or more connected, or the recording will fail.
        if(nSlaverCount > 0)
        {
            // Obtain relative information about Session from the event
            pSessionInfo = (pIPR_SessionInfo)pEvent->pvBuffer;

            wsprintf(szIPP, "%d.%d.%d.%d", pSessionInfo->PrimaryAddr.S_un_b.s_b1,
pSessionInfo->PrimaryAddr.S_un_b.s_b2,
```

```
                pSessionInfo->PrimaryAddr.S_un_b.s_b3, pSessionInfo->PrimaryAddr.S_un_b.s_b4);
            wsprintf(szIPS,  "%d.%d.%d.%d",  pSessionInfo->SecondaryAddr.S_un_b.s_b1,
pSessionInfo->SecondaryAddr.S_un_b.s_b2,
                pSessionInfo->SecondaryAddr.S_un_b.s_b3, pSessionInfo->SecondaryAddr.S_un_b.s_b4);

            // Step 2: Find idle channels in SynIPRecorder
            for(i=0; i<MaxLine; i++)
            {
                if(hannelInfo[i].nChType == IPRR_CH)
                {
                    ChannelInfo[i].nWorkState = SsmGetChState(i);
                    if(ChannelInfo[i].nWorkState == S_CALL_STANDBY)
                    {
                        bFind = TRUE;
                        break;
                    }
                }
            }
            if(!bFind)
                break;

            // Step 3: Invoke the API function SsmIPRActiveSession in SynIPRecorder
            // Pass the Session information to SynIPRecorder, and specify a Slaver to receive the message
            // And obtain the port in Slaver which receives the Session
            if(SsmIPRActiveSession(i, IPR_SlaverAddr[0].nRecSlaverID,
                                    pSessionInfo->dwSessionId,
                                    szIPP, pSessionInfo->PrimaryAddr.usPort,
                                    &pSessionInfo->nFowardingPPort,
pSessionInfo->nPrimaryCodec,
                                    szIPS, pSessionInfo->SecondaryAddr.usPort,
                                    &pSessionInfo->nFowardingSPort,
pSessionInfo->nSecondaryCodec) != 0)
            {
                SsmGetLastErrMsg(szErrMsg);
                nResult = -1;
            }

            //Step 4: Store the RTP transmitting port to the corresponding IPR channel for use with the function
            SsmIPRSendSession.
            ChannelInfo[pEvent->nReference].nFowardingPPort = pSessionInfo->nFowardingPPort;
            ChannelInfo[pEvent->nReference].nFowardingSPort = pSessionInfo->nFowardingSPort;
        }
        break;
    case E_RCV_IPR_AUX_MEDIA_SESSION_STARTED:
```

```
        // This event indicating the repeated detection of the event E_RCV_IPR_MEDIA_SESSION_STARTED is not necessary to be administrated.
        break;
    case E_RCV_IPR_MEDIA_SESSION_STOPED:
        // SynIPAnalyzer detects the stop of a new Session
        // Step1: Obtain the Session information from the event
        pSessionInfo = (pIPR_SessionInfo)pEvent->pvBuffer;

        // Step 2: Find the bound SynIPAnalyzer channel
        for(i=0; i<MaxLine; i++)
        {
            if(ChannelInfo[i].dwSessionId == pSessionInfo->dwSessionId
                && ChannelInfo[i].nChType == IPRA_CH)
            {
                bFind = TRUE;
                break;
            }
        }
        if(!bFind)
            break;

        // Step 3: Invoke the API function SsmIPRStopSendSession in SynIPAnalyzer
        // Stop forwarding RTP message
        SsmIPRStopSendSession(i);

        // Step 4: Find the SynIPRecorder channel which receives the RTP message
        for(i=0; i<MaxLine; i++)
        {
            if(ChannelInfo[i].dwSessionId == pSessionInfo->dwSessionId
                && ChannelInfo[i].nChType == IPRR_CH)
            {
                ChannelInfo[i].dwSessionId = 0;
                ChannelInfo[i].nPtlType = -1;
                ChannelInfo[i].nStationId = -1;
                bFind = TRUE;
                break;
            }
        }
        if(!bFind)
            break;

        // Step 5: Invoke the API function SsmStopRecToFile in SynIPRecorder
        // Inform the Slaver to stop recording
        if(SsmStopRecToFile(i) != 0)
```

```
        {
            SsmGetLastErrMsg(pIPRecorderDlg->szErrMsg);
            pIPRecorderDlg->bErrorOccurred = TRUE;
            nResult = -1;
        }
        break;
    case E_IPR_SLAVER_INIT_CB:
        // Response to the initialized information of Master at the time when the Slaver is connected to the
Master
        break;
    case E_IPR_START_SLAVER_CB:
        // The Slaver responds to the Master‘s function call of SsmIPRStartRecSlaver
        UpdateChannelState();
        break;
    case E_IPR_CLOSE_SLAVER_CB:
        // The Slaver responds to the Master’s function call of SsmIPRCloseRecSlaver
        break;

    case E_IPR_ACTIVE_SESSION_CB:
        // The Slaver responds to the Master‘s function call of SsmIPRActiveSession
        // Step 1: First check whether the call of SsmIPRActiveSession is successful in the Slaver
        if(pEvent->dwParam != 0)
        {
            // If failed
            //do something
            //...
            break ;
        }

        // Step 2: If the function call is successful, invoke the API function SsmRecToFile in SynIPRecorder
        // Inform the Slaver to start recording
        if(SsmRecToFile(pEvent->nReference, szRecFilePath, 6, 0, -1, -1, 0) != 0)
        {
            SsmGetLastErrMsg(szErrMsg);
            nResult = -1;
        }

         //Step 3: Invoke the API function SsmIPRSendSession in SynIPAnalyzer
        bFind = FALSE;
        for(i=0; i<MaxLine; i++)
        {
            if(ChannelInfo[i].dwSessionId == ChannelInfo[pEvent->nReference].dwSessionId
                && ChannelInfo[i].nChType == IPRA_CH)
            {
```

```
                if(ChannelInfo[i].dwSessionId)
                {
                    bFind = TRUE;
                    break;
                }
            }
        }
        if(!bFind)
            break;
        //Send the Session to the designated IP address and port.
        wsprintf(szIPP_Rec, "%d.%d.%d.%d", IPR_SlaverAddr[0].ipAddr.S_un_b.s_b1,
            IPR_SlaverAddr[0].ipAddr.S_un_b.s_b2,
            IPR_SlaverAddr[0].ipAddr.S_un_b.s_b3,
            IPR_SlaverAddr[0].ipAddr.S_un_b.s_b4);
        wsprintf(szIPS_Rec, "%d.%d.%d.%d", IPR_SlaverAddr[0].ipAddr.S_un_b.s_b1,
            IPR_SlaverAddr[0].ipAddr.S_un_b.s_b2,
            IPR_SlaverAddr[0].ipAddr.S_un_b.s_b3,
            IPR_SlaverAddr[0].ipAddr.S_un_b.s_b4);
        if(SsmIPRSendSession(i, szIPP, ChannelInfo[i].nFowardingPPort,
                szIPS, ChannelInfo[i].nFowardingSPort) != 0)
        {
            SsmGetLastErrMsg(szErrMsg);
            nResult = -1;
        }
        break;
    case E_IPR_DEACTIVE_SESSION_CB:
        // The Slaver responds to the Master for calling the function SsmIPRDeActiveSession
        break;
    case E_IPR_START_REC_CB:
        // The Slaver responds to the Master's function call of SsmRecToFile or SsmRecToMem
         If(SsmIPRDeActiveSession(pEvent->nReference)
        {
            SsmGetLastErrMsg(szErrMsg);
            nResult = -1;
        }
        break;
    case E_IPR_STOP_REC_CB:
        // The Slaver responds to the Master's function call of SsmStopRecToFile or SsmStopRecToMem
        break;
    case E_IPR_LINK_REC_SLAVER_CONNECTED:
        // The Master detects the connection of a Slaver
        break;
    case E_IPR_LINK_REC_SLAVER_DISCONNECTED:
        // The Master detects the disconnection of a Slaver
```

```
            break;
        default:
            break;
        }

        // Display the event information on the graphical interface
        EventDisplay(szEvtName, pEvent);

        return nResult;
}
```

1.24 HMP Series (CTI Series)

1.24.1 Brief Introduction of HMP Series

The HMP series product is a sort of special software which integrates the functions of SIP, SIP Server, voice playing and recording, data transcoding, faxing and conferencing etc.. It is used to construct the systems of media gateway, media server, IVR, call center, conference and so on. The HMP series product adopts the distributed architecture, supports up to 2400 media and signaling channels. It is a pure software, easy to obtain, develop, deploy, maintain and expand. It consists of two parts: HMP Client and HMP Server.

The HMP Server has the following processing capabilities:

- Voice recording and playing
- Conferencing
- Voice encoding and decoding
- Detecting and transmitting DTMF in signaling messages and REC2833
- Allows the detection of In-band DTMF
- SIP

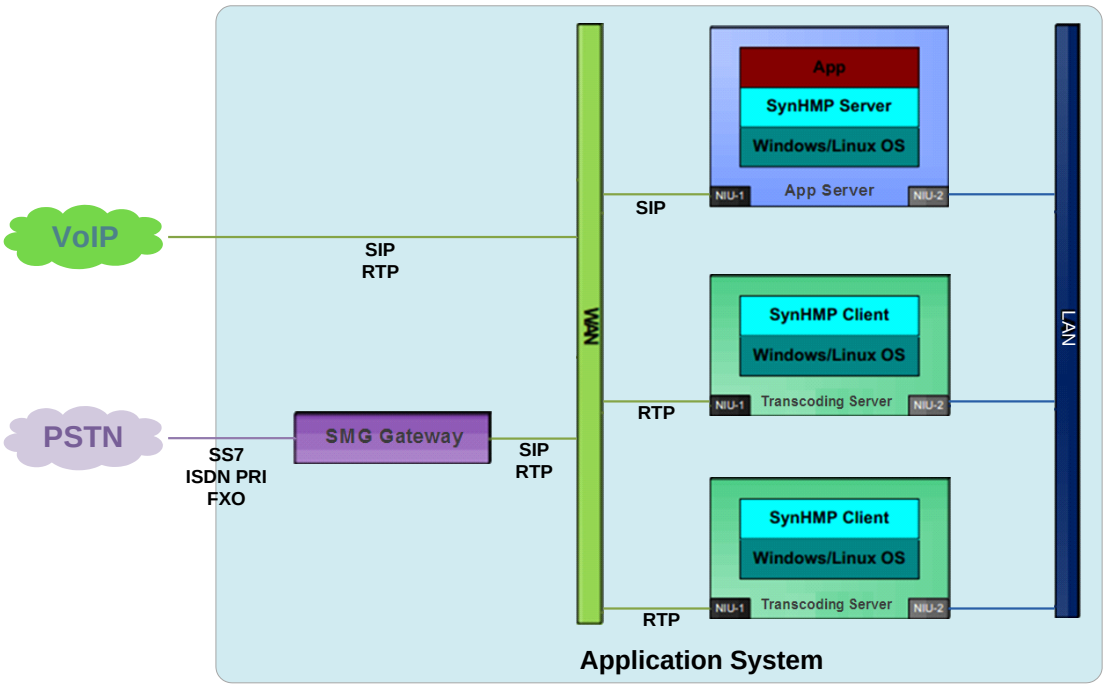
The HMP Client has the following processing capabilities:

- Distributed module deployment and expansion
- Receiving and transmitting RTP data in different encoding formats

The HMP series product uses USB KEY for encryption. See below for details.

Type	Installation	Operation	Upgrade File
USB-KEY	-	-	Independent EXE file.

1.24.2 System Structure of HMP Series



- Application Server
  - Dual Ethernet ports, Kilomega recommended
  - Operating System, Linux Centos or Windows Server recommended
  - SynHMP Server, supporting up to 2400 channels, 8-core 3.0G CPU or above recommended
  - APP, event programming mode recommended
- Transcoding Server
  - Dual Ethernet ports, Kilomega recommended
  - Operating System, Linux Centos recommended
  - SynHMP Client, with the supported channel quantity varying on CODEC, running on the application server with a few channels or on multiple transcoding servers with a lot of channels.

1.24.3 Configuration and Use of HMP Series

The HMP series product consists of two parts: HMP Client and HMP Server

HMP Client (composed of HMPCodec application and HMPLink):

HMP Client is used to receive the RTP data from the remote end and send it to HMP Server after encoding/decoding. Meanwhile, it is used to receive the RTP data from the HMP Server and send it to the remote end after transcoding it to the required format.

HMP Client can work in different machines and operating systems. Its service program is HMPCodec.exe, and the configuration file it uses is HMPCodec.ini.

HMP Server (composed of HMPVirtualDevice, HMPRouter, HMPLink, ShpPci, SynSip, SHP_A3 and application):

HMP Server runs on the host and controls one or more HMP Clients. HMP Server is used to receive the RTP data

from HMP Client, transform the RTP into voice data and record the voice. Meanwhile, it transforms the voice playing data to RTP data and forwards the data to HMP Client.

The basic function of HMP Server is similar to that of IP board, including SIP, conferencing, data exchange, voice recording and playing, DTMF receiving and transmitting, registration, etc.. The interface is the same as that of IP board.

The detection mode of HMP is similar to that of IPR, that is, they are both encrypted by USB KEY. However, the contents programmed into the USB key are distinguished from each other.

1.24.4 Special Configuration Item for HMP Client

Configuration Section	Configuration Item	
[HMPCodec]	*NICNum*	The amount of the network port used by HMP Client
	LocalIP[n]	IP address of the host where HMP Client locates
	RemoteIP[n]	IP address of the host where HMP Server locates
	LocalPort[n]	Port of the host where HMP Client locates
	RemotePort[n]	Port of the host where HMP Server locates
	MaxRtpThread	Thread amount of HMP Client for RTP CODEC
	LogType	Log output mode of HMP Client
	LogLocation	Log output path of HMP Client

1.24.5 Special Configuration Item for HMP Server

Configuration Section	Configuration Item	
[HMPRouter]	*TotalMediaForward*	Amount of HMP Client
	MaxMediaThread	Thread amount of HMP Server for media processing
	LocalIP[n]	IP address of host where HMP Server locates
	TotalCh[n]	Amount of channels supporting CODEC on HMP Client
	LocalPort[n]	Port number of HMP Server
	RemoteIP[n]	IP address of the host where HMP Client locates
	RemotePort[n]	Port number of HMP Client

1.24.6 Sample Configuration of HMP Server and HMP Client

1. Single PC, single network card and single client (Under this mode, up to 1200 channels are supported. If channels are more than 1200, you can use the mode of single PC multiple network cards or multiple PCs multiple network cards):

   HMP Server (IP address of Host A: 192.168.0.101)

   Configuration file: ShConfig.ini

   [HMPRouter]

   TotalMediaForward=1

   MaxMediaThread=4

   TotalCh[0]=600

   LocalIP[0]=127.0.0.1

   LocalPort[0]=5051

   RemoteIP[0]=127.0.0.1

   RemotePort[0]=5050

RtpIP[0]=192.168.0.101

HMP Client

Configuration file: HMPCodec.ini

[HMPCodec]

NICNum=1

LocalI[0]P=127.0.0.1

RemoteIP[0]=127.0.0.1

LocalPort[0]=5050

RemotePort[0]=5051

MaxRtpThread=4

LogType=2

2. Single PC, multiple network cards and single client (The client and server locate in the same host. Under this mode, each network card supports up to 800 channels):

HMP Server (IP address of Host A: 192.168.0.101, 192.168.0.102)

Configuration file: ShConfig.ini

[HMPRouter]

TotalMediaForward=2

MaxMediaThread=4

TotalCh[0]=600

LocalIP[0]=192.168.0.101

LocalPort[0]=5051

RemoteIP[0]=192.168.0.101

RemotePort[0]=5050

RtpIP[0]= 192.168.0.101

TotalCh[1]=600

LocalIP[1]=192.168.0.102

LocalPort[1]=5053

RemoteIP[1]=192.168.0.102

RemotePort[1]=5052

RtpIP[1]=192.168.0.102

HMP Client

Configuration file: HMPCodec.ini

[HMPCodec]

NICNum=2

LocalIP[0]=192.168.0.101

RemoteIP[0]=192.168.0.101

LocalPort[0]=5050

RemotePort[0]=5051

LocalIP[1]=192.168.0.102

RemoteIP[1]=192.168.0.102

LocalPort[1]=5052

RemotePort[1]=5053

MaxRtpThread=4

LogType=2

3. Multiple PC, multiple network cards and multiple clients (The client and server locate in the different host. Under this mode, each network card supports up to 800 channels):

HMP Server A (IP address of Host A: 192.168.0.101、192.168.0.102)

Configuration file: ShConfig.ini

[HMPRouter]

TotalMediaForward=2

MaxMediaThread=4

TotalCh[0]=600

LocalIP[0]=192.168.0.101

LocalPort[0]=5051

RemoteIP[0]=192.168.0.201

RemotePort[0]=5050

RtpIP[0]= 192.168.0.201

TotalCh[1]=600

LocalIP[1]=192.168.0.102

LocalPort[1]=5053

RemoteIP[1]=192.168.0.202

RemotePort[1]=5052

RtpIP[1]=192.168.0.202

HMP Client (IP address of Host B: 192.168.0.201)

Configuration file: HMPCodec.ini

[HMPCodec]

NICNum=1

LocalIP[0]=192.168.0.201

RemoteIP[0]=192.168.0.101

LocalPort[0]=5050

RemotePort[0]=5051

MaxRtpThread=4

LogType=2

HMP Client (IP address of Host C: 192.168.0.202)

Configuration file: HMPCodec.ini

[HMPCodec]

NICNum=1

LocalIP[0]=192.168.0.202

RemoteIP[0]=192.168.0.102

LocalPort[0]=5052

RemotePort[0]=5053

MaxRtpThread=4

LogType=2

2 SynCTI API Function Description

2.1 System Functions

2.1.1 Driver Platform Initialization Functions

2.1.1.1 SsmStartCti

SsmStartCti is the entry function for the entire SynCti API which can be used to initialize the SynCTI driver platform and voice boards.

Format:

int SsmStartCti(LPSTR *lpSsmCfgFileName*, LPSTR *lpIndexCfgFileName*)

Parameter Description:

lpSsmCfgFileName	The name of the configuration file (*.INI). If it's a NULL string, under Windows operating system, the driver will automatically use the configuration file 'ShConfig.ini' under the system directory; under Linux operating system, the driver will automatically use the configuration file 'ShConfig.ini' under the directory of '/etc/shcti/'.
lpIndexCfgFileName	The configuration file for preload voice data (*.INI). If it's a NULL string, under Windows operating system, the driver will automatically use the configuration file 'ShIndex.ini' under the system directory; under Linux operating system, the driver will automatically use the configuration file 'ShIndex.ini' under the directory of '/etc/shcti/'.

Return Value:

-2	Initialization is failed, because the driver has been loaded
-1	Initialization is failed, the failure reason can be obtained by SsmGetLastErrMsg
0	Initialization is successful

Function Description:

This function is the entry of the SynCTI driver program. After invoking this function, the following operations will be executed:

- Initializes the voice board;
- If lpIndexCfgFileName is not NULL, loads the preload voice data into the memory from the configuration file 'lpIndexCfgFileName'.
- Registers and initializes the voice boards and allocates the system resources based on the basic information such as the model and quantity of voice board, port address, and interruption number.
- Allocates system resources for voice recording and playback.
- Establishes an interrupt service routine (ISR).

Note:

- Only when this function or the function SsmStartCtiEx has been called successfully, can the other functions in the driver platform be called (except SsmGetLastErrMsg and SsmGetLastErrCode).
- Before exiting the application program, SsmCloseCti must be called to close the driver program in order to release the resources occupied by the driver.
- For detailed information about the configuration file, refer to Chapter 4.
- If multiple boards have been configured and only a part of them is successfully initialized, the driver also returns 0. The application program can determine whether all the boards are initialized successfully through the functions SsmGetMaxCfgBoard and SsmGetMaxUsableBoard.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Functions: SsmStartCtiEx, SsmCloseCti, SsmGetMaxCfgBoard , SsmGetMaxUsableBoard

2.1.1.2 SsmStartCtiEx

SsmStartCtiEx is the entry function for the entire SynCti API which can be used to initialize the SynCTI driver platform and voice boards. Besides the features of SsmStartCti, it can set the event output mode.

Format:

int SsmStartCtiEx(LPSTR *lpSsmCfgFileName*, LPSTR *lpIndexCfgFileName*, BOOL *bEnable*, PEVENT_SET_INFO *pEventSet*)

Parameter Description:

lpSsmCfgFileName	The name of the configuration file (*.INI). If it's a NULL string, under Windows operating system, the driver will automatically use the configuration file 'ShConfig.ini' under the system directory; under Linux operating system, the driver will automatically use the configuration file 'ShConfig.ini' under the directory of '/etc/shcti/'.
lpIndexCfgFileName	The configuration file for preload voice data (*.INI). If it's a NULL string, under Windows operating system, the driver will automatically use the configuration file 'ShIndex.ini' under the system directory; under Linux operating system, the driver will automatically use the configuration file 'ShIndex.ini' under the directory of '/etc/shcti/'.
bEnable	The control sign of the event output. If bEnable=FALSE, the driver will not output any event; if bEnable=TRUE, whether the driver will output any event is determined by the configuration item DefaultEventOutput.
pEventSet	Pointer which points to the object with the struct EVENT_SET_INFO. It's used for setting the event output mode. This parameter can't be NULL; otherwise the feature of setting event output mode is invalid and this function is absolutely the same as the function SsmStartCti. The declaration of the struct EVENT_SET_INFO is: typedef struct _EVENT_SET_INFO { DWORD dwWorkMode; LPVOID lpHandlerParam; DWORD dwOutCondition; DWORD dwOutParamVal; DWORD dwUser; } EVENT_SET_INFO,*PEVENT_SET_INFO; Below is the meaning of each parameter:

Parameter Name: *dwWorkMode*

Sets the event output mode of the driver. Range of value:

Value	Macro in event.h	Description
0	NO_EVENT	The driver doesn't output any events.
1	EVENT_POLLING	The driver works under the event polling mode. The parameters ***dwOutCondition*** and ***dwOutParamVal*** are valid, and the parameters ***lpHandlerParam*** and ***dwUser*** are invalid.
2	EVENT_CALLBACK	The driver works under the event callback mode. The parameters ***lpHandlerParam*** and ***dwUser*** are valid.
3	EVENT_MESSAGE	The driver works under the Windows message mode. All the messages will be sent to the windows message queue. This mode is invalid under Linux operating system.
4	EVENT_POLLINGA	The driver works under the event polling mode. The parameters ***dwOutCondition*** and

		dwOutParamVal are valid, the parameters ***lpHandlerParam*** and ***dwUser*** are invalid.
5	EVENT_CALLBACKA	The driver works under the event callback mode. ***lpHandlerParam*** and ***dwUser*** are valid.

Parameter Name: *lpHandlerParam*

Only when ***dwWorkMode***=EVENT_CALLBACK, EVENT_CALLBACKA or EVENT_MESSAGE is this parameter valid. If ***dwWorkMode*** equals to other values, this parameter will be ignored.

- ***dwWorkMode***=EVENT_CALLBACK
  lpHandlerParam is the pointer to the callback function.
  - The format of ***lpHandlerParam*** is:
    int CALLBACK CallBackFun(WORD wEvent, int nReference, DWORD dwParam, DWORD dwUser);
    The meanings of wEvent, nReference and dwParam are the same as the corresponding variables declared in the struct MESSAGE_INFO.
    For more details, refer to the function SsmWaitForEvent or SsmGetEvent. The parameter dwUser is the parameter ***dwUser*** which is passed from the application to the driver when invoking this callback function.
- ***dwWorkMode***=EVENT_CALLBACKA
  lpHandlerParam is the pointer to the callback function.
  - The format of ***lpHandlerParam*** is:
    int CALLBACK CallBackFunA(PSSM_EVENT pEvent):
    pEvent which returns the event information is the pointer pointing to the object of struct SSM_EVENT. For more details, refer to the function SsmWaitForEventA or SsmGetEventA.
    Note: When the callback function returns -1, all events are forbidden being thrown out. To throw out the events again, you should call and reset the function SsmSetEvent.
- ***dwWorkMode***=EVENT_MESSAGE
  lpHandlerParam is the window handle which is used by the application to receive the driver events.

Parameter Name: *dwUser*

User-defined parameter which is valid only when ***dwWorkMode***=EVENT_CALLBACK or EVENT_CALLBACKA. If ***dwWorkMode*** equals to other values, this parameter will be ignored. The driver will save this parameter. When the driver throws out the events, it will be directly passed to the parameter having the same name in the callback function pointed by ***lpHandlerParam***.

Parameter Name: *dwOutCondition, dwOutParamVal*

dwOutCondition sets the conditions on which the events are output. ***dwOutParamVal*** sets the types of the parameters of the output events. When the event is thrown out, the driver will assign the data types set by ***dwOutParamVal*** to the parameter dwParam in the struct MESSAGE_INFO or SSM_EVENT. There is no need to set the parameters ***dwOutCondition*** and ***dwOutParamVal*** in this function because only the default values of them will be used.

Return Value:

-2	Initialization is failed, because the driver has been loaded
-1	Initialization is failed, the failure reason can be obtained by SsmGetLastErrMsg
0	Initialization is successful

Function Description:

This function is the entry of the SynCTI driver program. After invoking this function, the following operations will be executed:

- Initializes the voice board.
- Sets the event output mode of the driver.
- If lpIndexCfgFileName is not NULL, loads the preload voice data into the memory from the configuration

file 'lpIndexCfgFileName'.

- Registers and initializes the voice boards and allocates the system resources based on the basic information such as the model and quantity of voice board, port address, and interruption number.
- Allocates system resources for voice recording and playback.
- Establishes an interrupt service routine (ISR).

Note:

- Only When this function or the function SsmStartCti has been called successfully, can the other functions in the driver platform be called (except SsmGetLastErrMsg and SsmGetLastErrCode).
- The last parameter pEventSet of this function uses the same struct as the parameter pEventSet of the function SsmSetEvent, but they differ slightly in usage. Please refer to the corresponding descriptions.
- Before exiting the application program, SsmCloseCti must be called to close the driver program in order to release the resources occupied by the driver.
- For detailed information about the configuration file, refer to Chapter 4.
- If multiple boards have been configured and only a part of them is successfully initialized, the driver also returns 0. The application program can determine whether all the boards are initialized successfully through the functions SsmGetMaxCfgBoard and SsmGetMaxUsableBoard.

Related Information:

Driver version	SynCTI Ver. 5.3.2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Functions: SsmStartCti, SsmCloseCti, SsmGetMaxCfgBoard, SsmGetMaxUsableBoard, SsmSetEvent

2.1.1.3 SsmCloseCti

Closes the SynCTI driver program.

Format:

int SsmCloseCti (void)

Parameter Description:

None

Return Value:

1	Successful to close the SynCTI driver program

Function Description:

Closes the SynCTI driver program.

Note:

- Before the application program exits the operating system, this function must be called.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmStartCti

2.1.2 Setting Functions for API and Event Output Mode

2.1.2.1 SsmSetLogEnable

Sets whether to output the logs of specified types and the way to output.

Format:

int WINAPI SsmSetLogEnable(int nLogType,int nLogEnable,int nLogCreateMode)

Parameter Description:

nLogType	Log type, with the value range of 0~8: =0: ShCtiApiLog =1: ShCtiSs1Log =2: ShCtiIsdnLog =3: ShCtiSpySs7Log =4: ShCtiSpySs1Log =5: ShCtiSpyIsdnLog =6: ShCtiSystemInfoLog =7: ShCtiDstLog =8: ShCtiFaxLog
nLogEnable	Sets whether to output logs. When nLogType=0, nLogEnable can be of the following values: =0: Not output logs =1: Output API function call information to DebugView =2: Output API function call information to the LOG file =3: Output event information to the LOG file =4: Output both API function call information and event information to the LOG file When nLogType=1, 2, 3, 4, 5, 6, 7, 8 nLogEnable can be of the following values: =0: Not output =1: Output to the LOG file =2: Output to DebugView
nLogCreateMode	The mode to create a log. It gets valid only when nLogEnable is set to output logs. =0: All channel information written into a log file =1: Each channel has an independent log file. This value gets valid only when nLogType=0, 1, 4, 7, 8

Return Value:

-1	Call failed
0	Call successful

Function Description:

Sets whether to output the logs of specified types and the way to output. No matter which type of logs the driver outputs, it will create the ShCtiLog folder and save all generated log files in this folder.

The log types described by nLogType are as follows.

Log Type	Definition
ShCtiApiLog	Records information about API function calls and events thrown out by the driver.
ShCtiSs1Log	Records received and sent ABCD codes and R2
ShCtiIsdnLog	Records a frame of messages on the data link layer of ISDN protocol
ShCtiSpySs7Log	Records a frame of messages that the driver receives
ShCtiSpySs1Log	Records information about SS1 CAS and R2 that the driver receives
ShCtiSpyIsdnLog	Records a frame of messages that the driver receives
ShCtiSystemInfoLog	Records system information, such as the link sync, the local-end impedance,

	E1/T1, CRC-4 and other running information
ShCtiDstLog	Records the digital call monitoring signaling message
ShCtiFaxLog	Records the fax interaction command

The following configuration items can be used to achieve the same purpose.

ApiLogEnable
ApiLogCreateMode
Ss1LogEnable
Ss1LogCreateMode
SpySs1LogEnable
SpySs1CreateMode
IsdnLogEnable
SpySs7LogEnable
SpyIsdnLogEnable
SystemInfoLogEnable
DSTLogEnable
DSTLogCreateMode
FaxLogEnable
FaxLogCreateMode

Note: None

Related Information:

Driver version	SynCTI Ver.5.1.1.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmSetLogAttribute, SsmGetLogAttribute, SsmSetApiLogRange

2.1.2.2 SsmSetLogAttribute

Sets the period to generate a log, the maximum number of saved latest logs, the number of cycles to stop log output and the log output path.

Format:

int WINAPI SsmSetLogAttribute(int nLogCreatePeriod,int nLogMaxKeep,int nLogMaxPeriod,LPSTR pLogFilePath)

Parameter Description:

nLogCreatePeriod	The generation period of the log (hour), with the value range of nLogCreatePeriod>0.
nLogMaxKeep	When the number of generated log files is greater than the set value of this parameter, the first generated files will be deleted automatically. The value range is nLogMaxKeep >0. If nLogMaxKeep =-1, it means there is no limit on the number to save log files and no log files will be deleted.
nLogMaxPeriod	When the number of log generating periods reaches the set value of this parameter, the log output will be stopped automatically. The value range is nLogMaxPeriod >0. If nLogMaxPeriod = -1, it means there is no limit on the number to generate log files.
pLogFilePath	The path to save logs. Note that it must be a path already existing, such as 'D:\', or the function call will fail. If pLogFilePath is set to null, it indicates the current working path; if pLogFilePath is set to a name, a folder with such name will be created under the current working path.

Return Value:

-1	Call failed
0	Call successful

Function Description:

Sets the period to generate a log, the maximum number of saved latest logs, the number of cycles to stop log output and the log output path.

Settings of this function are valid to the following logs.

Log Type	Definition
ShCtiApiLog	Records information about API function calls and events thrown out by the driver.
ShCtiSs1Log	Records received and sent ABCD codes and R2
ShCtiIsdnLog	Records a frame of messages on the data link layer of ISDN protocol
ShCtiSpySs7Log	Records a frame of messages that the driver receives
ShCtiSpySs1Log	Records information about SS1 CAS and R2 that the driver receives
ShCtiSpyIsdnLog	Records a frame of messages that the driver receives
ShCtiSystemInfoLog	Records system information, such as the link sync, the local-end impedance, E1/T1, CRC-4 and other running information
ShCtiDstLog	Records the digital call monitoring signaling message

The following configuration items can be used to achieve the same purpose.

LogCreatePeriod

LogMaxKeep

LogMaxPeriod

LogFilePath

Note: None

Related Information:

Driver version	SynCTI Ver.5.1.1.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmSetLogEnable, SsmGetLogAttribute, SsmSetApiLogRange

2.1.2.3 SsmSetApiLogRange

Sets the range of channels where events and API functions should be written into the log and the range of events that should be recorded into the log.

Format:

int WINAPI SsmSetApiLogRange(int nChStart, int nChEnd, int nEventStart, int nEventEnd)

Parameter Description:

nChStart	Indicates the start channel number for logging events and API functions. When nChStart=-1, it means the channel range is <nChEnd.	
nChEnd	Indicates the end channel number for logging events and API functions. When nChEnd=-1, it means the channel range is >nChStart.	
nEventStart	Indicates the start event number that should be written into the log. When nEventStart =-1, it means the event range is <nEventEnd.	
nEventEnd	Indicates the end event number that should be written into the log. When nEventEnd=-1, it means the event range is >nEventStart.	

Return Value:

-1	Call failed, the failure reason can be obtained from the function SsmGetLastErrMsg
0	Call successful

Function Description:

Sets the range of channels where events and API functions should be written into the log and the range of events that should be recorded into the log on condition that ApiLogEnable is set to output logs.

The following configuration items can be used to achieve the same purpose.

ApiLogSetEventRange

ApiLogSetChRange

Note: None

Related Information:

Driver version	SynCTI Ver. 5.1.1.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmSetLogEnable, SsmSetLogAttribute, SsmGetLogAttribute

2.1.2.4 SsmGetLogAttribute

Gets the period to generate a log, the maximum number of saved latest logs, the number of cycles to stop log output and the log output path.

Format:

int WINAPI SsmGetLogAttribute(PINT pLogCreatePeriod,PINT pLogMaxKeep,PINT pLogMaxPeriod,LPSTR pLogFilePath)

Parameter Description:

pLogCreatePeriod	The period to generate a log, calculated by hour, with the value range of pLogCreatePeriod>0.
pLogMaxKeep	When the number of generated log files is greater than the set value of this parameter, the first generated files will be deleted automatically. The value range is pLogMaxKeep >0. If pLogMaxKeep =-1, it means there is no limit on the number to save log files and no log files will be deleted.
pLogMaxPeriod	When the number of log generating periods reaches the set value of this parameter, the log output will be stopped automatically. The value range is pLogMaxPeriod >0. If pLogMaxPeriod = -1, it means there is no limit on the number to generate log files.
pLogFilePath	The path to save logs. Note that it must be a path already existing, such as 'D:\', or the function call will fail. If pLogFilePath is set to null, it indicates the current working path; if pLogFilePath is set to a name, a folder with such name will be created under the current working path.

Return Value:

-1	Call failed
0	Call successful

Function Description:

Gets the period to generate a log, the maximum number of saved latest logs, the number of cycles to stop log output and the log output path.

Note: None

Related Information:

Driver version	SynCTI Ver.5.1.1.0 or above
Header	shpa3api.h

Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmSetLogEnable, SsmSetLogAttribute, SsmSetApiLogRange

2.1.3 Timer Functions

2.1.3.1 SsmStartTimer

Is applicable to the driver for a system timer.

Format:

int SsmStartTimer(WORD wDelay, WORD wMode)

Parameter Description:

wDelay	The initial value (millisecond) of the timer, range of value: 0x0008~0xffff
wMode	Sets the timer operating mode: =0(TIMER_ONE): One-shot mode: Once the timer overflows, it stops automatically =1(TIMER_PERIODIC): Periodic mode: Every time when the timer overflows, the driver program automatically reloads the initial value wDelay until the application program calls SsmStopTimer

Return Value:

-1	Failed
≥0	The timer application is successful and 'SsmStartTimer' returns the timer ID

Function Description:

The application is applicable to the driver for a system timer and sets the operating parameter of the timer. Every time when the timer overflows, the driver throws out an event E_SYS_TIMEOUT For more details, refer to 'Driver-provided Timer' in Chapter 1.

Note:

- This function must be called after SsmSetEvent, otherwise the driver won't output the event TIMEOUT.

Related Information:

Driver version	SynCTI Ver.4.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Functions: SsmStopTimer, StartTimer

Related Events: E_SYS_TIMEOUT

2.1.3.2 SsmStopTimer

Stops a Timer.

Format:

int SsmStopTimer(int nTimer)

Parameters Description:

nTimer	Timer ID ,the return value of SsmStartTimer

Return Value:

-1	Failed
0	Successful

Function Description:

Stops a timer which is started by SsmStartTimer .

Note:

Related Information:

Driver version:	SynCTI Ver. 4.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmStartTimer

2.1.3.3 StartTimer

Starts a channel timer.

Format:

int StartTimer(int ch, WORD wClockID)

Parameter Description:

ch	Channel Number
wClockID	Timer ID, range of value: 0~9

Return Value:

-1	Failed
0	Successful

Function Description:

Starts a channel timer.

After the channel timer has been started, the driver only saves the current time in a variable and doesn't determine whether the timer overflows. Only when the application calls the function of ElapseTime, will the driver return the timer elapsed time.

Note:

- The timer started by StartTimer() doesn't output any events.

Related Information:

Driver version	SynCTI Ver. 4.7.1.5 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: ElapseTime

2.1.3.4 ElapseTime

Obtains the elapsed time since the channel timer is started.

Format:

DWORD ElapseTime(int ch, WORD ClockType)

Parameter Description:

ch	Channel number
ClockType:	Timer ID, value range: 0~9

Return Value:

Elapsed time, the value is n x 8ms, n=1, 2, 3…

Function Description:

Obtains the elapsed time since the channel timer is started.

Note:

Related Information:

Driver version	SynCTI Ver. 4.7.1.5 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: StartTimer

2.1.4 Functions to Get System Error Messages

2.1.4.1 SsmGetLastErrCode

Obtains the error code when the application is failed to call the API functions.

Format:

int SsmGetLastErrCode()

Parameter Description:

None

Return Value:

Below are the return values:

Return Value	Macro in shpa3api.h	Description
0	C_ERROR_INIT_FAILURE	Initialization is failed
1	C_ERROR_SSMAPI_UNOPENED	The API interface of the driver is not open to the application
2	C_ERROR_INVALID_APPCH	Invalid channel number
3	C_ERROR_UNSUPPORTED_OP	Unsupported operation
4	C_ERROR_INDEX_UNOPENED	The function of the voice playback via memory is not open to application
5	C_ERROR_INVALID_BUSCH	Invalid logical number of the CT-Bus channel
6	C_ERROR_OP_UNOPENED	The specified operation is not open to the application
7	C_ERROR_INVALID_FORMAT	Invalid voice CODEC format
8	C_ERROR_INVALID_PARAMETER	Invalid parameters
9	C_ERROR_FILEOP_FAILURE	File operation failed
10	C_ERROR_MEMORY_FAILURE	Memory access failed
11	C_ERROR_RESOURCE_USEUP	The related resource is exhausted
12	C_ERROR_SYSTEM	System error
13	C_ERROR_IdleChNotFound	No idle channel is available
14	C_ERROR_OP_FAILURE	Operation failed
15	C_ERROR_INVALID_APPSPYCIC	Invalid monitored CIC(Circuit Identification Code) number
16	C_ERROR_FAX_NOFILE	Fax file error
17	C_ERROR_VCH_INVALID_SCALE	Parameter value is out of range

Function Description:

Obtains the error code when the application is failed to call the API functions.

Note:

Related Information:

Driver version	SynCTI Ver.3.0 or above

Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmGetLastErrMsg

2.1.4.2 SsmGetLastErrMsg

Refer to SsmGetLastErrMsgA

2.1.4.3 SsmGetLastErrMsgA

Obtains the error message generated by the last failed API function call.

Format:

void SsmGetLastErrMsg(LPSTR szErrMsgBuf)

char* SsmGetLastErrMsgA(void)

Parameter Description:

szErrMsgBuf	The first address pointer pointing to the string which stores error messages. The storage buffer is allocated by the application and no less than 300 bytes

Return Value:

void	Nil
char*	The first address pointer pointing to the string which stores error messages in the driver

Function Description:

Obtains the error message generated by the last failed API function call.

Note:

- Needs to be called immediately upon the failure of the API function call.

Related Information:

Driver version	SynCTI Ver.2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmGetLastErrCode

2.1.4.4 fBmp_GetErrMsg

Obtains the error message generated by the last failed function call of Bmp.

Format:

void fBmp_GetErrMsg(char *buf)

Parameter Description:

buf	Pointer pointing to the buffer area storing the error information string

Return Value: None

Function Description:

When the function call of Bmp is failed, the driver will provide the detailed error message and the error code which can be obtained by invoking this function.

Note:

Related Information:

Driver Version	SynCTI Ver. 2.0 or above
Header	BmpApi.h
Library	BmpUtil.lib
DLL	BmpUtil.dll

Related Function:

2.1.5 Functions to Obtain Board Information

2.1.5.1 Obtaining Installed Board Information

2.1.5.1.1 SsmGetMaxCfgBoard

Obtains the total number of the voice boards which is set in the configuration file.

Format:

int SsmGetMaxCfgBoard()

Parameter Description: None

Return Value:

-1	Failed
≥0	Returns the total number of the boards

Function Description:

Obtains the total number of the voice boards which is set in the configuration file.

Note: None.

Related Information:

Driver version	SynCTI Ver. 3.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmGetMaxUsableBoard

2.1.5.1.2 SsmGetMaxUsableBoard

Obtains the total number of the boards which are successfully initialized by the driver.

Format:

int SsmGetMaxUsableBoard()

Parameter Description: None

Return Value:

-1	Failed
≥0	Maximum usable boards number

Function Description:

Obtains the total number of the boards which are successfully initialized by the driver.

Note: None.

Related Information:

Driver version	SynCTI Ver. 3.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmGetMaxCfgBoard

2.1.5.1.3 GetTotalPciBoard

Obtains the total number of the installed boards in the system.

Format:

int GetTotalPciBoard(int *pTotalBoards)

Parameter Description:

pTotalBoards	The integer pointer storing the total number of boards

Return Value:

-1	Call failed
0	Successful

Function Description:

Obtains the total number of the installed boards in the system. If this function is called successfully, the total number of the boards will be put into pTotalBoards.

Note:

This function call can be performed without the control of Shp_a3.dll. The total number of installed boards can be obtained directly. Because this function can be called before Shp_a3.dll is started, the configuration file of ShConfig.ini may be modified by some application program to ensure the successful call of the function SsmStartCti.

If you install or uninstall a board after the load of Shinitpci.dll, you should invoke ReloadPciBoardInfo before invoking this function to get the latest information.

Related Information:

Driver Version	Independent of SynCTI driver
Header	Shinitpci.h
Library	Shinitpci.lib
DLL	Shinitpci.dll

Related Functions: GetPciBoardSerialNo, GetPciBoardModel, ReloadPciBoardInfo

2.1.5.1.4 ReloadPciBoardInfo

Rereads the board information.

Format:

int ReloadPciBoardInfo()

Parameter Description: None

Return Value:

-1	Call failed

0	Successful

Function Description:

Rereads the board information. Usually it is invoked only after a board is installed or uninstalled
The information about GetPciBoardSerialNo, GetPciBoardModel and GetTotalPciBoard has been initialized in loading Shinitpci.dll.
If you install or uninstall a board after the load of Shinitpci.dll, you should invoke this function before invoking others in Shinitpci.dll.

Note:

This function call can be performed without the control of Shp_a3.dll.

Related Information:

Driver Version	Independent of SynCTI driver
Header	Shinitpci.h
Library	Shinitpci.lib
DLL	Shinitpci.dll

Related Functions: GetTotalPciBoard, GetPciBoardSerialNo, GetPciBoardModel

2.1.5.2 Obtaining Board Authorization Code

2.1.5.2.1 SsmGetAccreditId

Refer to SsmGetAccreditIdEx

2.1.5.2.2 SsmGetAccreditIdEx

Obtains the authorization code saved in the board firmware. SsmGetAccreditId() returns both the board model and the authorization code; SsmGetAccreditIdEx() only returns the authorization code.

Format:

int SsmGetAccreditId(int nBId)

int SsmGetAccreditIdEx(int nBId)

Parameter Description:

nBId	The board ID number specified in the configuration file

Return Value:

-1	Call failed, the failure reason can be acquired from the function SsmGetLastErrMsg	
0	There is no authorization code on the board	
>0	SsmGetAccreditId	The low 18 bits of the return value are valid, namely Bit_{17}…Bit_0: Bit_{17}~Bit_{10}：Board model Bit_9~Bit_0 ： Authorization code
	SsmGetAccreditIdEx	Returns the authorization code

Function Description:

Obtains the authorization code saved in the board firmware.

Note:

Related Information:

Driver version	SynCTI Ver.3.0 or above

Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmGetBoardModel

2.1.5.3 Obtaining Board Model

2.1.5.3.1 SsmGetBoardModel

Obtains the board model.

Format:

int SsmGetBoardModel(int nBId)

Parameter Description:

nBId	The board ID specified in the configuration file

Return Value:

If the return value is -1,the call failed (the failure reason can be obtained by function SsmGetLastErrMsg); If the return value is not -1, below are the corresponding models:

Return Value	Board Model
0x07	SHD-60A-CT/PCI/SS1
0x09	SHT-16A-CT/PCI
0x0a	SHT-8A/PCI
0x0b	SHD-60A-CT/PCI/SS7
0x0c	SHD-60A-CT/PCI/ISDN
0x0d	SHD-120A-CT/PCI/SS1
0x0e	SHD-120A-CT/PCI/SS7
0x0f	SHD-120A-CT/PCI/ISDN
0x14	SHT-16B-CT/cPCI
0x15	SHD-60A-CT/PCI/FJ
0x16	SHD-30A-CT/cPCI
0x17	SHD-60A-CT/cPCI
0x18	SHD-120A-CT/cPCI
0x1d	SHD-30A-CT/PCI/SS1
0x1e	SHD-30A-CT/PCI/SS7
0x1f	SHD-30A-CT/PCI/ISDN
0x20	SHD-30A-CT/PCI/FJ
0x24	SHT-16B-CT/PCI
0x25	SHT-16B-CT/PCI/FAX or SHT-16B-CT/PCI/MP3 [1]
0x26	SHT-8B/PCI
0x27	SHT-8B/PCI/FAX
0x28	SHD-30B-CT/PCI/FAX
0x29	SHD-60B-CT/PCI/FAX
0x2c	SHD-240A-CT/cPCI
0x2d	SHD-480A-CT/cPCI
0x2f	SHD-240S-CT/cPCI
0x30	SHD-480S-CT/cPCI
0x31	SHT-16B-CT/cPCI/MP3
0x32	SHT-16B-CT/cPCI/FAX or SHT-16B-CT/cPCI/MP3 [1]
0x36	SHD-60B-CT/cPCI/FAX
0x3b	SHR-16DA-CT/PCI
0x43	DST-1600

0x46	SHT-2A/USB
0x47	SHT-4A/USB
0x4b	SHD-30C-CT/PCI
0x4c	SHD-30C-CT/PCI/FAX
0x4d	SHD-60C-CT/PCI
0x4e	SHD-60C-CT/PCI/FAX
0x55	SHV-120A-CT/PCI
0x56	SHV-240A-CT/PCI
0x57	SHD-240D-CT/PCI
0x58	SHD-240D-CT/PCI/EC
0x59	SHD-120D-CT/PCI
0x5a	SHD-120D-CT/PCI/EC
0X5b	SHT-8C/PCI/FAX
0x5c	SHT-16C-CT/PCI/FAX
0x5d	SHN-32A-CT/PCI
0x5f	SHT-2B/USB
0x60	SHT-4B/USB
0x65	SHV-240A-CT/cPCI
0x66	SHD-30B-CT/PCI/FJ
0x67	SHD-60B-CT/PCI/FJ
0x68	ATP-24A/PCI
0x69	DST-24B/PCI
0x69	DST-24B/PCI(SSW)
0x6a	SHT-16C-CT/PCI/EC
0x6b	SHT-8C/PCI/EC
0x6c	ATP-24A/PCI+
0x6d	SHN-120B-CT/PCI+
0x6e	DST-24B/PCI+
0x6e	DST-24B/PCI+(SSW)
0x6f	DTP-30C/PCIe
0x70	DTP-30C/PCIe+
0x71	DTP-60C/PCIe
0x72	DTP-60C/PCIe+
0x73	DTP-120C/PCIe
0x74	DTP-120C/PCIe+
0x75	SHD-30E-CT/PCIe
0x76	SHD-30E-CT/PCIe/EC
0x77	SHD-30E-CT/PCIe/FAX
0x78	SHD-60E-CT/PCIe
0x79	SHD-60E-CT/PCIe/EC
0x7a	SHD-60E-CT/PCIe/FAX
0x7b	SHD-120E-CT/PCIe
0x7c	SHD-120E-CT/PCIe/EC
0x7d	SHD-120E-CT/PCIe/FAX
0x7e	SHD-240E-CT/PCIe
0x7f	SHD-240E-CT/PCIe/EC
0x80	SHD-240E-CT/PCIe/FAX
0x83	SHN-60B-CT/PCI+
0x84	ATP-24A/PCIe
0x85	ATP-24A/PCIe+
0x86	SHD-120D-CT/PCI/CAS
0x87	SHD-240D-CT/PCI/CAS
0x88	SHN-32B-CT/PCI+
0x89	SHN-16B-CT/PCI+
0x8a	SHN-8B-CT/PCI+

0x8b	DST-24B/PCIe
0x8c	DST-24B/PCIe+
0x8e	SHD-120E-CT/PCI(SSW)
0x8f	SHD-120E-CT/PCI/EC(SSW)
0x90	SHD-120E-CT/PCI/FAX(SSW)
0x91	SHD-240E-CT/PCI(SSW)
0x92	SHD-240E-CT/PCI/EC(SSW)
0x93	SHD-240E-CT/PCI/FAX(SSW)
0x94	SHD-30E-CT/PCI(SSW)
0x95	SHD-30E-CT/PCI/EC(SSW)
0x96	SHD-30E-CT/PCI/FAX(SSW)
0x97	SHD-60E-CT/PCI(SSW)
0x98	SHD-60E-CT/PCI/EC(SSW)
0x99	SHD-60E-CT/PCI/FAX(SSW)
0x9b	SHF-2D/PCI
0x9c	SHF-4D/PCI
0xB3	SHF-4D/PCIe
0x9d	DTP-30C/PCI
0x9e	DTP-30C/PCI+
0x9f	DTP-60C/PCI
0xa0	DTP-60C/PCI+
0xa1	DTP-120C/PCI
0xa2	DTP-120C/PCI+
0xa4	SHT-16D-CT/PCIe
0xa6	PCM1280E
0xa7	SHD-240E-CT/PCIe/VAR
0xa9	SHN-60B-CT/PCIe+
0xaa	SHN-120B-CT/PCIe+
0xad	SHN-120B-CT/PCIe/VAR
0xae	SHN-480C-CT/PCIe
0xb5	SHD-30D-CT/PCI
0xb6	SHD-60D-CT/PCI
0xfd	SynIPRecorder
0xfe	SynIPAnalyzer

Note[1]: The SHT-16B-CT/PCI/FAX board and the SHT-16B-CT/PCI/MP3 board differ in the hardware structure. However, they have the same board ID written in the firmware due to historical reasons.The configuration item DSP3WORKMODE is used to distinguish these two board models.

Function Description:

Obtains the board model.

Note:

Related Information:

Driver version	SynCTI Ver.3.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmGetAccreditIdEx, SsmGetAccreditId

2.1.5.3.2 GetPciBoardModel

Obtains the board model.

Format:

int GetPciBoardModel(int nBId, LPSTR lpBoardModel)

Parameter Description:

nBId	The ID number of the actually installed board
lpBoardModel	The model of the board with designated number, it's denoted in the below list of strings

Board Model & Corresponding String

Return Value	Meaning
"SHD-60A-CT/PCI/SS1"	60-channel PCI SS1 board
"SHD-60A-CT/PCI/SS7"	60-channel PCI SS7 board
"SHD-60A-CT/PCI/ISDN"	60-channel PCI ISDN board
"SHD-120A-CT/PCI/SS1"	120-channel PCI SS1 board
"SHD-120A-CT/PCI/SS7"	120-channel PCI SS7 board
"SHD-120A-CT/PCI/ISDN"	120-channel PCI ISDN board
"SHD-60B-CT/PCI/FAX"	60-channel PCI B-type board, configurable to be a signaling board with fax channels
"SHD-30B-CT/PCI/FAX"	30-channel PCI B-type board, configurable to be a signaling board with fax channels
"SHD-30A-CT/PCI/SS1"	30-channel PCI SS1 board
"SHD-30A-CT/PCI/SS7"	30-channel PCI SS7 board
"SHD-30A-CT/PCI/ISDN"	30-channel PCI ISDN board
"SHD-30A-CT/PCI/FJ"	30-channel PCI A-type board, supporting high-impedance parallel connection
"SHD-60A-CT/PCI/FJ"	60-channel PCI A-type board, supporting high-impedance parallel connection
"SHD-30B-CT/PCI/FJ"	30-channel PCI B-type board, supporting high-impedance parallel connection
"SHD-60B-CT/PCI/FJ"	60-channel PCI B-type board, supporting high-impedance parallel connection
"SHT-16B-CT/PCI/FAX or SHT-16B-CT/PCI/MP3 [1]"	16-channel PCI B-type call-recording board, supporting soft faxing or MP3 codec.
"SHT-8B/PCI"	8-channel PCI B-type board
"SHT-8B/PCI/FAX"	8-channel PCI B-type board with soft-faxing component
"SHT-8C/PCI/FAX"	8-channel PCI C-type board with soft-faxing component
"SHT-8C/PCI/EC"	8-channel PCI C-type board
"SHT-16C-CT/PCI/EC"	16-channel PCI C-type board
"SHT-16B-CT/PCI"	16-channel PCI B-type board
"SHT-16C-CT/PCI/FAX"	16-channel PCI C-type board with soft-faxing component
"SHT-16A-CT/PCI/FAX"	16-channel PCI A-type board with soft-faxing component
"DST-1600"	16-channel PCI B-type board
"SHT-16B-CT/cPCI"	16-channel cPCI B-type board
"SHD-30A-CT/cPCI"	30-channel cPCI board, supporting SS1, ISDN and SS7
"SHD-60A-CT/cPCI"	60-channel cPCI board, supporting SS1, ISDN and SS7
"SHT-16B-CT/cPCI/MP3 or SHT-16B-CT/cPCI/FAX"	16-channel cPCI B-type call-recording board, supporting MP3 Codec or soft faxing
"SHD-480A-CT/cPCI"	16E1 cPCI board, supporting teleconferencing
"SHD-480S-CT/cPCI"	16E1 cPCI board, not supporting teleconferencing
"SHD-60B-CT/cPCI/FAX"	60-channel cPCI B-type board, configurable to be a signaling board with fax channels
"SHR-16DA-CT/PCI"	16-channel PCI digital station tap board
"SHR-24DA-CT/PCI"	24-channel PCI digital station tap board
"SHT-120A-CT/PCI"	120-channel PCI high-capacity analog board
"SHT-2B/USB"	2-channel B-type analog board, including USB interface
"SHT-4B/USB"	4-channel B-type analog board, including USB interface
"SHT-2A/USB"	2-channel A-type analog board, including USB interface
"SHT-4A/USB"	4 channel A-type analog board, including USB interface
"SHT-120A-CT/cPCI"	120-channel cPCI high-capacity analog board
"SHD-30D-CT/PCI"	1E1/T1 PCI digital trunk board
"SHD-60D-CT/PCI"	2E1/T1 PCI digital trunk board

“SHD-240D-CT/PCI”	8E1/T1 PCI digital trunk board
“SHD-240D-CT/PCI/EC”	8E1/T1 PCI digital trunk board, having enhanced capability in echo cancellation
“SHD-120D-CT/PCI”	4E1/T1 PCI digital trunk board
“SHD-120D-CT/PCI/EC”	4E1/T1 PCI digital trunk board, having enhanced capability in echo cancellation
“SHD-120D-CT/PCI/CAS”	4E1 PCI digital trunk board, support of SS1
“SHD-240D-CT/PCI/CAS”	8E1 PCI digital trunk board, support of SS1
“SHD-30C-CT/PCI”	30-channel PCI C-type board
“SHD-30C-CT/PCI/FAX”	30-channel PCI C-type board with soft-faxing component
“SHD-60C-CT/PCI”	60-channel PCI C-type board
“SHD-60C-CT/PCI/FAX”	60-channel PCI C-type board with soft-faxing component
“SHF-2D/PCI”	2-channel PCI D-type analog fax board
“SHF-4D/PCI”	4-channel PCI D-type analog fax board
“SHF-4D/PCIe”	4-channel PCIe D-type analog fax board
“DTP-30C/PCI”	30-channel PCI board, supporting high-impedance parallel connection
“DTP-30C/PCI+”	30-channel PCI board, supporting high-impedance parallel connection
“DTP-60C/PCI”	60-channel PCI board, supporting high-impedance parallel connection
“DTP-60C/PCI+”	60-channel PCI board, supporting high-impedance parallel connection
“DTP-120C/PCI”	120-channel PCI board, supporting high-impedance parallel connection
“DTP-120C/PCI+”	120-channel PCI board, supporting high-impedance parallel connection
“DTP-30C/PCIe”	30-channel PCIe board, supporting high-impedance parallel connection
“DTP-30C/PCIe+”	30-channel PCIe board, supporting high-impedance parallel connection
“DTP-60C/PCIe”	60-channel PCIe board, supporting high-impedance parallel connection
“DTP-60C/PCIe+”	60-channel PCIe board, supporting high-impedance parallel connection
“DTP-120C/PCIe”	120-channel PCIe board, supporting high-impedance parallel connection
“DTP-120C/PCIe+”	120-channel PCIe board, supporting high-impedance parallel connection
“SHN-32A-CT/PCI”	32-channel VoIP A-type board
“SHN-8B-CT/PCI+”	8-channel VoIP B-type board
“SHN-16B-CT/PCI+”	16-channel VoIP B-type board
“SHN-32B-CT/PCI+”	32-channel VoIP B-type board
“SHN-60B-CT/PCI+”	60-channel VoIP B-type board
“SHN-120B-CT/PCI+”	120-channel VoIP B-type board
“SHN-60B-CT/PCIe+”	60-channel PCIe VoIP B-type board
“SHN-120B-CT/PCIe+”	120-channel PCIe VoIP B-type board
“SHN-120B-CT/PCIe/VAR”	120-channel PCIe VoIP B-type board with 60-channel voice alteration resources
“SHN-480C-CT/PCIe”	480-channel PCIe VoIP C-type board
“ATP-24A/PCI”	ATP-24A/PCI analog tap passive board
“ATP-24A/PCI+”	ATP-24A/PCI+ analog tap passive board
“ATP-24A/PCIe”	ATP-24A/PCIe analog tap passive board
“ATP-24A/PCIe+”	ATP-24A/PCIe+ analog tap passive board
“DST-24B/PCI”	24-channel PCI B-type digital station tap board
“DST-24B/PCI(SSW)”	24-channel PCI B-type digital station tap board
“DST-24B/PCI+”	24-channel PCI B-type digital station tap board with enhanced capability
“DST-24B/PCI+(SSW)”	24-channel PCI B-type digital station tap board with enhanced capability
“DST-24B/PCIe”	24-channel PCIe B-type digital station tap board
“DST-24B/PCIe+”	24-channel PCIe B-type digital station tap board with enhanced capability
“SHD-30E-CT/PCI(SSW)”	1 E1/T1/J1 PCI digital trunk board
“SHD-30E-CT/PCI/FAX(SSW)”	1 E1/T1/J1 PCI digital trunk board with soft-faxing component
“SHD-30E-CT/PCI/EC(SSW)”	1 E1/T1/J1 PCI digital trunk board, having enhanced capability in echo cancellation
“SHD-60E-CT/PCI(SSW)”	2 E1/T1/J1 PCI digital trunk board
“SHD-60E-CT/PCI/FAX(SSW)”	2 E1/T1/J1 PCI digital trunk board with soft-faxing component
“SHD-60E-CT/PCI/EC(SSW)”	2 E1/T1/J1 PCI digital trunk board, having enhanced capability in echo

	cancellation
“SHD-120E-CT/PCI(SSW)”	4 E1/T1/J1 PCI digital trunk board
“SHD-120E-CT/PCI/FAX(SSW)”	4 E1/T1/J1 PCI digital trunk board with soft-faxing component
“SHD-120E-CT/PCI/EC(SSW)”	4 E1/T1/J1 PCI digital trunk board, having enhanced capability in echo cancellation
“SHD-240E-CT/PCI(SSW)”	8 E1/T1/J1 PCI digital trunk board
“SHD-240E-CT/PCI/FAX(SSW)”	8 E1/T1/J1 PCI digital trunk board with soft-faxing component
“SHD-240E-CT/PCI/EC(SSW)”	8 E1/T1/J1 PCI digital trunk board, having enhanced capability in echo cancellation
“SHD-30E-CT/PCIe”	1 E1/T1/J1 PCIe digital trunk board
“SHD-30E-CT/PCIe/FAX”	1 E1/T1/J1 PCIe digital trunk board with soft-faxing component
“SHD-30E-CT/PCIe/EC”	1 E1/T1/J1 PCIe digital trunk board, having enhanced capability in echo cancellation
“SHD-60E-CT/PCIe”	2 E1/T1/J1 PCIe digital trunk board
“SHD-60E-CT/PCIe/FAX”	2 E1/T1/J1 PCIe digital trunk board with soft-faxing component
“SHD-60E-CT/PCIe/EC”	2 E1/T1/J1 PCIe digital trunk board, having enhanced capability in echo cancellation
“SHD-120E-CT/PCIe”	4 E1/T1/J1 PCIe digital trunk board
“SHD-120E-CT/PCIe/FAX”	4 E1/T1/J1 PCIe digital trunk board with soft-faxing component
“SHD-120E-CT/PCIe/EC”	4 E1/T1/J1 PCIe digital trunk board, having enhanced capability in echo cancellation
“SHD-240E-CT/PCIe”	8 E1/T1/J1 PCIe digital trunk board
“SHD-240E-CT/PCIe/FAX”	8 E1/T1/J1 PCIe digital trunk board with soft-faxing component
“SHD-240E-CT/PCIe/EC”	8 E1/T1/J1 PCIe digital trunk board, having enhanced capability in echo cancellation
“SHD-240E-CT/PCIe/VAR”	8 E1/T1/J1 PCIe digital trunk board with voice alteration resources
“SHT-16D-CT/PCIe”	16-channel PCIe D-type analog voice board
“PCM1280E”	128-channel PCIe PCM32 recording board

Return Value:

0	Successful
-1	Failed

Function Description:

This function obtains the model of the board with designated ID number. If this function is successfully called, the model will be put in lpBoardModel.

Note:

This function call can be performed without the control of Shp_a3.dll. The model of installed boards in the operating system can be obtained directly.

If you install or uninstall a board after the load of Shinitpci.dll, you should invoke ReloadPciBoardInfo before invoking this function to get the latest information.

Related Information:

Driver Version	Independent of SynCTI driver
Header	Shinitpci.h
Library	Shinitpci.lib
DLL	Shinitpci.dll

Related Function: GetTotalPciBoard, GetPciBoardSerialNo, ReloadPciBoardInfo

2.1.5.3.3 SsmGetBoardName

Obtains the board model.

Format:

int SsmGetBoardName (int nBId, LPSTR lpBoardModel)

Parameter Description:

nBId	The board ID specified in the configuration file
lpBoardModel	The model of the board with designated number, it's denoted in the below list of strings

Board Model & Corresponding String

Return Value	Meaning
"SHD-60A-CT/PCI/SS1"	60-channel PCI SS1 board
"SHD-60A-CT/PCI/SS7"	60-channel PCI SS7 board
"SHD-60A-CT/PCI/ISDN"	60-channel PCI ISDN board
"SHD-120A-CT/PCI/SS1"	120-channel PCI SS1 board
"SHD-120A-CT/PCI/SS7"	120-channel PCI SS7 board
"SHD-120A-CT/PCI/ISDN"	120-channel PCI ISDN board
"SHD-60B-CT/PCI/FAX"	60-channel PCI B-type board, configurable to be a signaling board with fax channels
"SHD-30B-CT/PCI/FAX"	30-channel PCI B-type board, configurable to be a signaling board with fax channels
"SHD-30A-CT/PCI/SS1"	30-channel PCI SS1 board
"SHD-30A-CT/PCI/SS7"	30-channel PCI SS7 board
"SHD-30A-CT/PCI/ISDN"	30-channel PCI ISDN board
"SHD-30A-CT/PCI/FJ"	30-channel PCI A-type board, supporting high-impedance parallel connection
"SHD-60A-CT/PCI/FJ"	60-channel PCI A-type board, supporting high-impedance parallel connection
"SHD-30B-CT/PCI/FJ"	30-channel PCI B-type board, supporting high-impedance parallel connection
"SHD-60B-CT/PCI/FJ"	60-channel PCI B-type board, supporting high-impedance parallel connection
"SHT-16B-CT/PCI/FAX or SHT-16B-CT/PCI/MP3 [1]"	16-channel PCI B-type call-recording board, supporting soft faxing or MP3 codec
"SHT-8B/PCI"	8-channel PCI B-type board
"SHT-8B/PCI/FAX"	8-channel PCI B-type board with soft-faxing component
"SHT-8C/PCI/FAX"	8-channel PCI C-type board with soft-faxing component
"SHT-8C/PCI/EC"	8-channel PCI C-type board
"SHT-16C-CT/PCI/EC"	16-chennel PCI C-type board
"SHT-16B-CT/PCI"	16-channel PCI B-type board
"SHT-16C-CT/PCI/FAX"	16-channel PCI C-type board with soft-faxing component
"SHT-16A-CT/PCI/FAX"	16-channel PCI A-type board with soft-faxing component
"DST-1600"	16-channel PCI B-type board
"SHT-16B-CT/cPCI"	16-channel cPCI B-type board
"SHD-30A-CT/cPCI"	30-channel cPCI board, supporting SS1, ISDN and SS7
"SHD-60A-CT/cPCI"	60-channel cPCI board, supporting SS1, ISDN and SS7
"SHT-16B-CT/cPCI/MP3 or SHT-16B-CT/cPCI/FAX"	16-channel cPCI B-type call-recording board, supporting MP3 codec or soft faxing
"SHD-480A-CT/cPCI"	16E1 cPCI board, supporting teleconferencing
"SHD-480S-CT/cPCI"	16E1 cPCI board, not supporting teleconferencing
"SHD-60B-CT/cPCI/FAX"	60-channel cPCI B-type board, configurable to be a signaling board with fax channels
"SHR-16DA-CT/PCI"	16-channel PCI digital station tap board
"SHR-24DA-CT/PCI"	24-channel PCI digital station tap board
"SHT-120A-CT/PCI"	120-channel PCI high-capacity analog board
"SHT-2B/USB"	2-channel B-type analog board, including USB interface
"SHT-4B/USB"	4-channel B-type analog board, including USB interface

“SHT-2A/USB”	2-channel A-type analog board, including USB interface
“SHT-4A/USB”	4-channel A-type analog board, including USB interface
“SHT-120A-CT/cPCI”	120-channel cPCI high-capacity analog board
“SHD-30D-CT/PCI”	1E1/T1 PCI digital trunk board
“SHD-60D-CT/PCI”	2E1/T1 PCI digital trunk board
“SHD-240D-CT/PCI”	8E1/T1 PCI digital trunk board
“SHD-240D-CT/PCI/EC”	8E1/T1 PCI digital trunk board, having enhanced capability in echo cancellation
“SHD-120D-CT/PCI”	4E1/T1 PCI digital trunk board
“SHD-120D-CT/PCI/EC”	4E1/T1 PCI digital trunk board, having enhanced capability in echo cancellation
“SHD-120D-CT/PCI/CAS”	4E1 PCI digital trunk board, support of SS1
“SHD-240D-CT/PCI/CAS”	8E1 PCI digital trunk board, support of SS1
“SHD-30C-CT/PCI”	30-channel PCI C-type board
“SHD-30C-CT/PCI/FAX”	30-channel PCI C-type board with soft-faxing component
“SHD-60C-CT/PCI”	60-channel PCI C-type board
“SHD-60C-CT/PCI/FAX”	60-channel PCI C-type board with soft-faxing component
“SHF-2D/PCI”	2-channel PCI D-type analog fax board
“SHF-4D/PCI”	4-channel PCI D-type analog fax board
“SHF-4D/PCIe”	4-channel PCIe D-type analog fax board
“DTP-30C/PCI”	30-channel PCI board, supporting high-impedance parallel connection
“DTP-30C/PCI+”	30-channel PCI board, supporting high-impedance parallel connection
“DTP-60C/PCI”	60-channel PCI board, supporting high-impedance parallel connection
“DTP-60C/PCI+”	60-channel PCI board, supporting high-impedance parallel connection
“DTP-120C/PCI”	120-channel PCI board, supporting high-impedance parallel connection
“DTP-120C/PCI+”	120-channel PCI board, supporting high-impedance parallel connection
“DTP-30C/PCIe”	30-channel PCIe board, supporting high-impedance parallel connection
“DTP-30C/PCIe+”	30-channel PCIe board, supporting high-impedance parallel connection
“DTP-60C/PCIe”	60-channel PCIe board, supporting high-impedance parallel connection
“DTP-60C/PCIe+”	60-channel PCIe board, supporting high-impedance parallel connection
“DTP-120C/PCIe”	120-channel PCIe board, supporting high-impedance parallel connection
“DTP-120C/PCIe+”	120-channel PCIe board, supporting high-impedance parallel connection
“SHN-32A-CT/PCI”	32-channel VoIP A-type board
“SHN-8B-CT/PCI+”	8-channel VoIP B-type board
“SHN-16B-CT/PCI+”	16-channel VoIP B-type board
“SHN-32B-CT/PCI+”	32-channel VoIP B-type board
“SHN-60B-CT/PCI+”	60-channel VoIP B-type board
“SHN-120B-CT/PCI+”	120-channel VoIP B-type board
“SHN-60B-CT/PCIe+”	60-channel PCIe VoIP B-type board
“SHN-120B-CT/PCIe+”	120-channel PCIe VoIP B-type board
“SHN-120B-CT/PCIe/VAR”	120-channel PCIe VoIP B-type board with 60-channel voice alteration resources
“SHN-480C-CT/PCIe”	480-channel PCIe VoIP C-type board
“ATP-24A/PCI”	ATP-24A/PCI analog tap passive board
“ATP-24A/PCI+”	ATP-24A/PCI+ analog tap passive board
“ATP-24A/PCIe”	ATP-24A/PCIe analog tap passive board
“ATP-24A/PCIe+”	ATP-24A/PCIe+ analog tap passive board
“DST-24B/PCI”	24-channel PCI B-type digital station tap board
“DST-24B/PCI+”	24-channel PCI B-type digital station tap board with enhanced capability
“DST-24B/PCIe”	24-channel PCIe B-type digital station tap board
“DST-24B/PCIe+”	24-channel PCIe B-type digital station tap board with enhanced capability
“SHD-30E-CT/PCI(SSW)”	1E1/T1/J1 PCI digital trunk board
“SHD-30E-CT/PCI/FAX(SSW)”	1E1/T1/J1 PCI digital trunk board with soft-faxing component
“SHD-30E-CT/PCI/EC(SSW)”	1E1/T1/J1 PCI digital trunk board, having enhanced capability in echo cancellation

“SHD-60E-CT/PCI(SSW)”	2E1/T1/J1 PCI digital trunk board
“SHD-60E-CT/PCI/FAX(SSW)”	2E1/T1/J1 PCI digital trunk board with soft-faxing component
“SHD-60E-CT/PCI/EC(SSW)”	2E1/T1/J1 PCI digital trunk board, having enhanced capability in echo cancellation
“SHD-120E-CT/PCI(SSW)”	4E1/T1/J1 PCI digital trunk board
“SHD-120E-CT/PCI/FAX(SSW)”	4E1/T1/J1 PCI digital trunk board with soft-faxing component
“SHD-120E-CT/PCI/EC(SSW)”	4E1/T1/J1 PCI digital trunk board, having enhanced capability in echo cancellation
“SHD-240E-CT/PCI(SSW)”	8E1/T1/J1 PCI digital trunk board
“SHD-240E-CT/PCI/FAX(SSW)”	8E1/T1/J1 PCI digital trunk board with soft-faxing component
“SHD-240E-CT/PCI/EC(SSW) ”	8E1/T1/J1 PCI digital trunk board, having enhanced capability in echo cancellation
“SHD-30E-CT/PCIe”	1E1/T1/J1 PCIe digital trunk board
“SHD-30E-CT/PCIe/FAX”	1E1/T1/J1 PCIe digital trunk board with soft-faxing component
“SHD-30E-CT/PCIe/EC”	1E1/T1/J1 PCIe digital trunk board, having enhanced capability in echo cancellation
“SHD-60E-CT/PCIe”	2E1/T1/J1 PCIe digital trunk board
“SHD-60E-CT/PCIe/FAX”	2E1/T1/J1 PCIe digital trunk board with soft-faxing component
“SHD-60E-CT/PCIe/EC”	2E1/T1/J1 PCIe digital trunk board, having enhanced capability in echo cancellation
“SHD-120E-CT/PCIe”	4E1/T1/J1 PCIe digital trunk board
“SHD-120E-CT/PCIe/FAX”	4E1/T1/J1 PCIe digital trunk board with soft-faxing component
“SHD-120E-CT/PCIe/EC”	4E1/T1/J1 PCIe digital trunk board, having enhanced capability in echo cancellation
“SHD-240E-CT/PCIe”	8E1/T1/J1 PCIe digital trunk board
“SHD-240E-CT/PCIe/FAX”	8E1/T1/J1 PCIe digital trunk board with soft-faxing component
“SHD-240E-CT/PCIe/EC”	8E1/T1/J1 PCIe digital trunk board, having enhanced capability in echo cancellation
“SHD-240E-CT/PCIe/VAR”	8 E1/T1/J1 PCIe digital trunk board with voice alteration resources
“SHT-16D-CT/PCIe”	16-channel PCIe D-type analog voice board
“PCM1280E”	128-channel PCIe PCM32 recording board
“SynIPAnalyzer”	SynIPAnalyzer
“SynIPRecorder”	SynIPRecorder

Return Value:

0	Successful
-1	Failed

Function Description:

This function obtains the model of the board with designated ID number. If this function is successfully called, the model will be put in lpBoardModel.

Note:

This function can achieve the same purpose as GetPciBoardModel. However, this function call is performed under the control of Shp_a3.dll.

Related Information:

Driver Version	SynCTI Ver.5.2.0.3
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmGetAccreditIdEx, SsmGetAccreditId, SsmGetBoardModel

2.1.5.4 Obtaining Board Serial Number

2.1.5.4.1 SsmGetPciSerialNo

Obtains the board serial number.

Format:

DWORD SsmGetPciSerialNo(int nBId)

Parameter Description:

nBId	Board ID, corresponding to the value of x in the section [BoardId=x] of the configuration file

Return Value:

0	Call failed, the failure reason can be acquired from the function SsmGetLastErrMsg
>0	Board serial number

Function Description:

Obtains the board serial number. The leaving factory boards have a label with serial number sticked on the back side. For more information, refer to the hardware manual of voice boards.

Note:

Related Information:

Driver version	SynCTI Ver.3.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmGetAccreditIdEx, SsmGetAccreditId

2.1.5.4.2 GetPciBoardSerialNo

Obtains the board serial number.

Format:

DWOR GetPciBoardSerialNo (int nBId, DWORD pSerialNo)

Parameter Description:

nBId	The ID number of the actually installed board
pSerialNo	The pointer storing the serial number of the board

Return Value:

-1	Call failed
0	Successful

Function Description:

Obtains the serial number of the designated board. If it's successfully called, the board serial number will be put into pSerialNo.

Note:

This function call can be performed without the control of Shp_a3.dll. The serial number of the installed board in the operating system can be obtained directly.

If you install or uninstall a board after the load of Shinitpci.dll, you should invoke ReloadPciBoardInfo before invoking this function to get the latest information.

Related Information:

Driver Version	Independent of SynCTI driver
Header	Shinitpci.h
Library	Shinitpci.lib
DLL	Shinitpci.dll

Related Functions: GetPciBoardModel , GetTotalPciBoard, ReloadPciBoardInfo

2.1.5.5 Obtaining Version Information

2.1.5.5.1 SsmGetDllVersion

Obtains the version information of the file 'shp_a3.dll'.

Format:

int SsmGetDllVersion(PSSM_VERSION pDLLVersion)

Parameter Description:

pDLLVersion	Struct pointer storing the file version information

Return Value:

-1	Call failed, the failure reason can be obtained from the function call of SsmGetLastErrMsg
0	Successful

Function Description:

Obtains the version information of the file 'shp_a3.dll'.

The structure of the returned version information is as follows:

```
typedef struct _VERSION
{   UCHAR ucMajor;// Major version
    UCHAR ucMinor;// Minor version
    USHORT usInternal;// Internal version
    USHORT usBuild;// Build number
    UCHAR ucRelease;// Compilation mode
    UCHAR ucFeature;// Compilation environment
}SSM_VERSION, *PSSM_VERSION;
```

For example, invoke this function based on SynCTI 5.0.3.0 and the following information appears:

```
typedef struct _VERSION
{    UCHAR ucMajor;//=5
     UCHAR ucMinor;//=0
     USHORT usInternal;//=3
     USHORT usBuild;   //=0
     UCHAR ucRelease; //= blank (Reserved)
     UCHAR ucFeature; //= blank (Reserved)
}SSM_VERSION, *PSSM_VERSION;
```

Note:

Related Information:

Driver Version	SynCTI Ver.3.1 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Functions:

2.1.6 Funtions to Obtain Channel Attributes

2.1.6.1 SsmGetMaxCh

Obtains the total channel number in the application program.

Format:

int SsmGetMaxCh(void)

Parameter Description: None

Return Value:

-1	Failed
≥0	Total channel number

Function Description:

Obtains the total channel number in the application program

Note:

Related Information:

Driver version	SynCTI Ver.3.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmGetChType

2.1.6.2 SsmGetChType

Obtains the channel type.

Format:

int SsmGetChType(int ch)

Parameter Description:

ch	The logical number of the channel

Return Value:

-1	Failed
0	Analog trunk channel
2	Station channel
3	Analog trunk recording channel
4	SS1 channel
6	TUP channel
7	ISDN channel (user side)
8	ISDN channel (network side)
9	Fax channel
10	Magnetic channel
11	ISUP channel (China SS7 signaling ISUP)

12	Digital trunk recording channel
13	Channel Bank EM channel
14	Voice-alteration channel
16	SIP channel
17	IP channel
19	DASS2 channel
20	SHT board channel without any module
21	EM Control channel
22	EM Voice channel
25	IPR channel, i.e. SynIPR recording channel
26	IPA channel, i.e. SynIPR channel for data package analysis
27	DPNSS channel

Function Description:

Obtains the channel type.

Note:

- To obtain more detailed information about the channel type, use this function with SsmGetAutoCallDirection.

Related Information:

Driver version	SynCTI Ver. 5.3.3.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmGetMaxCh

2.1.6.3 SsmGetChHdInfo

Obtains the ID of the board where the channel is located ,also obtains the channel's physical number on the board.

Format:

int SsmGetChHdInfo(int ch, int * pnBId, int * pnBCh)

Parameter Description:

ch	Channel number
pnBId	Returns the ID of the board where the channel is located. Its storage space is allocated by the application program
pnBCh	Returns the channel's physical number on the board. Its storage space is allocated by the application program

Return Value:

-1	Failed. The failure reason can be acquired by the function SsmGetLastErrMsg .
0	Successful

Function Description:

Obtains the ID of the board where the channel is located ,also obtains the channel's physical number on the board.

Note:

Related Information:

Driver version	SynCTI Ver. 4.7.1.5 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmGetAppChId

2.1.6.4 SsmGetAppChId

Inquires the channel logical number based on the board ID and the channel physical number.

Format:

int SsmGetAppChId(int * ch, int nBId, int nBCh)

Parameter Description:

ch	Returns the channel logical number. The storage space is allocated by the application program
nBId	The ID of the board where the channel is located
nBCh	The channel physical number

Return Value:

-1	Failed. The failure reason can be acquired by the function SsmGetLastErrMsg .
0	Successful

Function Description:

Inquires the channel logical number based on the board ID and the channel physical number.

Note:

Related Information:

Driver version	SynCTI Ver. 4.7.1.5 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmGetChHdInfo

2.1.7 Functions to Set Channel Attributes

2.1.7.1 SsmSetFlag

Sets the control attributes of the channel.

Format:

int SsmSetFlag(int ch, int nType, long lPara)

Parameter Description:

<table>
<tr><td>ch</td><td colspan="4">Channel Number</td></tr>
<tr><td>nType
lPara</td><td colspan="4">nType is the parameter type, lvalue is the parameter value.
The value range and the physical meaning of nType and lvalue are listed in the table below:
<table>
<tr><th colspan="2">nType</th><th>Description</th><th>lPara</th></tr>
<tr><th>Value</th><th>Macro in shpa3api.h</th><th></th><th></th></tr>
<tr><td>1</td><td>F_RCVDTMFSENS</td><td>Sets the minimum DTMF signal durations respectively at on and off states. For more information, refer to configuration item ReceiveDtmfSensitive</td><td>Please refer to the configuration item ReceiveDtmfSensitive</td></tr>
<tr><td>2</td><td>F_TXDTMFAMP</td><td>Sets the magnitude of the DTMF signal generated by the DTMF generator, By default, the value for SHT, ATP series is 0X8976; the value for SHD, DTP series is 0X3f3f.</td><td>High 8 bits and low 8 bits respectively indicate the two frequency magnitudes of the DTMF signal.</td></tr>
<tr><td>5</td><td>F_RXR2FILTERTIME</td><td>Sets the minimum duration for MFC R2 signal. For more information, refer to the configuration item RxR2FilterTime.</td><td>Range of value: 16~96 (must be integer times of 16). Calculated by ms.</td></tr>
<tr><td>9</td><td>F_ClearInVoiceOnRcvDtmf</td><td>Sets the DTMF clamping function. For more information, refer to the configuration item:</td><td>=0: Disabled
=1: Enabled</td></tr>
</table></td></tr>
</table>

		ClearInVoiceOnRxDtmf .	
10	F_MixerResToBus	Sets whether to use the output signal of 'off-bus mixer" as the incoming signal source of 'onto-bus mixer'. This function only supports SHT Series (8/16 Channels) and SHD Series voice boards. For more information, refer to the switch of K1-2 in the corresponding voice board's principle diagram.	=0: No =1: Yes
11	F_HighAndLowFreqEnScale	Sets the proportion between the high-frequency energy and the low-frequency energy in the DTMF signal. For more information, refer to the configuration item HighAndLowFreqEnScale.	Refer to the description of the configuration item HighAndLowFreqEnScale .
12	F_DualAndAllFreqEnScale	Sets the threshold percentage of the in-band energy in the overall frequency energy in the DTMF signal. For more details, refer to the configuration item DualAndAllFreqEnScale .	Refer to the configuration item of DualAndAllFreqEnScale .
13	F_EchoCancelInFsk	Sets whether to disable the echo-canceller when the FSK receiver is working. For more details, refer to the configuration item FskEchoCancelDelay.	=0: Disabled =1: Enabled
17	F_ClearInVoiceOnRcv450Hz	Sets whether to put tones onto bus.	=0: Onto bus (default) =1: Not onto bus
18	FSKMinGate	Sets an energy threshold value for FSK receiving.	=0: No threshold value n(n>0): The set threshold value
19	F_RECTOFILEA_CALLBACKTIMEA	Sets the callback time for the function SsmRecToFileA or SsmRecToFileB. For more information, refer to the corresponding function description. **Note: This parameter goes invalid if the software-based MP3 is used as the recording format.**	Value Range (Calculated by ms): A-law, μ-law: 1365>n≥16 ADPCM: 2730>n≥64 Hardware-based MP3: 10922>n≥16 G729A: 10922>n≥20 VOX: 2730>n≥64 Hardware-based GSM: 6553>n≥40 Software-based GSM: 1365>n≥16 For other encoding formats, the driver will invoke the callback function twice respectively at the times when half and full buffer are played out.
20	F_CALLERIDTYPE	Sets the mode for the remote PBX to send the calling party number on the analog phone line. For more information, refer to the description of the configuration item CallerIdStyle.	=0: DTMF mode =1: FSK mode
21	F_InVoiceToBusA	Determines whether to put incoming signals to the TDM bus. That is, sets the switch K3 in the figure of board operation principle. For more information, refer to the description of the configuration item InVoiceToBus.	=0: Disabled =1: Enabled
22	F_EchoCancelInFskA	Sets whether to disable the echo canceller when the FSK transceiver is working. For more information, refer to the description of the configuration item FskEchoCancelDelay.	=0: Disabled =1: Enabled
23	F_ChToRingingOnRingCntA	Sets the parameters for the ringing current detector. For more information, refer to the Ringing Current on Analog Phone Line section in Chapter	Refer to the configuration item ChToRingingOnRingCnt.

			1 or the description of the configuration item ChToRingingOnRingCnt.	
	24	F_SetAdjustCtlA	Sets whether to enable Echo Study.	=0: Disabled =1: Enabled
	25	F_RCVPHONUMHOLDUPA	Sets the control switch of the 'Called Number Hold-up' feature. For more information, refer to the description of the configuration item PhoNumHoldup.	Value Range: 0/1/2
	26	F_RELATIVEENGYHOOKDETECTA	Sets the working status of the 'Enhanced remote pickup detector' on the analog trunk channel. For more information, refer to 'Enhanced Remote Pickup Detector' in Chapter 1 or the configuration item RelativeEngyHookDetect.	=0: Disabled =1: Enabled
	27	F_CUTDTMFWHILERECORDING	Sets whether to clamp DTMF digits when the monitoring board is recording.	=0: Disabled =1: Enabled
	28	F_AlwaysDetectBargein	Dynamically sets the switch of Bargein feature.	=0: Disabled =1: Enabled
	29	F_NoiseFilteringMinGate	Dynamically sets the filtration threshold of the back noise.	=0: Disabled 0<n≤1000: Threshold value.
	30	F_SipSetSendDTMFType	Dynamically sets the DTMF transmission mode for the SIP channel.	=0: RFC2833; =1: Signaling; =2: In-band.
	31	F_FastPlayTime	Sets the Fast FWD/Fast BWD step for voice playing on the channel.	Refer to the configuration item FastPlayTime.

Return Value:

-1	Call Failed, this function is not supported
0	Call failed, the failure reason can be obtained from the function SsmGetLastErrMsg
1	Successful

Function Description

Sets the control attributes of the channel.

Note:

Related Information:

Driver Version	SynCTI Ver. 4.0.0.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmGetFlag

2.1.7.2 SsmGetFlag

Obtains the status of the functional switch on the channel.

Format:

int SsmGetFlag(int ch, int nType, long* plValue)

Parameter Description:

ch	Channel Number
nType plValue	nType: selects the parameter type to be set. For the value of the parameters, refer to the description of the functions. plValue returns the parameter's setup value. Below are the physical meanings of nType and plValue.

nType		Description
Value	MACRO in shpa3api.h	
1	F_RCVDTMFSENS	For more details, refer to the function SsmSetFlag.

	2	F_TXDTMFAMP	
	5	F_RXR2FILTERTIME	
	9	F_ClearInVoiceOnRcvDtmf	
	10	F_MixerResToBus	
	11	F_HighAndLowFreqEnScale	
	12	F_DualAndAllFreqEnScale	
	15	F_ISDNNet_WaitRemotePickup	plValue returns the detailed reason that the driver transfers the channel state to the state of S_CALL_WAIT_REMOTE_PICK, i.e, the type of the message which is received from remote PBX: ✧ plValue=4：CALL PROCEEDING; ✧ plValue=5：ALERTING; ✧ plValue=6：CONNECT; **Note:** This function requires SynCTI version 4.7.1.5 or above.
	17	F_ClearInVoiceOnRcv450Hz	Refer to the description of the function SsmSetFlag for details.
	18	F_FSKMinGate	
	19	F_RECTOFILEA_CALLBACKTIMEA	
	20	F_CALLERIDTYPE	
	21	F_InVoiceToBusA	
	22	F_EchoCancelInFskA	
	23	F_ChToRingingOnRingCntA	
	24	F_SetAdjustCtlA	
	25	F_RCVPHONUMHOLDUPA	
	26	F_RELATIVEENGYHOOKDETECTA	
	29	F_NoiseFilteringMinGate	
	30	F_SipSetSendDTMFType	
	31	F_FastPlayTime	

Return Value:

-1	Call failed, the failure reason can be acquired from the function SsmGetLastErrMsg.
0	Successful.

Function Description:

Obtains the status of the functional switch on the channel.

Note:

Related Information:

Driver Version	SynCTI Ver. 4.0.0.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmSetFlag

2.2 Functions for Event-mode Programming

2.2.1 Obtaining Events

2.2.1.1 SsmWaitForEvent

Refer to SsmWaitForEventA.

2.2.1.2 SsmWaitForEventA

Obtain the events thrown out by the Synway driver program. SsmWaitForEvent and SsmWaitForEventA are

asynchronous functions. If there are events generated, these functions will return immediately; if there are no events generated, the caller's thread will be blocked and these functions will not return until there are events generated or timeout.

Format:

int SsmWaitForEvent(DWORD dwTimeOut, PMESSAGE_INFO pEvent)

int SsmWaitForEventA(DWORD dwTimeOut, PSSM_EVENT pEventEx)

Parameter Description:

dwTimeOut	Set the operating mode of the function SsmWaitForEvent(), the value range: 0: SsmWaitForEvent works under the synchronization mode, i.e, the function will return immediately regardless of whether there are events or not. In this mode, SsmWaitForEvent and SsmGetEvent perform a same function. 0xffffffff: If there are no events, the function will get suspended and not return until there are events generated. Others: Set the maximum waiting time, calculated by millisecond. When waiting for events, if there are events generated, the function will return immediately; if there are no events, the function will not return until the waiting time exceeds the set value of dwTimeOut.
pEvent pEventEx	The pointer to the MESSAGE_INFO(pEvent) structure or the SSM_EVENT(pEventEx) structure, used to return the event information, and valid only when the return value of the function is 0. For more information, refer to 'MESSAGE_INFO' or 'SSM_EVENT' in Chapter 1.

Return Value:

-2	In any of the following situations: ✧ API interfaces are unopened; ✧ The event polling mode is not set as the programming mode; ✧ In-memory message queues are unallocated.
-1	Timeout
0	There are events generated and the event information is returned by pEvent or pEventEx.

Function Description:

Obtain the events thrown out by the Synway driver program. If the event buffer of the driver is not empty, SsmWaitForEvent or SsmWaitForEventA will return immediately; if there are no events available, the function will wait for the events and not return until the waiting time exceeds the set value of dwTimeOut.

Note:

- The functions can be invoked successfully only when the driver is working under the event polling mode.
- When MESSAGE_INFO is used, only SsmWaitForEvent can be invoked successfully. When SSM_EVENT is used, only SsmWaitForEventA can be invoked successfully.
- Functions SsmWaitForEvent and SsmWaitForEventA are the same except for the data structures of the output events. SsmWaitForEventA is usually used for the D-channel event programming of the DST Series boards.
- The total number of events to be saved in the event queue cannot exceed MaxEventNum. MaxEventNum=TotalAppCh*MaxEventPerChannel. TotalAppCh is used to set the total number of channels in the System, while MaxEventPerChannel is used to set the depth of a channel's event queue.
- If the amount of events exceeds MaxEventNum, no more event will be saved to the event queue.

Related Information:

Driver Version	SynCTI Ver. 4.0 or above for SsmWaitForEvent SynCTI Ver. 4.7.3.0 or above for SsmWaitForEventA
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmSetEvent

2.2.1.3 SsmGetEvent

Refer to SsmGetEventA.

2.2.1.4 SsmGetEventA

Obtain the events thrown out by the Synway driver program. SsmGetEvent and SsmGetEventA are synchronous functions which will return immediately regardless of whether there are events generated or not.

Format:

int SsmGetEvent(PMESSAGE_INFO pEvent)

int SsmGetEventA(PSSM_EVENT pEventEx)

Parameter Description:

pEvent pEventEx	The pointer to the MESSAGE_INFO(pEvent) structure or the SSM_EVENT(pEventEx) structure, used to return the event information and valid only when the return value of the function is 0. For more information, refer to 'MESSAGE_INFO' or 'SSM_EVENT' in Chapter 1.

Return Value:

-1	In any of the following situations: ✧ API interfaces are unopened; ✧ The event polling mode is not set as the programming mode; ✧ There is no event output.
0	There are events generated and the event information is returned by pEvent or pEventEx.

Function Description:

Obtain the events thrown out by the Synway driver program.

Note:

- The function can be called successfully only when the driver is working under the event polling mode.
- When MESSAGE_INFO is used, only SsmGetEvent can be called successfully. When SSM_EVENT is used, only SsmGetEventA can be called successfully.
- Functions SsmGetEvent and SsmGetEventA are the same except for the data structures of the output events. SsmGetEventA is usually used for the D-channel event programming of the DST Series boards.

Related Information:

Driver Version	SynCTI Ver. 4.0 or above for SsmGetEvent SynCTI Ver. 4.7.3.0 or above for SsmGetEventA
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmSetEvent

2.2.2 Filtering Output Events

2.2.2.1 SsmSetEvent

Sets the event output mode of the driver program and permits or forbids the driver to output one particular event.

Format:

int SsmSetEvent(WORD wEvent, int nReference, BOOL bEnable, PEVENT_SET_INFO pEventSet)

Parameter Description:

<table>
<tr><td>wEvent</td><td>Selects the event and the range of values:
0~0xfffe: Event code. For the value range of values, refer to MESSAGE INFO in Chapter 1. The function SsmSetEvent informs the driver program whether to throw out a specified event or not.
0xffff: Sets the working mode of the driver program, i.e. the event polling mode and the event callback mode, where the parameter nReference must be -1. The event queue depth in the driver can be set with the configuration item MaxEventPerChannel. For more information, please refer to 'Setting Event Output Mode' in Chapter 1.</td></tr>
<tr><td>nReference</td><td>The reference value of the event. If wEvent=0xffff, this parameter must be -1, used to set the working mode for the whole driver; if wEvent≠oxffff, the physical meaning of nReference depends on the type of wEvent as shown in the following list:
<table>
<tr><th>wEvent</th><th>The meaning of nReference and it's range of values</th></tr>
<tr><td>E_CHG_ISDNStatus
E_CHG_PcmLinkStatus
E_CHG_RemotePCMBlock</td><td>The logical number of the digital trunk:
nReference=-1: Sets all the digital trunks in the system.
0≤nReference≤M-1: Sets the digital trunk designated by this parameter.
Note: M is the total number of the digital trunks in the system.</td></tr>
<tr><td>E_RCV_Ss7Msu</td><td>0-Reserved</td></tr>
<tr><td>E_RCV_Ss7SpyMsu</td><td>0-Reserved</td></tr>
<tr><td>E_CHG_Mtp3State</td><td>DPC number:
nReference=-1: Sets all the DPCs in the system.
0≤nReference≤M-1: Sets the DPC designated by this parameter.</td></tr>
<tr><td>E_SYS_TIMEOUT</td><td>Timer ID:
nReference=-1: Sets all the timers in the system.
0≤nReference≤M-1: Sets the timer designated by this parameter</td></tr>
<tr><td>E_CHG_SpyState</td><td>Logical SpyCic number:
nReference=-1: Sets all the SpyCics in the system.
0≤nReference≤M-1: Sets the SpyCic designated by this parameter</td></tr>
<tr><td>E_CHG_SpyHangupInfo</td><td>Logical SpyCic number:
nReference=-1: Sets all monitored circuits in the system;
0≤nReference≤M-1: Sets the monitored circuits which are specified by this parameter.</td></tr>
<tr><td>E_CHG_CICRxPhoNumBuf
E_CHG_CICState
E_PROC_CICAutoDial</td><td>Virtual CIC number:
nReference=-1: Sets the virtual CIC in the system.
0≤nReference≤M-1: Sets the virtual CIC designated by this parameter</td></tr>
<tr><td>Other Events</td><td>Logical channel number:
nReference=-1: Sets all the channels (total is N) in the system.
0≤nReference≤N-1: Sets the channel designated by this parameter.</td></tr>
</table></td></tr>
<tr><td>bEnable</td><td>The control sign of the event output. When wEvent=0xffff and nReference=-1, if bEnable=FALSE, the driver will by default not output any event; if bEnable=TRUE, the</td></tr>
</table>

driver will output an default event specified by DefaultEventOutput. When wEvent≠0xffff, the value range of bEnable is as follows:
=TRUE: Throws out wEvent;
=FALSE: Not throws out wEvent.

pEventSet

Pointer which points to the struct object of EVENT_SET_INFO. It's used for setting up the output conditions of the events. If the parameter wEvent=0xffff, this parameter can't be NULL. The declaration of the struct EVENT_SET_INFO is:

```
typedef struct _EVENT_SET_INFO
{
    DWORD   dwWorkMode;
    LPVOID  lpHandlerParam;
    DWORD   dwOutCondition;
    DWORD   dwOutParamVal;
    DWORD   dwUser;
} EVENT_SET_INFO,*PEVENT_SET_INFO;
```

Below is the meaning of each parameter:

Parameter Name: *dwWorkMode*

Sets the working mode that the driver throws out the events. Only if wEvent=0xffff, is this parameter valid; If wEvent is other values, this parameter will be ignored. The range of values:

Value	Macro in event.h	Description
0	NO_EVENT	The driver doesn't output any events.
1	EVENT_POLLING	The driver works under the event polling mode. The parameters ***dwOutCondition*** and ***dwOutParamVal*** are valid, the parameters ***lpHandlerParam*** and ***dwUser*** are invalid.
2	EVENT_CALLBACK	The driver works under the event callback mode. ***lpHandlerParam*** and ***dwUser*** are valid.
3	EVENT_MESSAGE	The driver works under the windows message mode. All the messages will be sent to the windows message queue.
4	EVENT_POLLINGA	The driver works under the event polling mode. The parameters ***dwOutCondition*** and ***dwOutParamVal*** are valid, the parameters ***lpHandlerParam*** and ***dwUser*** are invalid.
5	EVENT_CALLBACKA	The driver works under the event callback mode. ***lpHandlerParam*** and ***dwUser*** are valid.

Parameter Name: *lpHandlerParam*

Only when ***dwWorkMode***=EVENT_CALLBACK, EVENT_CALLBACKA or EVENT_MESSAGE, is this parameter valid.
If ***dwWorkMode*** equals to other values, this parameter will be ignored.

- ***dwWorkMode***=EVENT_CALLBACK
  lpHandlerParam is the pointer to the callback function.
  The format of ***lpHandlerParam*** :
  int CALLBACK CallBackFun(WORD wEvent,int nReference,DWORD dwParam,DWORD dwUser);
  The meanings of wEvent, nReference and dwParam are the same as the corresponding variables declared in the struct MESSAGE_INFO.
  For more details, refer to the function SsmWaitForEvent or SsmGetEvent. The parameter dwUser is the parameter ***dwUser*** which is passed from the application to the driver when invoking this callback function.
- ***dwWorkMode***=EVENT_CALLBACKA
  lpHandlerParam is the pointer to the callback function.
  The format of ***lpHandlerParam***:
  int CALLBACK CallBackFunA(PSSM_EVENT pEvent):
  pEvent which returns the event information is the pointer pointing to the struct object SSM_EVENT.
  For more details, refer to the function SsmWaitForEventA or SsmGetEventA .
  Note: When the callback function returns -1, all events are forbidden being thrown out. After that, if it is necessary to throw them out, you should call the

function SsmSetEvent to reset them first.

➢ ***dwWorkMode***=EVENT_MESSAGE

lpHandlerParam is the window handle which is used by the application to receive the driver events.

Parameter Name: *dwUser*

User-defined parameter which is valid only when ***dwWorkMode***=EVENT_CALLBACK or EVENT_CALLBACKA; If ***dwWorkMode*** equals to other values, this parameter will be ignored. The driver will save this parameter. When the driver throws out events, it will be directly passed to the parameter having the same name in the callback function pointed by ***lpHandlerParam***.

Parameter Name: *dwOutCondition, dwOutParamVal*

dwOutCondition sets the conditions on which the events are output. ***dwOutParamVal*** sets the types of the parameters of the output events. When the event is generated, the driver will assign the data types set by ***dwOutParamVal*** to the parameter dwParam in the struct MESSAGE_INFO or SSM_EVENT. ***dwOutCondition*** and ***dwOutParamVal*** are optional parameters. When using the default value of ***dwOutCondition***, you can set pEventSet=NULL. But if the parameter wEvent is equal to one of the following values and pEventSet≠NULL, you have to set ***dwOutCondition*** each time before you invoke SsmSetEvent; otherwise, you shall meet unexpected consequence.

✧ **wEvent** = E_CHG_LineVoltage:

Only when the voltage change range on the analog phone line exceeds **dwOutCondition** (default value can be 5), will the driver throw out the event E_CHG_LineVoltage. **dwOutParamVal** is reserved but not used. When wEvent=0xffff and nReference=-1, **dwOutCondition** will be automatically set to 5**.**

✧ **wEvent** = E_PROC_PlayFile:

dwOutCondition sets the time interval (ms) that the driver throws out the event E_PROC_PlayFile (default value may be 1000). **dwOutParamVal** sets the types of the output parameters, the optional values are:

=0: Outputs the percentage of the played part.
=1: Outputs the time that finishes the play.
=2: Outputs the total number of played bytes.
=3: Outputs the total number of not-played bytes.

When wEvent=0xffff and nReference=-1, **dwOutCondition** will be automatically set to 1000 and **dwOutParamVal** set to 1**.**

✧ **wEvent** = E_PROC_PlayMem:

◆**dwOutParamVal**=END_HALF_BUFFER:
When the play pointer in the driver gets across the middle position or the tail part of the buffer, the driver throws out the event E_PROC_PlayMem.
If the play pointer points to the front part of the buffer, the output value (dwParam) is -1; If it points to the rear part, the output value is -2.

◆**dwOutParamVal**=END_BUFFER:
When the play pointer in the driver gets across the tail part of the buffer, the driver throws out the event E_PROC_PlayMem and outputs -2.

◆**dwOutParamVal**=MEM_OFFSET:
After a certain time period of voice playback, the driver throws out the event E_PROC_PlayMem and outputs the play pointer (i.e., the offset in the buffer) in the driver. The time length (ms) is specified by **dwOutCondition**.

◆**dwOutParamVal**=MEM_BYTES:
After playing a certain amount of voice-data bytes, the driver throws out the event E_PROC_PlayMem and outputs the play pointer (i.e., the offset in the buffer) in the driver. The amount of the bytes is specified by **dwOutCondition** (default value may be 64).

When wEvent=0xffff and nReference=-1, **dwOutParamVal** will be automatically set to MEM_OFFSET and **dwOutCondition** will be set to 64 ms.

✧ **wEvent** = E_PROC_RecordFile:

dwOutCondition is set to be the time interval (ms) that the driver throws out the event E_PROC_RecordFile (default value may be 1000).

dwOutParamVal is set to be the types of output values, the optional values are:

	=RECORD_TIME: Outputs the time (ms) spent on the finished recording; =RECORD_BYTES: Outputs the total finished recorded bytes(bytes). When wEvent=0xffff and nReference=-1, **dwOutCondition** will be automatically set to 1000 and **dwOutParamVal** set to RECORD_TIME**.** ✧ **wEvent** = E_PROC_RecordMem: ◆ **dwOutParamVal**=END_HALF_BUFFER: When the record pointer gets across the middle position or the tail part of the recording buffer, the driver throws out the event E_PROC_RecordMem. If the record pointer points to the front part of the buffer, the output value is -1; If the pointer points to the rear part, the output value is -2. ◆ **dwOutParamVal**=END_BUFFER: When the record pointer in the driver gets across the tail part of the buffer, the driver throws out the event of E_PROC_RecordMem and the output value is -2. ◆ **dwOutParamVal**=MEM_OFFSET: After a certain time period of voice recording, the driver throws out the event E_PROC_RecordMem and outputs the record pointer (i.e., the offset in the buffer) in the driver. The time length (ms) is specified by **dwOutCondition**. ◆ **dwOutParamVal**=MEM_BYTES: After recording a certain amount of voice-data bytes, the driver throws out the event of E_PROC_RecordMem and outputs record pointer (i.e., the offset in the buffer) in the driver. The bytes number is specified by **dwOutCondition** (default value may be 64). When wEvent=0xffff and nReference=-1, **dwOutParamVal** will be automatically set to MEM_OFFSET and **dwOutCondition** will be set to 64 ms. ✧ **wEvent** = E_OverallEnergy: When the change range of the full-frequency energy exceeds the set value of **dwOutCondition** (default value may be 1000) the driver throws out the event E_OverallEnergy. **dwOutParamVa**l is reserved and unused. When wEvent=0xffff and nReference=-1, **dwOutCondition** will be automatically set to 1000**.** ✧ **wEvent** = E_CHG_PeakFrq: When the change range of the peak frequency exceeds the set value of **dwOutCondition** (default value may be 100), the driver throws out the event E_CHG_PeakFrq. **dwOutParamVa**l is reserved and unused. When wEvent=0xffff and nReference=-1, dwOutCondition will be automatically set to 100. ✧ **wEvent** = Other events: Both of **dwOutParamVal** and **dwOutParamVal** are reserved and unused. They can be set to be 0.

Return Value:

-1	Failed
0	Successful

Function Description:

If wEvent=0xffff, this function is used to set the event output mode for the driver. For more information about the output mode supported by the SynCTI driver, refer to 'Setting Driver Event Output Mode' in Chapter 1.

If wEvent≠0xffff, this function is used to permit or forbid the driver to output one particular event. If there are some output parameters, they should be set. First set the output mode for the event. Then adjust the output rule for an undetermined event.

Notes:

- After the SynCTI driver is started, it automatically works under the polling mode. If the programming is done in the event polling or event callback mode, this function must be invoked to change the driver's event output mode after driver initialization.
- If wEvent=0xffff, the application can call this function only once and the parameter pEventSet can't be NULL otherwise the function returns failure.

- If wEvent≠0xffff, pEventSet can be NULL. If pEventSet is not NULL, its member variable dwWorkMode must be consistent with the previous set event output mode. At present, the driver does not support different modes to output multiple events.

Related Information:

Driver version	SynCTI Ver.4.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmWaitForEvent, SsmWaitForEventA, SsmGetEvent, SsmGetEventA, SsmPutUserEvent, SsmPutUserEventA

Sample Code: Refer to 'Programming demo for event polling mode' or 'Programming demo for event callback mode' in chapter 1.

2.2.3 Getting Programming Mode

2.2.3.1 SsmGetEventMode

Obtains the programming mode.

Format:

int SsmGetEventMode(WORD wEvent, int nReference, PWORD pwEnable, PEVENT_SET_INFO pEventSet)

Parameter Description:

wEvent	Event code: Regarding the value range, refer to the section MESSAGE_INFO in Chapter 1. Note that the value can not be 0xffff.
nReference	The reference value of the event: The physical meaning of nReference varies on the type of wEvent, just as the parameter nReference in the function SsmSetEvent. However, its value can not be -1.
pwEnable	Stores the flag to control the event output. Value range: =TRUE: Output wEvent; =FALSE: Not output wEvent
pEventSet	The point that points to the objects which have the EVENT_SET_INFO structure. It is used to save the event output condition set by the function SsmSetEvent. See SsmSetEvent for the EVENT_SET_INFO structure declaration.

Return Value:

-1	Failed
0	Successful

Function Description:

Obtains the driver event output mode set by the application.

Note:

- When this function is called to get the appplication's event programming mode, an event code but 0xffff should be assigned to the parameter wEvent .

Related Information:

Driver version	SynCTI Ver.4.8.0.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmSetEvent

2.2.4 Outputting Application's Self-defined Events by Driver

2.2.4.1 SsmPutUserEvent

Refer to SsmPutUserEventA

2.2.4.2 SsmPutUserEventA

Appends the application defined events to the internal event queue of the driver.

Format:

int SsmPutUserEvent(WORD wEvent, int nReference, DWORD dwParam)

int SsmPutUserEventA(PSSM_EVENT pEvent)

Parameter Description:

wEvent	User-defined event code. **Note: The user-defined event code should not be the same as the event code defined by the driver. Suggestion: Set the starting event code value to be 0xfffe and reduce it gradually.**
nReference	Event output reference. It will be transparently passed back to the application via the internal event queue of the driver.
dwParam	Event output parameter. It will be transparently passed back to the application via the internal event queue of the driver.
pEvent	Pointer pointing to the struct object SSM_EVENT. It inputs the event information to the driver. The struct object pointed by pEvent will be transparently passed back to the application via the internal event queue of the driver.

Return Value:

-1	Failed.
0	Successful.

Function Description:

Appends the application defined events to the internal event queue of the driver. Function SsmPutUserEvent and SsmPutUserEventA have the same functions except for the different data structures of the event. SsmPutUserEventA is normally used for the D-channel event programming of the DST Series board.

The driver puts the events submitted by the application in a temporary buffer in advance and appends the events to the event output queue of the driver when the hardware interrupt occurs. The configuration item MaxUserEventSize is used to set the depth of the temporary buffer.

Note:

- This function can only be invoked under 'event polling' programming mode. For more information about event polling mode, refer to 'Setting Driver Event Output Mode' in chapter 1.
- When MESSAGE_INFO is used, only the function SsmPutUserEvent can be used. When SSM_EVENT is used, only the function SsmPutUserEventA can be used.
- If the configuration MaxUserEventSize is set to be 0, the function returns failure immediately.

Related Information:

Driver version	SsmPutUserEvent requires SynCTI Ver. 4.0 or above; SsmPutUserEventA requires SynCTI Ver. 4.7.3.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmWaitForEvent, SsmWaitForEventA, SsmGetEvent, SsmGetEventA

2.3 Call State Machine Functions

2.3.1 Outgoing Call Functions (CTI Series)

2.3.1.1 Searching for Idle Channel

2.3.1.1.1 SsmSearchIdleCallOutCh

Searches for an idle channel in the channels which are able to make outgoing calls.

Format:

int SsmSearchIdleCallOutCh(WORD wSearchMode, DWORD dwPrecedence)

Parameter Description:

<table>
<tr><td>wSearchMode</td><td>Specify the searching mode based on the bit value:
<table>
<tr><th>Bit</th><th>Description</th></tr>
<tr><td>Bit0</td><td>Whether to search among analog trunk channels.
=1: Yes; =0: No</td></tr>
<tr><td>Bit1</td><td>Whether to search among SS1 channels.
=1: Yes; =0: No</td></tr>
<tr><td>Bit2</td><td>Whether to search among TUP channels.
=1: Yes; =0: No</td></tr>
<tr><td>Bit3</td><td>Whether to search among ISDN channels (user side).
=1: Yes; =0: No</td></tr>
<tr><td>Bit4</td><td>Whether to search among ISDN channels (network side).
=1: Yes; =0: No</td></tr>
<tr><td>Bit5</td><td>Whether to search among ISUP channels.
=1: Yes; =0: No</td></tr>
<tr><td>Bit6</td><td>Whether to search among those TUP channels corresponding to the specified signaling link set.
=1: Yes; =0: No</td></tr>
<tr><td>Bit7</td><td>Whether to search among those ISUP channels corresponding to the specified signaling link set.
=1: Yes; =0: No</td></tr>
<tr><td>Bit8</td><td>Whether to search among SIP channels.
=1: Yes; =0: No</td></tr>
<tr><td>Bit9</td><td>Reserved</td></tr>
<tr><td>Bit10</td><td>Whether to search among those SS7 channels corresponding to the specified PCM.
=1: Yes; =0: No</td></tr>
<tr><td>Bit11</td><td>Whether to search among those ISDN channels corresponding to the specified PCM.
=1: Yes; =0: No</td></tr>
<tr><td>Bit12</td><td>Whether to specify the searching range among ISDN channels or SIP channels.
=1: Yes; =0: No</td></tr>
<tr><td>Bit13</td><td>Whether to specify the searching range among analog trunk channels.
=1: Yes; =0: No</td></tr>
<tr><td>Bit14</td><td>Whether to search among analog station channels.
=1: Yes; =0: No</td></tr>
<tr><td>Bit15</td><td>Whether to specify the searching range among analog station channels.</td></tr>
</table></td></tr>
</table>

	=1: Yes; =0: No
dwPrecedence	Bit6 or Bit7 in wSearchMode (TUP or ISUP channel) set to 1: Select a signaling link set; Bit10 or Bit11 in wSearchMode (SS7 or ISDN channel) set to 1: Select a logical PCM number; Bit12 in wSearchMode set to 1: The 16 lower bits indicate the start channel and the 16 higher bits represent the end channel. Bit13 in wSearchMode set to 1: The 16 lower bits indicate the start channel and the 16 higher bits represent the end channel. Bit15 in wSearchMode set to 1: The 16 lower bits indicate the start channel and the 16 higher bits represent the end channel. Other types of channels: Set to 0.

Return Value:

-1	Failed. The failure reason can be acquired from the function SsmGetLastErrMsg.
≥0	The channel number which has been searched out.

Function Description:

Searches for an idle channel in the channels which are able to make outgoing calls. The driver maintains an idle channel queue.

Note:

- After this function is invoked, the channel state will remain idle for analog trunk channels, or turn into 'outgoing call locked' for ISDN, SS7, SS1 and SIP channels.
- To search channels in SS1, ISDN protocols, follow the time slots of E1 trunks in a 0, 1, 2, 3...31 sequence (except TS0 and TS16);
- To search channels in SS7 protocol, follow the size of OPC and DPC configured by this E1;
- The party with a large point code controls even time slots. To search channels, follow the even time slots in a 0, 2, 4, …, 30, 31, 29, 27, …, 1 sequence (except TS0, TS1 and TS16);
- The party with a small point code controls odd time slots. To search channels, follow the odd time slots in a 1, 3, 5, …, 31, 30, 28, 26, …, 0 sequence (except TS0, TS1 and TS16);
- For ISDN protocol, when the 11th bit of wSearchMode is set to 1 which means only the specified PCM will be searched for the idle channel, and dwPrecedence is set to -1, all PCMs will be searched for the idle channel.

Related Information:

Driver version	SynCTI Ver. 5.3.3.1 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmAutoDial, SsmChkAutoDial

2.3.1.2 Starting Outgoing Call

2.3.1.2.1 SsmAutoDial

Refer to SsmAutoDialEx

2.3.1.2.2 SsmAutoDialEx

Submits the AutoDial task to the driver to start an outgoing call. This function can set the address indicator in the calling line identification field in the TUP protocol (which can also be set by the configuration item CallerAddrInd), and is normally used for TUP channels.

Format:

int SsmAutoDial(int ch, LPSTR szPhoNum)

int SsmAutoDialEx(int ch, LPSTR szPhoNum, WORD wParam)

Parameter Description:

<table>
<tr><td>ch</td><td>Channel Number</td></tr>
<tr><td>szPhoNum</td><td>The pointer pointing to the buffer storing the called party number, with a maximum length of 50.
● Analog trunk channel: If the character in szPhoNum is neither standard DTMF character nor a designated flash character, it will be considered as a delay character. For more information about the flash character and delay character, refer to the function SsmTxDtmf.
● SS1 channel, TUP/ISUP channel and ISDN channel: the valid phone number must be digital characters between ‘0’ ~ ‘9’ and any non-digital character will be ignored by the driver.
● SIP channel: [“display-name” <][sip:/sips:][user@]host[:port][>]
a) display-name: local alias, with the default setting of NULL
b) sip:/sips: : the default setting is sip:. The setting sips: means to transmit the sip message in encryption mode
c) user: user name, with the default setting of NULL
d) host: allowed to be host name, IPv4 address or registered SIP server domain name
e) port: with the default value of 5060</td></tr>
<tr><td>wParam</td><td>This parameter is only applicable to the TUP channel.
The 4 higher bits should be set to be 0 and the 4 lower bits (namely DCBA) are valid.
<table>
<tr><th>Bit(s)</th><th>Meaning</th><th>Value</th><th>Description</th></tr>
<tr><td rowspan="4">BA</td><td rowspan="4">Nature of Address Indicator</td><td>00</td><td>Local subscriber number</td></tr>
<tr><td>01</td><td>Spare national number</td></tr>
<tr><td>10</td><td>Valid national number</td></tr>
<tr><td>11</td><td>International number</td></tr>
<tr><td rowspan="2">C</td><td rowspan="2">Calling party number presentation indicator</td><td>0</td><td>Presentation unrestricted</td></tr>
<tr><td>1</td><td>Presentation restricted</td></tr>
<tr><td rowspan="2">D</td><td rowspan="2">Calling party number incomplete indicator</td><td>0</td><td>Not incomplete</td></tr>
<tr><td>1</td><td>Incomplete</td></tr>
</table></td></tr>
</table>

Return Value:

-1	Failed. The failure reason can be acquired from the function SsmGetLastErrMsg.
0	Successful.

Function Description:

Submits the AutoDial task to the driver to start an outgoing call.

After the AutoDial task is started, the channel state machine automatically initiates an outgoing call connection. For related information, refer to the channel state transfer introductions in this manual.

If there is any progress with the outgoing call, the driver throws out the event E_PROC_AutoDial to the application. The application may also inquire the call progress status via the function call of SsmChkAutoDial. Because the call progress will be represented by the channel state transfer, the application can obtain the progress status of the AutoDial task by inquiring the channel state transfer information via the event E_CHG_ChState or the function

SsmGetChState.

➢ **ISUP Channel**

The driver originates an outgoing call with the IAM message. There are two methods to compose an IAM message.

(1) Automatically composed by the driver. Some fields of the IAM message can be customized by the following methods:

Field	Method
Nature of connection indicator [1]	Change the configuration item DefaultNatureOfConnectionInd.
Forward call indicator[1]	Change the configuration item DefaultIAM_ForwardCallInd.
Calling party category[1]	Change the configuration item DefaultIAM_CAT or invoke the function SsmSetISUPCAT.
Transmission medium requirement indicator [1]	Change the configuration item DefaultIAM_TransmissionMediumRequirment.
Called party number[2]	The first and second 8-bit group in the field can be set by the configuration item DefaultIAM_CalleeParam or the function SsmSetIsupFlag which has the parameter ISUP_PhoNumParam; The follow-up 8-bit group (i.e. the phone number) can be set by the parameter szPhoNum of the function SsmAutoDialEx.
Calling party number[3]	The first and second 8-bit group in the field can be set by the configuration item DefaultIAM_CallerParam or the function SsmSetIsupFlag which has the parameter ISUP_CallerParam; The follow-up 8-bit group (i.e the phone number) can be set by the configuration item CalloutCallerId or the function SsmSetTxCallerId. The configuration item SetSTSignal decides whether to transmit ST signal in the calling party number.
Original called party number[3]	The first and second 8-bit group can be set by the configuration item DefaultIAM_OriginalCalleeParam; The follow-up 8-bit group (i.e. the phone number) can be set by the function SsmSetTxOriginalCallerID.
User service information[3]	The configuration item bSubscriberSI decides whether to include this field, the configuration item SubscriberSI sets the value of this field.
Optional forward call indicator[3]	The configuration item bOptionalFCI decides whether to include this item, the configuration item OptionalFCI sets the value of this field.
User to user information[3]	Changes the configuration item Usr2UsrInfo or calls the function SsmSetIsupParameter.
Other optional parameters[3]	To set other parameters, call the function SsmSetIsupParameter.

Note: In the above table, ISUP_CallerParam and ISUP_ PhoNumParam are declared in the header file Shpa3api.h. [1] denotes essential length-fixed fields, [2] denotes essential length-unfixed fields, [3] denotes optional fields.

(2) The application itself creates an IAM message. Before invoking this function, the application itself can create an IAM message and submit the IAM message to the driver via the function call of SsmSetIsupUPPara. When the driver executes this function, it automatically covers the fields of SIO, DPC, OPC, SLC, CIC and the called party number.

➢ **TUP Channel**

The TUP channel can use the IAI message or IAM message as the initial address message to originate an outgoing call. Below are the functions and configuration items related with the IAM/IAI message:

Message Type	Related Function or Configuration Item	Description
IAM IAI	Configuration item: ConnectReqMsg	Decides which message to use, IAM or IAI
	Configuration item: CalloutIAM_CAT	Sets the Calling Party Category
	Configuration item: CalloutIAM_MsgPntr	Sets the field of the message indicator
IAI	Function: SsmSetTxOriginalCallerID	Sets the original calling party address
	Configuration item: OriginalCalleeAddrInd	Sets the original called party address indicator
	Configuration item: CalloutCallerId	Sets the calling party number
	Function: SsmSetTxCallerId	
	Configuration item: CallingIndicatorBit	Sets the additional calling party information indicator and calling line identification indicator in the 8-bit group of the first indicator.

	Configuration item: SetSTSignal	Sets whether the calling party number parameter ends with ST signal.
	Configuration item: CallerAddrInd	Sets the address indicator in the calling line identification field

If, before the application calls SsmAutoDialEx , the function SsmSetTxOriginalCallerID is invoked to set the original called party address, the driver only uses the IAI message to originate an outgoing call. The address indicator in the original called party address field of the IAI message can be set via the configuration item OriginalCalleeAddrInd.

- **ISDN Channel**

For the ISDN protocol, this function will trigger the driver to use the SETUP message to originate the outgoing call. Below are the functions and configuration items related with the SETUP message.

Message Type	Related Function or Configuration Item	Description
SETUP	Configuration item: UserCalledNoSet, NetCalledNoSet	Sets the called party number type and the numbering plan
	Configuration item: UserCallingNoSet, NetCallingNoSet	Sets the calling party number type and the numbering plan
	Configuration item: UserNumIsFull, NetNumIsFull	Sets whether to include the parameter of "Called number complete" in SETUP message.
	Configuration item: UserChIdentify, NetChIdentify	Sets the indication method for channel identification
	Configuration item: UserChPreference, NetChPreference	Sets whether to allow the preferential channel selection
	Configuration item: CalloutCallerId	Sets the calling party number (needs to be invoked before the call of this function)
	Function: SsmSetTxCallerId	
	Configuration item: UserHighLayerCompatible, NetHighLayerCompatible	Sets whether to include the "high layer compatibility" field in the SETUP message
	Configuration item: UserLowLayerCompatible, NetLowLayerCompatible	Sets whether to include the "low layer compatibility" field in the SETUP message
	Function: SsmISDNSetTxSubAddr	Sets the calling party subaddress (needs to be invoked before the call of this function)
	Function: SsmISDNSetDialSubAddr	Sets the called party subaddress (needs to be invoked before the call of this function)
	Function: SsmSetTxOriginalCallerID	Sets the original calling party number
	Function: SsmISDNSetCallerIdPresent	Sets the 'CallerID Present' field

- **SS1 Channel**

For SS1 channel, when the driver is executing the Autodial task, the application can partially control the outgoing call process via some functions or configuration items.

Related Function or Configuration Item	Description
Function: SsmSetKA Configuration Item: MfcKA	Sets the calling party category (i.e. KA signal)
Function: SsmSetKD Configuration Item: MfcKD	Sets the "Service nature of the original end" (i.e. KD signal)
Configuration Item: CalloutCallerId Function: SsmSetTxCallerId	Sets the calling party (original end) number.

- **IP Channel**

For IP channel, this function will trigger the driver to use a message specified by the protocol to start an outbound call. To be exact, INVITE message is used for SIP channel. Functions related to these messages are listed below.

Message Type	Related Function	Description

INVITE(SIP)	Function: SsmSetTxCallerId	Set the calling party number (should be invoked before the call of this function)

The function call result can be obtained via SsmChkAutoDial. In unusual cases when the IP channel fails to dial and the call of SsmChkAutoDial always returns 11, users can invoke SsmGetAutoDialFailureReason to get the exact failure reason.

Note:

- This function only supports TUP, ISUP, SS1, ISDN, IP and analog trunk channels;
- TUP channel: Maximum length of dial number is 30;
- ISUP channel: Maximum length of dial number is 49;
- SS1 channel: Maximum length of dial number is 50;
- ISDN channel: Maximum length of dial number is 30;
- IP channel: Maximum length of dial number is 128.

Related Information:

Driver version	SynCTI Ver. 3.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmAppendPhoNum, SsmChkAutoDial, SsmGetChState, SsmGetPendingReason

2.3.1.2.3 SsmAppendPhoNum

Appends the called party number to the AutoDial operation.

Format:

int SsmAppendPhoNum(int ch, LPSTR szPhoNum)

Parameter Description:

ch	Channel Number
szPhoNum	Pointer which points to the buffer storing phone number strings. For analog trunk channel, if the character in szPhoNum is neither standard DTMF character nor specified flash character, it will be regarded as the delay character; For SS1 outbound trunk channel, any non-number characters will be ignored by the driver. The maximum length of dial string is 50.

Return Value:

-1	Failed. The failure reason can be acquired via the function SsmGetLastErrMsg.
0	Successful.

Function Description:

Appends the called party number to the AutoDial operation. For more detailed information, refer to the channel state machine of each channel.

Note:

- This function only supports TUP, ISUP, SS1, ISDN and analog trunk channels;
- TUP channel: Maximum length of appended called party number is 16;
- ISUP channel: Maximum length of appended called party number is 50;
- SS1 channel: Maximum length of appended called party number is 50;
- ISDN channel: Maximum length of appended called party number is 20.

Related Information:

Driver version	SynCTI Ver. 3.0or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmAutoDial, SsmAutoDialEx

2.3.1.2.4 SsmChkAutoDial

Queries the execution status of the AutoDial operation.

Format:

int SsmChkAutoDial(int ch)

Parameter Description:

ch	Channel Number

Return Value:

-1: Call failed. Below are the meanings of other return values:

Return Value	Macro in shpa3api.h	Description
0	DIAL_STANDBY	Channel is idle and not executing the AutoDial task.
1	DIAL_DIALING	Sending the called party number.
2	DIAL_ECHOTONE	Ringback. ✧ Analog Trunk Channel: After Autodial, ringback tone is detected on the line. ✧ SS1 channel: During outgoing call, if the driver receives a backward KB=1 or KB=6 signal from the remote PBX, the channel is idle. ✧ ISDN channel: Indicates the driver receives the ALERTING message from the remote PBX. ✧ TUP/ISUP channel: Indicates that the address complete message (ACM) has been received from the remote PBX.
3	DIAL_NO_DIALTONE	If no dialtone is detected on the line, AutoDial failed. It's only applicable to the analog trunk channel.
4	DIAL_BUSYTONE	The called party is busy and Autodial failed. For analogy trunk channel, it indicates that the busy tone has been detected on the line.
5	DIAL_ECHO_NOVOICE	After the Autodial, the line will go into silence after detecting the ringback tone. Then the Autodial operation is terminated. It's only applicable to analog trunk channel.
6	DIAL_NOVOICE	After the Autodial, the line keeps silence without detecting ringback tones. Then the Autodial operation is terminated. It's only applicable to analog trunk channel.
7	DIAL_VOICE	The called party picks up the call and the Autodial task is finished.
8	DIAL_VOICEF1	The called party picks up the call (the answer signal at F1 frequency detected) and the AutoDial is finished. It's only applicable to analog trunk channel.
9	DIAL_VOICEF2	The called party picks up the call (the answer signal at F2 frequency detected) and the AutoDial is finished. It's only applicable to analog trunk channel.
10	DIAL_NOANSWER	The called party doesn't pick up the phone for a specified time interval and the Autodial fails. It is not applicable to SIP channels on VoIP boards.

11	DIAL_FAILURE	AutoDial fails. The failure reason can be acquired by the function SsmGetAutoDialFailureReason.
12	DIAL_INVALID_PHONUM	The called party number is unallocated. Autodial fails. It is applicable to neither SIP channels on VoIP boards nor ISDN channels.
13	DIAL_SESSION_PROCEEDING	The SIP channel on a VoIP board receives 18X messages (except 180).
14	DIAL_ISDN_PROGRESS	The ISDN channel receives the PROGRESS message from the remote PBX. To get the details of this message, invoke the function SsmISDNGetProgressMsg.

Function Description:

Queries the execution status of the AutoDial operation.

Note:

- This function does not support the station channel, magnet channel and recording channel.
- If the ISDN or IP channel fails to dial and this function call always returns 11, users can invoke SsmGetAutoDialFailureReason to get the exact failure reason.

Related Information:

Driver version	SynCTI Ver. 3.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmAutoDial, SsmAutoDialEx

2.3.1.2.5 SsmGetAutoDialFailureReason

Obtains the failure reason for the outgoing call.

Format:

int SsmGetAutoDialFailureReason(int ch)

Parameter Description:

ch	Channel Number

Return Value:

Return Value	Macro	Description
-1	-1	Failed.
0	ATDL_NULL	No outbound call operation.
1	ATDL_Cancel	AutoDial is cancelled by the application
2	ATDL_WaitDialAnsTimeout	Waiting for answer from called party is overtime (for TUP protocol only)
3	ATDL_WaitRemotePickupTimeout	Waiting for off-hook signal from called party is overtime
10	ATDL_Mtp3Unusable	SS7signaling: Signaling is unusable.
11	ATDL_RcvSSB	SS7 signaling: Receives SSB message from remote PBX
12	ATDL_RcvSLB	SS7 signaling: Receives SLB message from remote PBX
13	ATDL_RcvSTB	SS7 signaling: Receives STB message from remote PBX
14	ATDL_RcvUNN	SS7 signaling: Receives UNN message from remote PBX
15	ATDL_RcvSEC	SS7 signaling: Receives SEC message from remote

		PBX
16	ATDL_RcvCGC	SS7 signaling: Receives CGC message from remote PBX
17	ATDL_RcvNNC	SS7 signaling: Receives NNC message from remote PBX
18	ATDL_RcvCFL	SS7 signaling: Receives CFL message from remote PBX
19	ATDL_RcvLOS	SS7 signaling: Receives LOS message from remote PBX
20	ATDL_RcvSST	SS7 signaling: Receives SST message from remote PBX
21	ATDL_RcvACB	SS7 signaling: Receives ACB message from remote PBX
22	ATDL_RcvDPN	SS7 signaling: Receives DPN message from remote PBX
23	ATDL_RcvEUM	SS7 signaling: Receives EUM message from remote PBX
24	ATDL_RcvADI	SS7 signaling: Receives ADI message from remote PBX
25	ATDL_RcvBLO	SS7 signaling: Receives BLO message from remote PBX
26	ATDL_DoubleOccupy	SS7 signaling: Collision is detected
27	ATDL_CircuitReset	SS7 signaling: Receives the circuit/group reset signal from remote PBX
28	ATDL_BlockedByRemote	SS7 signaling: The circuit is blocked by remote PBX
40	ATDL_SS1WaitOccupyAckTimeout	SS1 signaling: Waiting for the occupy acknowledge is overtime
41	ATDL_SS1RcvCAS_HANGUP	SS1 signaling: Receives the backward disconnect signal
42	ATDL_SS1RcvA4	SS1 signaling: Receives the A4 signal (Keys congestion)
43	ATDL_SS1RcvA5	SS1 signaling: Receives the A5 signal (Unallocated number)
44	ATDL_SS1RcvUndefinedAx	SS1 signaling: Receives undefined backward A signal
45	ATDL_SS1RcvUndefinedAxOnTxCallerId	SS1 signaling: Receives undefined A signal during the transmission of Caller ID
46	ATDL_SS1WaitAxTimeout	SS1 signaling: Waiting for receiving backward A group signal is overtime
47	ATDL_SS1WaitAxStopTimeout	SS1 signaling: Waiting for backward A group signal to be stopped is overtime
48	ATDL_SS1WaitAxTimeoutOnTxCallerId	SS1 signaling: Waiting for A signal is overtime during the transmission of Caller ID
49	ATDL_SS1WaitAxStopTimeoutOnTxCallerId	SS1 signaling: Waiting for backward A signal to be stopped is overtime during the transmission of Caller ID.
50	ATDL_SS1RcvKB2	SS1 signaling: KB2 signal is received (subscriber 'local busy')
51	ATDL_SS1RcvKB3	SS1 signaling: KB3 is received (subscriber 'toll busy')
52	ATDL_SS1RcvKB4	SS1 signaling: KB4 is received (keys congestion)
53	ATDL_SS1RcvKB5	SS1 signaling: Receives KB5 signal (unallocated number)
54	ATDL_SS1RcvUndefinedKB	SS1 signaling: Receives undefined KB signal
55	ATDL_SS1WaitKBTimeout	SS1 signaling: Receiving backward KB signal is overtime
56	ATDL_SS1WaitKBStopTimeout	SS1 signaling: Waiting for remote end to stop sending KB signal is overtime.
60	ATDL_ISDNNETISBUS	ISDN: Network busy (no use any more)
61	ATDL_ISDNEMPTYNO	ISDN: Unallocated number.
65	ATDL_IllegalMessage	SS7 signaling: Receives the illegal message from remote PBX
66	ATDL_RcvREL	ISUP: Receives Release message (REL) from remote PBX
67	ATDL_RcvCBK	TUP: Receives CBK message from remote PBX

68	ATDL_IPInvalidPhonum	IP: Invalid dialed number
69	ATDL_IPRemoteBusy	IP: Remote end busy
70	ATDL_IPBeenRefused	IP: Refused
71	ATDL_IPDnsFail	IP: Invalid DNS
72	ATDL_IPCodecUnSupport	IP: Unsupported CODEC
73	ATDL_IPOutOfResources	IP: No available resources
74	ATDL_IPLocalNetworkErr	IP: Local network error
75	ATDL_IPRemoteNetworkErr	IP: Remote network error
4xx, 5xx, 6xx		IP: SIP status code is received from the remote end
100+n		Other reasons, n is the ISDN cause code (refer to Appendix 1 ISDN Release Cause Information Element for details)
Other		Reserved.

Function Description:

Obtains the failure reason for the outgoing call.

Note:

- So far the SynCTI driver does not support macros ATDL_IPRemoteBusy, ATDL_IPDnsFail, ATDL_IPCodecUnSupport, ATDL_IPLocalNetworkErr and ATDL_IPRemoteNetworkErr.

Related Information:

Driver version	SynCTI Ver. 3.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmAutoDial, SsmAutoDialEx

2.3.1.2.6 SsmSetTxCallerId

Sets the local end number (i.e. Caller ID) for an outgoing call.

Format:

int SsmSetTxCallerId(int ch, LPSTR pszTxCallerId)

Parameter Description:

ch	Channel Number
pszTxCallerId	• For SIP channels, pszTxCallerId sets the ‘displayname’ part in a complete SIP URI, with the maximum size of 511. The format of a complete SIP URI is ”displayname” <sip:user@host:port>; • For other channels, this parameter stores the Caller ID in ASCII string format. To be exact, the maximum size for TUP channels is 15 characters, for ISDN channels is 20 characters and for other types of channels is 50 characters. Each character must be a digit among ‘0’~'9', otherwise it will be ignored. • If the parameter pszTxCallerId is set to empty string the data stored in the interior buffer will be cleared once this function is invoked.

Return Value:

-1	Failed
≥0	The number of the CallerIDs which have been actually sent out

Function Description:

Sets the local end number (i.e. Caller ID) for an outgoing call.

During an outgoing call, if the remote PBX requires the local end to send the Caller ID, the driver will automatically

transmit the characters in pszTxCallerId to the remote PBX.

Note:

- This function must be called before the function SsmAutoDial or SsmAutoDialEx is invoked. Once the parameter value is set, it will be saved in the internal buffer area and used subsequently in the IAM message. To cancel it, invoke this function again and set pszTxCallerId to an empty string. Then the sent calling party number will be null.
- The configuration item TxCallerId for SS1 channel can implement the same feature. Its default setting is an empty string.
- This function only supports SS1 channel, TUP channel, ISUP channel, ISDN channel and IP channel.
- For TUP and ISUP protocols, some PBXes require that the Caller ID sent by the calling party include ST signal (Signal Termination), which can be set by the configuration item SetSTSignal.
- If the Caller ID has not successfully been sent due to the failure of the function call of SsmSetTxCallerId, the function SsmAutoDial will send the Caller ID according to the definition of the configuration item CalloutCallerId in the configuration file shconfig.ini. If the configuration item CalloutCallerId has a preset number, this number will be sent to the called party as the Caller ID. If no number has been set, the Caller ID will be null.

Related Information:

Driver version	SynCTI Ver. 3.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmGetPhoNumLen, SsmAutoDial, SsmAutoDialEx

2.3.1.2.7 SsmGetTxCallerId

Obtains the Caller ID which is set by the local end.

Format:

int SsmGetTxCallerId(int ch, LPSTR pszTxCallerId)

Parameter Description:

ch	Channel Number
pszTxCallerId	• Returns the character string set by SsmSetTxCallerId for SIP channel; • Returns the caller ID string for other channels.

Return Value:

-1	Failed
≥0	The total character number of pszTxCallerId

Function Description:

Obtains the Caller ID which is set by the local end.

Note:

- This function only supports SS1 channel, TUP channel, ISUP channel, ISDN channel and IP channel.

Related Information:

Driver version	SynCTI Ver. 3.0 or above
Header	shpa3api.h
Library	shp_a3.lib

DLL	shp_a3.dll

Related Function: SsmSetTxCallerId

2.3.1.2.8 SsmSetTxOriginalCallerID

Sets the redirecting number or original CalleeID in the call setup message.

Format:

int SsmSetTxOriginalCallerID(int ch, BYTE* pszTxCallerId)

Parameter Description:

ch	Channel Number
pszTxCallerId	Pointer pointing to the string storing the original CalleeID (for ISUP or TUP protocol) or redirecting number (for ISDN protocol). The maximum total character number is 16.

Return Value:

-1	Failed
>=0	The actual length of redirecting number/CalleeID denoted by pszTxCallerId

Function Description:

During outgoing call, sets the redirecting number (for ISDN protocol) or original CalleeID (for ISUP or TUP protocol) in the call setup message sent to the remote PBX.

Note:

- This function must be called before the function SsmAutoDial or SsmAutoDialEx. After the driver has finished establishing the outgoing call setup message, this buffer area will be cleared.
- For the TUP channel, if this function is called before SsmAutoDial or SsmAutoDialEx are invoked, the driver only uses IAI message to initiate the outgoing call. This function sets the original callee address of the original callee address field in the IAI message and the configuration item OriginalCalleeAddrInd sets the original callee address indicator.
- This function only supports the TUP channel, ISUP channel and ISDN channel.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmAutoDial, SsmAutoDialEx

2.3.1.2.9 SsmSetTxRedirectingNum

Sets the redirecting number in the call setup message.

Format:

int SsmSetTxRedirectingNum(int ch, BYTE* pszTxredirectingNum)

Parameter Description:

ch	Channel Number
pszTxredirectingNum	Pointer which points to the string storing the redirecting number. The maximum total character number is 16.

Return Value:

-1	Failed
≥0	The actual length of redirecting number denoted by pszTxredirectingNum

Function Description:

Sets the redirecting number in the call setup message sent to the remote PBX during an outgoing call.

Note:

- This function must be invoked before the call of the function SsmAutoDial or SsmAutoDialEx.
- The values of the parameters once set will be saved in the interior buffer of the driver and will be used in subsequent IAM messages.
- The parameters of this function will go invalid when the program is restarted and the configuration file is reloaded.
- How to cancel: set the second parameter of this function to a null string, that is, SsmSetTxRedirectingNum(CurCh, (LPBYTE)""). You can only cancel the number set by this function.
- For the ISUP channel, if this function is called before SsmAutoDial or SsmAutoDialEx, the driver only uses IAM message to initiate the outgoing call. This function only sets the redirecting number in the IAM message; other parameters like address indicator can be set by the configuration item DefaultIAM_RedirectingNumber.
- This function only supports the ISUP channel now.

Related Information:

Driver version	SynCTI Ver. 5.1.0.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmAutoDial, SsmAutoDialEx

2.3.1.2.10SsmSetWaitAutoDialAnswerTime

Sets the maximum time waiting for the called party to pick up the call.

Format:

BOOL SsmSetWaitAutoDialAnswerTime(WORD wSeconds)

Parameter Description:

wSeconds	Wait time (second)

Return Value:

FALSE	Failed
TRUE	Successful

Function Description:

Sets the maximum time waiting for the called party to pick up the call in an outgoing call.

Note:

- The configuration item MaxWaitAutoDialAnswerTime can implement the same feature.
- This function only supports the SS1 channel, TUP channel, ISUP channel, ISDN channel and analog trunk channel.

Related Information:

Driver version	SynCTI Ver. 3.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmGetWaitAutoDialAnswerTime

2.3.1.2.11SsmSetWaitAutoDialAnswerTimeEx

Sets the maximum time waiting for the called party to pick up the call on a designated channel.

Format:

Int WINAPI SsmSetWaitAutoDialAnswerTimeEx(int ch, int nSeconds)

Parameter Description:

ch	Channel number
nSeconds	Wait time (second)

Return Value:

-1	Failed
0	Successful

Function Description:

Sets the maximum time waiting for the called party to pick up the call on a designated channel during an outgoing call.

Note:

- The parameter nSeconds must be larger than or equal to 0, otherwise the system will prompt error. When nSeconds=0, the configuration item MaxWaitAutoDialAnswerTime or the function SsmSetWaitAutoDialAnswerTime works; when nSeconds>0, this function SsmSetWaitAutoDialAnswerTimeEx gets valid.
- This function supports the SS1 channel, TUP channel, ISUP channel, ISDN channel, IP channel and analog trunk channel, but it is valid only when invoked before the function SsmAutoDial.

Related Information:

Driver version	SynCTI Ver. 5.3.1.1 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related function: SsmSetWaitAutoDialAnswerTime

2.3.1.2.12SsmGetWaitAutoDialAnswerTime

Obtains the preset maximum time waiting for the called party to pick up the call.

Format:

BOOL SsmGetWaitAutoDialAnswerTime(WORD* pwSeconds)

Parameter Description:

pwSeconds	Returns the wait time (second)

Return Value:

FALSE	Failed
TRUE	Successful

Function Description:

Obtains the preset maximum time waiting for the called party to pick up the call during an outgoing call.

Note:

- This function only supports the SS1 channel, TUP channel, ISUP channel and ISDN channel.

Related Information:

Driver version	SynCTI Ver. 3.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmSetWaitAutoDialAnswerTime

2.3.1.2.13SsmGetKB

Obtains the called party's state.

Format:

int SsmGetKB(int ch)

Parameter Description:

ch	Channel Number

Return Value:

-1	Failed
≥0	The detailed meanings of the called party's state is related with the channel type and call direction: ➢ Outgoing call: ✧ SS1 Channel: Returns the value of the KB signal which is sent from the remote end to local end. For detailed value and its meanings, refers to the description of function SsmSetKB. ✧ TUP Channel: Returns the message indicator field (it includes 8 bits) in the ACM message received from the remote PBX. For more information, refer to TUP protocol related documents. ✧ ISUP Channel: Returns the lower 8 bits of the backward call indicator field (it includes 2 bytes) in the ACM message received from the remote PBX, for more information, refer to ISUP protocol related documents. ✧ ISDN Channel: Not supported. ➢ Incoming call: Returns the parameter value of the KB signal when the application calls the function SsmSetKB, for more information, refer to the description of the function SsmSetKB.

Function Description:

Obtains the called party's state.

Note:

- For outgoing call, it's recommended to invoke this function when the channel transfers to the state of S_CALL_WAIT_REMOTE_PICKUP

Related Information:

Driver version	SynCTI Ver. 2.0 or above

Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmSetKB

2.3.1.3 SS1 Channel Functions

2.3.1.3.1 SsmSetKA

Sets the 'Calling Party Category' signal during an outgoing call.

Format:

int SsmSetKA(int ch, BYTE btSigKA)

Parameter Description:

<table>
<tr><td>ch</td><td>Channel Number</td></tr>
<tr><td>btSigKA</td><td>KA signal, range of value: 1≤btSigKA≤10, see below for the details:
<table>
<tr><td rowspan="3">KA Code</td><td colspan="3">KA signal's content (including KOA)</td></tr>
<tr><td colspan="2">Local office of step-by-step system</td><td colspan="2">Crossbar and SPC local office (including quasi-electronic telephone switching system)</td><td colspan="2"></td></tr>
<tr><td colspan="2">KA</td><td colspan="2">KA</td><td colspan="2">KOA</td></tr>
<tr><td>1</td><td rowspan="3">Common</td><td>Period</td><td rowspan="3">Common</td><td>Period</td><td rowspan="3">Common</td><td>Period</td></tr>
<tr><td>2</td><td>User table, immediate</td><td>User table, immediate</td><td>User table, immediate</td></tr>
<tr><td>3</td><td>Printer, immediate</td><td>Printer, immediate</td><td>Printer, immediate</td></tr>
<tr><td>4</td><td colspan="2">Reserved</td><td colspan="2">Reserved</td><td colspan="2">Reserved</td></tr>
<tr><td>5</td><td colspan="2">Common, no charge</td><td colspan="2">Common, no charge</td><td colspan="2">Common, no charge</td></tr>
<tr><td>6</td><td colspan="2">Reserved</td><td colspan="2">Reserved</td><td colspan="2">Reserved</td></tr>
<tr><td>7</td><td colspan="2">Reserved</td><td colspan="2">Reserved</td><td colspan="2">Reserved</td></tr>
<tr><td>8</td><td colspan="2">Reserved</td><td colspan="2">Priority, period</td><td colspan="2">Priority, period</td></tr>
<tr><td>9</td><td colspan="2">suburb call with right automatically, toll without right automatically</td><td colspan="2">Reserved</td><td colspan="2">Reserved</td></tr>
<tr><td>10</td><td colspan="2">toll suburb call without right automatically</td><td colspan="2">Priority, no charge</td><td colspan="2">Priority, no charge</td></tr>
<tr><td>11</td><td colspan="2" rowspan="5"></td><td colspan="4" rowspan="2">Reserved</td></tr>
<tr><td>12</td></tr>
<tr><td>13</td><td colspan="4">Plan for test</td></tr>
<tr><td>14</td><td colspan="4">Reserved</td></tr>
<tr><td>15</td><td colspan="4">Reserved</td></tr>
</table></td></tr>
</table>

Return Value:

-1	Failed
0	Successful

Function Description:

During outgoing call, sets the KA signal value (calling party category) of R2 signaling. For more information, refer to 'China SS1 State Machine' in Chapter 1.

Note:

- This function only supports the SS1 channel of the SHD Series board;
- This function needs to be called before the function SsmAutoDial or SsmAutoDialEx;

- The configuration item MfcKA can implement the same feature.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmGetKA

2.3.1.3.2 SsmSetKD

During the outgoing call, sets the KD signal value.

Format:

int SsmSetKD(int ch, BYTE btSigKD)

Parameter Description:

ch	Channel Number
btSigKD	Code value of the KD signal and its range of value is listed below:

KD Code	Description
1	Toll operator semi-auto call
2	Toll auto call (phone communication or user fax, user data communication)
3	Local call (user calls semi-auto operator and international semi-auto operator)
4	Local user fax or user data communication and user with priority
5	Semi-auto callerID verification
6	Test call

Return Value:

-1	Failed
0	Successful

Function Description:

During outgoing call, sets the KD signal value (transmit end service nature). For more information, refer to 'China SS1 State Machine' in Chapter 1.

Note:

- This function only supports the SS1 channel of the SHD Series board;
- This function needs to be called before the function SsmAutoDial or SsmAutoDialEx;
- The configuration item MfcKD can implement the same feature.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmGetKD

2.3.2 Incoming Call Functions

2.3.2.1 Obtaining Calling Party Information

2.3.2.1.1 SsmChkOpCallerId

Queries whether a designated channel supports receiving the callerID information.

Format:

int SsmChkOpCallerId (int ch)

Parameter Description:

ch	Channel Number

Return Value:

-1	Failed
0	The designated channel doesn't support receiving the callerID information
1	The designated channel supports receiving the callerID information

Function Description:

Queries whether a designated channel supports receiving the callerID information.

Note:

- Station channel and magnet channel doesn't support this operation.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmGetCallerId

2.3.2.1.2 SsmGetCallerId

Refer to SsmGetCallerIdEx

2.3.2.1.3 SsmGetCallerIdA

Refer to SsmGetCallerIdEx

2.3.2.1.4 SsmGetCallerIdEx

Obtains the callerID information of the incoming call. SsmGetCallerId and SsmGetCallerIdA obtain the callerID information stored in the basic buffer area; SsmGetCallerIdEx obtains the callerID information stored in the extension buffer area and it's only applicable to analog trunk channel and analog trunk recording channel.

Format:

int SsmGetCallerId(int ch, LPSTR szCallerId)

int SsmGetCallerIdEx(int ch, LPSTR szCallerIdEx)

char* SsmGetCallerIdA(int ch)

Parameter Description:

ch	Channel Number
szCallerId	String pointer pointing to the buffer storing basic callerID information. The storage space is allocated by the application and should be no less than 255 bytes.
szCallerIdEx	String pointer pointing to the buffer storing extentional callerID information. The storage space is allocated by the application and should be no less than 255 bytes. **Note: For analog trunk channel and analog trunk recording channel, the original data of the callerID stored in szCallerIdEx is in FSK format, for more information, refer to 'Caller ID on Analog Phone Line' in Chapter 1.**

Return Value:

Function Name	Return Value	Description
SsmGetCallerId SsmGetCallerIdEx	-1	Failed
	≥0	The length of the callerID information
SsmGetCallerIdA	NULL	Failed
	Other	String pointer pointing to the buffer storing the basic callerID information in the driver

Function Description:

During incoming call, obtains the callerID information stored in the internal buffer area.

For SS1 channel, if the configuration item MfcR2ToRxCallerIdBuf is set to 1, SsmGetCallerIdEx can obtain the R2 signal which is sent from the remote PBX and stored in the callerID extension information buffer of the driver. For more information, refer to 'China SS1 State Machine' in Chapter 1.

➢ **IP Channel**

1) For a SIP channel, the CallerID obtained by this function is SIP URI (the From field in SIP message), shown in the format: [sip:/sips:]user[@host[:port]].

Note:

- Station channel and magnet channel doesn't support this operation.

Related Information:

Driver version	SynCTI Ver.4.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmClearCallerId, SsmClearCallerIdEx

2.3.2.1.5 SsmGetCallerName

Obtains the name string in the FSK caller information on the analog phone line.

Format:

int SsmGetCallerName(int ch, LPSTR pszCallerName)

Parameter Description:

ch	Channel Number
pszCallerName	Pointer which points to the buffer storing the name string. The buffer space is allocated by the application and should be no less than 50 characters.

Return Value:

-1	Failed
≥0	Returns the total number of characters in pszCallerName

Function Description:

Obtains the name string in the FSK caller information on the analog phone line. For more information, refer to 'Caller ID on Analog Phone Line' in Chapter 1.

Note:

- This function only supports the SHT Series analog trunk channel and ATP Series analog recording channel.

Related Information:

Driver version	SynCTI Ver. 3.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmGetCallerId, SsmGetCallerIdA, SsmGetCallerIdEx

2.3.2.1.6 SsmClearCallerId

Refer to SsmClearCallerIdEx

2.3.2.1.7 SsmClearCallerIdEx

Clears the buffer area storing callerID information in the driver.

Format:

int SsmClearCallerId (int ch)

int SsmClearCallerIdEx(int ch)

Parameter Description:

ch	Channel Number

Return Value:

-1	Failed
0	Successful

Function Description:

Clears the buffer area storing callerID information in the driver. SsmClearCallerId clears the basic buffer area and SsmClearCallerIdEx clears the extension buffer area.

Note:

- This function supports all but ISDN and IP channels on SHD, ATP, DTP and DST series boards and the analog trunk channels on SHT series boards.

Related Information:

Driver version	SynCTI Ver. 3.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmGetCallerId, SsmGetCallerIdA, SsmGetCallerIdEx

2.3.2.1.8 SsmGetKA

During incoming call, obtains the calling party category.

Format:

int SsmGetKA(int ch)

Parameter Description:

ch	Channel Number

Return Value:

-1	Failed
≥0	Returns the 'calling party category'. The detailed meanings are associated with channel types. ✧ SS1 channel: If it's an incoming call, the return value is the KA signal value. For more information, refer to the description of function SsmSetKA. If it's an outgoing call, the returned value represents the 'calling party category' which is sent from the local end to the remote end during the call progress. ✧ ISUP channel: During the incoming call, the return value represents the parameter value of 'calling party category' in the IAM message sent from the remote PBX.

Return Value	Description
9	National operator (for domestic use)
10	Ordinary subscriber, it's used for toll to toll and toll to local office calls
11	Calling subscriber with priority
12	Data call (voice with data)
13	Test call
14	Payphone
15	Ordinary calling subscriber, it's used for local office to local office calls

✧ TUP Channel: During incoming call, the return value represents the parameter value of 'calling party category' in the IAI/IAM message sent from the remote PBX.

Function Description:

During incoming call, obtains the calling party category.

Note:

- This function only supports SS1 channel, TUP channel and ISUP channel;
- It's recommended that the application invokes this function when the channel state is transferred to S_CALL_RINGING.

Related Information:

Driver version	SynCTI Ver. 4.7.1.5 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmSetKA, SsmGetCallerId, SsmGetChState

2.3.2.1.9 SsmGetKD

During incoming call, obtains the service nature of the original end.

Format:

int SsmGetKD(int ch)

Parameter Description:

ch	Channel number

Return Value:

-1	Failed
≥0	Returns the value of the KD signal sent from the calling party. For more detailed information, refer to the description of function SsmSetKD.

Function Description:

Obtains the KD signal value (i.e. the service nature of the initiator) from the calling party during an incoming call.

Note:

- This function only supports SS1 channels of SHD Series boards and the called party's channels in the SpyCic of DTP Series boards.
- It's recommended that the application invokes this function when the channel state is transferred to S_CALL_RINGING.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmGetChState

2.3.2.2 Obtaining Called Party Information

2.3.2.2.1 SsmGetPhoNumStr

Refer to SsmGetPhoNumLen

2.3.2.2.2 SsmGetPhoNumStrA

Refer to SsmGetPhoNumLen

2.3.2.2.3 SsmGetPhoNumLen

During incoming call, obtains the called party's phone number information. SsmGetPhoNumLen gets the length of the CalleeID; SsmGetPhoNumStr and SsmGetPhoNumStrA get the detailed CalleeID.

Format:

int SsmGetPhoNumLen(int ch)

int SsmGetPhoNumStr(int ch, LPSTR pszPhoNum)

char* SsmGetPhoNumStrA(int ch)

Parameter Description:

ch	Channel Number
pszPhoNum	The pointer which points to the buffer storing the ASCII formatted CalleeID. The storage space is allocated by the application and should be no less than 50 characters.

Return Value:

Function Name	Return Value	Description
SsmGetPhoNumStr	-1	Failed
SsmGetPhoNumLen	≥0	The length of the CalleeID.
SsmGetPhoNumStrA	NULL	Failed
	Other	Returns the pointer which points to the buffer storing the CalleeID in the driver.

Function Description:

During incoming call, the remote PBX sends the call setup message to the local end. Upon receiving the message, the driver separates the CalleeID from the message and stores it in the buffer. This function is applicable to obtain the CalleeID in the buffer.

- **IP Channel**

1) For a SIP channel, the CalleeID obtained by this function is SIP URI (the To field in SIP message), shown in the format: [“displayname“] [sip:/sips:]user[@host[:port]].

Note:

- SsmGetPhoNumLen , SsmGetPhoNumStr and SsmGetPhoNumStrA are only applicable to incoming calls on the SS1, TUP, ISUP, ISDN and IP channel.
- SsmGetPhoNumStrA is only applicable to incoming calls on the SS1, TUP, ISUP and ISDN channel, inapplicable to those on the IP channel.
- The length of the calleeID buffer in the driver is 50 characters. If the length of the received CalleeID exceeds 50, the redundant number will be discarded.

Related Information:

Driver version	SynCTI Ver. 3.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Event: E_CHG_RxPhoNumBuf

2.3.2.3 Obtaining 1st Called Party Number Information

2.3.2.3.1 SsmGet1stPhoNumLen

Refer to SsmGet1stPhoNumStrA

2.3.2.3.2 SsmGet1stPhoNumStr

Refer to SsmGet1stPhoNumStrA

2.3.2.3.3 SsmGet1stPhoNumStrA

Obtains relative information from the call setup message such as the redirecting number information for ISDN channel, the original called party number information for TUP channel, and the original called party number/the redirecting number information for ISUP channel. SsmGet1stPhoNumLen gets the length of the information; SsmGet1stPhoNumStr and SsmGet1stPhoNumStrA get the information itself.

Format:

int SsmGet1stPhoNumStr(int ch, LPSTR pszPhoNum)

char *SsmGet1stPhoNumStrA(int ch)

int SsmGet1stPhoNumLen(int ch)

Parameter Description:

ch	Channel Number
pszPhoNum	Pointer which points to the buffer area storing phone number strings. The storage space is allocated by the application and should be no less than 50 characters.

Return Value:

Function Name	Return Value	Description
SsmGet1stPhoNumStr SsmGet1stPhoNumLen	-1	Failed
	≥0	The length of the phone number string
SsmGet1stPhoNumStrA	NULL	Failed
	Other	Returns the pointer which points to the buffer storing the phone number in the driver.

Function Description:

During incoming call, obtains the called number information in the call setup message.

- **ISDN Channel**

Obtains the redirecting number information from the SETUP message.

- **TUP Channel**

Obtains the original called party number information from the IAI or IAM message.

- **ISUP Channel**

Obtains the original called party number information or the redirecting number information from the IAM message. The configuration item SaveRGNTo1stPhoNumStr determines the type of the return value:

- SaveRGNTo1stPhoNumStr is set to be 0: If there is an original called number field in the IAM message, it returns original called party number information, otherwise, it returns NULL;
- SaveRGNTo1stPhoNumStr is set to be 1: If there is only the redirecting number field in the IAM message, this function returns redirecting number information. If there is only the original called number field, it returns original called number information. If there are both, it returns the information contained in the latter field of the message. For example, if these two fields are sequenced ‘redirecting number field | original called number’, it returns the original called number information; If they are sequenced ‘original called number | redirecting number field’, it returns the redirecting number information.

If it is required to obtain the redirecting number information and the original called number information separately from the IAM message of the ISUP channel, use the function SsmGetIsupParameter.

Note:

- The length of the buffer in the driver is 50 characters. If the length of the received number exceeds 50, the redundant number will be discarded.

Related Information:

Driver version	SynCTI Ver.3.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmGetChState

2.3.2.4 Setting Parameters for Incoming Call Progress

2.3.2.4.1 SsmEnableAutoSendKB

Sets the 'auto-receive inbound call' feature.

Format:

int SsmEnableAutoSendKB(int ch, BOOL bEnable)

Parameter Description:

ch	Channel Number
bEnable	=TRUE: Enable =FALSE: Disable

Return Value:

-1	Failed
0	Successful

Function Description:

Sets the 'auto-receive inbound call' feature.

Note:

- This function is only applicable to SHD Series boards.
- For SS1 inbound trunk channel, the parameters set by this function can also be set by the configuration item AutoSendKB. For more information, refer to 'China SS1 State Machine' in Chapter 1.
- For ISUP/TUP channel, the parameters set by this function can also be set by the configuration item AutoSendACM. For more information, refer to 'ISUP Channel State Machine' or 'TUP Channel State Machine' in Chapter 1.
- For ISDN channel, the parameters set by this function can also be set by the configuration item UserSideAutoSendAck (user side) or NetSideAutoSendAck (network side). For more information, refer to 'ISUP Channel State Machine' in Chapter 1.

Related Information:

Driver version	SynCTI Ver. 3.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmSetKB, SsmGetAutoSendKBFlag

2.3.2.4.2 SsmGetAutoSendKBFlag

Obtains the flag of the 'auto-receive incoming call' feature.

Format:

int SsmGetAutoSendKBFlag(int ch)

Parameter Description:

ch	Channel Number

Return Value:

-1	Failed
0	Disabled
1	Enabled

Function Description:

Obtains the flag of the 'auto-receive incoming call' feature.

Note:

- This function is only applicable to SHD Series boards.

Related Information:

Driver version	SynCTI Ver. 4.7.1.5 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmEnableAutoSendKB

2.3.2.4.3 SsmSetKB

During incoming call, sends backward answer message to the remote PBX and sets the called party's state.

Format:

int SsmSetKB(int ch, BYTE btSigKB)

Parameter Description:

<table>
<tr><td>ch</td><td>Channel Number</td></tr>
<tr><td>btSigKB</td><td>The called party state indicator. The detailed setting is related with the channel type.
➢ SS1 channel: btSigKB indicates the backward KB signal sent to remote PBX. Below are the KB values:
<table>
<tr><th rowspan="2">btSigKB</th><th colspan="2">KB</th></tr>
<tr><th>Received KD=1,2 or 6</th><th>Received KD=3 or 4</th></tr>
<tr><td>1</td><td>Called subscriber free</td><td>Called subscriber free
Circuit reset: independent control</td></tr>
<tr><td>2</td><td>Called subscriber 'local busy'</td><td>Reserved</td></tr>
<tr><td>3</td><td>Called subscriber 'toll busy'</td><td></td></tr>
<tr><td>4</td><td>Keys congestion</td><td>Called subscriber busy or keys congestion</td></tr>
<tr><td>5</td><td>Unallocated CalleeID</td><td>Unallocated CalleeID</td></tr>
<tr><td>6</td><td>Reserved</td><td>Called subscriber free
Circuit reset: Calling party control</td></tr>
</table>
➢ TUP channel: This parameter represents the TUP messages sent by the driver:
<table>
<tr><th>btSigKB</th><th>Meanings</th><th>Message Sent by the Driver</th></tr>
<tr><td>1</td><td>Called subscriber free, charge</td><td>ACM</td></tr>
<tr><td>2</td><td>Called subscriber 'local busy'</td><td>SLB</td></tr>
</table></td></tr>
</table>

3	Called subscriber 'toll busy'	STB
4	Keys congestion	SEC
5	Unallocated CalleeID	UNN
6	Reserved	
7	Called subscriber free, no charge	ACM
8	Request callerID	GRQ (The state will be transferred to 'waiting GSM')
9	Switching Equipment Congestion signal	SEC
10	Circuit Group Congestion signal	CGC
11	Address incomplete signal	ADI
12	Call failure signal	CFL
13	Line out-of-service signal	LOS
14	Send special information tone signal	SST
15	Access barred signal	ACB
16	Digital path not provided signal	DPN
17	Busy	SSB

- ISUP channel: This parameter represents the ISUP messages sent by the driver:

btSigKB	Meanings	Driver Operation
1/6	Called subscriber free, charge	Send ACM Message, accept incoming call
2	Called subscriber busy	Send REL message, reject incoming call
3	Incomplete address	Send REL message, reject incoming call
4	Call rejection	Send REL message, reject incoming call
5	No answer	Send REL message, reject incoming call
7	Called subscriber free, no charge	Send ACM Message, accept incoming call
8	Request callerID	Send INR message, transfer the state to 'waiting INF'.
9	User absent	Send REL message, reject incoming call.

- ISDN channel: This parameter represents the ISDN messages sent by the driver:

btSigKB	Meanings	Driver Operation
1	Called subscriber free	Send ALERT message
2	Called subscriber busy	Send DISCONNECT message, reason=Called subscriber busy
4	Call rejection	Send DISCONNECT message, reason=Call rejection
5	No answer	Send DISCONNECT message, reason=No answer from Called party
Other	Reserved	

- SIP channel: This parameter represents the SIP messages sent by the driver:

btSigKB	Meanings	Driver Operation
1	Called subscriber free	Send 180 message
2	Called subscriber busy	Send 486 message, reason=Called subscriber busy
Other	Reserved	

Return Value:

-1	Failed (Note: For the ISUP channel, it returns -1 or 0 if the call is failed)
0	Successful (Note: For the ISUP channel, it returns 1 if the call is successful)

Function Description:

During an incoming call, after the driver finishes receiving such information as calleeID based on the preset number receiving rule, if the application disables the feature 'Auto-reception of incoming call' via the function SsmEnableAutoSendKB or the configuration item AutoSendACM, the channel state will be transferred to S_CALL_PENDING and the driver will give the control right back to the application. The application then sends the backward answer message to the remote PBX and sets the called party's state. The application can call this function to accept or reject the current incoming call.

For more information, refer to related part in Chapter 1 according to the channel type.

- SS1 channel: 'China SS1 State Machine '

- ISUP channel: 'ISUP Channel State Machine'
- TUP channel: 'TUP Channel State Machine'
- ISDN channel: 'ISDN Channel State Machine'

Note:

- For ISUP channels, if the parameter bigSigKB is set to be 1, 6, or 7, the driver will send ACM message to the remote PBX. The ACM message is constructed by the following method: If, before this function is invoked, the function SsmSetIsupUPPara (with parameter C_ISUP_ACM) is called to submit a self-constructed ACM message to the driver, this ACM message will be used; otherwise, the driver will construct an ACM message automatically. For more information, refer to 'ISUP Channel State Machine' in Chapter 1.
- For ISDN channels, SsmSetCharge can be used to set whether a call is charged or free of charge.

Related Information:

Driver version	SynCTI Ver. 4.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmEnableAutoSendKB

2.3.3 Channel State Transition Events

2.3.3.1 SsmGetChState

Obtains the current state of the channel.

Format:

int SsmGetChState(int ch)

Parameter Description:

ch	Channel Number

Return Value:

The return value -1 indicates call failed. Below are the meanings of other values.

Value	MACRO in shpa3api.h	State Description
0	S_CALL_STANDBY S_FAX_Wait	idle
1	S_CALL_PICKUPED	off-hook
2	S_CALL_RINGING	ringing
3	S_CALL_TALKING	talking
4	S_CALL_ANALOG_WAITDIALTONE	Analog trunk channel: outgoing call, wait for dialing tone
5	S_CALL_ANALOG_TXPHONUM	Analog trunk channel: outgoing call, dialing
6	S_CALL_ANALOG_WAITDIALRESULT	Analog trunk channel: outgoing call, wait for dialing result.
7	S_CALL_PENDING	pending . The pending cause can be got via function SsmGetPendingReason
8	S_CALL_OFFLINE	off-line
9	S_CALL_WAIT_REMOTE_PICKUP	Outgoing call, wait answer, wait called subscriber pickup
10	S_CALL_ANALOG_CLEAR	Analog trunk channel: internal state
11	S_CALL_UNAVAILABLE	channel unusable
12	S_CALL_LOCKED	outgoing call locked
19	S_CALL_RemoteBlock	blocked by remote
20	S_CALL_LocalBlock	blocked locally
30	S_CALL_Ss1InWaitPhoNum	SS1 Channel: receive called subscriber number
31	S_CALL_Ss1InWaitFwdStop	SS1 Channel: wait remote PBX to stop sending forward signal
32	S_CALL_Ss1InWaitCallerID	SS1 Channel: receive Caller ID

33	S_CALL_Ss1InWaitKD	SS1 Channel: receive KD signal
34	S_CALL_Ss1InWaitKDStop	SS1 Channel: wait remote PBX to stop sending KD signal
35	S_CALL_SS1_SAYIDLE	SS1 Channel: send idle signal to remote PBX
36	S_CALL_SS1WaitIdleCAS	SS1 Channel: wait idle signal from remote PBX
37	S_CALL_SS1PhoNumHoldup	SS1 Channel: phone number hold-up
38	S_CALL_Ss1InWaitStopSendA3p	SS1 Channel: wait remote PBX to stop sending A3 pulse signal
40	S_CALL_Ss1OutWaitBwdAck	SS1 Channel: wait remote PBX to answer seizure confirmation signal
41	S_CALL_Ss1OutTxPhoNum	SS1 Channel: send CalleeID
42	S_CALL_Ss1OutWaitAppendPhoNum	SS1 Channel: wait application to append phone number
43	S_CALL_Ss1OutTxCallerID	SS1 Channel: send CallerID
44	S_CALL_Ss1OutWaitKB	SS1 Channel: wait KB signal from remote PBX
45	S_CALL_Ss1OutDetectA3p	SS1 Channel: wait A3 pulse signal from remote PBX
50	S_FAX_ROUND	FAX channel: state transition is in progress
51	S_FAX_PhaseA	FAX channel: fax call setup (Phase A)
52	S_FAX_PhaseB	FAX channel: handling before fax message transmission (Phase B)
53	S_FAX_SendDCS	FAX channel: send DCS signal to the receiver during transmission
54	S_FAX_Train	FAX channel: train before fax message transmission
55	S_FAX_PhaseC	FAX channel: fax message transmitting (Phase C)
56	S_FAX_PhaseD	FAX channel: handling after fax message transmission (Phase D)
57	S_FAX_NextPage	FAX channel: transmit next page
58	S_FAX_AllSent	FAX channel: fax message transmission is completed
59	S_FAX_PhaseE	FAX channel: fax call is released (Phase E)
60	S_FAX_Reset	FAX channel: reset MODEM
61	S_FAX_Init	FAX channel: initialize MODEM
62	S_FAX_RcvDCS	FAX channel: Receiving fax, receive DCS signal from sender
63	S_FAX_SendFTT	FAX channel: Receiving fax, send FTT signal indicating training failure
64	S_FAX_SendCFR	FAX channel: Receiving fax, send CFR signal confirming the request is acceptable
65	S_FAX_SendPPS	FAX channel: Fax transmission. Successive fax negotiation is undergone in the ECM mode.
66	S_FAX_RcvPPR	FAX channel: Receive PPR signal after sending PPS signal.
67	S_FAX_RepeatECMPage	FAX channel: Fax transmission. Resend fax data in the ECM mode.
68	S_FAX_CTC_CTR	FAX channel: Positive negotiation is undergone in the ECM mode after 4 times data resending.
69	S_FAX_SendPPR	FAX channel: The sender is required to resend the fax data in the ECM mode.
70	S_TUP_WaitPcmReset	TUP channel: wait circuit group to reset
71	S_TUP_WaitSAM	TUP channel: wait for subsequent address message of remote PBX
72	S_TUP_WaitGSM	TUP channel: wait for GSM message from remote PBX
73	S_TUP_WaitCLF	TUP channel: wait for disconnect message from remote PBX
74	S_TUP_WaitPrefix	TUP channel: inbound office prefix
75	S_TUP_WaitDialAnswer	TUP channel: wait for message from remote PBX
76	S_TUP_WaitRLG	TUP channel: wait for release-guard signal from remote PBX
77	S_TUP_WaitSetCallerID	TUP channel: wait for the application to set callerID
81	S_ISDN_OUT_WAIT_NET_RESPONSE	ISDN channel: wait for network response
82	S_ISDN_OUT_PLS_APPEND_NO	ISDN channel: wait for the application to append the number
83	S_ISDN_IN_CHK_CALL_IN	ISDN channel: incoming call is detected
84	S_ISDN_IN_RCVING_NO	ISDN channel: number receiving
85	S_ISDN_IN_WAIT_TALK	ISDN channel: ready for talk
86	S_ISDN_OUT_WAIT_ALERT	ISDN channel: wait for the alerting signal from remote end
87	S_ISDN_CALL_BEGIN	ISDN channel: originate an outgoing call or detect an incoming call
88	S_ISDN_WAIT_HUANGUP	ISDN channel: wait for hangup
89	S_ISDN_IN_CALL_PROCEEDING	ISDN channel: call proceeding
100	S_CALL_SENDRING	Magnetic channel: send ring
105	S_SPY_RCVPHONUM	Monitoring channel: receive number
110	S_SPY_SS1RESET	SS1 channel: reset
111	S_SPY_SS1WAITBWDACK	SS1 channel: waiting for the backward acknowledgement
112	S_SPY_SS1WAITKB	SS1 channel: waiting for the KB signal
120	S_ISUP_WaitSAM	ISUP channel: wait for SAM message from remote PBX
121	S_ISUP_WaitRLC	ISUP channel: wait for release complete signal from remote PBX

122	S_ISUP_WaitReset	ISUP channel: wait for circuit to reset
123	S_ISUP_LocallyBlocked	ISUP channel: locally blocked
124	S_ISUP_RemotelyBlocked	ISUP channel: remotely blocked
125	S_ISUP_WaitDialAnswer	ISUP channel: wait for message from remote PBX
126	S_ISUP_WaitINF	ISUP channel: wait for INF message from remote PBX
127	S_ISUP_WaitSetCallerID	ISUP channel: wait for the application to set caller ID
128	S_DTRC_ACTIVE	DTR channel: monitored voice channel is in active state
129	S_ISUP_Suspend	ISUP channel: suspended
130	S_CALL_EM_TXPHONUM	E/M channel: dial or the voice alteration resource is used
131	S_CALL_EM_WaitIdleCAS	E/M channel: wait for the idle signal from remote end
132	S_CALL_VOIP_DIALING	IP channel: VoIP calling party is dialing
133	S_CALL_VOIP_WAIT_CONNECTED	IP channel: VoIP called party picks up the phone and waits for the channel to enter talking state
134	S_CALL_VOIP_CHANNEL_UNUSABLE	IP channel: IP channel is unusable now
135	S_CALL_DISCONECT	USB is disconnected
136	S_CALL_SS1WaitFlashEnd	SS1 channel: wait for the end of flash
137	S_CALL_FlashEnd	SS1 channel: flash ends
139	S_CALL_SIGNAL_ERROR	DTR channel: frame synchronization normal but signal incomplete
140	S_CALL_FRAME_ERROR	DTR channel: frame synchronization abnormal but signal complete
150	S_CALL_VOIP_SESSION_PROCEEDING	IP channel: In session, it is a state when 183 received.
151	S_CALL_VOIP_REG_ING	IP channel: registering a SIP channel
152	S_CALL_VOIP_REG_FAILED	IP channel: fail to register a SIP channel
153	S_CALL_VOIP_CALL_ON_HOLD	IP channel: call on hold state of SIP channel
160	S_IP_MEDIA_LOCK	MEDIA channel: locked
161	S_IP_MEDIA_OPEN	MEDIA channel: unlocked
162	S_SPY_RBSWAITACK	RBS monitoring channel: wait for answer
163	S_SPY_RBSSENDACK	RBS monitoring channel: send answer
170	S_IPR_USING	IPR channel: the channel is used
171	S_IPR_COMMUNICATING	IPR channel: the channel is in the communicating state
172	S_ISUP_WaitCOT	ISUP channel: Wait for COT message
300	S_FAX_EOR_ERR	FAX channel: Negative processing is done in the ECM mode after several times fax resending.
301	S_FAX_RNR_RR	FAX channel: The fax receiver is busy in the ECM mode.
302	S_FAX_RTN	FAX channel: Fax reception, receiving denied and retraining.
303	S_FAX_NextPage_EOM	FAX channel: Sending of the next page should start from Phase B, retraining
400	S_FAX_V34_PhaseV8	FAX channel: V.8 training phase in the V.34 fax mode.
other		Reserved

Function Description:

Obtains the current channel state.

Note:

- In order to enhance the application's operation efficiency, it's recommended to use the event of E_CHG_ChState to obtain the information of channel state changes.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmGetChStateKeepTime, SsmGetPendingReason

2.3.3.2 SsmGetChStateKeepTime

Retrieves the duration while the channel keeps in the current state.

Format:

long SsmGetChStateKeepTime(int ch)

Parameter Description:

ch	Channel number

Return Value:

-1	Failed
≥0	The duration (ms) while the channel keeps in the current state

Function Description:

Retrieves the duration while the channel keeps in the current state.

More information can be obtained via the flexible implementation of this function:

- When the incoming call channel stays in the state of S_CALL_RINGING, this function can be used to retrieve the ringing tone duration.
- When the current channel stays in the state of S_CALL_TALKING, this function can be used to retrieve the call duration.
- When the outgoing call channel stays in the state of S_CALL_WAIT_REMOTE_PICKUP, this function can be used to retrieve the time waiting for the called subscriber to pick up the phone.

Note:

- This function does not support VoIP boards.

Related Information:

Driver version	SynCTI Ver.3.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmGetChState

2.3.4 Obtaining Pending Reason

2.3.4.1 SsmGetPendingReason

Obtains the detailed cause for the channel entering the state of S_CALL_PENDING.

Format:

int SsmGetPendingReason(int ch)

Parameter Description:

ch	Channel Number

Return Value:

-1 represents a failed call, below are the meanings of other values.

Return Value	Macro	Applicable Channel Type	Description
0	ANALOGOUT_NO_DIALTONE	Analog trunk channel	Outgoing call failed: no dial tone detected
1	ANALOGOUT_BUSYTONE	Analog trunk channel	Outgoing call failed: busy tone detected
2	ANALOGOUT_ECHO_NOVOICE	Analog trunk channel	Outgoing call: after the ringback tone is detected, the phone line keeps silence. The driver can't determine whether the called subscriber picks up the phone
3	ANALOGOUT_NOANSWER	Analog trunk channel	Outgoing call: after the ringback tone is detected, the called subscriber doesn't answer within the time

			specified in the configuration item MaxWaitAutoDialAnswerTime
4	ANALOGOUT_TALKING_REMOTE_HANGUPED	Analog trunk channel	The channel in the 'connected' state detects that the remote subscriber has hanged up.
5	ANALOGOUT_NOVOICE	Analog trunk channel	Outgoing call: the autodial is finished and no ringback tone or other voice signals have been detected. The driver can't determine whether the called subscriber has hanged up.
10	PEND_WaitBckStpMsg	ISUP / TUP/ ISDN/ SS1	Incoming call: if, after the driver has finished receiving the information such as calleeID based on the preset number receiving rule, the feature of 'automatically receive incoming call' is disabled, the application itself needs to decide whether to accept this call or not.
11	SS1IN_BWD_KB5	SS1	Incoming call: after the KD signal from the remote PBX has been received, the application sets KB=5 (unallocated number) and waits for the release signal from the calling party.
12	PEND_RemoteHangupOnTalking	ISUP / TUP/ SS1	Incoming call: the remote end hangs up the phone while talking.
13	PEND_AutoDialFailed	ISUP / TUP/ ISDN	Outgoing call failed. You can invoke the function SsmGetAutoDialFailureReason() to get the exact reason why this auto dial fails.
14	PEND_SsxUnusable	ISUP / TUP	SS7 signaling is unavailable
15	PEND_CircuitReset	ISUP / TUP	The event of circuit reset occurs
16	PEND_PcmSyncLos	ISUP / TUP SS1	The basic frame (timeslot0) synchronization signal of the digital trunk is lost.
20	SS1OUT_TALKING_REMOTE_HANGUPED	SS1	Outgoing call: the remote end hangs up the phone while talking.
	PEND_CalleeHangupOnTalking	ISUP / TUP	
21	SS1OUT_NOANSWER	SS1	Outgoing call failed: the called subscriber doesn't answer the call within the time specified by the configuration item MaxWaitAutoDialAnswerTime
22	SS1OUT_NOBWDACK	SS1	Outgoing call failed: waiting for the 'Seizure Ack' signal from the remote PBX is overtime
23	SS1OUT_DIALING_BWD_HANGUP	SS1	Outgoing call failed: remote PBX cancels the call.
24	SS1OUT_BWD_A5	SS1	Outgoing call failed: receives the A5 signal (unallocated number signal) from the remote PBX
25	SS1OUT_BWD_KB5	SS1	Outgoing call failed: receives the KB=5 (unallocated number signal) from the remote PBX.
26	SS1OUT_BWD_KB2	SS1	Outgoing call failed: receives KB=2(Called subscriber 'local busy')from the remote PBX
27	SS1OUT_BWD_KB3	SS1	Outgoing call failed: receives KB=3(Called subscriber 'toll busy') from the remote PBX
28	SS1OUT_BWD_A4	SS1	Outgoing call failed: receives A4 signal (keys congestion) from remote PBX
29	SS1OUT_BWD_KB4	SS1	Outgoing call failed: receives KB=4 signal (keys congestion) from remote PBX
30	SS1OUT_TIMEOUT_BWD_A	SS1	Outgoing call failed: the wait for backward group A signals from the remote PBX is time out.
31	SS1OUT_TIMEOUT_BWD_A_STOP	SS1	Outgoing call failed: the wait for the remote PBX to stop sending backward group A signals is time out.
32	SS1OUT_TIMEOUT_BWD_KB	SS1	Outgoing call failed: the wait for KB signal from the remote PBX is time out.
33	SS1OUT_TIMEOUT_BWD_KB_STOP	SS1	Outgoing call failed: the wait for the remote PBX to stop sending KB signal is time out.

34	SS1OUT_TIMEOUT_CALLERID_BWD_A1	SS1	Outgoing call failed: when sending calling party number to the remote PBX, the wait for backward group A signal from the remote PBX is time out.
35	SS1OUT_TIMEOUT_CALLERID_BWD_A1_STOP	SS1	Outgoing call failed: when sending calling party number to the remote PBX, the wait for the remote PBX to stop sending backward group A signals is timeout and the autodial fails.
36	SS1OUT_UNDEFINED_CALLERID_BWD_A	SS1	Outgoing call failed: receive undefined backward group A signal when sending calling party number to the remote PBX,
37	SS1OUT_UNDEFINED_BWD_A		Outgoing call failed: receive undefined backward group A signal
38	SS1OUT_UNDEFINED_BWD_KB		Outgoing call failed: receive the undefined KB signal
41	ISDN_CALLOVER	ISDN	The call is over and the remote end hangs up first.
42	ISDN_WAIT_RELEASE	ISDN	Receives the 'disconnected' message from the remote end and waits for the release of the local end.
43	ISDN_HANGING	ISDN	The local end hangs up first, being disconnecting.
44	ISDN_RELEASING	ISDN	Releasing the call
45	ISDN_UNALLOCATED_NUMBER	ISDN	Unallocated number
46	ISDN_NETWORK_BUSY	ISDN	Network busy
47	ISDN_CIRCUIT_NOT_AVAILABLE	ISDN	Designated circuit is unavailable
48	PEND_CalleeHangupOnWaitRemotePickUp	TUP	Outgoing call failed: receives the 'disconnect' message from the remote PBX while waiting for the called subscriber to pick up.
49	ISUP_HardCircuitBlock	ISUP	Receive the hardware blocking message from the remote PBX
50	ISUP_RemoteSuspend	ISUP	Timer T6 overflows. For more information about the timer T6, refer to the 'ISUP Channel State Machine' in chapter 1.
51	PEND_RcvHGBOrSGB	TUP	Receive the blocking message from the remote PBX(SGB/HGB)
52	ISDN_NO_ANSWER	ISDN	No answer
53	ISDN_CALL_REJ	ISDN	Call rejection
54	PEND_RemoteHangupOnRinging	ISUP / TUP	Incoming call: when the channel is in the 'ringing' state, the remote PBX cancels the call
55	ISDN_NO_ROUTE	ISDN	No route to destination, the cause may be 'the mobile phone is not in the service area'
56	ISDN_NO_ROUTE_TO_DEST	ISDN	No route to the destination, the cause may be 'the mobile phone is power off'
57	EM_USER_BUSY	E/M	User busy
58	EM_CH_ERROR	E/M	Channel error
59	EM_LOCAL_HANGUP	E/M	Local end hangs up first
60	EM_LOCAL_NOANSWER	E/M	No answer at Local end
61	EM_REMOTE_HANGUP	E/M	Remote end hangs up first
62	EM_REMOTE_NOANSWER	E/M	No answer from remote end
63	PEND_RemoteHangupOnSuspend	ISUP	When the channel is in the suspended state, the remote end hangs up
64	PEND_CalleeHangupOnSuspend	ISUP	When the channel is in the suspended state, the called party hangs up
65	ISDN_NORMAL_UNSPEC	ISDN	The call is finished normally
66	ISDN_USER_ABSENCE	ISDN	User absent
67	ISDN_INVALID_NUMBER_FOEMAT	ISDN	Number format invalid

68	ISDN_NO_CIRCUIT_AVAILABLE	ISDN	No circuit/channel available
69	IP_REMOTE_CRASHED	SIP	Remote end crashes
70	IP_REMOTE_CLOSED	SIP	Remote end sends the BYE or CANCEL message
71	IP_DIAL_TIMEOUT	SIP	Call from the VoIP board is time-out
72	IP_REMOTE_REJECTED	SIP	Call from the VoIP board is rejected by the remote end
73	IP_REFER_SECCEED	SIP	Call is pending after it is successfully transferred by the VoIP board
74	IP_REFER_REFUSED	SIP	Call is pending after it is unsuccessfully transferred by the VoIP board
75	IP_STUN_FAILED	SIP	The stun traversal fails in the call from the VoIP board
76	IP_NOTRCV_ACK	SIP	ACK message is not received after the VoIP board picks up the call
77	IP_REDIRECT_FAIL	SIP	Call redirecting via the VoIP board is failed
Others	Reserved		

Function Description:

Obtains the detailed cause for the channel entering the state of S_CALL_PENDING.

Note:

Related Information:

Driver version	SynCTI Ver. 3.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function:

2.3.5 Channel Pickup Functions (CTI Series)

2.3.5.1 SsmPickup

Refer to SsmPickupANX

2.3.5.2 SsmPickupANX

Executes the pickup operation on the channel. SsmPickupANX not only includes all the features of SsmPickup, but also indicates the charge information of the called party in the answer message; however, it is only applicable to the TUP and ISUP channels.

Format:

int SsmPickup(int ch)

int SsmPickupANX(int ch, int nANX)

Parameter Description:

ch	Channel Number
nANX	=0: No indication =1: Charge =2: Free of charge

Return Value:

-1	Failed
0	Successful

Function Description:

Executes the pickup operation on the channel.

✧ **ISUP channel:**

For ISUP channel, when the application calls this function, if the channel is in the 'ringing' state, based on the settings of the configuration item DefaultCalledPickupMsg, the driver will send CON (address is complete and pickup) or ANM (answer) message, in which the backward call indicator field can be set by the configuration item DefaultBackwardCallInd, to the remote PBX. If the channel is in 'idle' or 'blocked' state, the driver will not send any messages to the remote PBX.

✧ **ISDN channel:**

For the ISDN channel, when the application calls this function, if the channel is in the 'ringing' state, the driver will send CONNECT message to the remote PBX and start the Timer T313 simultaneously; then it will transfer the channel state to S_ISDN_IN_WAIT_TALK. If the channel is in the state of 'idle' or 'blocked', the driver will not send any messages to the remote PBX.

✧ **Analog Trunk Channel**

For the incoming calls in the analog trunk channel, when the application calls this function, if the channel is in the state of S_CALL_RINGING and the ringing current stays at on state, in order to avoid the damage to the voice board caused by the on-line ringing current voltage, the driver will not send the pickup command to the hardware circuit. Only when the voltage of the ringing current turns to the off state will the driver execute the pickup command and throw out the event E_SYS_ActualPickup to the application. The function SsmCheckActualPickup can be used to check whether the pickup command has been finished.

✧ **IP Channel**

An idle IP channel will turn into the 'Outgoing Call Locked' state after the call of this function. Then users can invoke SsmHangup to send the channel back to the 'Idle" state or invoke dial functions to call out. Invoking this function for an IP channel which is in the 'Ringing' state means to accept the incoming call. When this function is called for an IP channel in other states, it will return error.

Note:

- SsmPickup is applicable to SS1 channel, TUP channel, ISUP channel, ISDN channel, IP channel, analog trunk channel and EM Control channel. SsmPickupANX is only applicable to TUP channel and ISUP channel.
- For analog trunk channel, after the application calls this function, the driver will reset the ringing current detector and also reset and start the tone detector.

Related Information:

Driver version	SsmPickup requires SynCTI Ver. 3.0 or above SsmPickupANX requires SynCTI Ver. 4.7.1.5 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmHangup, SsmHangupEx

2.3.6 Channel Hangup Functions (CTI Series)

2.3.6.1 SsmHangup

Refer to SsmHangupEx

2.3.6.2 SsmHangupEx

Sends the hangup command to the channel. SsmHangupEx can also send the information such as the cause of hangup to the driver, but it is only applicable to ISUP channel.

Format:

int SsmHangup(int ch)

int SsmHangupEx(int ch, UCHAR ucCauseVal)

Parameter Description:

ch	Channel Number
ucCauseVal	The cause of hangup release. For detailed values, refer to the description on ISUP protocol.

Return Value:

-1	Fail to hang up the call. The failure reason can be acquired via function SsmGetLastErrMsg.
0	Hangup operation succeeds.

Function Description:

Sends the hangup command to the channel. The exact driver behavior depends on the channel type.

- **ISUP Channel**

This function will trigger the driver to send REL (release) message to the remote PBX which can be used either to reject the incoming call or to cancel the outgoing call. If the application calls SsmHangupEx, the release cause in the REL message is determined by the parameter of ucCauseVal. If the application calls SsmHangup, the release cause varies according to the channel state:

Channel State	Release Cause in REL Message
S_CALL_PICKUPED	Does not send REL message
S_ISUP_WaitDialAnswer S_CALL_WAIT_REMOTE_PICKUP S_CALL_TALKING S_ISUP_WaitSetCallerID	C_ISUP_REL_NORMAL_REL
S_ISUP_Suspend	C_ISUP_REL_DENY
S_ISUP_WaitSAM S_ISUP_WaitINF S_CALL_RINGING S_CALL_PENDING (the hangup cause is PEND_WaitBckStpMsg)	Determined by the preset value of the configuration item DefaultHangupRELInd or function SsmSetIsupFlag (with parameter ISUP_REL_DENY_SetToOther)

Note: The macros used in the above table are declared in the header file shpa3api.h.

- **TUP Channel**

If the channel is in the state of S_CALL_RINGING, when this function is invoked, the application will reject the incoming call and the driver will send the call rejection message to the remote PBX. The call rejection message could be either CBK or CFL, which is determined by the configuration item HangupRingSendCBK.

➢ **ISDN Channel**

For the ISDN channel, after invoking this function:

- ✧ If the channel is in the state of S_CALL_PENDING, the driver will send 'RELEASE' message to the remote PBX.
- ✧ If the channel is in the state of S_ISDN_OUT_PLS_APPEND_NO, the current call will be hanged up. The driver will send the 'DISCONNECT' message to the remote PBX and the hangup cause field will be set to be 'unallocated number'.
- ✧ If the channel is in the state of S_CALL_TALKING, it indicates that the local end hangs up first. The driver will send the 'DISCONNECT' message to the remote PBX and the hangup cause is set to be 'the call is finished normally'.
- ✧ If the channel is in the state of S_CALL_RINGING, it indicates that the application rejects the current incoming call. The driver will send the 'DISCONNECT' message to the remote PBX and the hangup cause in the message is set to be 'call rejection'.

➢ **IP Channel**

- ✧ If this function is called for an IP channel which stays in the 'Outgoing Call Locked' state, the channel state will turn into idle;
- ✧ If this function is called for a non-idle IP channel, the IP state machine will trigger different protocol commands according to the current state and terminate the IP session.

Note:

- The function SsmHangupEx is applicable to ISUP channel, TUP channel, ISDN channel, analog trunk channel and SS1 channel. The VoIP channel and the EM Control channel can only use the function SsmHangup.
- For the ISDN channel, before invoking this function, the application can set the hangup reason via the function SsmISDNSetHangupRzn.

Related Information:

Driver version	SynCTI Ver. 2.0 or above SsmHangupEx: SynCTI Ver. 4.7.1.5 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmPickup, SsmPickupANX

2.3.7 Setting Channel's Call Direction

2.3.7.1 SsmSetAutoCallDirection

Sets the call direction of the channel.

Format:

int SsmSetAutoCallDirection(int ch, BOOL bEnAutoCall, int nDirection)

Parameter Description:

ch	Channel Number
bEnAutoCall	Whether the call is processed by the driver automatically. =TRUE: Yes; =FALSE: No. In this case, nDirection will be ignored.
nDirection	Call direction, it's valid only when bEnAutoCall is set to be TRUE. Range of Value:

	=0: Only incoming calls supported; =1: Only outgoing calls supported; =2: Both outgoing and incoming calls supported (invalid to SS1).

Return Value:

-1	Failed, the failure reason can be acquired via the function SsmGetLastErrMsg
≥0	Successful

Function Description:

Sets the call direction of the channel.

Note:

- This function only supports the SS1 and ISDN channels of SHD Series boards, and the SIP channel of SHN Series boards.
- For the SS1 and ISDN channels of SHD Series boards, only when the channel is in the state of S_CALL_STANDBY can this function be called. For the SIP channel of SHN Series boards, this function can be called regardless of the channel state.
- This function has no effect on SS7 trunk channels. The call direction of SS7 channel can be set via blocking functions if necessary. For more details, refer to the function SsmBlockLocalCh.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmGetChState, SsmGetAutoCallDirection

2.3.7.2 SsmGetAutoCallDirection

Obtains the call direction of the channel.

Format:

int SsmGetAutoCallDirection(int ch, int* pnDirection)

Parameter Description:

ch	Channel Number
pnDirection	Returns the call direction, it's valid only when bEnAutoCall is set to be TRUE. Range of value: =0: Only incoming call supported; =1: Only outgoing call supported; =2: Both incoming and outgoing call supported; =3: Runtime under ISDN or TUP.

Return Value:

-1	Failed
0	In this channel, the call isn't processed by the driver automatically.
1	In this channel, the call is processed by the driver automatically.

Function Description:

Obtains the call direction of the channel.

Note:

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h

Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmSetAutoCallDirection

2.3.8 Obtaining Release Reason

2.3.8.1 SsmGetReleaseReason

Obtains the release cause value in the release message sent from the remote PBX.

Format:

WORD SsmGetReleaseReason(int ch)

Parameter Description:

ch	Channel Number

Return Value:

0	Unknown reason
>0	The cause value in the release message, the detailed message types and meanings are related with the channel type. ➢ ISUP channel: Returns the release cause in REL message. For detailed information, refer to *Appendix 1 ISDN Release Cause Information Element.* The commonly used values are: 0x01: unallocated number 0x10: normal disconnection 0x11: user busy 0x12: no answer 0x15: rejected 0x1c: incomplete address 0x1f: normal 0x2a: switching device congested 0x14: subscriber absent 0x16: number changed ➢ ISDN channel: Returns the 'release cause information element' field in the ISDN release message. For detailed information, refer to *Appendix 1 ISDN Release Cause Information Element*.

Function Description:

Obtains the release cause value in the release message sent from the remote PBX.

Note:

- Before processing the incoming call (i.e. when the driver receives a call setup message from the remote PBX), the driver sets the release cause value to be 0.
- This function is only applicable to ISDN channel (including network side and user side) and ISUP channel.

Related Information:

Driver version	SynCTI Ver. 4.7.1.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function:

2.3.9 TUP Channel Functions

2.3.9.1 SsmSetCalleeHoldFlag

Sets the 'caller holding' feature for the TUP channel.

Format:

int SsmSetCalleeHoldFlag(int ch, BOOL bFlag)

Parameter Description:

ch	Channel Number
bFlag	=TRUE: Enable =FALSE: Disable

Return Value:

-1	Failed. The failure reason can be acquired via the function SsmGetLastErrMsg
0	Successful

Function Description:

Sets the 'caller holding' feature for the TUP channel. For more information, refer to 'CALLER HOLDING Feature' in Chapter 1.

Note:

- This function is only applicable to the TUP channels of SHD Series board
- Only when the channel is in the state of S_CALL_TALKING, can this function be called.
- During the incoming call, before the driver transfers the channel to the state of S_CALL_TALKING, it automatically disables the 'caller holding' feature.

Related Information:

Driver version	SynCTI Ver. 4.7 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function:

2.3.9.2 SsmGetTupFlag

Obtains the value of the specified parameter or field in the TUP message.

Format:

int SsmGetTupFlag(int ch,int nType,DWORD *pdwParaValue)

Parameter Description:

ch	Channel Number
nType	Select the type of the parameter or field. Range of value: =1(Tup_ANX): During an outgoing call, if the called subscriber accepts this call when the channel stays in the S_CALL_WAIT_REMOTE_PICKUP state, the remote PBX will send the Answer Signal to the local end and the driver will transfer the channel state to S_CALL_TALKING. At this point, the detailed answer message type can be obtained via the parameter pdwParaValue.
pdwParaValue	Return value of the parameter or field. Detailed meaning is related with nType. nType=Tup_ANX: 0x06:ANU Message (No indication) 0x16:ANC Message (Charge) 0x26:ANN Message (No charge)

Return Value:

1	Successfully obtains the message.
-1	Failed, the failure reason can be acquired via the function SsmGetLastErrMsg

Function Description:

Obtains the value of the specified parameter or field in the TUP message.

Note:

- This function is only applicable to the TUP channels.

Related Information:

Driver version	SynCTI Ver. 4.7.1.9 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: None

2.3.9.3 SsmSetTupParameter

Sets the optional parameters of the Tup message in the outgoing call channel.

Format:

int SsmSetTupParameter (int nBCh, UCHAR ucMsgTypeCode, UCHAR ucParamTypeCode, WORD wLength, PUCHAR pucContent)

Parameter Description:

nBCh	Channel Number
ucMsgTypeCode	Message type. The range of value: 0x21: IAI message 0x11: IAM message Other: Reserved
ucParamTypeCode	Parameter type. The type value is related with the ucMsgTypeCode: ucMsgTypeCode = IAI message: 0x00: the nature of address indicator for the original called party in IAI 0x01: Sets the calling party's category field in the IAI message 0x02: Sets the address indicator for the calling line identification field in the IAI message ucMsgTypeCode = IAM message: 0x01: Sets the calling party's category field in the IAM message Other: Reserved
wLength	The length of parameter's content
pucContent	Pointer pointing to the buffer area storing the content of the parameter

Return Value:

-1	Failure in parameter setting
0	Successful

Function Description:

Sets the optional parameters of the TUP message in the outgoing call channel. This function is only effective to the nature of address indicator for the original called party in IAI, the address indicator for the calling line identification field in IAI and the calling party's category field in IAI/IAM . Relative configuration item: CalloutIAM_CAT.

Related Information:

Driver Version	SynCTI Ver. 4.7.3.1 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function： None

2.3.9.4 SsmGetTupParameter

Obtains a specified field from the received IAM message.

Format:

int SsmGetTupParameter (int nBCh, UCHAR ucMsgTypeCode, UCHAR ucParamTypeCode,PUCHAR pucContent, WORD wNumberOfBytesToWrite, LPWORD lpNumberOfBytesWritten);

Parameter Description:

nBCh	Channel Number
ucMsgTypeCode	Message type. The range of value: 0x11: IAM message Other: Reserved
ucParamTypeCode	Parameter type. The type value is related with the ucMsgTypeCode: 0xE3: obtains the CIC field of the IAM message Other: Reserved
pucContent	Pointer pointing to the buffer area storing the content of the parameter
wNumberOfBytesToWrite	Size of the buffer area
lpNumberOfBytesWritten	Length of the actually returned data

Return Value:

-1	Failed
0	Successful

Function Description:

Obtains a specified field from the received IAM message. At present this function only supports to obtain the CIC field of the IAM message.

Related Information:

Driver Version	SynCTI Ver.5.3.1.2 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function：None

2.3.10 ISUP Channel Functions

2.3.10.1 SsmSetISUPCAT

Sets the calling party category in the ISUP IAM message.

Format:

int SsmSetISUPCAT(int nch, UCHAR ucCallerCAT)

Parameter Description:

nch	Channel Number
ucCallerCAT	Hexadecimal calling party category

Return Value: 0

Function Description:

Sets the calling party category in the ISUP IAM message. The parameters set by this function can also be set by the configuration item DefaultIAM_CAT.

Note:

- This function should be called before calling the function SsmAutoDial or SsmAutoDialEx.
- The parameters set in this function will go invalid once the program is restarted and the configuration file

is reloaded.

Related Information:

Driver version	SynCTI Ver. 4.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function:

2.3.10.2 SsmSetIsupParameter

Sets the optional fields in the ISUP message sent from the local end to the remote PBX.

Format:

int SsmSetIsupParameter(int ch, UCHAR ucMsuType, UCHAR ucParamType, WORD wLength, PUCHAR pucContent)

Parameter Description:

ch	Channel Number
ucMsuType	Message type. The range of value: 0x01: IAM message 0X09: ANM message Other: Reserved
ucParamType	Parameter type. The type value is related to the usMsuType: ucMsuType= IAM message: set optional fields ucMsuType= ANM message: set other optional fields except the following one: ✧ Backward call indicator For the parameter type of each optional field, refer to Appendix 2 Optional ISUP Parameters and Descriptions. ucMsuType=other: reserved
wLength	Parameter length (byte). If ucMsuType=IAM, setting wLength to 0 means that the IAM message no longer includes the optional fields designated by ucParamType
pucContent	The content of the parameter. If ucMsuType=IAM, setting pucContent to a NULL string means to remove the optional fields designated by ucParamType from the IAM message. You can only remove the field set by this function.

Return Value:

-1	Failed
0	Successful

Function Description:

Sets the optional fields in the ISUP message sent from the local end to the remote PBX.

Note:

- If ucMsuType is set to be the IAM message, this function should be called before invoking the function SsmAutoDial. Once the parameter value is set, it will be saved in the internal buffer area and used in the subsequent IAM message. For the user to user information field in the IAM message, it can also be set by the configuration item Usr2UsrInfo.
- The parameters set in this function will go invalid once the program is restarted and the configuration file is reloaded.
- To cancel the settings, you should set wLength to 0 and pucContent to a NULL string at the same time; otherwise, invoking this function will fail.
- Only to a channel which has been set with an optional field can you invoke this function to cancel the settings; otherwise, invoking this function will fail.

Related Information:

Driver version	SynCTI Ver. 4.7.2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmGetIsupParameter

Sample code:

The redirection information field is added in the IAM message. The parameter type of the redirection information field is 0x13, which contains 2 bytes. Below is the meaning of the first 8-bit group (namely HGFEDCBA, A is the lowest bit) of the redirection information field:

Bit	Meaning	Value	Description
CBA	Redirection indicator	000	No redirection
		001	Call reroute
		010	Call reroute, presentation or restriction of all the redirection information
		011	In-call modification
		100	In-call modification, presentation or restriction of all the redirection information
		110	In-call modification, presentation or restriction of the redirection number
		other	reserved
D	Reserved		
HGFE	Original redirection cause	0000	Unknown/unusable
		0001	User busy
		0010	No answer
		0011	No condition
		other	reserved

Below is the meaning of the second 8-bit group (namely PONMLKJI, I is the lowest bit):

Bit	Meaning	Value	Description
KJI	Redirection counter	1~5	The redirection times of the call is expressed by a binary number between 1 and 5.
L	Reserved		Needs to be set 0
PONM	Cause of redirection	0000	Unknown/unusable
		0001	User busy
		0010	No answer
		0011	No condition
		0110	Unavailable mobile user
		other	Reserved

If the redirection information is set to be call forwarding on no answer, the related sample code is below:

```
UCHAR IsupParamRI[2];
IsupParamRI[0] = 0x03;          //set the content of the redirection information
IsupParamRI[1] = 0x21;
SsmSetIsupParameter(ch, 0x01, 0x13, 2, IsupParamRI);
SsmAutoDial(ch,…);
```

2.3.10.3 SsmGetIsupParameter

Obtains the optional fields in the ISUP message sent from the remote PBX to the local end.

Format:

int SsmGetIsupParameter(int nBCh, UCHAR ucMsuType, UCHAR ucParamType, PUCHAR pucContent, WORD wBufSize, LPWORD lpNumberOfBytesWritten)

Parameter Description:

nBCh	Channel Number
ucMsuType	Message type. Range of value: 0x01: IAM message 0x06: ACM message Others: reserved
ucParamType	● Fields of IAM message: 0xE3: Obtain the CIC field of the IAM message 0xE7: Obtain the TMR field of the IAM message 0xE8: Obtain the calling party subaddress of the IAM message

	0xE9: Obtain the called party subaddress of the IAM message ● The type of the optional field. This value is related with ucMsuType. For more information, refer to Appendix 2 Optional ISUP Parameters and Descriptions.
pucContent	The pointer which points to the buffer area storing the content of the optional fields. The buffer area for storage is allocated by the application and its length is recommended to be 272. **Note:** The returned information doesn't include the code of the type (1 byte) and the length (1byte) of the optional field.
wBufSize	The length of pucContent (byte)
lpNumberOfBytesWritten	The actual length (Byte) of the data which is written into pucContent by the driver.

Return Value:

-1	Failed
0	Successful

Function Description:

Obtains the optional fields in the ISUP message sent from the remote PBX to the local end.

Note:

Related Information:

Driver version	SynCTI Ver. 4.7.2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmSetIsupParameter

2.3.10.4 SsmSetIsupFlag

Sets the parameters in the ISUP user part.

Format:

int SsmSetIsupFlag(int ch, int nType, DWORD dwValue, PVOID pV)

Parameter Description:

ch	Channel number
nType	Choose the type of the parameter, below are the available values:

	ISUP_PhoNumParam=2: Sets the callee parameter of the called party number field in the IAM message, at this point, the parameter dwValue is the set value and the parameter pV is invalid. ISUP_CallerParam=1: Sets the caller parameter of the calling party number field, at this point, the parameter dwValue is the set value and the parameter pV is invalid. ISUP_REL_DENY_SetToOther=100: When the application calls the function SsmHangup and triggers the driver to send REL message to the remote PBX, the parameter dwValue can set the release cause carried in the REL message and the parameter pV is invalid. ISUP_PhoNumREL=3: Sends the disconnection message including the number redirection information to the remote PBX, at this point, the parameter pV is valid. ISUP_IAM_TMR=4: Sets the TMR (Transmission Medium Requirement) parameter in the IAM message, where the parameter dwValue is set and the parameter pV is invalid. DefaultIAM_OriginalCalleeParam=101: The parameter of the original called party. ISUP_PhoNumRELEx=5: This type is similar to PhoNumREL. It sends the disconnection message including the number redirection information to the remote PBX and sets the release cause. Now the parameter pV is valid.
dwValue	Sets the parameter value which is related with the value of nType

Value of nType	Value and meaning of dwValue
ISUP_PhoNumParam	Refer to the configuration item DefaultIAM_CalleeParam
ISUP_CallerParam	Refer to the configuration item DefaultIAM_CallerParam
ISUP_REL_DENY_SetToOther	Refer to the configuration item DefaultHangupRELInd
ISUP_IAM_TMR	Refer to the configuration item DefaultIAM_TransmissionMediumRequirment
DefaultIAM_OriginalCalleeParam	Parameter value of Original Called Party: Spare: 0x1000 Subscriber Number: 0x1001 Unknown: 0x1002 National Number: 0x1003 International Number: 0x1004
ISUP_PhoNumREL	Reserved, unused

pV	The meaning of pV is related with the value of nType:
	➢ nType=ISUP_PhoNumREL:
	pV is the pointer pointing to the struct object ISUP_RIREL, which is used to customize the parameters in the REL message. The struct ISUP_RIREL in Shpa3api.h is declared as below:
	typedef struct tag_ISUP_RIREL{
	WORD **wRIMsg**;
	WORD **wRIPhoNumPara**;
	WORD **wPhoNumLen**;
	UCHAR **ucRIPhoNum**[20];
	}ISUP_RIREL,*PISUP_RIREL;
	The parameter meanings in the above struct are:
	wRIMsg: The 'redirection information' field consists of the bits PONMLKJIHGFEDCBA;
	wRIPhoNumPara: The 'redirection number' field consists of the bits Bit15…Bit0;
	wPhoNumLen: The character number of ucRIPhoNum
	ucRIPhoNum: The string of the redirection phone number.
	➢ nType=ISUP_PhoNumRELEx:
	pV is the pointer pointing to the struct object ISUP_RIRELEX which is used to customize the parameters in the REL message. The struct ISUP_RIRELEX in Shpa3api.h is declared as below:
	typedef struct tag_ISUP_RIRELEX
	{
	UCHAR **ucCauseInd**;
	WORD **wRIMsg**;
	WORD **wRIPhoNumPara**;
	WORD **wPhoNumLen**;
	UCHAR **ucRIPhoNum**[20];
	}ISUP_RIRELEX, *PISUP_RIRELEX;
	The parameter meanings in the above struct are:
	ucCauseInd: Release cause;
	wRIMsg : The 'redirection information' field consists of the bits PONMLKJIHGFEDCBA;
	wRIPhoNumPara: The 'redirection number' field consists of the bits Bit15…Bit0;
	wPhoNumLen: The character number in ucRIPhoNum
	ucRIPhoNum: The string of the redirection phone number.
	➢ nType=other: Reserved, unused.

Note: All the constants used in the table are declared in the header file Shpa3api.h.

Return Value:

-1	Failed
1	Successful

Function Description:

Sets the parameters in the ISUP user part.

Note:

- ISUP_IAM_TMR=4 is only supported by SynCTI Ver. 5.1.0.0 or above. ISUP_PhoNumRELEx=5 is only supported by SynCTI Ver. 5.3.2.0 or above.
- When nType is set to ISUP_PhoNumREL, the operation of invoking SsmSetIsupFlag on the channel ch to set user parameters only has an effect on the current call. When nType is set to other values, the parameters set in this function will go invalid once the program is restarted and the configuration file is

reloaded.

Related Information:

Driver version	SynCTI Ver. 4.7.1.5 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmGetIsupFlag

2.3.10.5 SsmGetIsupFlag

Obtains the parameters in the ISUP user part.

Format:

int SsmGetIsupFlag(int ch, int nType, DWORD *pd)

Parameter Description:

ch	Channel number
nType	Select the parameter type, for more details, refer to the function SsmSetIsupFlag.
*pd	Parameter value, for more details, refer to the function SsmSetIsupFlag.

Return Value:

-1	Failed
1	Successful

Function Description:

Obtains the parameters in the ISUP user part.

Note:

Related Information:

Driver version	SynCTI Ver. 4.7.1.5 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmSetIsupFlag

2.3.10.6 SsmGetIsupUPPara

Obtains the original ISUP message saved in the driver.

Format:

int SsmGetIsupUPPara(int nBCh, WORD wMsuType, LPWORD pwMsuSize, PUCHAR pucRawMsu)

Parameter Description:

nBCh	Channel Number
wMsuType	Message Type **Value** / **Macro in shpa3api.h** / **Message Corresponding with parameter pucContent** 0x01 / C_ISUP_IAM / Initial address message (IAM) 0x06 / C_ISUP_ACM / Address complete message (ACM)
pwMsuSize	Returns the actual length (bytes) of the message.
pucRawMsu	Pointer which points the buffer area storing the ISUP message. The buffer area is allocated by the application and its length must be greater than or equal to 281 bytes.

Return Value:

1	Successful

-1	Incorrect channel number
-2	Call failed, the parameter wMsuType is out of bound

Function Description:

Obtains the original ISUP message saved in the driver. For more information, refer to 'ISUP Channel State Machine' in Chapter 1.

Note:

- This function only supports ISUP channel.
- The ISUP message stored in the driver will be cleared up following the hangup of the ISUP call.

Related Information:

Driver version	SynCTI Ver. 4.7.1.5 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmSetIsupUPPara

2.3.10.7 SsmSetIsupUPPara

Submits a self-constructed ISUP message to the driver.

Format:

int SsmSetIsupUPPara(int nBCh, WORD wMsuType, LPWORD pwMsuSize, PUCHAR pucRawMsu)

Parameter Description:

nBCh	Channel number
wMsuType	Message Type Value / Macro in shpa3api.h / The Message Corresponding to Parameter pucContent 0x01 / C_ISUP_IAM / Initial address message(IAM) 0x06 / C_ISUP_ACM / Address complete message (ACM)
pwMsuSize	The length (bytes) of the ISUP message (including SIO field), with the maximum length of 281 bytes
pucRawMsu	Pointer which points to the initial address of the message data

Return Value:

1	Successful
-1	Incorrect channel number
-2	Failed, the parameter wMsuType is out of bound
-3	Verification failed, the message content doesn't qualify the ISUP protocol rules

Function Description:

Submits a self-constructed ISUP message to the driver.

Note:

- This function only supports ISUP channel.
- If wMsuType is C_ISUP_IAM, this function must be called before SsmAutoDial or SsmAutoDialEx is invoked.
- If wMsuType is C_ISUP_ACM, only when the channel is in the 'pending' state and the cause for pending is PEND_WaitBckStpMsg, can the ACM message submitted by the application be sent to the remote PBX. For more information, refer to 'ISUP Channel State Machine' in Chapter 1.
- When the driver sends ISUP messages submitted by the application, it automatically covers the SIO, DPC, OPC, SLC and CIC fields. If wMsuType is C_ISUP_IAM, the driver covers the corresponding

fields in pucRawMsu with the calleeID in SsmAutoDial or SsmAutoDialEx.

- The parameters set in this function are only valid in the current function call.

Example:

WORD acmmsglen=18;

UCHAR acmmsg[20]={0,0,0,0,0,0,0,0,0,0,6,22,20,1,41,1,3,0};

The value of acmmsg[13] must be 1.

SsmSetIsupUPPara(channel, C_ISUP_ACM, &acmmsglen, acmmsg);

Related Information:

Driver version	SynCTI Ver. 4.7.1.5 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmAutoDial, SsmAutoDialEx, SsmGetIsupUPPara, SsmSendIsupMsg

2.3.10.8 SsmSendIsupMsg

Sends one ISUP message to the remote PBX.

Format:

int SsmSendIsupMsg(int nBCh, WORD wMsuType)

Parameter Description:

nBCh	Channel number		
wMsuType	Message Type		
	Value	**Macro in shpa3api.h**	**Message Type**
	0x06	C_ISUP_ACM	Address Complete Message (ACM)

Return Value:

1	Successful
-1	Incorrect channel number
-2	Failed, the parameter wMsuType is out of bound
-3	Failed in sending message

Function Description:

Sends one ISUP message to the remote PBX. The function SsmSetIsupUPPara must be called in advance so that the ISUP message can be submitted to the driver.

Note:

- This function only supports ISUP channel.
- Only when the channel is in the S_CALL_PENDING state and the cause for pending is PEND_WaitBckStpMsg, can this function be called.

Related Information:

Driver version	SynCTI Ver. 4.7.1.5 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmSetIsupUPPara

2.3.10.9 SsmGetRedirectionInfReason

During incoming call, for ISUP channel, obtains the cause value of redirection from the received IAM message; for ISDN channel, obtains the cause value of redirection from the received Setup message.

Format:

int SsmGetRedirectionInfReason(int ch)

Parameter Description:

ch	Channel number

Return Value:

-1	Failed
≥0	Cause value of redirection 0: Unknown/unavailable 1: User busy 2: No answer 3: Unconditional 4: Redirect the call when notified 5: Immediately redirect the call 6: Mobile subscriber unreachable

Function Description:

During incoming call, for ISUP channel, obtains the redirection reason from the received IAM message; for ISDN channel, obtains the redirection reason of the information element Facility (0x1c) or Redirecting Number (0x74) from the received Setup message. To obtain the redirection reason of which information element is determined by the configuration item GetRedirectionReason.

Note:

Related Information:

Driver version	SynCTI Ver. 5.3.0.3 or above (To obtain the redirection reason of the information element Facility (0x1c), SynCTI Ver. 5.3.2.3 or above is required)
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmGetIsupUPPara

2.3.10.10 SsmGetRedirectionInfNum

Obtains the redirection number from an ISDN channel.

Format:

int WINAPI SsmGetRedirectionInfNum(int ch, LPSTR szRedirectNum)

Parameter Description:

ch	Channel number
szRedirectNum	The pointer pointing to the buffer area storing the redirection number. The storage space, allocated by the application, should be not less than 32 bytes.

Return Value:

≥0	Call successful. Returns the length of the redirection number (bytes)
-1	Call failed

Function Description:

Obtains the redirection number from an ISDN channel.

Note:

Related Information:

Driver version	SynCTI Ver. 5.3.0.3 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function:

2.3.10.11 SsmIsHaveCpg

Refer to SsmGetCpg

2.3.10.12 SsmGetCpg

SsmIsHaveCpg checks whether ACM message is followed by the CPG message. If yes, SsmGetCpg retrieves the original CPG message stored in the driver.

Format:

int SsmIsHaveCpg(int ch)

int SsmGetCpg(int ch, char* pszContent, int* piLength)

Parameter Description:

ch	Channel Number
pszContent	Returns the content of the CPG message. For detailed information about the CPG message, refer to relative documents of the ISUP protocol
piLength	Returns the length of the CPG message

Return Value:

-1	Failed. The failure reason can be acquired via the function SsmGetLastErrMsg
0	SsmIsHaveCpg: no CPG message SsmGetCpg: Successful
1	SsmIsHaveCpg: CPG message exists. Only when SsmIsHaveCpg returns 1 can the function SsmGetCpg be invoked.

Function Description:

During outgoing call, after the remote PBX sends ACM message, it may continue sending the CPG message (call progress). When the driver has received the CPG message, it saves the message in the internal buffer area. This function checks whether the ACM message is followed by the CPG message.

Note:

- Only when the channel is in the S_CALL_WAIT_REMOTE_PICKUP state, can this function be called.
- When the following events happen, the driver automatically cleans the CPG message in the internal buffer:
  - ✧ The channel is transferred to the S_CALL_STANDBY state.
  - ✧ Receives the ACM message from the remote PBX.
- This function only supports the ISUP channel.

Related Information:

Driver version	SynCTI Ver. 4.7.1.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Event: E_CHG_ChState, E_RCV_Ss7IsupCpg

2.3.10.13 SsmIsupGetUsr

Checks and obtains the USR message.

Format:

BOOL SsmIsupGetUsr(int *ch, PUCHAR pucData, PUCHAR ucLen)

Parameter Description:

ch	The pointer pointing to the number of the channel receiving messages
pucData	The buffer area storing the 'user-to-user information' in the received USR message, which occupies 130 bytes
ucLen	The length of pucData

Return Value:

0	Obtains new data
Other	No new data has been received

Function Description:

Checks and obtains the USR message. After the execution of this function, the new message will be deleted from the system.

Note: none.

Related Information:

Driver Version	SynCTI Ver. 4.7.3.1 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Event: E_RCV_Ss7IsupUtuinf

2.3.10.14 SsmIsupSendUsr

Sends the USR message.

Format:

BOOL SsmIsupSendUsr(int ch, PUCHAR pucData, UCHAR ucLen)

Parameter Description:

ch	Channel Number
pucData	The data to be sent as the 'user-to-user information' in the USR message
ucLen	The length of pucData, the defined valid value in the protocol is 2~130

Return Value:

1	Successful
Other	Call failed

Function Description:

Sends the USR message.

Note: None.

Related Information:

Driver Version	SynCTI Ver. 4.7.3.1 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Event:

2.3.11 ISDN Channel Function

The functions provided in this section are specially used for ISDN channel calls.

2.3.11.1 Setting Parameters for Setup Message

2.3.11.1.1SsmISDNSetDialSubAddr

Refer to SsmISDNSetDialSubAddrEx

2.3.11.1.2SsmISDNSetDialSubAddrEx

Sets the subaddress of the called party in the SETUP message during an outgoing call.

Format:

int SsmISDNSetDialSubAddr(int ch, LPSTR lpSubAddress)

int WINAPI SsmISDNSetDialSubAddrEx(int ch, LPBYTE lpSubAddressEx, UCHAR ucSubAddressLen)

Parameter Description:

ch	Channel Number
lpSubAddress	The subaddress of the called party. It's comprised with characters '0'~'9', and the maximum length is 20 bytes. The storage space is allocated by the application
lpSubAddressEx	The subaddress of the called party, including the subaddress type and the buffer area storing the subaddress messages
ucSubAddressLen	The total length of the subaddress type and the buffer area storing the subaddress messages

Return Value:

0	Successful
-1	Failed, the failure reason can be obtained by the function SsmGetLastErrMsg.

Function Description:

During outgoing call, sets the subaddress of the called party.

Note:

- This function must be called before SsmAutoDial is invoked.
- The parameters set in this function will go invalid once the configuration file is reloaded.

Related Information:

Driver version	SynCTI Ver. 3.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmAutoDial

2.3.11.1.3SsmISDNSetTxSubAddr

Refer to SsmISDNSetTxSubAddrEx

2.3.11.1.4SsmISDNSetTxSubAddrEx

Sets the subaddress of the calling party in the SETUP message during an outgoing call.

Format:

int SsmISDNSetTxSubAddr(int ch, LPSTR lpSubAddress)

int WINAPI SsmISDNSetTxSubAddrEx(int ch, LPBYTE lpSubAddressEx,UCHAR ucSubAddressLen)

Parameter Description:

ch	Channel Number
lpSubAddress	The subaddress of the calling party, comprised with characters '0'~'9', and the maximum length is 20 bytes. The storage space is allocated by the application.
lpSubAddressEx	The subaddress of the calling party, including the subaddress type and the buffer area storing the subaddress messages
ucSubAddressLen	The total length of the subaddress type and the buffer area storing the subaddress messages

Return Value:

0	Successful
-1	Failed, the failure reason can be obtained by the function SsmGetLastErrMsg.

Function Description:

During outgoing call, sets the subaddress of the calling party in the SETUP message.

Note:

- This function must be called before SsmAutoDial is called.
- The parameters set in this function are only valid in the current function call.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmAutoDial, SsmISDNGetTxCallerSubAddr

2.3.11.1.5SsmISDNSetCallerIdPresent

Sets the 'CallerID Present' field in the signaling message.

Format:

int SsmISDNSetCallerIdPresent(int ch, UCHAR ucPresentation)

Parameter Description:

ch	Channel Number
ucPresentation	=0: not allowed to present =1: allowed to present, provided by users =2: allowed to present, provided by network

Return Value:

0	Call successful
-1	Call failed. The failure reason can be obtained by the function SsmGetLastErrMsg.

Function Description:

Sets the 'CallerID Present' field in the signaling message.

Note:

- The parameters set in this function will go invalid once the program is restarted and the configuration file is reloaded to initialize the link.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function:

2.3.11.1.6 SsmSetIsdnParameter

Adds a field to the SETUP message or sends a user-defined non-SETUP message dynamically.

Format:

int WINAPI SsmSetIsdnParameter(int nBCh, UCHAR ucMsgTypeCode, UCHAR ucParamTypeCode, PUCHAR pucContent, WORD wNumberOfBytesToWrite)

Parameter Description:

nBCh	Channel Number
ucMsgTypeCode	Message type, support the SETUP message (0x05) and other user-defined messages
ucParamTypeCode	Field type code
pucContent	Buffer storing the field content (excluding field type code and content length)
wNumberOfBytesToWrite	Actual length of the field content

Return Value:

0	Call successful
-1	Call failed. The failure reason can be obtained by the function SsmGetLastErrMsg.

Function Description:

Adds a field to the SETUP message or sends a user-defined non-SETUP message dynamically.

Note:

- For the SETUP message, the parameters set in this function will go invalid once the AutoDial operation starts.
- For other user-defined messages, this function will add a field to the message and send it at a time.
- This function can be invoked multiple times to add more than one user-defined field.

Related Information:

Driver version	SynCTI Ver. 5.3.2.2 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function:

Sample code:

```
//Sends a self-defined non-SETUP message: 08 02 80 01 7b 70 06 81 54 32 30 30 30
```

```
char msu [100]="";
msu [0]=0x81 ;
msu [1]=0x54 ;
msu [2]=0x32 ;
msu [3]=0x30 ;
msu [4]=0x30 ;
msu [5]=0x30;
SsmSetIsdnParameter( 0, 0x7b, 0x70, msu, 6);
```

2.3.11.1.7SsmSetIsdnParameterA

Adds a field to the SETUP message or sends a user-defined non-SETUP message dynamically.

Format:

int WINAPI SsmSetIsdnParameterA(int nBCh, UCHAR ucMsgTypeCode, PUCHAR pucParamTypeCode,PUCHAR pucContent, PUCHAR pucNumberOfBytesToWrite,WORD wNumberOfUnit)

Parameter Description:

nBCh	Channel Number
ucMsgTypeCode	Message type, support the SETUP message (0x05) and other user-defined messages
pucParamTypeCode	Type code for each information element
pucContent	Buffer storing the content of each information element (excluding field type code and content length)
pucNumberOfBytesToWrite	Actual length of the content of each information element.
wNumberOfUnit	The number of the information elements to be added.

Return Value:

0	Call successful
-1	Call failed. The failure reason can be obtained by the function SsmGetLastErrMsg.

Function Description:

Adds a field to the SETUP message or sends a user-defined non-SETUP message dynamically.

Note:

- For the SETUP message, the parameters set in this function will go invalid once the AutoDial operation starts.
- For other user-defined messages, this function may add more than one user-defined field to the message and send it at a time.

Related Information:

Driver version	SynCTI Ver. 5.3.3.1 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function:

Sample code:

```
// Sends the non-SETUP message which contains more than one self-defined information element at a time: 08 02 80 01 7d 08 03 82 e4 6c 14 01 01

Parsing:
08 // protocol discriminator: isdn
02 80 01 //02 call reference length, 80 01 call reference content
7d // message type: state
08 03 82 e4 6c //08 the first information element: cause, 03 length, 82 e4 6c content
```

14 01 01 //14 the second information element: call state, 01 length, 01 content

```
char msu [100]="";
msu [0]=0x82 ;// The content of the first information element
msu [1]=0xe4 ;
msu [2]=0x6c ;
msu [3]=0x01 ;// The content of the second information element

char ParamType[50]="";// Type code for the information element

ParamType[0]=0x08;// The first information element

ParamType[1]=0x14;// The second information element


char NumberOfBytesToWrite [50]="";// The actual length of the information element

NumberOfBytesToWrite [0]=0x03;// The length of the first information element
NumberOfBytesToWrite [1]=0x01;// The length of the second information element


SsmSetIsdnParameterA (0, 0x7d, ParamType, msu, NumberOfBytesToWrite, 2);
```

2.3.11.2 Obtaining Incoming Call Information

2.3.11.2.1 Obtaining Subaddress Information

2.3.11.2.1.1 SsmISDNGetSubAddr

Obtains the subaddress information of the called party.

Format:

int SsmISDNGetSubAddr(int ch, LPSTR lpSubAddress)

Parameter Description:

ch	Channel Number
lpSubAddress	Buffer area storing the subaddress of the called party. The storage space is allocated by the application, and its length should be no less than 21 bytes.

Return Value:

≥0	The actual length (bytes) of the string in lpSubAddress
-1	Failed, the failure reason can be obtained by the function SsmGetLastErrMsg.

Function Description:

During incoming call, the message from the remote PBX includes the subaddress information of the called party. This function retrieves the ASCII character formatted subaddress information of the called party from the message.

Note: None

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmISDNGetCallerSubAddr

2.3.11.2.1.2 SsmISDNGetCallerSubAddr

Obtains the subaddress information of the calling party.

Format:

int SsmISDNGetCallerSubAddr(int ch, LPSTR lpSubAddress)

Parameter Description:

ch	Channel Number
lpSubAddress	Buffer area storing the subaddress of the calling party. The storage space is allocated by the application and its length should be no less than 21 bytes.

Return Value:

≥0	The length of the calling party's subaddress
-1	Call failed. The failure reason can be obtained by the function SsmGetLastErrMsg

Function Description:

During incoming call, the message from the remote PBX includes the subaddress information of the calling party. This function retrieves the ASCII character formatted subaddress information of the calling party from the message.

Note: None

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmISDNGetSubAddr

2.3.11.2.2 Obtaining User-defined Calling Party Number

2.3.11.2.2.1 SsmGetUserCallerId

Obtains the user-defined calling party number.

Format:

int SsmGetUserCallerId(int ch, LPSTR szCallerId)

Parameter Description:

ch	Channel Number:
szCallerId	The buffer area storing the user-defined calling party number during incoming call. The maximum length is 21 bytes.

Return Value:

-1	Call failed, the failure reason can be obtained via the function call of SsmGetLastErrMsg
≥0	Length of the user-defined calling party number

Function Description:

Obtains the user-defined calling party number.

Note:

Related Information:

Driver Version	SynCTI Ver. 4.7.3.0 or above

Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function:

2.3.11.2.3 Obtaining Designated Field from SETUP Message

2.3.11.2.3.1 SsmGetIsdnParameter

Obtains the content of a designated signaling unit from the SETUP message.

Format:

int WINAPI SsmGetIsdnParameter(int nBCh, UCHAR ucMsgTypeCode, UCHAR ucParamTypeCode,UCHAR ucParamIndex, PUCHAR pucContent, WORD wNumberOfBytesToWrite, LPWORD lpNumberOfBytesWritten)

Parameter Description:

nBCh	Channel Number
ucMsgTypeCode	Message type, only the SETUP message (0x05) is supported now
ucParamTypeCode	Code of the designated signaling unit
ucParamIndex	Serial number of the designated signaling unit. Commonly it is 0. If there are several signaling units of the same type, the serial number of the signaling unit to be obtained is determined by this parameter
pucContent	Buffer storing the content of the signaling unit (excluding signaling unit code and content length)
wNumberOfBytesToWrite	Size of the buffer. It should be no less than 300 bytes
lpNumberOfBytesWritten	Length of the actually obtained signaling unit

Return Value:

-1	Call failed, the failure reason can be obtained by the function of SsmGetLastErrMsg
0	Call successful

Function Description:

Obtains the content of a designated signaling unit from the SETUP message.

Note:

Related Information:

Driver Version	SynCTI Ver. 5.3.1.4 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function:

2.3.11.3 Setting Hangup Reason

2.3.11.3.1 SsmISDNSetHangupRzn

Sets the hangup cause.

Format:

int SsmISDNSetHangupRzn(int ch, int nReason)

Parameter Description:

ch	Channel Number
nReason	Hangup cause, the range of value: 0: Normal hangup 1: Subscriber busy 2: Unallocated number 3: Reject the current call

Return Value:

0	Successful
-1	Call failed, the failure reason can be obtained by the function SsmGetLastErrMsg

Function Description:

Sets the hangup cause.

Note:

- This function must be called before the function SsmHangup is invoked.
- The parameters set in this function will go invalid once the configuration file is reloaded.

Related Information:

Driver version	SynCTI Ver. 4.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmGetPendingReason

2.3.11.4 Other Functions

2.3.11.4.1SsmISDNGetDisplayMsg

Obtains the display information of the ISDN channel.

Format:

int SsmISDNGetDisplayMsg(int ch, LPSTR lpDispMsg)

Parameter Description:

ch	Channel Number
lpDispMsg	The buffer area storing the display information. The maximum length is 100 bytes

Return Value:

≥0	The length of the display information
-1	Call failed, the failure reason can be obtained by the function SsmGetLastErrMsg

Function Description:

Obtains the display information of the ISDN channel. The display information is the ASCII formatted string, which is normally sent from the network side to the user side. It may also include greetings or other supplementary information.

Note: None

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h

Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: None

2.3.11.4.2SsmISDNGetTxCallerSubAddr

Obtains the subaddress information of the calling party configured at the local end.

Format:

int SsmISDNGetTxCallerSubAddr(int ch, LPSTR lpSubAddress)

Parameter Description:

ch	Channel Number
lpSubAddress	The buffer area storing the subaddress information of the calling party during outgoing call. The maximum length is 21 bytes.

Return Value:

≥0	The length of the calling party's subaddress.
-1	Call failed, the failure reason can be obtained by the function SsmGetLastErrMsg

Function Description:

Obtains the subaddress information of the calling party configured at the local end.

Note:

- This function is only applicable to the outgoing call.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmISDNGetSubAddr

2.3.11.4.3SsmSetNumType

Sets the calling/called party number type for a specified channel.

Format:

int SsmSetNumType(int ch, int nNumClass, int nNumType)

Parameter Description:

ch	Channel Number
nNumClass	Calling/called Party Number: 1: Calling party number 2: Called party number
nNumType	Number Type: 0x00: Unknown 0x10: International number 0x20: National number 0x30: Specified network number 0x40: Specified number or subscriber number Others: Spare

Return Value:

-1	Call failed
0	Call successful

Function Description:

Sets the calling/called party number type for a specified channel.

Note:

- This function is only applicable to ISDN channels.

Related Information:

Driver version	SynCTI Ver. 5.0.2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmGetNumType, NetCallingNoSet

2.3.11.4.4SsmGetNumType

Gets the calling/called party number type for a specified channel.

Format:

int SsmGetNumType(int ch, int nNumClass, int* pNumType)

Parameter Description:

ch	Channel Number
nNumClass	Calling/called Party Number: 1: Calling party number 2: Called party number
pNumType	Number Type: 0x00: Unknown 0x10: International number 0x20: National number 0x30: Specified network number 0x40: Specified number or subscriber number Others: Spare

Return Value:

-1	Call failed
0	Call successful

Function Description:

Gets the calling/called party number type for a specified channel.

Note:

- This function is only applicable to ISDN channels.

Related Information:

Driver version	SynCTI Ver. 5.0.0.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmSetNumType, NetCallingNoSet

2.3.11.4.5SsmSetCharge

Sets a call on the corresponding channel to be charged or free of charge.

Format:

int SsmSetCharge(int ch,int ChargeFlag)

Parameter Description:

ch	Channel number
ChargeFlag	0: Free; 1: Charged; Others: Reserved

Return Value:

-1	Call failed
0	Call successful

Function Description:

Sets a call on the corresponding channel to be charged or free of charge.

Note:

- This function only supports ISDN channels for some particular PBX models;
- It can be invoked in the state of ringback or talking for an outbound call;
- It can be invoked in the state of talking for an inbound call.

Related Information:

Driver version	SynCTI Ver. 5.0.0.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function:

2.4 Tone Generator Functions (CTI Series)

2.4.1 Setting Parameters for Tone Generator

2.4.1.1 SsmSetTxTonePara

Sets the frequency and volume of the tone generated by the tone generator.

Format:

int SsmSetTxTonePara(int ch, int nFreq1, int nVolume1, int nFreq2, int nVolume2)

Parameter Description:

ch	Channel Number
nFreq1	Sets the frequency (Hz) of the 1st tone, range of value: 300~3400, the default value: 450.
nVolume1	Sets the volume of the 1st tone, range of value: -7~+6. Greater than 0 denotes volume increasing, less than 0 denotes volume decreasing, -7 denotes turning off the 1st tone. The default value is 0. The volume value multiplies 3 equals the decibel value.
nFreq2	Sets the frequency (Hz) of the 2nd tone, range of value: 300~3400, with the default value of 0 which means the tone generator does not use this frequency and only sends

	the single frequency tone.
nVolume2	Sets the volume of the 2nd tone, range of value: -7~+6. Greater than 0 denotes volume increasing, less than 0 denotes volume decreasing, -7 denotes turning off the 2nd tone. The default value is -7 (the 2nd tone is not used). The volume value multiplies 3 equals the decibel value.

Return Value:

-1	Call failed, the failure reason can be obtained by the function SsmGetLastErrMsg
0	Successful

Function Description:

Sets the frequency and volume of the tone generated by the tone generator.

Note:

- The configuration items DefaultSendToneFrequence and DefaultSendToneVolume can implement the same feature. The default tone generated is a 450Hz single frequency tone.

Related Information:

Driver version	SynCTI Ver. 2.0.0.0or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmSendTone

2.4.1.2 SsmQueryOpSendTone

Checks if there is a tone generator on the channel.

Format:

int SsmQueryOpSendTone (int ch)

Parameter Description:

ch	Channel Number

Return Value:

-1	Failed
0	No
1	Yes

Function Description:

Checks if there is a tone generator on the channel.

Note:

Related Information:

Driver version	SynCTI Ver. 2.0.0.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function:

2.4.1.3 SsmGetTxTonePara

Obtains the frequency and volume of the tone from the tone generator.

Format:

int SsmGetTxTonePara(int ch, int* nFreq1, int* nVolume1, int* nFreq2, int* nVolume2)

Parameter Description:

ch	Channel number
nFreq1	Returns the frequency (Hz) of the 1st tone.
nVolume1	Returns the volume of the 1st tone, range of value: -7~+6. Greater than 0 denotes volume increasing, less than 0 denotes volume decreasing, -7 means this frequency is not used. The volume value multiplies 3 equals the decibel value.
nFreq2	Returns the frequency (Hz) of the 2nd tone.
nVolume2	Returns the volume of the 2nd tone, range of value: -7~+6. Greater than 0 denotes volume increasing, less than 0 denotes volume decreasing, -7 means that this frequency is not used. The volume value multiplies 3 equals the decibel value.

Return Value:

-1	Call failed, the failure reason can be obtained by the function SsmGetLastErrMsg
0	Successful

Function Description:

Obtains the frequency and volume of the tone from the tone generator.

Note:

Related Information:

Driver version	SynCTI Ver. 2.0.0.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmSetTxTonePara

2.4.2 Sending Tones

2.4.2.1 SsmSendTone

Refer to SsmSendToneEx

2.4.2.2 SsmSendToneEx

Starts the tone generator to generate tones on the line. SsmSendTone is used to send those tones in the driver-predefined type. SsmSendToneEx specifies the signal duration at on or off states.

Format:

int SsmSendTone(int ch, int nToneType)

int SsmSendToneEx(int ch, DWORD dwOnTime, DWORD dwOffTime)

Parameter Description:

ch	Channel Number
nToneType	Tone type, range of value: 0: dial tone 1: busy tone 2: ringback tone 3: howler tone
dwOnTime	Signal duration (ms) at on state. It must be integral times of 8
dwOffTime	Signal duration (ms) at off state. It must be integral times of 8

Return Value:

-1	Call failed, the failure reason can be obtained by the function SsmGetLastErrMsg
0	Successful

Function Description:

Generates single or dual frequency tones on the specified channel.

Note:

- The frequency and volume of the tone can be set via the function SsmSetTxTonePara or the configuration items DefaultSendToneFrequence and DefaultSendToneVolume.
- Once the tone generator is started, the specified tone will be continuously presented on the line until the application calls the function SsmStopSendTone.

Related Information:

Driver version	SynCTI Ver. 2.0.0.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmStopSendTone, SsmSetTxTonePara, SsmChkSendTone

2.4.2.3 SsmStopSendTone

Stops the tone generator.

Format:

int SsmStopSendTone(int ch)

Parameter Description:

ch	Channel Number

Return Value:

-1	Call failed, the failure reason can be obtained by the function SsmGetLastErrMsg
0	Successful

Function Description:

Stops the tone generator.

Note:

Related Information:

Driver version	SynCTI Ver. 2.0.0.0or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmSendTone, SsmSendToneEx

2.4.2.4 SsmChkSendTone

Checks whether the tone generator is started, and obtains the type of the tone being sent.

Format:

int SsmChkSendTone(int ch, int* pnToneType)

Parameter Description:

ch	Channel Number
pnToneType	Returns the type of the tone which is being sent on the channel: 0: dial tone 1: busy tone 2: ringback tone 3: howler tone 4: self-defined tone (The tone generated by SsmSendToneEx is detected)

	5: no tone detected

Return Value:

-1	Call failed, the failure reason can be obtained by the function SsmGetLastErrMsg
0	The channel is not sending the tone
1	The channel is sending the tone

Function Description:

Checks whether the tone generator is started, and obtains the type of the tone being sent.

Note:

Related Information:

Driver version	SynCTI Ver. 2.0.0.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmSendTone, SsmSendToneEx

2.5 Tone Detector Functions

2.5.1 Commonly Used Tone Detector Functions

2.5.1.1 Setting Working Status

2.5.1.1.1 SsmStartToneAnalyze

Starts the tone detector.

Format:

int SsmStartToneAnalyze(int ch)

Parameter Description:

ch	Channel Number

Return Value:

-1	Call failed, the failure reason can be obtained by the function SsmGetLastErrMsg
0	Call successful

Function Description:

Starts the tone detector. For more information, refer to Tone Detector in Chapter 1.

Note:

- By default this function is only applicable to analog trunk channels, magnet channels and recording channels.
- If required to manually start or stop the tone analysis for digital boards or VoIP boards, you should first configure DefaultToneCheckState=1 under the section [BoardId=x] in the file ShConfig.ini, and then call the function SsmStartToneAnalyze/SsmCloseToneAnalyze to start or stop the tone detector.

Related Information:

Driver version	SynCTI Ver. 2.0.0.0 or above
Header	shpa3api.h

Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmCloseToneAnalyze, SsmStart2ndToneAnalyzer

2.5.1.1.2 SsmCloseToneAnalyze

Stops the tone detector.

Format:

int SsmCloseToneAnalyze (int ch)

Parameter Description:

ch	Channel Number

Return Value:

-1	Call failed, the failure reason can be obtained by the function SsmGetLastErrMsg
0	Successful

Function Description:

Stops the tone detector. For more information, refer to Tone Detector in Chapter 1.

Note:

Related Information:

Driver version	SynCTI Ver. 2.0.0.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmStartToneAnalyze

2.5.1.1.3 SsmQueryOpToneAnalyze

Checks whether the channel includes the tone detector.

Format:

int SsmQueryOpToneAnalyze(int ch)

Parameter Description:

ch	Channel Number

Return Value:

-1	Call failed, the failure reason can be obtained by the function SsmGetLastErrMsg
0	No
1	Yes, it includes the tone detector

Function Description:

Checks whether the channel includes the tone detector.

Note:

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib

DLL	shp_a3.dll

Related Function:

2.5.1.2 Obtaining Detected Result

2.5.1.2.1 SsmGetToneAnalyzeResult

Obtains the detected result of the tone detector.

Format:

int SsmGetToneAnalyzeResult (int ch)

Parameter Description:

ch	Channel Number

Return Value:

-1	Failed.
0	Tone analysis is in progress.
1	Dial tone is detected.
2	Busy tone is detected.
3	Ringback tone is detected.
4	After the ringback tone is detected, the line keeps silent.
5	Silence is detected.
6	The answer signal is detected. For analog phone lines, it usually denotes that the call is answered (the called party picks up the phone).
7	The single frequency tone with F1 frequency is detected. The parameter F1, which is set by the function SsmSetVoiceFxPara, is often used to detect the fax tone.
8	The single frequency tone with F2 frequency is detected. The parameter F2, which is set by the function SsmSetVoiceFxPara, is often used to detect the fax tone.
9	The user-defined tone type is detected.
n>9	Some particular user-defined tones (e.g. SIT) are detected. For more information, see the section 'Enhanced Tone Detector'.

Function Description:

Obtains the detected result of the tone detector.

Note:

- By default this function is only applicable to analog trunk channels, magnet channels and recording channels;
- For digital boards, you should do the following configurations to analyze the tones: set the configuration item DefaultToneCheckState=1 under the section [BoardId＝x] in the file ShConfg.ini.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmClearToneAnalyzeResult

2.5.1.3 Clearing Detected Result

2.5.1.3.1 SsmClearToneAnalyzeResult

Clears the detected result of the tone detector.

Format:

int SsmClearToneAnalyzeResult (int ch)

Parameter Description:

ch	Channel Number

Return Value:

-1	Failed
0	Successful

Function Description:

Clears the detected result of the tone detector.

Note:

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmGetToneAnalyzeResult

2.5.2 FFT Functions

2.5.2.1 Setting Parameters

2.5.2.1.1 SsmSetPeakFrqDetectBW

Sets the bandwidth used to detect the peak frequency in the incoming call signal.

Format:

int SsmSetPeakFrqDetectBW(int ch, WORD nPeakBW)

Parameter Description:

ch	Channel Number
nPeakBW	Bandwidth (Hz) , range of value: 40~80, with the default value of 80Hz.

Return Value:

-1	Failed
0	Successful

Function Description:

Sets the bandwidth used to detect the peak frequency in the incoming call signal. For more information, refer to Tone Detector in Chapter 1.

Note:

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmQueryOpPeakFrqDetect, SsmGetPeakFrqDetectBW

2.5.2.2 Obtaining Parameters

2.5.2.2.1 SsmQueryOpPeakFrqDetect

Checks whether the channel supports the operation of detecting the peak frequency.

Format:

int SsmQueryOpPeakFrqDetect (int ch)

Parameter Description:

ch	Channel Number

Return Value:

-1	Failed
0	Not support
1	Support

Function Description:

Checks whether the channel supports the operation of detecting the peak frequency.

Note:

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function:

2.5.2.2.2 SsmGetPeakFrqDetectBW

Obtains the bandwidth parameter used to detect the peak frequency.

Format:

int SsmGetPeakFrqDetectBW(int ch)

Parameter Description:

ch	Channel Number

Return Value:

-1	Failed
≥0	The set value of the bandwidth

Function Description:

Obtains the bandwidth parameter used to detect the peak frequency.

Note:

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmQueryOpPeakFrqDetect, SsmSetPeakFrqDetectBW

2.5.2.3 Obtaining Detected Result

2.5.2.3.1 SsmGetPeakFrq

Obtains the peak frequency in the incoming call signal.

Format:

int SsmGetPeakFrq(int ch)

Parameter Description:

ch	Channel Number

Return Value:

-1	Failed
≥0	The peak frequency in the incoming call signal

Function Description:

Obtains the peak frequency in the incoming call signal. For more information, refer to 'Tone Detector' in Chapter 1.

Note:

- The event of E_CHG_PeakFrq can implement the same feature.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmQueryOpPeakFrqDetect, SsmSetPeakFrqDetectBW

2.5.2.3.2 SsmGetPeakFrqEnergy

Obtains the energy value of the peak frequency in the incoming call signal.

Format:

long SsmGetPeakFrqEnergy(int ch)

Parameter Description:

ch	Channel Number

Return Value:

-1	Failed
≥0	The energy value of the peak frequency in the incoming call signal

Function Description:

Obtains the energy value of the peak frequency in the incoming call signal. For more information, refer to 'Tone

Detector' in Chapter 1.

Note:

- The event of E_CHG_PeakFrq can implement the same feature.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmSetPeakFrqDetectBW, SsmGetPeakFrq

2.5.2.3.3 SsmGetOverallEnergy

Refer to SsmGetOverallEnergyAllCh

2.5.2.3.4 SsmGetOverallEnergyAllCh

Obtains the overall energy value of the incoming call signal after FFT transformation. SsmGetOverallEnergy is only applicable to a single channel, SsmGetOverallEnergyAllCh can simultaneously obtain the overall energy of multiple channels.

Format:

int SsmGetOverallEnergy(int ch)

int SsmGetOverallEnergyAllCh (int nBeginCh, int nChNum, PDWORD pdwEnergyTable)

Parameter Description:

ch	Channel Number
nBeginCh	Starting channel number
nChNum	Channel amount
PdwEnergyTable	Stores the energy value of the nChNum channels starting from nBeginCh. The space is allocated by the application

Return Value:

Function Name	Return Value	Description
SsmGetOverallEnergy	-1	Call failed
	≥0	Overall energy value
SsmGetOverallEnergyAllCh	-1	Call failed
	0	Successful

Function Description:

Obtains the overall energy value of the incoming call signal after FFT transformation.

Note:

- For more information about the unit conversion of overall energy to DB (decibel), refer to the 'FFT component' in 'Tone Detector' of Chapter 1.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib

DLL	shp_a3.dll

Related Function: SsmGetPeakFrqEnergy

2.5.3 Setting Parameters for Noise Filter

2.5.3.1 SsmSetMinVocDtrEnergy

Sets the threshold value for noise detection on the line.

Format:

int SsmSetMinVocDtrEnergy(int ch, DWORD dwMinVocDtrEnergy)

Parameter Description:

ch	Channel Number
dwMinVocDtrEnergy	Threshold value for noise detection, range of value: ≥700, with the default value of 100000. For more information about the unit conversion of this parameter to DB (decibel), refer to the 'FFT component' in 'Tone Detector' of Chapter 1.

Return Value:

-1	Call failed, the failure reason can be obtained by the functionSsmGetLastErrMsg
0	Successful

Function Description:

Sets the threshold value for noise detection on the line. If the energy of the incoming call signal on the line is less than the set value of dwMinVocDtrEnergy, it will be regarded as noise by the driver.

Note:

- The configuration item MinimumVoiceDetermineEnergy can implement the same feature.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmGetMinVocDtrEnergy

2.5.3.2 SsmGetMinVocDtrEnergy

Obtains the threshold value of energy for the driver to determine whether the signal on the line is noise or voice.

Format:

int SsmGetMinVocDtrEnergy(int ch, PDWORD pdwMinVocDtrEnergy)

Parameter Description:

ch	Channel Number
pdwMinVocDtrEnergy	Returns the threshold value of energy which determines whether the signal is noise or voice.

Return Value:

-1	Failed
0	Successful

Function Description:

Obtains the threshold value of energy for the driver to determine whether the signal on the line is noise or voice.

Note:

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmSetMinVocDtrEnergy

2.5.4 Call Progress Tone Detector Functions

2.5.4.1 Setting Working Status of 2nd Call Progress Tone Detector

2.5.4.1.1 SsmStart2ndToneAnalyzer

Starts the 2nd call progress tone detector.

Format:

int SsmStart2ndToneAnalyzer(int ch,BOOL bEnable)

Parameter Description:

ch	Channel Number
bEnable	The 2nd tone detector enabling switch. = TRUE: Start; = FALSE: Stop

Return Value:

-1	Call failed, the failure reason can be obtained by the function SsmGetLastErrMsg
0	Successful

Function Description:

Starts the 2nd call progress tone detector.

Note:

- The configuration item Enable2ndToneAnalyzer can implement the same feature.
- When the 2nd call progress tone detector is started, it occupies the FFT analyzer which is used by the function SsmSetVoiceFxPara, therefore the driver no longer detects the special tones designated by the function SsmSetVoiceFxPara.
- The 2nd call progress tone detector can be started only if the 1st call progress tone detector is turned on.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmGet2ndToneAnalyzerState

2.5.4.1.2 SsmGet2ndToneAnalyzerState

Checks the working status of the 2nd call progress tone detector.

Format:

int SsmGet2ndToneAnalyzerState(int ch)

Parameter Description:

ch	Channel Number

Return Value:

-1	Failed
0	Turned off
1	Turned on

Function Description:

Checks the working status of the 2nd call progress tone detector.

Note:

- By default this function is only applicable to analog trunk channels, magnet channels and recording channels.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmStart2ndToneAnalyzer

2.5.4.2 Obtaining Result of 2nd Call Progress Tone Detector

2.5.4.2.1 SsmGet2ndToneAnalyzeResult

Obtains the result of the 2nd call progress tone detector.

Format:

int SsmGet2ndToneAnalyzeResult(int ch)

Parameter Description:

ch	Channel Number

Return Value:

-1	Failed
0	Tone analysis is in progress
1	Dial tone detected
2	Busy tone detected
3	Ringback tone detected
4	The line keeps silent after the detection of ringback tone
5	Silence detected
6	Answer signal detected, which usually indicates for analog phone lines that the called party answers (picks up) the call.
7	Detection of single tones with the frequency of F_1. F_1, which can be set via the function SsmSetVoiceFxPara, is often used for detection of fax tones.
8	Detection of single tones with the frequency of F_2. F_2, which can be set via the function SsmSetVoiceFxPara, is often used for detection of fax tones.
9	User-defined tone detected
>9	Particular user-defined tone detected (e.g. SIT tone). For more information, refer to the section Enhanced Tone Detector in Chapter 1.

Function Description:

Obtains the result of the 2nd call progress tone detector.

Note:

- If the result of the second tone detector is the same as that of the first tone detector, by default the driver will regard that only one tone detector actually works.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmClear2ndToneAnalyzeResult

2.5.4.2.2 SsmClear2ndToneAnalyzeResult

Clears the result of the 2nd call progress tone detector.

Format:

int SsmClear2ndToneAnalyzeResult(int ch)

Parameter Description:

ch	Channel Number

Return Value:

-1	Failed
0	Successful

Function Description:

Clears the result of the 2nd call progress tone detector.

Note:

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmGet2ndToneAnalyzeResult

2.5.4.3 Setting Parameters for Frequency Detector

2.5.4.3.1 SsmSetTonePara

Refer to SsmSet2ndTonePara

2.5.4.3.2 SsmSet2ndTonePara

Sets the operating parameters of the frequency detector in the call progress tone detector. SsmSetTonePara and SsmSet2ndTonePara are used to set the frequency parameters of the 1st and 2nd call progress tone detector respectively.

Format:

int SsmSetTonePara(int ch, WORD wF1, WORD wBw1, WORD wF2, WORD wBw2, DWORD dwMinEnrgRatio)

int SsmSet2ndTonePara(int ch, WORD wF1, WORD wBw1, WORD wF2, WORD wBw2, DWORD dwMinEnrgRatio)

Parameter Description:

ch	Channel Number
wF1	The 1st center frequency (Hz), range of value: 300~3400
wBw1	The bandwidth (Hz) of the 1st center frequency, range of value: 40~120
wF2	The 2nd center frequency (Hz), range of value: 0 or 300~3400
wBw2	The bandwidth (Hz) of the 2nd center frequency, range of value: 0 or 40~120
dwMinEnrgRatio	Threshold value, calculated by percentage (%), range of value: 15~80

Return Value:

-1	Call failed, the failure reason can be obtained by the function SsmGetLastErrMsg
0	Successful

Function Description:

Sets the operating parameters of the frequency detector in the call progress tone detector. For more information, refer to Call Progress Tone Detector in Chapter 1.

Note:

- The configuration item TonePara can implement the same feature as the function SsmSetTonePara, with the default value of 450Hz;
- The configuration item 2ndTonePara can implement the same feature as the function SsmSet2ndTonePara.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmGetTonePara, SsmGet2ndTonePara

2.5.4.3.3 SsmGetTonePara

Refer to SsmGet2ndTonePara

2.5.4.3.4 SsmGet2ndTonePara

Obtains the operating parameters of the frequency detector in the call progress tone detector. SsmGetTonePara and SsmGet2ndTonePara are used to obtain the frequency parameters of the 1st and 2nd call progress tone detector respectively.

Format:

int SsmGetTonePara(int ch, PWORD pwF1, PWORD pwBw1, PWORD pwF2, PWORD pwBw2, PDWORD pdwMinEnrgRatio)

int SsmGet2ndTonePara(int ch, PWORD pwF1, PWORD pwBw1, PWORD pwF2, PWORD pwBw2, PDWORD pdwMinEnrgRatio)

Parameter Description:

ch	Channel Number

pwF1	Returns the center frequency (Hz) of the 1st tone
pwBw1	Returns the bandwidth (Hz) of the 1st tone
pwF2	Returns the center frequency (Hz) of the 2nd tone
pwBw2	Returns the bandwidth (Hz) of the 2nd tone
pdwMinEnrgRatio	Returns the preset threshold percentage of the in-band energy in the overall energy for judging the signal tones

Return Value:

-1	Failed
0	Successful

Function Description:

Obtains the operating parameters of the frequency detector in the call progress tone detector.

Note:

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmSetTonePara, SsmSet2ndTonePara

2.5.4.4 Setting Parameters for Dial Tone Detector

2.5.4.4.1 SsmSetIsDialToneDtrTime

Refer to SsmSet2ndIsDialToneDtrTime

2.5.4.4.2 SsmSet2ndIsDialToneDtrTime

Sets the minimum duration of the dial tone detector. SsmSetIsDialToneDtrTime and SsmSet2ndIsDialToneDtrTime set the 1st and 2nd call progress tone detector respectively.

Format:

int SsmSetIsDialToneDtrTime(int ch, WORD wIsDialToneDtrTime)

int SsmSet2ndIsDialToneDtrTime(int ch, WORD wIsDialToneDtrTime)

Parameter Description:

ch	Channel Number
wIsDialToneDtrTime	The minimum duration (ms) of the dial tone, range of value: ≥1300ms

Return Value:

-1	Call failed, the failure reason can be obtained by the function SsmGetLastErrMsg
0	Successful

Function Description:

Sets the minimum duration of the Dial Tone Detector.

Note:

- The configuration item IsDialToneDetermineTime can implement the same feature as the function SsmSetIsDialToneDtrTime.

- The configuration item 2ndIsDialToneDetermineTime can implement the same feature as the function SsmSet2ndIsDialToneDtrTime.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmGetIsDialToneDtrTime, SsmGet2ndIsDialToneDtrTime

2.5.4.4.3 SsmGetIsDialToneDtrTime

Refer to SsmGet2ndIsDialToneDtrTime

2.5.4.4.4 SsmGet2ndIsDialToneDtrTime

Obtains the minimum duration of the Dial Tone Detector. SsmGetIsDialToneDtrTime and SsmGet2ndIsDialToneDtrTime obtain the parameters of the 1st and 2nd call progress tone detector respectively.

Format:

int SsmGetIsDialToneDtrTime(int ch, PWORD pwIsDialToneDtrTime)

int SsmGet2ndIsDialToneDtrTime(int ch, PWORD pwIsDialToneDtrTime)

Parameter Description:

ch	Channel Number
pwIsDialToneDtrTime	Returns the minimum duration (ms)

Return Value:

-1	Failed
0	Successful

Function Description:

Obtains the minimum duration of the Dial Tone Detector.

Note:

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmSetIsDialToneDtrTime, SsmSet2ndIsDialToneDtrTime

2.5.4.5 Setting Parameters for Ringback Tone Detector

2.5.4.5.1 SsmSetRingEchoTonePara

Refer to SsmSet2ndRingEchoTonePara

2.5.4.5.2 SsmSet2ndRingEchoTonePara

Sets the signal durations at on and off states for the ringback tone detector. SsmSetRingEchoTonePara is applicable to the 1st call progress tone detector, SsmSet2ndRingEchoTonePara is applicable to applicable to the 2nd call progress tone detector.

Format:

int SsmSetRingEchoTonePara(int ch, WORD wTon, WORD wToff)

int SsmSet2ndRingEchoTonePara(int ch, WORD wTon, WORD wToff)

Parameter Description:

ch	Channel Number
wTon	The duration (ms) of the tone at on state, range of value: 500~2500
wToff	The duration (ms) of the tone at off state, range of value: 1500 ~ 6000

Return Value:

-1	Call failed, the failure reason can be obtained by the function SsmGetLastErrMsg
0	Successful

Function Description:

Sets the signal durations respectively at on and off states for the Ringback Tone Detector.

Note:

- The parameters set by SsmSetRingEchoTonePara can also be set by the configuration item RingEchoTonePara. Default value: The signal duration at on state is 1000ms, the signal duration at off state is 4000ms. The parameters set by SsmSet2ndRingEchoTonePara can also be set by the configuration item 2ndRingEchoTonePara. Default value: The signal duration at on state is 1000ms, the signal duration at off state is 4000ms.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmGetRingEchoTonePara, SsmGet2ndRingEchoTonePara

2.5.4.5.3 SsmGetRingEchoToneTime

Obtains the duration of the ringback tone.

Format:

int SsmGetRingEchoToneTime(int ch)

Parameter Description:

ch	Channel Number

Return Value:

-1	Failed
≥0	Duration time (ms)

Function Description:

Obtains the duration of the ringback tone.

Note:

- This function is only applicable to the analog trunk channel.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function:

2.5.4.5.4 SsmGetRingEchoTonePara

Refer to SsmGet2ndRingEchoTonePara

2.5.4.5.5 SsmGet2ndRingEchoTonePara

Obtains the signal durations respectively at on and off states for the ringback tone detector. SsmGetRingEchoTonePara and SsmGet2ndRingEchoTonePara are applicable to the 1st and 2nd call progress tone detector respectively.

Format:

int SsmGetRingEchoTonePara(int ch, PWORD pwTon, PWORD pwToff)

int SsmGet2ndRingEchoTonePara(int ch, PWORD pwTon, PWORD pwToff)

Parameter Description:

ch	Channel Number
pwTon	Returns the signal duration (ms) at on state
pwToff	Returns the signal duration (ms) at off state

Return Value:

-1	Failed
0	Successful

Function Description:

Obtains the signal durations respectively at on and off states for the ringback tone detector.

Note:

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmSetRingEchoTonePara, SsmSet2ndRingEchoTonePara

2.5.4.6 Setting Parameters for Busy Tone Detector

2.5.4.6.1 SsmSetBusyTonePeriodEx

Refer to SsmSet2ndBusyTonePeriod

2.5.4.6.2 SsmSetBusyTonePeriod

Refer to SsmSet2ndBusyTonePeriod

2.5.4.6.3 SsmSet2ndBusyTonePeriod

Sets the busy tone cycle of busy tone detector in the call progress tone detector. SsmSetBusyTonePeriod is applicable to the 1st call progress tone detector, SsmSet2ndBusyTonePeriod is applicable to the 2nd call progress tone detector, and SsmSetBusyTonePeriodEx is applicable to both.

Format:

int SsmSetBusyTonePeriodEx(int ch, int nCptdNo, WORD wMax, PWORD pwPeriod)

int SsmSetBusyTonePeriod(int ch, WORD wPeriod)

int SsmSet2ndBusyTonePeriod(int ch, WORD wPeriod)

Parameter Description:

ch	Channel Number
nCptdNo	=1: Sets the 1st call progress tone detector =2: Sets the 2nd call progress tone detector
wMax	Number of different types of busy tone cycles, range of value: 1~4
pwPeriod	The pointer storing the parameter of busy tone cycle, it's storage space is allocated by the application
wPeriod	Busy tone cycle (ms), range of value: 200~2000. This parameter can also be specified in the configuration file, with the default value of 700ms (350ms/ON,350ms/OFF)

Return Value:

-1	Failed
0	Successful

Function Description:

Sets the busy tone cycle of Busy Tone Detector in the Call Progress Tone Detector. SsmSetBusyTonePeriodEx is a new version function which supports the configuration of at most 4 different busy tone cycles simultaneously. For more information, refer to 'Busy Tone Detector'.

Note:

- The busy tone cycle can also be set via the configuration item BusyTonePeriod or 2ndBusyTonePeriod .
- It's recommended to use the function SsmSetBusyTonePeriodEx.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmGetBusyTonePeriodEx, SsmGetBusyTonePeriod, SsmGet2ndBusyTonePeriod

Sample Code:

The application needs to simultaneously detect 4 types of busy tones (with the cycles of 700ms, 800ms, 900ms and 1000ms respectively) in the 1st call progress tone detector and only detects the busy tone with 700ms period in the 2nd call progress tone detector, below is the implementing sample code:

```
WORD BusyTonePeriod[4] = {700,800,900,1000};
SsmSetBusyTonePeriodEx(ch, 1, 4, BusyTonePeriod);
SsmSet2ndBusyTonePeriod(ch, 700);
```

2.5.4.6.4 SsmSetIsBusyToneDtrCnt

Refer to SsmSet2ndIsBusyToneDtrCnt

2.5.4.6.5 SsmSet2ndIsBusyToneDtrCnt

Sets the minimum number of the busy tone cycles. SsmSetIsBusyToneDtrCnt is applicable to the 1st call progress tone detector, SsmSet2ndIsBusyToneDtrCnt is applicable to the 2nd call progress tone detector.

Format:

int SsmSetIsBusyToneDtrCnt(int ch, WORD wIsBusyToneDtrCnt)

int SsmSet2ndIsBusyToneDtrCnt(int ch, WORD wIsBusyToneDtrCnt)

Parameter Description:

ch	Channel Number
wIsBusyToneDtrCnt	The minimum number of the busy tone cycles, range of value: 1~50

Return Value:

-1	Call failed, the failure reason can be obtained by the function SsmGetLastErrMsg
0	Successful

Function Description:

Sets the minimum number of the busy tone cycles. For more information, refer to Busy Tone Detector.

Note:

- The configuration items IsBusyToneDetermineCount and 2ndIsBusyToneDetermineCount can implement the same feature.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmGetIsBusyToneDtrCnt, SsmGet2ndIsBusyToneDtrCnt

2.5.4.6.6 SsmGetBusyTonePeriodEx

Refer to SsmGet2ndBusyTonePeriod

2.5.4.6.7 SsmGetBusyTonePeriod

Refer to SsmGet2ndBusyTonePeriod

2.5.4.6.8 SsmGet2ndBusyTonePeriod

Obtains the set value judging the busy tone cycle. SsmGetBusyTonePeriod is applicable to the 1st call progress tone detector, SsmGet2ndBusyTonePeriod is applicable to the 2nd call progress tone detector, and SsmGetBusyTonePeriodEx is applicable to both.

Format:

int SsmGetBusyTonePeriodEx(int ch, int nCptdNo, PWORD pwPeriod)

int SsmGetBusyTonePeriod(int ch, PWORD pwBusyTonePeriod)

int SsmGet2ndBusyTonePeriod(int ch, PWORD pwBusyTonePeriod)

Parameter Description:

ch	Channel Number
nCptdNo	=1: sets the 1st call progress tone detector =2: sets the 2nd call progress tone detector
pwPeriod	Returns the pointer of the set value of busy tone cycle (ms), the storage space is allocated by the application, the size is 4 WORD
pwBusyTonePeriod	Returns the set value of busy tone cycle (ms)

Return Value:

-1	Failed
0	Successful
>0	The preset number of busy tone cycles

Function Description:

Obtains the set value judging the busy tone cycle. For more information, refer to 'Busy Tone Detector'.

Note:

- It's recommended to use the function SsmGetBusyTonePeriodEx.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmSetBusyTonePeriodEx, SsmSetBusyTonePeriod, SsmSet2ndBusyTonePeriod

2.5.4.6.9 SsmGetIsBusyToneDtrCnt

Refer to SsmGet2ndIsBusyToneDtrCnt

2.5.4.6.10SsmGet2ndIsBusyToneDtrCnt

Obtains the set value of the minimum number of busy tone cycles. SsmGetIsBusyToneDtrCnt is applicable to the 1st call progress tone detector, SsmGet2ndIsBusyToneDtrCnt is applicable to the 2nd call progress tone detector.

Format:

int SsmGetIsBusyToneDtrCnt(int ch, PWORD pwIsBusyToneDtrCnt)

int SsmGet2ndIsBusyToneDtrCnt(int ch, PWORD pwIsBusyToneDtrCnt)

Parameter Description:

ch	Channel Number
pwIsBusyToneDtrCnt	Returns the set value of the minimum number of busy tone cycles

Return Value:

-1	Failed
0	Successful

Function Description:

Obtains the set value of the minimum number of busy tone cycles.

Note:

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmSetIsBusyToneDtrCnt, SsmSet2ndIsBusyToneDtrCnt

2.5.4.7 Obtaining Detected result

2.5.4.7.1 SsmGetBusyToneLen

Refer to SsmGet2ndBusyToneLen

2.5.4.7.2 SsmGet2ndBusyToneLen

Obtains the duration time of busy tone. SsmGetBusyToneLen is applicable to the 1st call progress tone detector, SsmGet2ndBusyToneLen is applicable to the 2nd call progress tone detector.

Format:

int SsmGetBusyToneLen(int ch)

int SsmGet2ndBusyToneLen(int ch)

Parameter Description:

ch	Channel Number

Return Value:

-1	Failed
≥0	The duration time (s) of busy tone

Function Description:

Obtains the duration time of busy tone. For more information, refer to 'Busy Tone Detector' in Chapter 1.

Note:

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmGetBusyAMDToneCount

2.5.4.7.3 SsmGetBusyAMDToneCount

Refer to SsmGet2ndBusyAMDToneCount

2.5.4.7.4 SsmGet2ndBusyAMDToneCount

Obtains the number of busy tone cycles. SsmGetBusyAMDToneCount is applicable to the 1st call progress tone detector, SsmGet2ndBusyAMDToneCount is applicable to the 2nd call progress tone detector.

Format:

int SsmGetBusyAMDToneCount(int ch)

int SsmGet2ndBusyAMDToneCount(int ch)

Parameter Description:

ch	Channel Number

Return Value:

-1	Failed
≥0	The number of continuous busy tone cycles

Function Description:

Obtains the number of busy tone cycles.

Note:

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmGetBusyToneLen

2.5.4.8 Obtaining Result Gained by Using Back-to-Back Busy Tone Detection

2.5.4.8.1 SsmGetBusyToneEx

Obtains the detected result of the busy tone detector which adopts the back-to-back busy tone detecting algorithm.

Format:

int SsmGetBusyToneEx(int ch)

Parameter Description:

ch	Channel Number

Return Value:

-1	Failed
0	No busy tone detected
1	Busy tone detected

Function Description:

Obtains the detected result of the busy tone detector which adopts the back-to-back busy tone detecting algorithm. For more information, refer to 'Busy Tone Detector' in Chapter 1.

Note:

- The event of E_CHG_BusyToneEx can implement the same feature.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmClearToneAnalyzeResult

2.5.4.9 Obtaining Result of Selcall Tone Detector

2.5.4.9.1 SsmGetSelcallToneStr

Obtains the Selcall Tone characters saved in the buffer area of the Selcall Tone detector.

Format:

int SsmGetSelcallToneStr(int ch, LPSTR pszBuf)

Parameter Description:

ch	Channel Number
pszBuf	The pointer storing the received Selcall Tone string. Its storage space is allocated by the application and its length must be greater than 256 bytes.

Return Value:

Function Name	Return Value	Description
SsmGetDtmfStr	-1	Call Failed
	≥0	The number of characters in the Selcall Tone detector buffer
SsmGetDtmfStrA	NULL	Call failed

Function Description:

Obtains the standard ASCII-formatted Selcall Tone characters saved in the buffer area of the Selcall Tone detector.

Note:

- The buffer area of the Selcall Tone detector will not be cleared by the driver when this function is invoked. It will be cleared only when the function SsmClearSelcallToneBuf is called.

Related Information:

Driver version	SynCTI Ver. 5.3 3.0 or above

Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmClearSelcallToneBuf, SsmGetSelcallToneLen

Related Event: E_CHG_RCV_SELCALL

2.5.4.9.2 SsmGetSelcallToneLen

Obtains the number of DTMF characters saved in the buffer area of the Selcall Tone detector.

Format:

int SsmGetSelcallToneLen(int ch)

Parameter Description:

ch	Channel Number

Return Value:

-1	Failed. The failure reason can be acquired via the function SsmGetLastErrMsg
≥0	The number of Selcall Tone characters saved in the buffer area of the Selcall Tone detector

Function Description:

Obtains the number of DTMF characters saved in the buffer area of the Selcall Tone detector.

Note:

Related Information:

Driver version	SynCTI Ver. 5.3.3.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmClearSelcallToneBuf, SsmGetSelcallToneStr

Related Event: E_CHG_RCV_SELCALL

2.5.4.9.3 SsmClearSelcallToneBuf

Clears the Selcall Tone reception buffer area in the driver.

Format:

Int SsmClearSelcallToneBuf(int ch)

Parameter Description:

ch	Channel Number

Return Value:

-1	Failed. The failure reason can be acquired via the function SsmGetLastErrMsg.
0	Successful

Function Description:

Clears the Selcall Tone reception buffer area in the driver.

Related Information:

Driver version	SynCTI Ver. 5.3.3.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmGetSelcallToneLen, SsmGetSelcallToneStr

2.5.5 Fax Progress Tone Detector Functions

2.5.5.1 Setting Working Parameters

2.5.5.1.1 SsmSetVoiceFxPara

Sets the operating parameters of the fax progress tone detector.

Format:

int SsmSetVoiceFxPara(int ch, WORD wSelFx, WORD wFx,WORD wFxBW, DWORD dwIsVocFxRatio, WORD wIsVocFxDtrTime)

Parameter Description:

ch	Channel Number
wSelFx	Selects the fax progress tone detector, range of value: 1: sets the parameter of 1st fax progress tone detector 2: sets the parameter of 2nd fax progress tone detector
wFx	Center frequency (Hz), range of value: 300~3400
wFxBW	Bandwidth (Hz), range of value: 40~120
dwIsVocFxRatio	The minimum threshold percentage (%) of in-band energy in overall energy, range of value: 15~80
wIsVocFxDtrTime	The minimum duration time (ms) , range of value: 16~1024, it must be integral times of 16

Return Value:

-1	Call failed, the failure reason can be obtained by the function SsmGetLastErrMsg
0	Successful

Function Description:

Sets the operating parameters of the fax progress tone detector. For more information, refer to 'Fax Progress Tone Detector' in Chapter 1.

Note:

- The configuration items VoiceFreqF1Para and VoiceFreqF2Para can implement the same feature.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmGetVocFxFlag, SsmGetVoiceFxPara

2.5.5.1.2 SsmGetVoiceFxPara

Obtains the operating parameters of the fax progress tone detector.

Format:

int SsmGetVoiceFxPara(int ch, WORD wSelFx, PWORD pwFx, PWORD pwFxBW, PDWORD pdwIsVocFxRatio, PWORD pwIsVocFxDtrTime)

Parameter Description:

ch	Channel Number
wSelFx	Selects the fax progress tone detector, range of value: 1: sets the parameter of 1st fax progress tone detector 2: sets the parameter of 2nd fax progress tone detector
pwFx	Returns the center frequency (Hz)
pwFxBW	Returns the bandwidth (Hz)
pdwIsVocFxRatio	Returns the minimum threshold percentage (%) of in-band energy in overall energy
wIsVocFxDtrTime	Returns the minimum duration (ms)

Return Value:

-1	Failed
0	Successful

Function Description:

Obtains operating parameters of the fax progress tone detector.

Note:

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmSetVoiceFxPara

2.5.5.2 Obtaining Detected Result

2.5.5.2.1 SsmGetVocFxFlag

Obtains the detected result of the fax progress tone detector.

Format:

int SsmGetVocFxFlag(int ch, int nSelFx, BOOL bClear)

Parameter Description:

ch	Channel Number
nSelFx	Selects the frequency of voice to be detected: =1: selects F1; =2: selects F2
bClear	Indicates whether the driver clears the detected result after reading the designated sign =TRUE: clear the detected result; =FALSE: not clear the detected result

Return Value:

-1	Failed
0	The single tone voice with specified frequency is not detected
1	The single tone voice with specified frequency is detected

Function Description:

Obtains the detected result of the fax progress tone detector. For more information, refer to 'Fax Progress Tone Detector'.

Note:

- It's recommended to use the event of E_CHG_VocFxFlag to implement the same feature.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmSetVoiceFxPara

2.5.6 AMD Function

2.5.6.1 Obtaining and Clearing Result Gained by AMD

2.5.6.1.1 SsmGetAMDResult

Obtains the result of AMD detection.

Format:

int SsmGetAMDResult (int ch)

Parameter Description:

ch	Channel Number

Return Value:

-1	Call failed or AMD invalid
0	The detection task is over because the phone is picked up by man
1	Signal tone detected and the detection task continues
2	Ringback music or cue tone detected and the detection task continues
3	The detection task is over due to timeout
4	The detection task is over because the line keeps silent after the detection of the ringback music or cue tone
5	The detection task is over because the line keeps silent after the detection of the dial tone
6	The detection task is over due to the detection of busy tone
7	The detection task is over due to the detection of fax
8	The detection task is over due to the detection of cue tone

Function Description:

Obtains the result of AMD detection.

Note:

- It is recommended to use the event E_CHG_AMD to implement the same feature.
- The return value 8 is valid only when the configuration item EnableAMDBeep is set to 1, and in such case, the return value 2 indicates the detection task is over due to the detection of ringback music.

Related Information:

Driver version	SynCTI Ver. 5.3.3.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

2.5.6.1.2 SsmClearAMDResult

Clears the detection result of AMD.

Format:

int SsmClearAMDResult (int ch)

Parameter Description:

ch	Channel Number

Return Value:

-1	Failed
0	Successful

Function Description:

Clears the detection result of AMD.

Note:

- It is recommended to call this function before starting AMD.

Related Information:

Driver version	SynCTI Ver. 5.0.3.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

2.5.6.1.3 SsmControlAMD

Manually turns on or turns off the AMD detector.

Format:

int WINAPI SsmControlAMD(int ch,BOOL bStart)

Parameter Description:

ch	Channel Number
bStart	Indicates whether to turn on the control switch of the AMD detector. TRUE： turn on FALSE：turn off

Return Value:

-1	Failed
0	Successful

Function Description:

Manually turns on or turns off the AMD detector.

Note:

- The channel state will also have an effect on the turnon and turnoff of AMD detector when this function is used.

Related Information:

Driver Version	SynCTI Ver. 5.0.3.0 or above
Header	Shpa3api.h
Library	shp_a3.lib

DLL	shp_a3.dll

2.5.6.1.4 SsmSetAMDPara

Sets the parameters used to judge whether the phone is picked up by a man or not for AMD detector.

Format:

int SsmSetAMDPara(int ch, DWORD AMDTimeA, DWORD AMDTimeB, DWORD AMDTimeC, DWORD AMDTimeD)

Parameter Description:

ch	Channel Number
AMDTimeA	The shortest silence duration before the phone is picked up by a man, calculated by ms
AMDTimeB	The shortest duration for the man who picked up the phone to speak, calculated by ms
AMDTimeC	The longest duration for the man who picked up the phone to speak, calculated by ms
AMDTimeD	The shortest silence duration after the phone is picked up by a man, calculated by ms

Return Value:

-1	Failed. The failure reason can be acquired via the function SsmGetLastErrMsg.
0	Successful

Function Description:

Sets the parameters used to judge whether the phone is picked up by a man or not for AMD detector, such as: the shortest silence duration before the phone is picked up by a man, the shortest and the longest duration for the man who picked up the phone to speak, as well as the shortest silence duration after the phone is picked up by a man.

Note:

- The configuration items AMDTimeA, AMDTimeB, AMDTimeC, AMDTimeD can implement the same feature.

Related Information:

Driver Version	SynCTI Ver. 5.3.3.1 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

2.6 DTMF Detector Functions

DTMF-reception buffer area is the internal buffer area in the driver used to receive DTMF strings. Its size can be specified in the configuration file and the default value is 200. It is necessary that the DTMF characters are read by the application in time before the buffer area overflows. Otherwise, the driver receives a new character, the first received DTMF character will be discarded.

For the station channel or recording channel, when the driver detects pickup or hangup, it will automatically clear the DTMF-reception buffer area. For other types of channels, the driver will automatically clear the buffer area upon the reception of the hangup command.

All the functions in this section don't support magnet channels.

2.6.1 Starting/Terminating DTMF Detector

2.6.1.1 SsmEnableRxDtmf

Starts or stops the DTMF detector.

Format:

int SsmEnableRxDtmf(int ch, BOOL bRun)

Parameter Description:

ch	Channel Number
bRun	The control switch determining whether to start the DTMF detector TRUE: start FALSE: stop

Return Value:

-1	Failed. The failure reason can be acquired via the function SsmGetLastErrMsg
0	Successful

Function Description:

Starts or stops the DTMF detector.

Note:

- The driver automatically enables or disables the DTMF detector according to the channel state transition. Only if you want to control the switch manually does this function need to be invoked. Note that in SS1 mode, only when the application controls the call process by itself can this function be invoked.
- This function is only applicable to SHD, DST and SHN series.
- The configuration item AlwaysEnableRxDtmf can implement the same feature.
- This function goes invalid once the channel state changes.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmSetRxDtmfHandler, SsmGetDtmfStr

2.6.2 Setting Callback Functions

2.6.2.1 SsmSetRxDtmfHandler

Sets the callback function when DTMF Detector outputs DTMF signals.

Format:

int SsmSetRxDtmfHandler(int ch, RXDTMFHANDLER pfnOnRcvDtmf, PVOID pV)

Parameter Description:

ch	Channel Number
pfnOnRcvDtmf	The pointer of the callback function. The type of the callback function is: RXDTMFHANDLER, it's declared in shpa3api.h as: typedef void (*RXDTMFHANDLER) (int ch, char cDtmf, int nDTStatus, PVOID pV); In the above format, ch: Channel Number; cDtmf: DTMF character (ASCII formatted) nDTStatus: DTMF signal type, 0 means the DTMF signal disappears, 1 means

	pV:	the DTMF signal appears; The pointer of PVOID type. It's usually used by the application to pass its data structure to the callback function. This pointer is directly passed to the callback function by the application
pV	The pointer of PVOID type. The application passes the needed object or data structure to the callback function via this parameter	

Return Value:

-1	Failed. The failure reason can be acquired via the function SsmGetLastErrMsg
0	Successful

Function Description:

Sets the callback function when DTMF Detector outputs DTMF signals.

Note:

- Each time when DTMF detector detects a complete DTMF character, it will invoke the callback function set by this function twice: upon the appearance and disappearance of the DTMF signal respectively. For more information, refer to DTMF Detector in Chapter 1.
- The received DTMF character will still be put into the buffer area of the DTMF detector.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function:

2.6.3 Obtaining DTMF Digits

2.6.3.1 SsmClearRxDtmfBuf

Clears the DTMF-reception buffer area in the driver.

Format:

int SsmClearRxDtmfBuf(int ch)

Parameter Description:

ch	Channel Number

Return Value:

-1	Failed. The failure reason can be acquired via the function SsmGetLastErrMsg.
0	Successful

Function Description:

Clears the DTMF-reception buffer area in the driver.

Note:

- When the channel is transferred to the state of standby, the driver will automatically clear the buffer area of the DTMF detector.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmGetDtmfStr

2.6.3.2 SsmGetDtmfStr

Refer to SsmGetDtmfStrA

2.6.3.3 SsmGetDtmfStrA

Obtains the DTMF characters saved in the buffer area of the DTMF detector.

Format:

int SsmGetDtmfStr(int ch, LPSTR pszDtmf)

char* SsmGetDtmfStrA(int ch)

Parameter Description:

ch	Channel Number
pszDtmf	The pointer storing the received DTMF string, its storage space being allocated by the application

Return Value:

Function Name	Return Value	Description
SsmGetDtmfStr	-1	Failed
	≥0	The number of characters in the DTMF detector
SsmGetDtmfStrA	NULL	Call failed
	Other	Returns the pointer which points to the buffer area of the DTMF detector

Function Description:

Obtains the standard ASCII-formatted DTMF characters saved in the buffer area of the DTMF Detector. SsmGetDtmfStr and SsmGetDtmfStrA are the same in features except for the ways of returning the buffer area of the DTMF detector.

Note:

- When this function is invoked, the driver will not clear the buffer area of the DTMF detector.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	Shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmGetRxDtmfLen, SsmGet1stDtmf, SsmGet1stDtmfClr, SsmGetLastDtmf

2.6.3.4 SsmGetRxDtmfLen

Obtains the number of DTMF characters saved in the buffer area of the DTMF detector.

Format:

int SsmGetRxDtmfLen(int ch)

Parameter Description:

ch	Channel Number

Return Value:

-1	Failed. The failure reason can be acquired via the function SsmGetLastErrMsg
≥0	The number of DTMF characters saved in the buffer area of the DTMF detector

Function Description:

Obtains the number of DTMF characters saved in the buffer area of the DTMF detector.

Note:

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmGetDtmfStr, SsmGetDtmfStrA

2.6.3.5 SsmGet1stDtmf

Refer to SsmGet1stDtmfClr

2.6.3.6 SsmGet1stDtmfClr

Obtains the first received DTMF character in the buffer area of the DTMF detector.

Format:

int SsmGet1stDtmf(int ch, char* pcDtmf)

int SsmGet1stDtmfClr(int ch, char* pcDtmf)

Parameter Description:

ch	Channel Number
pcDtmf	The initial address of the buffer area which stores the DTMF characters

Return Value:

-1	Failed. The failure reason can be acquired via the function SsmGetLastErrMsg
0	The buffer area of the DTMF detector is empty
1	The buffer area of the DTMF detector is not empty

Function Description:

Obtains the first received DTMF character in the buffer area of the DTMF detector. After obtaining the DTMF character, the function SsmGet1stDtmf will leave it in the buffer area, while the function SsmGet1stDtmfClr will clear it from the buffer.

Note:

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	Shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmGetDtmfStr, SsmGetDtmfStrA

2.6.3.7 SsmGetLastDtmf

Obtains the last received DTMF character in the buffer area of the DTMF detector.

Format:

int SsmGetLastDtmf(int ch, char* pcDtmf)

Parameter Description:

ch	Channel Number
pcDtmf	The initial address of the buffer area storing the DTMF characters

Return Value:

-1	Failed. The failure reason can be acquired via the function SsmGetLastErrMsg
0	The buffer area of the DTMF detector is empty
1	The buffer area of the DTMF detector is not empty

Function Description:

Obtains the last received DTMF character in the buffer area of the DTMF detector.

Note: None

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmGetRxDtmfLen, SsmGet1stDtmfClr, SsmGetDtmfStr, SsmGetDtmfStrA

2.6.4 Using WaitDtmf

2.6.4.1 SsmSetWaitDtmf

Refer to SsmSetWaitDtmfExB

2.6.4.2 SsmSetWaitDtmfEx

Refer to SsmSetWaitDtmfExB

2.6.4.3 SsmSetWaitDtmfExA

Refer to SsmSetWaitDtmfExB

2.6.4.4 SsmSetWaitDtmfExB

Starts the WaitDtmf task. The precision of the maximum waiting time of SsmSetWaitDtmf is 1 second, while that of SsmSetWaitDtmfEx and SsmSetWaitDtmfExA is 1 millisecond; SsmSetWaitDtmfEx only supports one terminating character, while SsmSetWaitDtmfExA supports multiple terminating characters. SsmSetWaitDtmfExB implements the same feature as SsmSetWaitDtmfExA except that it supports a larger value range for the parameter wTimeOut2.

Format:

int SsmSetWaitDtmf(int ch, WORD wTimeOut1, WORD wMaxLen, char cEndChar, BOOL bWithEndChar)
int SsmSetWaitDtmfEx(int ch, WORD wTimeOut2, WORD wMaxLen, char cEndChar, BOOL bWithEndChar)
int SsmSetWaitDtmfExA(int ch, WORD wTimeOut2, WORD wMaxLen, char* szEndChar, BOOL bWithEndChar)
int SsmSetWaitDtmfExB(int ch, DWORD wTimeOut2, WORD wMaxLen, char* szEndChar, BOOL bWithEndChar)

Parameter Description:

ch	Channel Number
wTimeOut1	Maximum waiting time (s), with the value range of 1~524
wTimeOut2	Maximum waiting time (ms), with the value range of 0~65535 (SsmSetWaitDtmfExA) or 0~2^{32}-1 (SsmSetWaitDtmfExB)
wMaxLen	The maximum length of the DTMF to be received, range of value: 1~350
cEndChar	Terminating character. If the driver receives this character, it will stop the task of WaitDtmf
szEndChar	Terminating character set, at most 16 characters. If the driver receives any character in the set, it will stop the task of WaitDtmf

bWithEndChar	After the WaitDtmf task is finished, this parameter determines whether the string returned from the driver to the application includes the received terminating character: =TRUE: include =FALSE: not include

Return Value:

-1	Failed. The failure reason can be acquired via the function SsmGetLastErrMsg
0	Successful

Function Description:

Submits a task of receiving specified DTMF string, i.e. the WaitDtmf task. When this task is started successfully, the driver will check the received data already in the DTMF-reception buffer area and refuse any new received DTMF digit to enter this buffer area. This task will not be cancelled until one of the following conditions is met:

- ✧ The terminating character specified in the parameter of cEndChar or szEndChar is received.
- ✧ The number of the received DTMF characters equals the maximum length which is set in the parameter wMaxLen
- ✧ The waiting time exceeds the time specified in the parameter wTimeOut1 or wTimeOut2.

When the WaitDtmf task is finished, the driver throws out the event of E_PROC_WaitDTMF to the application. The execution result of the WaitDtmf task can be obtained by the function SsmChkWaitDtmf.

The function SsmCancelWaitDtmf can be used to cancel the WaitDtmf task.

Note:

- Each time when the driver receives one DTMF character on the designated channel, it automatically resets and restarts the timer.
- When this function is invoked, the previously received digits in the DTMF buffer area will not be cleared. That is, the remaining DTMF digits in the buffer area will affect this function's execution result.
- If the WaitDtmf task is already started on a channel, the DTMF digits received afterwards will no longer be put into the DTMF reception buffer area and the event E_CHG_RcvDTMF will not be generated during the execution of this task.
- The end character and the end character unit are case sensitive.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmChkWaitDtmf, SsmCancelWaitDtmf

2.6.4.5 SsmCancelWaitDtmf

Cancels the task of WaitDtmf.

Format:

int SsmCancelWaitDtmf(int ch)

Parameter Description:

ch	Channel Number

Return Value:

-1	Error occurs
0	Successful

Function Description:

Cancels the task of WaitDtmf.

Note:

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmSetWaitDtmf, SsmSetWaitDtmfEx, SsmSetWaitDtmfExA, SsmChkWaitDtmf

2.6.4.6 SsmChkWaitDtmf

Queries the execution status of the WaitDtmf task.

Format:

int SsmChkWaitDtmf(int ch, LPSTR pszDtmf)

Parameter Description:

ch	Channel Number
pszDtmf	The pointer pointing to the buffer area which stores DTMF, the storage space is allocated by the application

Return Value:

-1	Error occurs
0	Receiving
1	The task is finished due to receiving timeout; or no WaitDtmf task on the channel.
2	The task is finished due to the reception of the designated terminating character.
3	The task is finished due to the reception of the DTMF string with designated length.

Function Description:

Queries the execution status of the WaitDtmf task.

Note:

- The event of E_PROC_WaitDTMF can implement the same feature.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Event: E_PROC_WaitDTMF

2.7 DTMF Generator Functions (CTI Series)

2.7.1 Setting Parameters for DTMF Generator

2.7.1.1 SsmSetTxDtmfPara

Sets the durations of the DTMF signal generated by the DTMF Generator respectively at on and off states.

Format:

int SsmSetTxDtmfPara(int ch, WORD wOnTime, WORD wOffTime)

Parameter Description:

ch	Channel Number
wOnTime	The duration (ms) of DTMF at on state, the range of value: ≥40, must be integral times of 8
wOffTime	The duration (ms) of DTMF at off state, the range of value: ≥40, must be integral times of 8

Return Value:

-1	Call failed, the failure reason can be obtained by the function SsmGetLastErrMsg
0	Successful

Function Description:

Sets the durations of the DTMF signal generated by the DTMF Generator respectively at on and off states.

Note:

- The configuration item TxDtmfTimePara can implement the same feature.
- For IP channel, this function call is valid only when DTMF signals are sent by in-band method.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmGetTxDtmfPara, SsmTxDtmf

2.7.1.2 SsmQueryTxDtmf

Checks whether the channel includes DTMF Generator.

Format:

int SsmQueryTxDtmf(int ch, LPSTR pszReserved)

Parameter Description:

ch	Channel Number
PszReserved	Reserved string pointer, it is not necessary to be used

Return Value:

-1	Failed
0	The designated channel doesn't support the operation of sending the standard DTMF characters
1	The designated channel supports the operation of sending the standard DTMF characters

Function Description:

Checks whether the channel includes DTMF Generator.

Note:

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function:

2.7.1.3 SsmGetTxDtmfPara

Obtains the duration of the DTMF signal generated by the DTMF Generator respectively at on and off states.

Format:

int SsmGetTxDtmfPara(int ch, PWORD pwOnTime, PWORD pwOffTime)

Parameter Description:

ch	Channel Number
pwOnTime	Returns the duration (ms) at on state
pwOffTime	Returns the duration (ms) at off state

Return Value:

-1	Failed
=0	Successful

Function Description:

Obtains the durations of the DTMF signal generated by the DTMF Generator respectively at on and off states.

Note:

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmSetTxDtmfPara

2.7.2 Setting Working Status of DTMF Generator

2.7.2.1 SsmTxDtmf

Transmits DTMF string in the channel's outgoing call direction.

Format:

int SsmTxDtmf(int ch, LPSTR pszDtmf)

Parameter Description:

ch	Channel Number
pszDtmf	● Pointer to the string, must be terminated with '\0'. The number of characters can't exceed the set value in the configuration item TxDtmfBufSize. The characters can be: a) '0'~'9','*','#','A','B','C','D': Standard DTMF signal; b) '!': Generates one flash on the line, only applicable to the analog trunk channel; c) Other characters: Keeps silence for a period on the line. The duration time is set by the configuration item DefaultTxDelayTime, with the default value of 2000ms. ● For IP channel, the meaning of this parameter varies on DTMF transmission modes as follows: a) When DTMF signals are sent by in-band or RFC2833 method, only 15 standard DTMF digits '0'~'9','*','#','A','B','C','D' are supported; b) When DTMF signals are sent by out-of-band method, all characters in the ASCII character set can be used.

Return Value:

-1	Call failed, the failure reason can be obtained by the function SsmGetLastErrMsg
0	Successful

Function Description:

Transmits DTMF string in the channel's outgoing call direction.

The driver will start the DTMF transmission immediately after copying the characters in pszDtmf to the internal transmission buffer area. When all the characters in the transmission buffer area have been sent out, the driver throws out the event of E_PROC_SendDTMF to the application. The function SsmChkTxDtmf can be used to query the transmission status, the function SsmStopTxDtmf can be used to terminate the transmission task.

Note:

- The size of the transmission buffer area of the DTMF generator can be set by the configuration item TxDtmfBufSize and the default value is 50 characters.
- This function only supports SHT Series, SHD Series and SHN Series boards.
- For IP channel,
  a) When DTMF signals are sent by out-of-band method, it is a synchronous function; otherwise it is an asynchronous function.
  b) Only when DTMF signals are sent by in-band method can the function SsmStopTxDtmf be invoked to terminate the transmission.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmChkTxDtmf, SsmStopTxDtmf, SsmSetTxDtmfPara

2.7.2.2 SsmStopTxDtmf

Stops the transmission task of the DTMF generator.

Format:

int SsmStopTxDtmf(int ch)

Parameter Description:

ch	Channel Number

Return Value:

-1	Call failed, the failure reason can be obtained by the function SsmGetLastErrMsg
0	Successful

Function Description:

Stops the transmission task of the DTMF generator.

Note:

- For IP channel, this function call is valid only when DTMF signals are sent by in-band method.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmTxDtmf

2.7.2.3 SsmChkTxDtmf

Queries the transmission status of the DTMF generator.

Format:

int SsmChkTxDtmf(int ch)

Parameter Description:

ch	Channel Number

Return Value:

-1	Error occurs
0	The transmission of DTMF string is completed
1	The transmission of DTMF string is in progress

Function Description:

Queries the transmission status of the DTMF generator.

Note:

- The event of E_PROC_SendDTMF can implement the same feature.
- For IP channel, this function call is valid only when DTMF signals are sent by in-band method.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmTxDtmf, SsmStopTxDtmf

2.8 Barge-in Detector Functions

2.8.1 Setting Parameters for Barge-in Detector

2.8.1.1 SsmSetBargeInSens

Sets the sensitivity of the Barge-in detector.

Format:

int SsmSetBargeInSens(int ch, int nSens)

Parameter Description:

ch	Channel Number
nSens	Sensitivity, range of value: 0~31, more sensitive with the larger value

Return Value:

-1	Call failed, the failure reason can be obtained by the function SsmGetLastErrMsg
0	Successful

Function Description:

Sets the sensitivity of the Barge-in detector. For more information, refer to Barge-in Detector.

Note:

- The configuration item BargeInSensitive can implement the same feature.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmGetBargeInSens, SsmSetIsBargeInDtrmTime, SsmSetNoSoundDtrmTime

2.8.1.2 SsmSetIsBargeInDtrmTime

Sets the minimum duration time of Barge-in.

Format:

int SsmSetIsBargeInDtrmTime(int ch, WORD wMinKeepTime)

Parameter Description:

ch	Channel Number
wMinKeepTime	Minimum duration time (ms), range of value: ≥16, must be integral times of 16ms

Return Value:

-1	Not supported or parameter setting error
0	Call failed, the failure reason can be obtained by the function SsmGetLastErrMsg
1	Successful

Function Description:

Sets the minimum duration time of Barge-in.

When the voice signal is detected in the incoming call and the duration time is greater than the set value of this function, the Barge-in detector outputs the detected result of 'BargeIn' and throws out the event of E_SYS_BargeIn to the application. For more information, refer to 'Barge-in Detector'.

Note:

- The configuration item BargeInDtrmTime can implement the same feature, with the default value of 32ms.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmGetIsBargeInDtrmTime, SsmSetBargeInSens, SsmSetNoSoundDtrmTime

2.8.1.3 SsmSetVoiceEnergyMinValue

Sets the threshold value for noise judgment.

Format:

int SsmSetVoiceEnergyMinValue(int ch, DWORD nVoiceEnergyMinValue)

Parameter Description:

ch	Channel Number
nVoiceEnergyMinValue	The threshold value for noise judgment, range of value: 0<nVoiceEnergyMinValue<=0x7fffffff

Return Value:

-1	Call failed, the failure reason can be obtained from the function SsmGetLastErrMsg
0	Successful

Function Description:

Sets the threshold value of Barge-in detector for noise judgment.

Note:

- The configuration item VoiceEnergyMinValue can implement the same feature, with the default value of 100000.

Related Information:

Driver version	SynCTI Ver. 5.1.0.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmGetVoiceEnergyMinValue

2.8.1.4 SsmSetNoSoundDtrmTime

Sets the minimum silence duration.

Format:

int SsmSetNoSoundDtrmTime(int ch, DWORD wMinKeepTime)

Parameter Description:

ch	Channel Number
wMinKeepTime	Minimum duration time (ms), range of value: ≥16, must be integral times of 16ms

Return Value:

-1	Call failed, the failure reason can be obtained from the function SsmGetLastErrMsg
0	Successful

Function Description:

Sets the minimum silence duration.

When the voice signal disappears (i.e. the channel becomes silent) and the silence lasts for a period of time larger than the set value of this function, the BargeIn detector outputs the detected result of 'No Sound' and throws out the event of E_SYS_NoSound to the application. For more information, refer to 'Barge in Detector' in Chapter 1.

Note:

- The configuration item IsNoSoundDtrmTime can implement the same feature, with the defaulted value of 5000ms.

Related Information:

Driver version	SynCTI Ver. 2.0 or higher
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmGetIsBargeInDtrmTime, SsmSetBargeInSens, SsmGetNoSoundDtrmTime

2.8.1.5 SsmGetBargeInSens

Obtains the sensitivity of the Barge-in detector.

Format:

int SsmGetBargeInSens(int ch)

Parameter Description:

ch	Channel Number

Return Value:

-1	Call failed, the failure reason can be obtained by the function SsmGetLastErrMsg
≥0	The sensitivity of the Barge In detector

Function Description:

Obtains the preset sensitivity of the Barge In detector.

Note:

Related Information:

Driver version	SynCTI Ver. 2.0 or higher
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmSetBargeInSens

2.8.1.6 SsmGetIsBargeInDtrmTime

Obtains the preset value of the minimum duration time of BargeIn.

Format:

int SsmGetIsBargeInDtrmTime(int ch)

Parameter Description:

ch	Channel Number

Return Value:

-1	Failed
≥0	Minimum duration time (ms)

Function Description:

Obtains the preset value of the minimum duration time of BargeIn.

Note:

Related Information:

Driver version	SynCTI Ver. 2.0 or higher
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmSetIsBargeInDtrmTime

2.8.1.7 SsmGetVoiceEnergyMinValue

Gets the threshold value for noise judgment.

Format:

DWORD SsmGetVoiceEnergyMinValue(int ch)

Parameter Description:

ch	Channel Number

Return Value:

-1	Call failed, the failure reason can be obtained from the function SsmGetLastErrMsg
≥0	Successful, return the threshold value for noise judgment.

Function Description:

Gets the threshold value of Barge-in detector for noise judgment.

Note:

Related Information:

Driver version	SynCTI Ver. 5.1.0.0 or above
Header	shpa3api.h
Library	shp_a3.lib

DLL	shp_a3.dll

Related Function: SsmSetVoiceEnergyMinValue

2.8.1.8 SsmGetNoSoundDtrmTime

Obtains the preset value of the minimum duration time of silence.

Format:

int SsmGetNoSoundDtrmTime(int ch)

Parameter Description:

ch	Channel Number

Return Value:

-1	Failed
≥0	Minimum duration time (ms)

Function Description:

Obtains the preset value of the minimum duration time of silence.

Note:

Related Information:

Driver version	SynCTI Ver. 2.0 or higher
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmSetNoSoundDtrmTime

2.8.2 Obtaining Detected Result of Barge-in Detector

2.8.2.1 SsmDetectBargeIn

Obtains the detected result of the Barge-in Detector.

Format:

int SsmDetectBargeIn(int ch)

Parameter Description:

ch	Channel Number

Return Value:

-1	Failed
0	Barge-in is not detected
1	Barge-in is detected

Function Description:

Obtains the detected result of the Barge-in detector. For more information, refer to 'Barge in Detector' in Chapter 1.

Note:

- It's recommended to use the event of E_SYS_BargeIn to get the detected result.

Related Information:

Driver version	SynCTI Ver. 2.0 or higher
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmDetectNoSound

2.8.2.2 SsmDetectNoSound

Obtains the detected results of the silence detector.

Format:

int SsmDetectNoSound(int ch)

Parameter Description:

ch	Channel Number

Return Value:

-1	Failed
0	'No Sound' is not detected
1	'No sound' is detected

Function Description:

Obtains the detected results of the silence detector.

Note:

- It's recommended to use the event of E_SYS_NoSound to get the detected result.

Related Information:

Driver version	SynCTI Ver. 2.0 or higher
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmDetectBargeIn

2.8.2.3 SsmGetNoSoundTime

Obtains the duration time of silence.

Format:

int SsmGetNoSoundTime(int ch)

Parameter Description:

ch	Channel Number

Return Value:

-1	Failed
≥0	The duration time of silence (ms)

Function Description:

Obtains the duration time of silence. For more information, refer to 'Barge in Detector' in chapter 1.

Note:

Related Information:

Driver version	SynCTI Ver. 2.0 or higher
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmDetectNoSound

2.9 Voice Playing Functions

2.9.1 Setting Playing Data's Destination

2.9.1.1 SsmSetPlayDest

Sets whether to send the voice playing data onto the TDM bus.

Format:

int SsmSetPlayDest(int ch, int nSelDest)

Parameter Description:

ch	Channel Number
nSelDest	The selector switch determining whether to send the voice playing data onto the TDM bus. For SHD and SHT series boards, this parameter controls the status of the switch K1-1 in both operation principle diagrams of SHD and SHT series boards. Value range: 0: Not send onto the bus (switch off); 1: Send onto the bus (switch on).

Return Value:

-1	Call failed, the failure reason can be obtained by the function SsmGetLastErrMsg
0	Call successful

Function Description:

Sets whether to send the voice playing data onto the TDM bus. For more information, refer to 'Operation Principle of SHD Series' or 'Operation Principle of SHT Series' in Chapter 1.

Note:

- This function only supports SHT, SHD and SHN series boards.

Related Information:

Driver version	SynCTI Ver. 4.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function:

2.9.2 Setting Playing Volume

2.9.2.1 SsmSetPlayVolume

Refer to SsmSetPlayGain

2.9.2.2 SsmSetPlayGain

Sets the voice playing volume for SHD/SHT/SHN Series boards. SsmSetPlayGain achieves more accurate control than SsmSetPlayVolume.

Format:

int SsmSetPlayVolume(int ch, int nGain)

int SsmSetPlayGain(int ch, WORD wGainLevel)

Parameter Description:

ch	Channel Number

nGain	Volume gain, range of value: -7~+6. Greater than 0 denotes volume increasing, less than 0 denotes volume decreasing, -7 denotes completely turning off the audio output; the default value is 0. The volume value multiplies 3 equals DB value.

wGainLevel	Volume gain expressed in hexadecimal form, lower 8 bits are valid, higher 8 bits are reserved and needs to be set to 0. Range of value:
	wGainLevel=0: completely turns off the audio output;
	wGainLevel=1: decreases 16 times;
	wGainLevel=16: gain keeps unchanged (0DB);
	32≤wGainLevel≤255: divides by 16 equals the gain amplification times;
	wGainLevel=255: 16 times amplification

Return Value:

-1	Call failed, the failure reason can be obtained by the function SsmGetLastErrMsg
0	Successful

Function Description:

The above 2 functions are used to set the gain of the volume adjustor A1 in SHD/SHT/ATP/SHN Series boards. For more information about the volume adjustor A1, refer to Operation Principle of SHD Series or Operation Principle of SHT Series.

Note:

- The configuration item DefaultPlayVolume can implement the same function as SsmSetPlayVolume.

Related Information:

Driver version	SynCTI Ver. 3.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function:

2.9.3 Setting Termination Condition of Voice Playing

2.9.3.1 SsmSetDtmfStopPlay

Sets whether the voice playing needs to be terminated because of the detection of DTMF character by the DTMF Detector.

Format:

int SsmSetDtmfStopPlay(int ch, BOOL bTerminatePlaybackOnDTMF)

Parameter Description:

ch	Channel Number
bTerminatePlaybackOnDTMF	Enabling indicator, range of value: TRUE: turn-on FALSE: turn-off

Return Value:

-1	Call failed, the failure reason can be obtained by the function SsmGetLastErrMsg
0	Successful

Function Description:

Sets whether the voice playing needs to be terminated because of the detection of DTMF character by the DTMF Detector.

In case that the parameter bTerminatePlaybackOnDTMF is set to be TRUE: while the voice is playing on the channel, if the DTMF Detector detects the DTMF character in the incoming call signals, and the detected DTMF

character is included in the character set preset by the function SsmSetDTMFStopPlayCharSet or the configuration item DtmfStopPlayCharSet, the driver will stop the voice playing immediately

Note:

- The configuration item DefaultDtmfStopPlay can implement the same feature; the default value of the configuration item is turn-off.
- The indicator set by this function has effect on the loaded file to be played, single file, multiple files, single buffer, Pingpong buffer and multiple buffers.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmSetDTMFStopPlayCharSet, SsmGetDtmfStopPlayFlag

2.9.3.2 SsmSetDTMFStopPlayCharSet

Sets the DTMF character set which is used to terminate the voice playing.

Format:

int SsmSetDTMFStopPlayCharSet(int ch, LPSTR lpstrDtmfCharSet)

Parameter Description:

ch	Channel Number
lpstrDtmfCharSet	Pointer pointing to the character set which is used to terminate the voice playing. The usable DTMF characters include: ‘0’, ‘1’, ‘2’, ‘3’, ‘4’, ‘5’, ‘6’, ‘7’, ‘8’, ‘9’, ‘*’, ‘#’, ‘a’, ‘b’, ‘c’, 'd'

Return Value:

-1	Failed, the failure reason may be parameter errors or unsupported operation by the channel. The detailed reason can be acquired via the function SsmGetLastErrMsg
0	Failed, the failure reason is that the command can’t be transferred to the board. The detailed reason can be obtained by the function SsmGetLastErrMsg
1	Successful

Function Description:

Sets the DTMF character set which is used to terminate the voice playing.

Note:

- The configuration item DtmfStopPlayCharSet can implement the same feature. The default value is the full set.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmSetDtmfStopPlay, SsmGetDTMFStopPlayCharSet

2.9.3.3 SsmGetDtmfStopPlayFlag

Obtains the indicator which denotes whether the voice playing stops because the DTMF Detector detects the DTMF character.

Format:

int SsmGetDtmfStopPlayFlag(int ch)

Parameter Description:

ch	Channel Number

Return Value:

-1	Failed
0	The indicator is in the turn-off state
1	The indicator is in the turn-on state

Function Description:

Obtains the indicator which denotes whether the voice playing stops because the DTMF Detector detects the DTMF character.

Note:

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmSetDtmfStopPlay

2.9.3.4 SsmGetDTMFStopPlayCharSet

Obtains the DTMF character set which is used to terminate the voice playing.

Format:

int SsmGetDTMFStopPlayCharSet(int ch, LPSTR lpstrDtmfCharSet)

Parameter Description:

ch	Channel Number
lpstrDtmfCharSet	Returns the preset DTMF character set

Return Value:

-1	Failed, the failure reason can be acquired via the function SsmGetLastErrMsg
≥0	The length of the character set

Function Description:

Obtains the DTMF character set which is used to terminate the voice playing.

Note:

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmSetDTMFStopPlayCharSet

2.9.3.5 SsmSetBargeinStopPlay

Sets whether the voice playing stops because the voice activities are detected by the Barge-in Detector.

Format:

int SsmSetBargeinStopPlay(int ch, BOOL bTerminatePlaybackOnBargein)

Parameter Description:

ch	Channel Number
bTerminatePlaybackOnBargein	Enabling indicator, range of value: TRUE: Turn-on FALSE: Turn-off

Return Value:

-1	Call failed, the failure reason can be obtained by the function SsmGetLastErrMsg
0	Successful

Function Description:

Sets whether the voice playing stops because the voice activities are detected by the Barge-in Detector.

In the case that the parameter bTerminatePlaybackOnBargein is set to be TRUE: When the voice is playing on the channel, if the Barge-in Detector detects voice activities in the incoming call signals, the driver terminates the voice playing.

Note:

- The configuration item DefaultBargeInStopPlay can implement the same feature; the default value is turn-off.
- The indicator set by this function has effect on the loaded file to be played, single file, multiple files, single buffer, Pingpong buffer and multiple buffers.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmGetBargeinStopPlayFlag

2.9.3.6 SsmGetBargeinStopPlayFlag

Obtains the indicator which denotes whether the voice playing is stops because the Barge-in Detector detects the voice activities.

Format:

int SsmGetBargeinStopPlayFlag(int ch)

Parameter Description:

ch	Channel Number

Return Value:

-1	Failed
0	The indicator is in turn-off state
1	The indicator is in turn-on state

Function Description:

Obtains the indicator which denotes whether the voice playing stops because the Barge-in Detector detects the voice activities.

Note:

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib

DLL	shp_a3.dll

Related Function: SsmSetBargeinStopPlay

2.9.3.7 SsmSetHangupStopPlayFlag

Sets whether the voice playing stops because the driver detects the hangup of the remote end.

Format:

int SsmSetHangupStopPlayFlag(int ch, BOOL bHangupStopPlayFlag)

Parameter Description:

ch	Channel Number
bHangupStopPlayFlag	The indicator denoting whether the voice playing stops when the driver detects the hangup of the remote end: TRUE: turn-on FALSE: turn-off

Return Value:

-1	Call failed, the failure reason can be obtained by the function SsmGetLastErrMsg
0	Successful

Function Description:

Sets whether the voice playing stops because the driver detects the hangup of the remote end on Channel ch.

Note:

- The configuration item HangupStopPlay can implement the same feature; the default value is turn-off.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function:

2.9.4 Functions for Playing in Single File Mode

2.9.4.1 SsmPlayFile

Plays a single file, starts the file playing task.

Format:

int SsmPlayFile(int ch, LPSTR pszFileName, int nFormat, DWORD dwStartPos, DWORD dwLen)

int SsmPlayFileW(int ch, LPCWSTR lpFileName, int nFormat, DWORD dwStartPos, DWORD dwLen)

Parameter Description:

ch	Channel Number
pszFileName lpFileName	Voice file name (pszFileName is UTF8 formatted; lpFileName is UNICODE formatted). It could be either a standard wav file or a non-header file. If it's a standard wav file, the voice encoding format is achieved from the file header, the parameter nFormat is ignored. If it's a non-header file, the voice encoding format is designated by nFormat.

<table>
<tr><td>nFormat</td><td>Encoding format of the voice data, range of value and its meaning are listed below:
<table>
<tr><th>Value</th><th>Encoding format</th><th>Remarks</th></tr>
<tr><td>-2</td><td>PCM16</td><td></td></tr>
<tr><td>1</td><td>PCM8</td><td></td></tr>
<tr><td>6</td><td>A-Law</td><td></td></tr>
<tr><td>7</td><td>μ-Law</td><td></td></tr>
<tr><td>17</td><td>IMA ADPCM</td><td>The board is required to have the hardware decoding capability</td></tr>
<tr><td>23</td><td>VOX</td><td></td></tr>
<tr><td>49</td><td>GSM</td><td></td></tr>
<tr><td>85</td><td>MP3</td><td></td></tr>
<tr><td>131</td><td>G.729A</td><td></td></tr>
</table>
Note: nFormat = -1 represents the default encoding format which is related to the item DefaultRecordFormat in ShConfig.ini. For more information about whether a specific board model supports the above encoding format and in which way it supports, refer to ‘Board Supported CODECs’ in Chapter 1</td></tr>
<tr><td>dwStartPos</td><td>The starting position counted from the actual voice data part in the file. For the standard wav file, the starting position 0 represents the position of the first voice data byte following the file header; For the non-header file, it’s counted from the head part of the file.</td></tr>
<tr><td>dwLen</td><td>The length of the voice data (bytes) to be played. If this parameter is larger than the length from dwStartPos to the tail part of the file, the latter is used to be the actual playing length.</td></tr>
</table>

Return Value:

-1	Call failed, the failure reason can be obtained by the function SsmGetLastErrMsg
0	Successful

Function Description:

Plays a single file.

After invoking this function, the driver starts playing a single file. The file playing task stops if one of the following events happens:

- ✧ All the data in the file have been played out.
- ✧ The application invokes SsmStopPlayFile or SsmStopPlay;
- ✧ DTMF, Barge in or remote end hangup has been detected. For more information, refer to Setting Termination Condition in chapter 1.
- ✧ For some models in the SHD series, the configuration item of LinkFromStopPlayAndTone is set to 1 and the off-bus operation is performed on the channel. For more information, refer to ‘Operation Principle of SHD Series’ in chapter 1.

After the file playing task is finished, the driver closes the played file and throws out the event of E_PROC_PlayEnd to the application. The function SsmCheckPlay can also be used to enquire the status of the file playing.

During the execution of file playing, the driver throws out the event of E_PROC_PlayFile to the application at every time interval. The function SsmSetEvent (with the parameter E_PROC_PlayFile) sets whether to throw out this event by the driver and the time interval of throwing out events. By default, every 1000ms the driver throws out the event and outputs the time length of the file played.

During the execution of file playing, the application can call the below functions to pause the file playing or adjust the file playing position.

- ✧ SsmPausePlay
- ✧ SsmFastFwdPlay
- ✧ SsmFastBwdPlay
- ✧ SsmSetPlayTime
- ✧ SsmSetPlayPrct

The following functions can be called to obtain the related information:

- ✧ SsmGetDataBytesToPlay
- ✧ SsmGetPlayedTime
- ✧ SsmGetPlayedPercentage

Note:

- The file is opened with 'shared read' mode, so this function supports the feature of 'record while play';
- If this function is called on channel 0, for ATP/DST Series boards, the played voice is sent to the port of the on-board speaker, but not to the line; For SHT Series boards, the played voice is sent not only to the line but also to the port of the on-board speaker;
- Due to historical reasons, the coding value 65411 which ever represented G.729A is still valid in early versions and backward compatible. For G.729A recording in new versions, we suggest you use the coding value 131.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmStopPlayFile, SsmStopPlay, SsmCheckPlay, SsmPausePlay, SsmRestartPlay, SsmFastFwdPlay, SsmFastBwdPlay, SsmSetPlayTime, SsmSetPlayPrct, SsmGetDataBytesToPlay, SsmGetPlayedTime, SsmGetPlayedPercentage

2.9.4.2 SsmStopPlayFile

Stops the file playing task.

Format:

int SsmStopPlayFile(int ch)

Parameter Description:

ch	Channel Number

Return Value:

-1	Call failed, the failure reason can be obtained by the function SsmGetLastErrMsg
0	Successful

Function Description:

Stops the file playing task started by the function call of SsmPlayFile.

Note:

- This function can be totally replaced by the function SsmStopPlay.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmPlayFile, SsmPausePlay, SsmRestartPlay, SsmFastFwdPlay, SsmFastBwdPlay, SsmSetPlayTime, SsmSetPlayPrct, SsmStopPlay

2.9.4.3 SsmPausePlay

Pauses (suspends) the file playing.

Format:

int SsmPausePlay(int ch)

Parameter Description:

ch	Channel Number

Return Value:

-1	Call failed, the failure reason can be obtained by the function SsmGetLastErrMsg
0	Successful

Function Description:

Pauses (suspends) the file playing. When the file playing is paused, the application can only execute the below operations:

- Invokes the function SsmStopPlayFile or SsmStopPlay to stop the file playing.
- Invokes the function SsmRestartPlay to restart the file playing.

Note:

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmRestartPlay, SsmStopPlayFile, SsmStopPlay

2.9.4.4 SsmRestartPlay

Restarts the file playing which is paused by the function call of SsmPausePlay.

Format:

int SsmRestartPlay(int ch)

Parameter Description:

ch	Channel Number

Return Value:

-1	Call failed, the failure reason can be obtained by the function SsmGetLastErrMsg
0	Successful

Function Description:

Restarts the file playing which is paused by the function call of SsmPausePlay.

Note:

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmPausePlay, SsmStopPlayFile, SsmStopPlay

2.9.4.5 SsmFastFwdPlay

Skips 1 second of voice data and continues to play.

Format:

int SsmFastFwdPlay(int ch)

Parameter Description:

ch	Channel Number

Return Value:

-1	Call failed, the failure reason can be obtained by the function SsmGetLastErrMsg
0	Successful

Function Description:

Starting from the current playing position, skips 1 second (configurable with the configuration item FastPlayTime) of voice data, and then continues the single file playing.

Note:

- If the remaining voice data is less than 1 second, the driver regards that the entire voice data has been played out and stops the current playing.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmFastBwdPlay, SsmStopPlayFile, SsmStopPlay

2.9.4.6 SsmFastBwdPlay

Skips back 1 second of voice data and continues to play.

Format:

int SsmFastBwdPlay(int ch)

Parameter Description:

ch	Channel Number

Return Value:

-1	Call failed, the failure reason can be obtained by the function SsmGetLastErrMsg
0	Successful

Function Description:

Starting from the current playing position, skips back 1 second (configurable with the configuration item FastPlayTime), and then continues the file playing.

Note:

- If the playing position returns to the start of the file, the file will be played from the beginning.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmFastFwdPlay, SsmPausePlay, SsmStopPlayFile, SsmStopPlay

2.9.4.7 SsmSetPlayTime

Jumps to a specified time position in the file and continues to play.

Format:

int SsmSetPlayTime(int ch, DWORD dwTime)

Parameter Description:

ch	Channel Number
dwTime	Time (ms) based new playing position

Return Value:

-1	Call failed, the failure reason can be obtained by the function SsmGetLastErrMsg
0	Successful

Function Description:

Jumps to the specified time position in the file and continues to play. The function SsmGetPlayingFileInfo can be used to obtain the file information such as the duration of the file.

Note:

- If the file offset specified in the parameter dwTime exceeds the length of the voice data, the driver regards that the entire voice data has been played out and stops the current playing.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmGetPlayingFileInfo, SsmSetPlayPrct, SsmStopPlayFile, SsmStopPlay

2.9.4.8 SsmSetPlayPrct

Jumps to the specified percentage position in the file and continues to play.

Format:

int SsmSetPlayPrct(int ch, DWORD dwPercentage)

Parameter Description:

ch	Channel Number
dwPercentage	The new playing position based on the percentage of the file, range of value: 0~100

Return Value:

-1	Call failed, the failure reason can be obtained by the function SsmGetLastErrMsg
0	Successful

Function Description:

Jumps to the position of a specified percentage in the file and continues to play. The function SsmGetPlayingFileInfo can be used to obtain the file duration.

Note:

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmSetPlayTime, SsmStopPlayFile, SsmStopPlay

2.9.4.9 SsmGetDataBytesToPlay

Queries the length (bytes) of the voice data which has not been played.

Format:

int SsmGetDataBytesToPlay(int ch)

Parameter Description:

ch	Channel Number

Return Value:

-1	Failed
≥0	The length (bytes) of the voice data which has not been played

Function Description:

Queries the length (bytes) of the voice data which has not been played.

Note:

- This function only supports the voice playing task started by the function SsmPlayFile.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmGetDataBytesPlayed, SsmGetPlayedTime, SsmGetPlayedPercentage

2.9.4.10 SsmGetDataBytesPlayed

Queries the length of the voice data which has been played.

Format:

int SsmGetDataBytesPlayed(int ch)

Parameter Description:

ch	Channel Number

Return Value:

-1	Failed
≥0	The length (bytes) of voice data which has been played

Function Description:

Queries the length of the voice data which has been played.

Note:

- This function only supports the voice playing task started by the function SsmPlayFile.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmGetDataBytesToPlay, SsmGetPlayedTime, SsmGetPlayedPercentage

2.9.4.11 SsmGetPlayedTimeEx

Obtains the playtime starting from the 1st voice data byte to the current playing position in the file.

Format:

int SsmGetPlayedTimeEx(int ch)

Parameter Description:

ch	Channel Number

Return Value:

-1	Call failed, the failure reason can be obtained by the function SsmGetLastErrMsg
≥0	Returns the playtime (ms) starting from the 1st voice data byte to the current playing position in the file

Function Description:

Obtains the playtime starting from the 1st voice data byte to the current playing position in the file.

For example, after starting file playing by calling the function SsmPlayFile(ch, 'Voice.voc', 7, 16000,), if the driver initiates playing from the file offset 16000 and has played 10 seconds of voice data, invoke this function and its return value will be 10+2=12 seconds (the playing duration for 16000 bytes μ-law format voice data is 2 seconds).

Note:

- This function is only applicable to the file playing which is started by the function call of SsmPlayFile.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmGetPlayedTime

2.9.4.12 SsmGetPlayingFileInfo

Obtains the related information of the playing file.

Format:

int SsmGetPlayingFileInfo(int ch , int *pnFormat , long *pnTime)

Parameter Description:

ch	Channel Number
pnFormat	Returns the voice encoding format of the playing file. For the code value of the voice CODEC format in the SynCTI driver platform, refer to SynCTI Supported CODECs in Chapter 1
pnTime	Total time length (ms) of the voice data in the file

Return Value:

-1	Call failed, the failure reason can be obtained by the function SsmGetLastErrMsg
0	Successful

Function Description:

After the application calls the function of SsmPlayFile, this function is used to obtain the related information of the playing file.

Note:

- This function is only applicable to the playing task initiated by the function SsmPlayFile.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib

DLL	shp_a3.dll

Related Function: SsmPlayFile, SsmGetPlayedTime, SsmGetPlayedTimeEx, SsmGetPlayedPercentage

2.9.4.13 SsmPlayFileW

Refer to SsmPlayFile

2.9.5 Functions for Playing in File List Mode

2.9.5.1 SsmClearFileList

Closes and clears all the voice files submitted to the driver.

Format:

int SsmClearFileList(int ch)

Parameter Description:

ch	Channel Number

Return Value:

-1	Call failed, the failure reason can be obtained by the function SsmGetLastErrMsg
0	Successful

Function Description:

Closes and clears all the voice files submitted to the driver.

Note:

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmAddToFileList

2.9.5.2 SsmAddToFileList

Submits a voice file to be played to the driver.

Format:

int SsmAddToFileList(int ch, LPSTR pszFileName, int nFormat, DWORD dwStartPos, DWORD dwLen)

Parameter Description:

ch	Channel Number
pszFileName	Voice file name, it is either a standard wav file or a non-header file. If it's a standard wav file, the voice encoding format is achieved from the file header, the parameter nFormat is ignored. If it's a non-header file, the voice encoding format is designated by nFormat.
nFormat	Encoding format of the voice data, range of value and its meaning are listed below:

Value	Encoding format	Remark
-2	PCM16	
1	PCM8	
6	A-Law	
7	μ-Law	
17	IMA ADPCM	The board is required to have the hardware decoding capability
23	VOX	

	85 MP3 For more information about whether a specific board model supports the above encoding formats and in which way it supports, refer to 'Board Supported CODECs' in Chapter 1
dwStartPos	The starting position counted from the actual voice data part in the file. For the standard wav file, the starting position 0 represents the position of the first voice data byte following the file header; For the non-header file, it's counted from the head part of the file
dwLen	The length (bytes) of the voice data to be played. If this parameter is larger than the actual length calculated from dwStartPos, the driver regards that the file will be played until the end

Return Value:

-1	Call failed, the failure reason can be obtained by the function SsmGetLastErrMsg
0	Successful

Function Description:

Submits a voice file to be played to the driver. After the file is submitted, the multi-file playing task can be started by the function call of SsmPlayFileList.

Note:

- After this function is successfully executed, the submitted file is opened with 'shared read' mode. The file keeps open until the function SsmClearFileList is called to close it.
- This function can be invoked continuously to submit multiple files, but the total number can't exceed the value set in the configuration item MaxPlayFileList. All the files must have the same voice CODEC format.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmPlayFileList, SsmStopPlay, SsmClearFileList

2.9.5.3 SsmPlayFileList

Starts the multi-file playing task.

Format:

int SsmPlayFileList(int ch)

Parameter Description:

ch	Channel Number

Return Value:

-1	Call failed, the failure reason can be obtained by the function SsmGetLastErrMsg
0	Successful

Function Description:

Starts the multi-file playing task. After the application calls this function, the driver starts the task of multi-file playing and plays the first file. Each time when one file is played out, the driver throws out the event of E_PROC_PlayFileList to the application, and then starts to play the next file.

The driver stops playing the file if one of the following events happens:

- ✧ All data in the file have been played out.
- ✧ The application calls SsmStopPlayFileList or SsmStopPlay;

- ✧ DTMF, Barge-in or remote hang-up is detected by the driver. For more information, refer to Setting Termination Condition in chapter 1.
- ✧ For some models in the SHD series, the configuration item of LinkFromStopPlayAndTone is set to 1 and the off-bus operation is performed on the channel. For more information, refer to 'Operation Principle of SHD Series' in chapter 1.

After the file finishes playing, the driver throws out the event of E_PROC_PlayEnd to the application. The function of SsmCheckPlay can also be used to query the status of the file playing.

Note:

- Before invoking this function, you can call the function SsmAddToFileList to add files to the file list if it is empty.
- The driver won't close the submitted file after the file playing task is terminated. In such case, you can call the function SsmClearFileList to clear the file list.
- If this function is invoked on Channel 0, for the ATP/DST Series boards, the played voice is sent directly to the port of the on-board speaker, but not to the line; for the SHT Series boards, the played voice is not only sent to the line, but also to the port of the on-board speaker.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmStopPlayFileList, SsmStopPlay, SsmCheckPlay, SsmAddToFileList, SsmClearFileList

2.9.5.4 SsmStopPlayFileList

Stops the multi-file playing task.

Format:

int SsmStopPlayFileList(int ch)

Parameter Description:

ch	Channel Number

Return Value:

-1	Call failed, the failure reason can be obtained by the function SsmGetLastErrMsg
0	Successful

Function Description:

Stops the task of multi-file playing, the task must be started by the function of SsmPlayFileList.

Note:

- This function can be completely replaced by the function of SsmStopPlay.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmPlayFileList, SsmClearFileList, SsmStopPlay

2.9.6 Functions for Playing in Single Buffer Mode

2.9.6.1 SsmPlayMem

Starts the task of single-buffer playing.

Format:

int SsmPlayMem(int ch, int nFormat, LPBYTE pBuf, DWORD dwBufSize, DWORD dwStartOffset, DWORD dwStopOffset)

Parameter Description:

<table>
<tr><td>ch</td><td>Channel Number</td></tr>
<tr><td>nFormat</td><td>Encoding format of the voice data, range of value and its meaning are listed below:
<table>
<tr><th>Value</th><th>Encoding format</th><th>Remarks</th></tr>
<tr><td>1</td><td>PCM8</td><td></td></tr>
<tr><td>6</td><td>A-Law</td><td></td></tr>
<tr><td>7</td><td>μ-Law</td><td></td></tr>
<tr><td>17</td><td>IMA ADPCM</td><td>The board is required to have the hardware decoding capability</td></tr>
<tr><td>-2</td><td>PCM16</td><td></td></tr>
<tr><td>85</td><td>MP3</td><td></td></tr>
</table>
For more information about whether a specific board model supports the above encoding formats and in which way it supports, refer to ‘Board Supported CODECs’ in Chapter 1</td></tr>
<tr><td>pBuf</td><td>The address of the buffer storing voice data</td></tr>
<tr><td>dwBufSize</td><td>The size (bytes) of the buffer storing voice data; it should be no less than 768 bytes</td></tr>
<tr><td>dwStartOffset</td><td>Starting position (bytes) of voice playing (offset), it’s calculated from the initial address of the buffer</td></tr>
<tr><td>dwStopOffset</td><td>Ending position (bytes) of voice playing (offset), it’s calculated from the initial address of the buffer. If this parameter is less than the buffer length, when the playing pointer in the driver reaches the position designated by dwStopOffset, the current voice playing is finished; Otherwise, the buffer is circularly played until the application calls SsmStopPlayMem to stop the voice playing.</td></tr>
</table>

Return Value:

-1	Call failed, the failure reason can be obtained by the function SsmGetLastErrMsg
0	Successful

Function Description:

Starts the task of single-buffer playing.

After calling this function, the driver will start the task of single-buffer playing. Meanwhile, the driver maintains a playing pointer (offset) with the initial value equaling to dwStartOffset, which is used to read data from the submitted buffer. The single-buffer playing task stops if one of the following events happens:

- ✧ The value of dwStopOffset is less than the submitted buffer size and the playing pointer in the driver reaches the ending position specified by the parameter dwStopOffset, which means the specified data in the buffer have been played out.
- ✧ The application calls SsmStopPlayMem or SsmStopPlay;
- ✧ The driver detects DTMF, Barge in or remote hang-up. For more information, refer to ‘Setting Termination Condition’ in Chapter 1.
- ✧ For some models in the SHD series, the configuration item of LinkFromStopPlayAndTone is set to 1 and the off-bus operation is performed on the channel. For more information, refer to ‘Operation Principle of SHD Series’ in chapter 1.

After the task of single-buffer playing is finished, the driver throws out the event of E_PROC_PlayEnd to the application. The function SsmCheckPlay can also be used to query the status of single-buffer playing.

During the execution of single-buffer playing, each time when the preset condition is met, the driver outputs the event of E_PROC_PlayMem to the application. The output conditions of the event E_PROC_PlayMem can be set by the function SsmSetEvent to one of the three methods:

- ✧ The driver finishes the playing of a designated length of voice data
- ✧ The playing pointer in the driver gets across 1/2 part of the buffer.
- ✧ The playing pointer reaches to the terminating position of the buffer

For the settings of output conditions of the event of E_PROC_PlayMem, refer to MESSAGE_INFO in Chapter 1.

The function SsmSetStopPlayOffset can be used to reset the ending position of playing. The function SsmStopPlayMem or SsmStopPlay can be used to terminate the single-buffer playing, the function SsmGetPlayOffset can be used to obtain the playing pointer in the driver.

Note:

- If this function is invoked on the channel 0, for the ATP/DST Series boards, the played voice is sent directly to the port of the on-board speaker, but not to the line; For the SHT Series boards, the played voice is not only sent to the line, but also to the port of the on-board speaker.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmStopPlayMem, SsmStopPlay, SsmSetStopPlayOffset, SsmGetPlayOffset, SsmSetEvent

2.9.6.2 SsmStopPlayMem

Terminates the task of single-buffer playing initiated by the function SsmPlayMem.

Format:

int SsmStopPlayMem(int ch)

Parameter Description:

ch	Channel Number

Return Value:

-1	Call failed, the failure reason can be obtained by the function SsmGetLastErrMsg
0	Successful

Function Description:

Terminates the task of single-buffer playing initiated by the function SsmPlayMem.

Note:

- After this function is successfully called, the driver will still throw out the event of E_PROC_PlayEnd;
- The function SsmStopPlay can implement the same features.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmPlayMem, SsmSetStopPlayOffset, SsmStopPlay

2.9.6.3 SsmSetStopPlayOffset

Resets the ending position for playback in the buffer.

Format:

int SsmSetStopPlayOffset(int ch, DWORD dwStopPlayOffset)

Parameter Description:

ch	Channel Number
dwStopPlayOffset	Ending offset of the buffer

Return Value:

-1	Call failed, the failure reason can be obtained by the function SsmGetLastErrMsg
0	Successful

Function Description:

Resets the ending position for playback in the buffer.

Note:

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmGetPlayOffset

2.9.6.4 SsmGetPlayOffset

Obtains the playing pointer in the driver during the single-buffer playing.

Format:

int SsmGetPlayOffset(int ch, DWORD* pdwPlayOffset)

Parameter Description:

ch	Channel Number
pdwPlayOffset	Returns the playing pointer in the driver, an offset (bytes) based on the initial address of the buffer

Return Value:

-1	Failed
0	Successful

Function Description:

Obtains the playing pointer in the driver during the single-buffer playing.

Note:

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmSetStopPlayOffset

2.9.7 Functions for Playing in Buffer List Mode

2.9.7.1 SsmAddToPlayMemList

Submits a buffer to the driver.

Format:

int SsmAddToPlayMemList(LPBYTE pBuf, DWORD dwDataLen, int nFormat)

Parameter Description:

pBuf	Pointer pointing to the voice data buffer. The storage space of buffer area is allocated by the application
dwDataLen	The size (byte) of the voice data buffer
nFormat	Voice encoding format. Range of value: 6:A-law 7:μ-law 17:IMA ADPCM

Return Value:

-1	Call failed, the failure reason can be obtained by the function SsmGetLastErrMsg
0	Successful

Function Description:

Submits a buffer with the voice data to be played to the driver. For each submitted buffer, the driver assigns a serial number to it. The first one is assigned with 0, and the rest may be deduced by analogy.

This function can be called multiple times continuously, but the maximum number of times cannot exceed the set value of the configuration item MaxPlayMemList. The driver plays the buffers in the submitting sequence.

Note:

- The buffer submitted by this function is not specific to a particular channel. It can be used by all channels.
- All the buffers submitted must use the same voice encoding format.
- When the driver is executing this function, it only records the information in the buffer but does not copy the data in the buffer to the driver itself. Therefore, the application must ensure that no channel is executing the task of playing multiple buffers before releasing the submitted buffer.
- If the data is in ADPCM format, special attentions should be paid to the frame structure because disordered frame data will cause noises.

Related Information:

Driver version	SynCTI Ver. 3.0or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmClearPlayMemList, SsmPlayMemList

2.9.7.2 SsmClearPlayMemList

Clears all the buffers submitted to the driver.

Format:

int SsmClearPlayMemList()

Parameter Description: None

Return Value:

-1	Call failed, the failure reason can be obtained by the function SsmGetLastErrMsg

0	Successful

Function Description:

Clears all the buffers submitted to the driver.

Note:

- Ensure that no channel is executing the task of playing multiple buffers before invoking this function.

Related Information:

Driver version	SynCTI Ver. 3.0or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmAddToPlayMemList

2.9.7.3 SsmPlayMemList

Starts voice playing in Buffer List Mode.

Format:

int SsmPlayMemList(int ch, PWORD pMemList, WORD wMemListLen)

Parameter Description:

ch	Channel Number
pMemList	Array pointer storing the serial number of the buffer to be played. The serial number of the buffer must be submitted to the driver in advance by the function SsmAddToPlayMemList .
wMemListLen	Length of the array

Return Value:

-1	Call failed, the failure reason can be obtained by the function SsmGetLastErrMsg
0	Successful

Function Description:

Starts voice playing in Buffer List Mode.

After calling this function, the driver starts the task of playing multiple buffers in order according to the sequence designated in pMemList. The task will terminate if one of the following events happens:

- ✧ All voice data in the buffer designated by pMemList have been played out.
- ✧ The application calls SsmStopPlayMemList or SsmStopPlay.
- ✧ The driver detects DTMF, Barge in or remote hangup. For more details, refer to Setting Termination Condition in chapter 1.
- ✧ For some models in the SHD series, the configuration item of LinkFromStopPlayAndTone is set to 1 and the off-bus operation is performed on the channel. For more information, refer to 'Operation Principle of SHD Series' in chapter 1.

After the task of playing multiple buffers is stopped, the driver throws out the event of E_PROC_PlayEnd to the application. The function SsmCheckPlay can also be used to enquire the completion status of this task.

Note:

- If the channel does not support the hardware decoding to the encoding format designated by the function SsmAddToPlayMemList which submits the buffers to the driver, this function will fail. Whether a channel supports the hardware decoding to a specific encoding format is determined by the model of the board. For more details, refer to the related information in Chapter 1.
- If this function is called on channel 0, for ATP/DST Series boards, the voice is sent to the port of the

on-board speaker, but not to the line; for SHT Series boards, the voice is sent not only to the line but also to the port of the on-board speaker.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmStopPlayMemList, SsmStopPlay, SsmCheckPlay, SsmAddToPlayMemList

2.9.7.4 SsmStopPlayMemList

Stops voice playing in Buffer List Mode which is initiated by the function SsmPlayMemList.

Format:

int SsmStopPlayMemList(int ch)

Parameter Description:

ch	Channel Number

Return Value:

-1	Call failed, the failure reason can be obtained by the function SsmGetLastErrMsg
0	Successful

Function Description:

Stops voice playing in Buffer List Mode which is initiated by the function SsmPlayMemList.

Note:

- After this function is called successfully, the driver will still throw out the event of E_PROC_PlayEnd to the application.
- This function can be completely substituted by the function SsmStopPlay.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmPlayMemList, SsmStopPlay

2.9.8 Preload Voice Data Playing Functions (CTI Series)

2.9.8.1 Preload Voice Information Initialization Functions

2.9.8.1.1 SsmSetMaxIdxSeg

Sets the amount of preload voice segments.

Format:

int SsmSetMaxIdxSeg(WORD wMaxIdxSeg)

Parameter Description:

wMaxIdxSeg	The maximum number of index segments

Return Value:

-1	Call failed, the failure reason can be obtained by the function SsmGetLastErrMsg
0	Successful

Function Description:

Sets the amount of preloaded voice segments. The configuration item MaxIndexSeg can implement the same feature.

Note:

- The existing preload voice segments will be unloaded once this function is called.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmGetTotalIndexSeg

2.9.8.1.2 SsmGetTotalIndexSeg

Obtains the amount of preload voice segments.

Format:

int SsmGetTotalIndexSeg ()

Parameter Description: None

Return Value:

-1	Failed
≥0	The amount of index segments

Function Description:

Obtains the amount of preload voice data segments.

Note: None.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmSetMaxIdxSeg

2.9.8.1.3 SsmLoadIndexData

Loads the designated voice segments to the memory.

Format:

int SsmLoadIndexData(int nSegNo, LPSTR pAlias, int nCodec, LPSTR pVocFile, long lStartPos, long lLen)

Parameter Description:

nSegNo	The number assigned to the voice segment, range of value: 0~N. N is the preset amount of the voice data segments which can be obtained by the function SsmGetTotalIndexSeg

<table>
<tr><td>pAlias</td><td>The alias assigned to the voice segment whose maximum length is 20 characters. The numbers ‘0’ ~ ‘9’ cannot be used as the first character of the alias string</td></tr>
<tr><td>nCodec</td><td>Encoding format of the voice data, range of value and the corresponding meanings are listed below:
<table>
<tr><th>Value</th><th>Encoding Format</th><th>Remark</th></tr>
<tr><td>1</td><td>PCM_8</td><td></td></tr>
<tr><td>-2</td><td>PCM_16</td><td></td></tr>
<tr><td>6</td><td>A-Law</td><td></td></tr>
<tr><td>7</td><td>μ-Law</td><td></td></tr>
<tr><td>17</td><td>IMA ADPCM</td><td>The board is required to have the hardware decoding capability</td></tr>
<tr><td>23</td><td>VOX</td><td>The board is required to have the hardware decoding capability</td></tr>
</table>
For more information about whether each specific board model supports the above encoding format and in which way it supports, refer to ‘Board Supported CODECs’ in chapter 1</td></tr>
<tr><td>pVocFile</td><td>The file containing voice segments. It can be a standard wav file or a non-header file without the file header. If it is a standard wav file, the format specified by the parameter nCodec will be ignored. Otherwise, it will be regarded as a non-header file, and the encoding format will be determined by the parameter nCodec</td></tr>
<tr><td>lStartPos</td><td>The starting position counted from the actual voice data in the file. For the standard wav file, the starting position 0 represents the position of the first voice data byte following the file header. For the non-header file, it’s counted from the head part of the file.</td></tr>
<tr><td>lLen</td><td>The length (bytes) of the voice segment.
Note:
● If this parameter is larger than the data length from the position lStartPos to the end of the file, then the latter is adopted as the actual load length.
● If lLen=-1, then data is loaded from the position lStartPos to the end of the file.
The conversion between the byte length and time length of voice segment is determined by the encoding format. For the related information, refer to ‘SynCTI Supported CODECs’ in chapter 1</td></tr>
</table>

Return Value:

-1	Call failed, the failure reason can be obtained by the function SsmGetLastErrMsg
0	Successful

Function Description:

Loads the voice segment data from the specified file to the memory. The encoding format of the voice segment data is determined by the following rule:

1. If pVocFile is a standard WAV file, the parameter nCodec will be ignored.
2. If pVocFile is not a standard WAV file, it will be regarded as a non-header file and the encoding format will be determined by the parameter nCodec.

Note:

- This function should be invoked for each preload voice segment.
- The section SegNo in the configuration file ShIndex.ini can implement the same feature.
- The application must ensure that all the preload voice segments have the same encoding format.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function:

SsmFreeIndexData

2.9.8.1.4 SsmFreeIndexData

Unloads the voice segments in the memory.

Format:

int SsmFreeIndexData(int nSegNo)

Parameter Description:

nSegNo	The number assigned to the voice segment

Return Value:

-1	Call failed, the failure reason can be obtained by the function SsmGetLastErrMsg
0	Successful

Function Description:

Unloads the voice segments in the memory. After the segments being unloaded, the segment number (nSegNo) can be reused for new voice segment data.

Note:

- When the driver uninstalls the function SsmCloseCti, it will automatically release all the resources occupied by preload voice segment data, such as the memory.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmLoadIndexData

2.9.8.1.5 SsmLoadChIndexData

Loads voice segment data from the specified file to the memory of the corresponding channel.

Format:

int WINAPI SsmLoadChIndexData(int ch, int nSegNo, LPSTR pAlias, int nCodec, LPSTR pVocFile, long lStartPos, long lLen)

Parameter Description:

ch	Channel number
nSegNo	The number assigned to voice segment in the memory index
pAlias	The alias assigned to voice segment in the memory index. The maximum length of the alias is 20 characters
nCodec	Encoding format for voice segment data, with the optional values of 6 (A-law), 7(μ-law) and 17(IMA ADPCM)
pVocFile	The file containing voice segment data
lStartPos	The start position of voice segment data in the voice file (the file header excluded)
lLen	The length (bytes) of the voice segment data to be loaded. If this parameter is larger than the data length from the position lStartPos to the end of the file, then the latter is adopted as the actual load length. If lLen=-1, then the data is just loaded from the position lStartPos to the end of the file

Return Value:

-1	Call failed, the failure reason can be obtained by the function SsmGetLastErrMsg
0	Call successful

Function Description:

Loads voice segment data from the specified file to the memory of the corresponding channel. The encoding format of the voice segment data is determined by the following rule:

1. If pVocFile is a standard WAV file, the parameter nCodec will be ignored.
2. If pVocFile is not a standard WAV file, it will be regarded as a non-header file and the encoding format will be determined by the parameter nCodec.

Note:

- When the driver uninstalls the function SsmCloseCti, it will automatically release all the resources occupied by preload voice segment data, such as the memory.

Related Information:

Driver version	SynCTI Ver. 4.8.0.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmFreeChIndexData

2.9.8.1.6 SsmFreeChIndexData

Unloads the voice segment data in the memory.

Format:

int WINAPI SsmFreeChIndexData(int ch, int nSegNo)

Parameter Description:

ch	Channel number
nSegNo	The number assigned to voice segment in the memory index

Return Value:

-1	Call failed, the failure reason can be obtained by the function SsmGetLastErrMsg
0	Call successful

Function Description:

Unloads the voice segment data in the memory. After the segments being unloaded, the segment number (nSegNo) can be reused for new voice segment data.

Note:

- When the driver uninstalls the function SsmCloseCti, it will automatically release all the resources occupied by preload voice segment data, such as the memory.

Related Information:

Driver version	SynCTI Ver. 4.8.0.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmLoadChIndexData

2.9.8.2 Functions for Playing Preload Voice Segments

2.9.8.2.1 SsmPlayIndexString

Refer to SsmPlayIndexList.

2.9.8.2.2 SsmPlayIndexList

Plays one or multiple designated preload voice segments in sequence.

Format:

int SsmPlayIndexList(int ch, WORD wIdxListLen, PWORD pwIdxList)

int SsmPlayIndexString(int ch, LPSTR pszIdxStr)

Parameter Description:

ch	Channel Number
wIdxListLen	The amount of preload voice segments to be played, range of value: 1~N-1. N is set by the configuration item MaxPlayIndexList.
pwIdxList	The array storing the number of the preload voice segments. The voice segments must have the same encoding format.
pszIdxStr	The string containing the information of preload voice segments. It is used for storing the index of the alias or number of voice segments to be played. The preload voice segments are separated by ',', and each of them can be denoted by a number or an alias. The corresponding relationship between the number (or alias) and the voice segment is determined by the system initialization function SsmStartCti according to the preload voice configuration file ShIndex.ini; also it can be loaded by the function SsmLoadIndexData and unloaded by the function SsmFreeIndexData respectively. The amount of the preload voice segments designated in pszIdxStr cannot exceed the set value in the configuration item MaxPlayIndexList. Note: All the voice segments designated in pszIdxStr must have the same encoding format.

Return Value:

-1	Call failed, the failure reason can be obtained by the function SsmGetLastErrMsg
0	Successful

Function Description:

Plays the preloaded voice segments designated by pwIdxList or pszIdxStr.

After calling this function, the driver will start the task of playing the preload voice segments. The task will terminate if one of the following events happens:

- ✧ All voice data designated by pwIdxList or pszIdxStr have been played out.
- ✧ The application calls the function SsmStopPlayIndex or SsmStopPlay.
- ✧ The driver detects DTMF, Barge in or remote hangup. For more information, refer to Setting Termination Condition in chapter 1.
- ✧ For some models in the SHD series, the configuration item of LinkFromStopPlayAndTone is set to 1 and the off-bus operation is performed on the channel. For more information, refer to 'Operation Principle of SHD Series' in chapter 1.

After the task of playing the preload voice data is stopped, the driver will throw out the event of E_PROC_PlayEnd to the application. The function SsmCheckPlay can also be used to check the completion status of this task.

The application can invoke the function of SsmStopPlayIndex or SsmStopPlay to stop the voice playing at any

moment.

Note:

- This function only supports the channels which are designated by pwIdxList or pszIdxStr with the hardware decoding capability. For more information, refer to the function SsmLoadIndexData.
- If this function is called on channel 0, for ATP/DST Series boards, the voice is sent to the port of the on-board speaker, but not to the line; For SHT Series boards, the voice is sent not only to the line but also to the port of the on-board speaker.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmStopPlayIndex, SsmStopPlay, SsmCheckPlay, SsmLoadIndexData

2.9.8.2.3 SsmStopPlayIndex

Stops playing preload voice segments.

Format:

int SsmStopPlayIndex(int ch)

Parameter Description:

ch	Channel Number

Return Value:

-1	Call failed, the failure reason can be obtained by the function SsmGetLastErrMsg
0	Successful

Function Description:

Stops playing preload voice segments. After the task of playing preload voice segments is stopped, the driver will set the cause of termination to be CHKPLAY_APPLICATION_END and throw out the event of E_PROC_PlayEnd to the application.

Note:

- The task of voice playing should be started by the function SsmPlayIndexString or SsmPlayIndexList.
- This function can be completely substituted by the function SsmStopPlay.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmPlayIndexString, SsmPlayIndexList, SsmStopPlay

2.9.9 Functions for Playing in Pingpong Buffer Mode(CTI Series)

2.9.9.1 SsmPlayMemBlock

Submits a buffer to the driver and starts voice playing in Pingpong Buffer Mode.

Format:

int SsmPlayMemBlock(int ch, int nFormat, LPBYTE pBuf, DWORD dwBufSize, PLAYMEMBLOCKHANDLER pfnCallback, PVOID pV)

Parameter Description:

ch	Channel Number
nFormat	Voice encoding format. Range of value: -2:PCM16 1:PCM8 6:A-law 7:μ-law 17:IMA ADPCM (The board is required to have the hardware decoding capability)
pBuf	The pointer pointing to the voice data buffer
dwBufSize	The size (Bytes) of the voice data buffer. The minimum value of dwBufSize is determined by the following rules: ● If the configuration item RecordAndPlayUseAsIP is set 0 (default), dwBufSize should be no less than 768 bytes. ● If RecordAndPlayUseAsIP is set 1, for A-law, μ-law, PCM8 and PCM16, dwBufSize should be no less than 192 bytes (recommended value: under Windows it should be no less than 192 bytes; under Linux it should be no less than 384 bytes); For IMA ADPCM, the size of the buffer should be no less than 768 bytes.
pfnCallback	Pointer of callback function. Once the task of voice playing of a buffer is finished or terminated, the driver will automatically invoke the callback function set by this parameter. NULL means not to set the callback function. The prototype of the callback function is declared in shpa3api.h: typedef BOOL (*PLAYMEMBLOCKHANDLER) (int **ch**, int **nEndReason**, PUCHAR **pucBuf**, DWORD **dwStopOffset**, PVOID **pV**); in which,**ch**: Channel Number; **nEndReason**: The reason of termination, range of value: 1: All data in the buffer have been played out; 2: The playing task is stopped because the DTMF characters are detected. 3: The playing task is stopped because barge-in is detected. 4: The playing task is stopped because remote hangup is detected. 5: The playing task is stopped by the application. **pucBuf**: The pointer pointing to the buffer where the playing task is finished. **dwStopOffset**: The offset (bytes) of the driver's playing pointer in the buffer when the playing task is finished. **pV**: The pointer passed by the application upon invoking this function.
pV	Pointer of PVOID type. It is commonly used by the application for transmitting necessary data, such as the data structure, to the callback function. This pointer is passed to the callback function directly.

Return Value:

-1	Call failed, the failure reason can be obtained by the function SsmGetLastErrMsg
0	Successful

Function Description:

Submits a buffer to the driver and starts voice playing in Pingpong Buffer Mode. For more information about voice playing in Pingpong Buffer Mode, refer to Pingpong Buffer Mode in Chapter 1.

When it is the first time to invoke this function, the driver will start voice playing in Pingpong Buffer Mode in the sequence of buffer submitting. Upon a complete play of voice data in the first buffer, the driver will automatically play those in the second buffer and meanwhile invoke the callback function from the application to pass an end-of-play message to the application. In the callback function, the application will fill data into the next buffer and restart such process.

The task will terminate if one of the following events happens:

- ✧ All voice data in the submitted buffer have been played out.
- ✧ The application calls the function SsmStopPlayMemBlock or SsmStopPlay.
- ✧ The driver frtects DTMF, Barge in or remote hangup. For more information, refer to Setting Termination Condition in chapter 1.
- ✧ For some models in the SHD series, the configuration item of LinkFromStopPlayAndTone is set to 1 and the off-bus operation is performed on the channel. For more information, refer to 'Operation Principle of SHD Series' in chapter 1.

The function SsmCheckPlay can also be used to check the completion status of the task of voice playing in Pingpong Buffer Mode.

Note:

- If the channel does not support the hardware decoding to the encoding format designated by the parameter nFormat, this function will fail. Whether a channel supports the hardware decoding to a specific encoding format is determined by the model of the board. For more details, refer to the related information in Chapter 1.
- In the callback function, the time for the application to update the voice data and execute the submitted code should be minimized and not exceed the interruption time of the driver (8ms). Otherwise it may cause driver error.
- The application can invoke the function SsmStopPlayMemBlock or SsmStopPlay to stop the double-buffer voice playing, but these two functions can't be called in the callback function.
- After the task of voice playing in Pingpong Buffer Mode is stopped, the driver will throw out the event of E_PROC_PlayEnd to the application.
- If this function is called on Channel 0, for ATP/DST Series boards, the voice is sent to the port of the on-board speaker, but not to the line; For SHT Series boards, the voice is sent not only to the line but also to the port of the on-board speaker.
- It is not allowed to submit more than two buffers to the driver at the same time.
- It is not allowed to use the timer to invoke the function SsmPlayMemBlock.

Sample Code:

```
......//initialize and fill the buffers szPlayBuf1, szPlayBuf2
BOOL PlayStart(int nCh)
{
if(SsmPlayMemBlock(nCh, 6, szPlayBuf1, dwBufSize, PlayCallBack, NULL) != 0)//submit the buffer szPlayBuf1 to the driver and start voice playing in Pingpong Buffer Mode
{
SsmGetLastErrMsg(szErrMsg);
cout<<szErrMsg<<endl;
return FALSE;
}
if(SsmPlayMemBlock(nCh, 6, szPlayBuf2, dwBufSize, PlayCallBack, NULL) != 0)//submit the buffer szPlayBuf2 to the driver
{
SsmGetLastErrMsg(szErrMsg);
cout<<szErrMsg<<endl;
return FALSE;
```

```
}
return TRUE;
}
BOOL PlayCallBack(int ch, int nEndReason, PUCHAR pucBuf, DWORD dwStopOffset, PVOID pV)
{
        int nResult = 0;
        if(pucBuf == szPlayBuf1)
{
                .....//refill the buffer szPlayBuf1
nResult = SsmPlayMemBlock(ch, 6, szPlayBuf1, dwBufSize, PlayCallBack, pV);)//submit the buffer szPlayBuf1 to the driver
}
else if(pucBuf == szPlayBuf2)
{
                .....//refill the buffer szPlayBuf2
nResult = SsmPlayMemBlock(ch, 6, szPlayBuf2, dwBufSize, PlayCallBack, pV);//submit the buffer szPlayBuf2 to the driver
}
return nResult;
}
```

Related Information:

Driver version	SynCTI Ver. 3.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmStopPlayMemBlock, SsmStopPlay, SsmPlayMem, SsmPlayMemList

2.9.9.2 SsmStopPlayMemBlock

Stops the task of voice playing in Pingpong Buffer Mode initialized by SsmPlayMemBlock.

Format:

int SsmStopPlayMemBlock(int ch)

Parameter Description:

ch	Channel Number

Return Value:

-1	Call failed, the failure reason can be obtained by the function SsmGetLastErrMsg
0	Successful

Function Description:

Stops the task of voice playing in Pingpong Buffer Mode initialized by SsmPlayMemBlock.

Note:

- This function cannot be invoked in the callback function which is set by the function SsmPlayMemBlock.
- This function can be completely substituted by the function SsmStopPlay.

Related Information:

Driver version	SynCTI Ver. 3.0or above

Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmPlayMemBlock, SsmStopPlay

2.9.10 General Functions for Terminating Voice Playing

2.9.10.1 SsmStopPlay

Stops voice playing in any mode.

Format:

int SsmStopPlay(int ch)

Parameter Description:

ch	Channel Number

Return Value:

-1	Call failed, the failure reason can be obtained by the function SsmGetLastErrMsg
0	Successful

Function Description:

Stops voice playing in any mode.

Note:

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmStopPlayFile, SsmStopPlayFileList, SsmStopPlayMem, SsmStopPlayMemList, SsmStopPlayIndex, SsmStopPlayMemBlock

2.9.11 Obtaining Relative Information about Voice Playing

2.9.11.1 SsmQueryPlayFormat

Queries if the channel supports the designated voice playing format.

Format:

int SsmQueryPlayFormat (int ch, int nFormat)

Parameter Description:

ch	Channel Number
nFormat	Voice playing format, for the range of value and more detailed information, refer to SynCTI Supported CODECs in Chapter 1

Return Value:

0	Not supported
1	Supported with hardware decoding
2	Supported with software decoding

Function Description:

Queris if the channel supports the designated voice playing format.

Note:

- This function is applicable to the channels of any type.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function:

2.9.11.2 SsmGetPlayType

Queries the mode of voice playing performed currently on the channel.

Format:

int SsmGetPlayType(int ch)

Parameter Description:

ch	Channel Number

Return Value:

-1	Call failed
0	No voice playing
1	Playing a single file
2	Playing a single file, but the file playing is suspended by the application
3	Playing the file list
4	Playing the preload voice
5	Playing a single buffer
6	Playing the buffer list
7	Transmitting FSK data (Note: Transmitting FSK data will occupy the resource of voice playing)
8	Using the voice-playing operation to achieve voice exchange between the channels without CT-bus
9	Playing the Pingpong buffer

Function Description:

Queries the mode of voice playing performed currently on the channel.

Note:

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function:

2.9.11.3 SsmCheckPlay

Obtains the completion status of voice playing.

Format:

int SsmCheckPlay(int ch)

Parameter Description:

ch	Channel Number

Return Value:

-1	Call failed
0	Voice playing in progress
1	Voice playing is ended: All voice data played out or no voice playing task on the channel
2	The playing task is stopped because DTMF characters are detected. For more information, refer to Setting Termination Condition in Chapter 1.
3	The playing task is stopped because bargein is detected. For more information, refer to Setting Termination Condition in Chapter 1.
4	The playing task is stopped because remote hangup is detected. For more information, refer to Setting Termination Condition in Chapter 1.
5	The playing task is terminated by the application
6	The playing task is suspended by the function SsmPausePlay which is invoked by the application. To start the voice playing task, invoke the function SsmPlayFile
7	The playing task is terminated due to the off-bus operation on the channel. For more information, refer to Setting Termination Condition in Chapter 1
8	The playing task is terminated due to network disconnection. When the application is invoking the function SsmPlayFile to play voice files in other computers located in the network, and the driver fails to read the files due to such reasons as network disconnection, it returns this value.

Function Description:

Obtains the completion status of the voice playing.

Note:

- The voice playing task can be initialized by the following functions: SsmPlayFile, SsmPlayFileList, SsmPlayMemList, SsmPlayIndexList, SsmPlayIndexString, SsmPlayMem, and SsmPlayMemBlock.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Functions: SsmPlayFile, SsmPlayFileList, SsmPlayMemList, SsmPlayIndexList, SsmPlayIndexString, SsmPlayMem, SsmPlayMemBlock

Related Events: E_PROC_PlayEnd, E_PROC_PlayFile, E_PROC_PlayFileList, E_PROC_PlayMem

2.9.11.4 SsmGetPlayedPercentage

Obtains the percentage of the played data to the whole.

Format:

int SsmGetPlayedPercentage(int ch)

Parameter Description:

ch	Channel Number

Return Value:

-1	Call failed
≥0	The percentage of the played bytes to the whole

Function Description:

Obtains the percentage of the played data to the whole.

Note:

- This function supports the voice playing task initialized by the function SsmPlayIndexList, SsmPlayIndexString, SsmPlayFile, SsmPlayFileList, and SsmPlayMemList.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmGetPlayedTime

2.9.11.5 SsmGetPlayedTime

Obtains the duration of the current voice playing.

Format:

int SsmGetPlayedTime(int ch)

Parameter Description:

ch	Channel Number

Return Value:

-1	Call failed, the failure reason can be obtained by the function SsmGetLastErrMsg
≥0	The duration (ms) of the voice playing

Function Description:

Obtains the duration of the current voice playing.

Note:

- This function supports the voice playing tasks initialized by the function SsmPlayFile, SsmPlayFileList, SsmPlayMemList, SsmPlayIndexList, SsmPlayIndexString, and SsmPlayMem.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmGetPlayedPercentage, SsmGetDataBytesToPlay, SsmGetPlayedTimeEx, SsmGetPlayingFileInfo.

2.10 Voice Recording Functions

2.10.1 Setting Recording Parameters

2.10.1.1 Setting Signal Source and Its Volume

2.10.1.1.1SsmSetRecVolume

Refer to SpySetRecVolume.

2.10.1.1.2SpySetRecVolume

Sets the gain of the volume adjuster A3 for the incoming signal entering the recording mixer M3.

Format:

int SsmSetRecVolume(int ch, int nVolume)

int SpySetRecVolume(int nCic, WORD wDirection, int nVolume)

Parameter Description:

ch	Channel Number
nCic	The logic number of SpyCic. For more information, refer to 'DTP Series' in Chapter 1
wDirection	The direction of the volume adjuster, range of value: 0: Sets the calling party's volume, used for the occasion to record the calling party only 1: Sets the called party's volume, used for the occasion to record the called party only 2: Sets both the calling party's and the called party's volume simultaneously, used for the occasion to record the calling party or the called party.
nVolume	Recording volume, range of value: -7~+6, -7 means turn-off, and the gain results correspond to: —infinite, -18DB, -12DB, -8.5DB, -6DB, -4DB, -2.5DB, 0DB, +2.8DB, +6DB, +9.2DB, +12DB, +15DB, +18DB,. The negative value means volume decreasing and the positive value means the volume increasing

Return Value:

-1	Call failed, the failure reason can be obtained by the function SsmGetLastErrMsg
0	Successful

Function Description:

Sets the gain of the volume adjuster A3 for the incoming signal entering the recording mixer M3. For more details about M3 and A3, refer to the corresponding operation principles in Chapter 1.

Note:

- The configuration item DefaultRecordVolume can implement the same feature, with the default value of 0.
- The parameters set by this function are applicable to all types of recording functions except for SynIPRecorder. To set the volume for SynIPRecorder, use the function SsmIPRSetRecVolume.

Related Information:

Driver version	SsmSetRecVolume requires SynCTI Ver. 2.0 or above; SpySetRecVolume requires SynCTI Ver.5.3.1.3 or above.
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmSetRecMixer, SsmSetRecBack, SsmIPRSetRecVolume

2.10.1.1.3SsmSetRecMixer

Refer to SpySetRecMixer.

2.10.1.1.4SpySetRecMixer

Sets the gain of the volume adjuster A2 for other signal sources entering the recording mixer M3.

Format:

int SsmSetRecMixer(int ch, BOOL bEnRecMixer, int nMixerVolume)

int SpySetRecMixer(int nCic, BOOL bEnRecMixer, int nMixerVolume)

Parameter Description:

ch	Channel Number
nCic	The logic number of SpyCic. For more information, refer to 'DTP Series' in Chapter 1
bEnRecMixer	=FALSE: turns off the volume adjuster A2, the incoming signal of the current channel is the only data source for recording; =TRUE: turns on the volume adjuster A2, the data source for the recording operation includes both the incoming signal of the current channel and other sources
nMixerVolume	The volume gain of other recording signal sources, it is valid only when the value of bEnRecMixer is set to True. Range of value: -7~+6, in which, -7: turns off A2, equivalent to setting the value of bEnRecMixer to False. -6~6: volume gain (multiplied by 3 equals the dB value), The negative value means volume decreasing and the positive value means the volume increasing

Return Value:

-1	Call failed
0	Successful

Function Description:

Sets the gain of the volume adjuster A2 for the other signal sources entering the recording mixer M3. For more detailed information about M3 and A2, refer to the corresponding operation principles in Chapter 1.

Note:

- The configuration item DefaultRecordMixerVolume can implement the same features, A2 is turned off by default.
- The parameters set by this function are valid to the recording functions of all types.
- When the function SpySetRecMixer is used, if the current SpyCic is not recording, this function only sets the volume of the mixer on the calling party's channel but do not switch it on, and bEnRecMixer is invalid; if the current SpyCic is performing a mix recording, this function sets the switch and the volume of the mixer on the calling party's channel, and bEnRecMixer is valid; if the current SpyCic is recording in other modes, this function will fail.
- This function does not support DST Series boards.
- To prevent noise, the volume adjuster A2 must be switched off before the channels on two different boards start bus exchanging and switching using the soft-switch feature.
- SsmSetRecMixer does not support ATP-24A Series boards.

Related Information:

Driver version	SsmSetRecMixer requires SynCTI Ver. 2.0 or above; SpySetRecMixer requires SynCTI Ver.5.3.1.3 or above.
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmSetRecVolume, SsmSetRecBack, SsmGetRecMixerState, SsmQueryOpRecMixer

2.10.1.1.5SsmQueryOpRecMixer

Queries if the channel supports the gain set of the volume adjuster A2.

Format:

int SsmQueryOpRecMixer(int ch)

Parameter Description:

ch	Channel Number

Return Value:

-1	Call failed, the failure reason can be obtained by the function SsmGetLastErrMsg
0	Not supported
1	Supported

Function Description:

Queries if the channel supports the gain set of the volume adjuster A2.

Note:

- This function doesn't support DST Series boards.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmSetRecMixer

2.10.1.1.6SsmGetRecMixerState

Obtains the gain settings of the volume adjuster A2.

Format:

int SsmGetRecMixerState(int ch, int* pnEnRecMixer, int* pnMixerVolume)

Parameter Description:

ch	Channel Number
pnEnRecMixer	The pointer storing the switching status of the recording mixer
pnMixerVolume	The pointer storing the volume of the voice signal from another channel which enters the recording mixer to join the incoming signal of the current channel. The obtained volume multiplied by 3 equals its dB value

Return Value:

-1	Failed
0	Successful

Function Description:

Obtains the gain settings of the volume adjuster A2.

Obtains the status of the recording mixer on the specified channel. If the switching status of the recording mixer is turned on, the parameter pnEnRecMixer is set 1, otherwise, it is set 0. The parameter pnMixerVolume is used to return the volume of the voice signal from another channel which enters the recording mixer to join the incoming signal of the current channel.

Note:

- The settings of the recording mixer switch are valid to both File Mode and Buffer Mode recording functions.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmSetRecMixer, SsmQueryOpRecMixer

2.10.1.1.7DTRSetMixerVolum

Sets the gain of the volume adjusters A4-1, A4-2 for the incoming/outgoing signal on DST Series boards.

Format:

int DTRSetMixerVolume(int ch, int nGroup, int nInboundGain, int nOutboundGain)

Parameter Description:

ch	Channel Number
nGroup	Chooses B channel. The digital phone operates in 2B+D mode. Because the current driver can't process two B channels at the same time, this parameter must be set to 0
nInboundGain	The gain of the volume adjuster for incoming signals (i.e. volume adjuster A4-1 in Operation Principle of DST Series in Chapter 1), range of value: -7~+6, the negative value means volume decreasing, the positive value means the volume increasing , -7 means turn-off, this parameter multiplied by 3 equals dB value
nOutboundGain	The gain of the volume adjuster for outgoing signals (i.e. volume adjuster A4-2 in Operation Principle of DST Series in Chapter 1), range of value: -7~+6, the negative value means volume decreasing, the positive value means the volume increasing, -7 means turn-off, this parameter multiplied by 3 equals dB value

Return Value:

-1	Failed
0	Successful

Function Description:

Sets the gain of the volume adjusters A4-1, A4-2 for the incoming/outgoing voice signal on DST Series boards.

For more detailed information about A4-1, A4-2, refer to Operation Principle of DST Series in Chapter 1.

Note:

- This function only supports DST Series boards.

Related Information:

Driver version	SynCTI Ver. 4.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmSetRecVolume, DTRGetMixerVolume

2.10.1.1.8DTRGetMixerVolume

Obtains the gain of the volume adjusters A4-1, A4-2 for the incoming/outgoing signal on DST Series boards.

Format:

int DTRGetMixerVolume(int ch, int nGroup, int* pnInboundGain, int* pnOutboundGain)

Parameter Description:

ch	Channel Number
nGroup	Chooses B channel, it must be set 0
pnInboundGain	Returns the gain of the volume adjuster for incoming signal (i.e. volume adjuster A4-1 in Operation Principle of DST Series in Chapter 1), range of value: -6~+6, the negative value means volume decreasing, the positive value means the volume increasing, this parameter multiplied by 3 equals dB value

pnOutboundGain	Returns the gain of the volume adjuster for outgoing signal (i.e. volume adjuster A4-2 in Operation Principle of DST Series in Chapter 1), range of value: -6~+6, the negative value means volume decreasing, the positive value means the volume increasing, this parameter multiplied by 3 equals dB value

Return Value:

-1	Failed
0	Successful

Function Description:

Obtains the gain of the volume adjusters A4-1, A4-2 for the incoming/outgoing signal on DST Series boards. For more detailed information about A4-1, A4-2, refer to Operation Principle of DST Series in Chapter 1.

Note:

- This function only supports DST Series boards.

Related Information:

Driver version	SynCTI Ver. 4.0or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: DTRSetMixerVolume

2.10.1.1.9SsmSetRecBack

Sets whether to take the output of the bus mixer and the voice playing data as the input signals of the recording mixer.

Format:

int SsmSetRecBack(int ch, int nSelector)

Parameter Description:

ch	Channel Number
nSelector	Sets the status of the switches k6-2 and k6-1, range of value: 0: k6-2 switched off, k6-1 switched off 1: k6-2 switched on, k6-1 switched on 2: k6-2 switched off, k6-1 switched on 3: k6-2 switched on, k6-1 switched off The default value is 1, i.e. both k6-2 and k6-1 switched on. For more information about these two switches, refer to Operation Principle of SHD Series in Chapter 1

Return Value:

-1	Call failed, the failure reason can be obtained by the function SsmGetLastErrMsg
0	Successful

Function Description:

Sets whether to take the output of the bus mixer and the voice playing data as the input signals of the recording mixer.

Note:

- This function only supports SHD and SHN Series boards.

Related Information:

Driver version	SynCTI Ver. 4.0 or above
Header	shpa3api.h

Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmSetRecVolume, SsmSetRecMixer

2.10.1.1.10 SsmSetNoModuleChBusRec

Uses the non-module channels of ATP Series boards as recording channels.

Format:

int SsmSetNoModuleChBusRec(int ch, int nUsedAsResourceRecCh)

Parameter Description:

ch	Channel Number
nUsedAsResourceReCh	1: enables this feature 0: disables this feature

Return Value:

-1	Call failed, the failure reason can be obtained by the function SsmGetLastErrMsg
0	Successful

Function Description:

Uses the non-module channels of ATP Series boards as recording channels. For more information, refer to 'special application of non-module channel' in Operation Principle of ATP Series of Chapter 1.

Note:

- Before recording, invoke the function SsmSetRecVolume or set the configuration item DefaultRecordVolume to turn off the volume adjuster of the incoming signal.
- The signal source for recording only comes from the TDM bus.
- The configuration item NoModuleChBusRec can implement the same feature.
- This function only supports ATP Series boards.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmSetRecVolume

2.10.1.2 Setting Termination Condition for Voice Recording

2.10.1.2.1SsmSetDTMFStopRecCharSet

Sets whether the recording stops when the DTMF Detector detects DTMF character.

Format:

int SsmSetDTMFStopRecCharSet(int ch, LPSTR lpstrDtmfCharSet)

Parameter Description:

ch	Channel Number
lpstrDtmfCharSet	Pointer pointing to the ASCII-formatted DTMF character set which is used to stop the recording.

	The DTMF character set can include ‘0’, ’1’, ’2’, ’3’, ’4’, ’5’, ’6’, ’7’, ’8’, ’9’, ’*’, ’#’, ’a’, ’b’, ’c’, ’d’, it can also be set NULL to indicate that this feature is disabled.

Return Value:

-1	Call failed. The failure reason can be obtained by the function SsmGetLastErrMsg
0	Successful

Function Description:

Sets whether the recording stops when the DTMF Detector detects DTMF character.

Provided that the parameter lpstrDtmfCharSet is not null, if the DTMF detector detects DTMF characters designated in lpstrDtmfCharSet in the incoming signals while recording, the driver will stop the recording immediately.

Note:

- The configuration item DtmfStopRecCharSet can implement the same feature.
- This feature supports recording in File Mode, Buffer Mode and Pingpong Buffer Mode.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmSetHangupStopRecFlag, SsmGetDTMFStopRecCharSet

2.10.1.2.2SsmGetDTMFStopRecCharSet

Queries if the recording stops when the DTMF Detector detects the DTMF character.

Format:

int SsmGetDTMFStopRecCharSet(int ch, LPSTR lpstrDtmfCharSet)

Parameter Description:

ch	Channel Number
lpstrDtmfCharSet	Returns ASCII-formatted DTMF character set

Return Value:

-1	Call failed. The failure reason can be obtained by the function SsmGetLastErrMsg
0	This feature is disabled
>0	The number of preset characters in the DTMF character set

Function Description:

Queries if the recording stops when the DTMF Detector detects the DTMF character.

Note:

Related Information:

Driver version	SynCTI Ver. 3.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmSetDTMFStopRecCharSet

2.10.1.2.3SsmSetHangupStopRecFlag

Sets whether the recording is stops when the driver state machine detects remote end hangup.

Format:

int SsmSetHangupStopRecFlag(int ch, BOOL bHangupStopRecFlag)

Parameter Description:

ch	Channel Number
bHangupStopRecFlag	The flag denotes whether the recording stops when the driver detects remote end hangup. Range of value: TRUE: enable this feature FALSE: disable this feature

Return Value:

-1	Call failed. The failure reason can be obtained by the function SsmGetLastErrMsg
0	Successful

Function Description:

Sets whether the recording stops when the driver state machine detects remote end hangup.

Note:

- The configuration item HangupStopRec can implement the same feature. This feature is disabled by default.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmSetDTMFStopRecCharSet

2.10.1.3 Setting Prerecord Feature (REC Series)

2.10.1.3.1SsmSetPrerecord

Sets the prerecord feature.

Format:

int SsmSetPrerecord(int ch, BOOL bEnable, int nMode, WORD wInsertTime, int nFormat)

Parameter Description:

ch	Channel Number
bEnable	TRUE: enable the prerecord feature FALSE: disable the prerecord feature **Note:** The corresponding configuration item to this parameter is PrerecordEnable
nMode	Sets the operating mode of the prerecord feature: 0: Starts writing recording data into the buffer when the Barge-in Detector detects barge in 1: Starts writing recording data into the buffer when the analog recording channel detects pick-up on the line **Note:** The corresponding configuration item to this parameter is PrerecordMode
nInsertTime	Sets the time length (ms) of writing the prerecord data to into the file after the prerecord

	feature is enabled and the File Mode recording is started. **Note:** The corresponding configuration item to this parameter is PrerecordInsertTime
nFormat	Sets the encoding format for recording, range of value: 1: PCM8 6: A-law 7: μ-law 17: IMA ADPCM(Note: the board is required to have hardware encoder) 49: GSM 6.10 85: MP3 **Note:** The corresponding configuration item to this parameter is PrerecordCodec

Return Value:

-1	Call failed
0	Successful

Function Description:

Sets the prerecord feature. For more details about the prerecord feature, refer to Setting Prerecord Feature in Chapter 1.

Note:

- This function is applicable to ATP Series boards, the analog trunk channels of SHT and SHF Series boards, the TUP, ISUP and ISDN channels of SHD Series boards;
- Prerecord feature only supports functions for File Mode recording.

Related Information:

Driver version	SynCTI Ver. 5.3.3.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmGetPrerecordState

2.10.1.3.2SsmGetPrerecordState

Obtains the settings of prerecord feature.

Format:

int SsmGetPrerecordState (int ch , int* pnMode, PWORD pwInsertTime, int* pnFormat)

Parameter Description:

ch	Channel Number
pnMode	Returns the operating mode of prerecord
pnInsertTime	Returns the time length (ms) of writing the prerecord data into the file
pnFormat	Returns the recording format for prerecord

Return Value:

-1	Call failed
0	Disable the prerecord feature
1	Enable the prerecord feature. The return values of the above parameters are valid only when this function returns 1

Function Description:

Obtains the settings of prerecord feature.

Note:

- For the meanings of related parameters, refer to the function SsmSetPrerecord;
- This function is applicable to ATP Series boards and the analog trunk channels of SHT and SHF Series boards;
- The prerecord feature is only supported for File Mode recording functions.

Related Information:

Driver version	SynCTI Ver. 2.1or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmSetPrerecord

2.10.1.4 Setting Tail-truncation Feature

2.10.1.4.1SsmSetTruncateTail

Sets the operating parameters for tail-truncation feature.

Format:

int SsmSetTruncateTail(int ch, DWORD dwTime)

Parameter Description:

ch	Channel Number
dwTime	Sets the length (ms) of the recording data to be truncated. 0 means to disable this feature. For SynCTI Ver. 3.4.x, the time length of tail-truncation is the duration of DTMF character which stops the file recording; For Ver. 3.5 and later versions, if dwTime is greater than duration of the DTMF character which stops the recording, the time length of the truncated data is dwTime; Otherwise, it's the actual duration of the DTMF character. This parameter can also be set via the configuration item TruncateTailOnRecordToFile

Return Value:

-1	Call failed, the failure reason can be obtained by the function SsmGetLastErrMsg
0	Successful

Function Description:

Sets the parameters for tail-truncation feature. For more information about tail-truncation feature, refer to Setting Tail-Truncation Feature in Chapter 1.

Provided the tail-truncation feature and the feature of recording termination by DTMF character are enabled, once the DTMF detector detects a DTMF character which is in the DTMF character set designated by the function SsmSetDTMFStopRecCharSet or the configuration item DtmfStopRecCharSet during File Mode recording, the driver will stop the recording immediately and truncate the last section of the voice data. The length of the truncated voice data is set by this function.

Note:

- If the file recording is terminated normally, i.e. it completes the recording of the voice data of the designated length, the driver will not perform the tail-truncation operation;
- The tail-truncation feature only supports the functions for File Mode recording.

Related Information:

Driver version	SynCTI Ver. 3.0 or above

Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmRecToFile, SsmRecToFileA, SsmRecToFileB, SsmRecToFileEx, SsmGetTruncateTailTime

2.10.1.4.2SsmGetTruncateTailTime

Obtains the preset parameters for the tail-truncation feature.

Format:

long SsmGetTruncateTailTime(int ch)

Parameter Description:

ch	Channel Number

Return Value:

-1	Call failed, the failure reason can be obtained by the function SsmGetLastErrMsg
0	This feature is disabled
>0	Returns the time length (ms) of the voice data to be truncated

Function Description:

Obtains the preset parameters for the tail-truncation feature.

Note:

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmSetTruncateTail

2.10.1.5 Setting AGC Feature

2.10.1.5.1SsmSetRecAGC

Enables or disables the AGC feature.

Format:

int SsmSetRecAGC(int ch, int nAGCSwitch)

Parameter Description:

ch	Channel Number
nAGCSwitch	Automatic gain control switch for recording (AGC). For DST series B-type digital station tap boards, Bit0, Bit1 and Bit2 are valid. Bit0=0, the AGC downlink is disabled; Bit0=1, the AGC downlink is enabled. Bit1=0, the AGC uplink is disabled; Bit1=1, the AGC uplink is enabled. Bit2=0, DC isolation is disabled; Bit2=1, DC isolation is enabled. For other boards, Bit0 is valid. Bit0=0, the AGC is disabled; Bit0=1, the AGC is enabled.

Return Value:

-1	Call failed, the failure reason can be obtained by the function SsmGetLastErrMsg

0	Successful

Function Description:

Enables or disables the AGC (Automatic Gain Control) feature. If the AGC is enabled, the driver will automatically adjust the voice signal according to the amplitude of the input signal: increase the small signal gain and decrease the large signal gain.

Note:

- The AGC switch is valid to recordings of all types.
- The configuration item AutoRecAgcSwitch can implement the same feature with the default value of 0.
- For SHT Series, if the AGC feature should always be enabled, besides using this function, you should set the configuration item OpenRecEnAndPlayEnOnIdle to 1.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmGetRecAGCSwitch

2.10.1.5.2SsmGetRecAGCSwitch

Obtains the working status of the recording AGC.

Format:

int SsmGetRecAGCSwitch(int ch)

Parameter Description:

ch	Channel Number

Return Value:

-1	Call failed
>0	For DST series B-type digital station tap boards, Bit0, Bit1 and Bit2 are valid. Bit0=0, the AGC downlink is disabled; Bit0=1, the AGC downlink is enabled. Bit1=0, the AGC uplink is disabled; Bit1=1, the AGC uplink is enabled. Bit2=0, DC isolation is disabled; Bit2=1, DC isolation is enabled. For other boards, Bit0 is valid. Bit0=0, the AGC is disabled; Bit0=1, the AGC is enabled.

Function Description:

Obtains the working status of the recording AGC module on the designated channel.

Note:

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmSetRecAGC

2.10.1.6 Setting Stereo Recording

2.10.1.6.1SsmSetRecStereo

Enables or disables the stereo recording.

Format:

int WINAPI SsmSetRecStereo(int ch, BOOL bRecStereo)

Parameter Description:

ch	Channel Number
bRecStereo	Switch of the stereo recording TRUE: Enable; FALSE: Disable

Return Value:

-1	Call failed, the failure reason can be obtained by the function SsmGetLastErrMsg
0	Successful

Function Description:

Enables or disables the stereo recording. If this feature is enabled, the generated recording file is a stereo audio file; otherwise, the generated recording file is a mono audio file.

Note:

- This function is only applicable to DST series B-type boards.
- This function only supports A-law and μ-law recording formats.

Related Information:

Driver version	SynCTI Ver. 5.3.2.2 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function:

2.10.2 Obtaining Recording Information

2.10.2.1 SsmQueryRecFormat

Queries if the channel supports the designated recording format.

Format:

int SsmQueryRecFormat (int ch, int nFormat)

Parameter Description:

ch	Channel Number
nFormat	Recording format

Return Value:

0	Not supported
1	Supported with hardware encoding
2	Supported with software encoding

Function Description:

Queries if the channel supports the designated recording format.

Note:

- This function supports boards and channels of all types.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function:

2.10.2.2 SsmGetRecType

Obtains the type of the current recording task.

Format:

int SsmGetRecType(int ch)

Parameter Description:

ch	Channel Number

Return Value:

-1	Call failed
0	No recording on the channel
1	Recording in File Mode
2	The File Mode recording is suspended
3	Recording in Buffer Mode
4	Rrecording in Pingpong Buffer Mode

Function Description:

Obtains the type of the current recording task.

Note:

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function:

2.10.2.3 SsmGetRecTime

Obtains the duration of the recording task.

Format:

int SsmGetRecTime(int ch)

Parameter Description:

ch	Channel Number

Return Value:

-1	Call failed
≥0	the duration (ms) of the recording task

Function Description:

Obtains the duration of the recording task.

Note:

- This function only supports recording in File Mode and Buffer Mode.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmRecToFile, SsmRecToFileA, SsmRecToFileB, SsmRecToFileEx, SsmRecToMem

2.10.2.4 SsmCheckRecord

Queries the completion status of the recording task.

Format:

int SsmCheckRecord (int ch)

Parameter Description:

ch	Channel Number

Return Value:

-1	Call failed
0	Recording in process
1	Recording is terminated by the application
2	Recording is terminated upon detection of DTMF characters
3	Recording is terminated upon detection of remote hangup
4	Recording finished in a normal way
5	File Mode recording is suspended
6	File Mode recording is terminated due to the failure of writing data into file
7	RTP timeout

Function Description:

Queries the completion status of the recording task.

Note:

- This function is applicable to recording tasks initiated by any modes including SsmRecToFile, SsmRecToFileA, SsmRecToFileB, SsmRecToFileEx, SsmRecToMem and SsmRecordMemBlock.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmRecToFile, SsmRecToFileA, SsmRecToFileB, SsmRecToFileEx, SsmRecToMem, SsmRecordMemBlock

2.10.3 Functions for Recording in File Mode

2.10.3.1 SsmRecToFile

Refer to SpyRecToFile

2.10.3.2 SsmRecToFileA

Refer to SpyRecToFile

2.10.3.3 SsmRecToFileB

Refer to SpyRecToFile

2.10.3.4 SsmRecToFileEx

Refer to SpyRecToFile

2.10.3.5 SpyRecToFileA

Refer to SpyRecToFile

2.10.3.6 SpyRecToFileB

Refer to SpyRecToFile

2.10.3.7 SpyRecToFile

Starts the recording in File Mode on a designated channel.

In addition to having all the features of the function SsmRecToFile, SsmRecToFileA and SsmRecToFileB also can obtain the recording data stream saved in the internal buffer area of the driver via a callback function.

In addition to having all the features of the function SsmRecToile, SsmRecToFileEx also provides the capability of using the Barge in signal to trigger the driver to write the recording data into the file.

SpyRecToFile, SpyRecToFileA and SpyRecToFileB initiate the File Mode recording task based on the circuit number. These functions are only applicable to DTP Series boards.

Compared with SsmRecToFile, SsmRecToFileA, SsmRecToFileB and SsmRecToFileEx, the names of the recording files SsmRecToFileW, SsmRecToFileAW, SsmRecToFileBW and SsmRecToFileExW are UNICODE formatted.

Format:

int **SsmRecToFile**(int ch, LPSTR pszFileName, int nFormat, DWORD dwStartPos, DWORD dwBytes, DWORD dwTime, int nMask)
int **SsmRecToFileW**(int ch, LPCWSTR lpFileName, int nFormat, DWORD dwStartPos, DWORD dwBytes, DWORD dwTime, int nMask)
int **SsmRecToFileA**(int ch, LPSTR pszFileName, int nFormat, DWORD dwStartPos, DWORD dwBytes, DWORD dwTime, int nMask, LPRECTOMEM pfnCallbackA)
int **SsmRecToFileA**W(int ch, LPCWSTR lpFileName, int nFormat, DWORD dwStartPos, DWORD dwBytes, DWORD dwTime, int nMask, LPRECTOMEM pfnCallbackA)
int **SsmRecToFileB**(int ch,LPSTR pszFileName,int nFormat,DWORD dwStartPos,DWORD dwBytes,DWORD dwTime,int nMask,LPRECTOMEMB pfnCallbackB, PVOID pVoid)
int **SsmRecToFileBW**(int ch,LPCWSTR lpFileName,int nFormat,DWORD dwStartPos,DWORD dwBytes,DWORD dwTime,int nMask,LPRECTOMEMB pfnCallbackB, PVOID pVoid)
int **SsmRecToFileEx**(int ch, LPSTR pszFileName, int nFormat, DWORD dwStartPos, DWORD dwBytes, DWORD dwTime, int nMask, BOOL bSaveToFileOnBargin, DWORD dwRollbackTime)
int **SsmRecToFileExW**(int ch, LPCWSTR lpFileName, int nFormat, DWORD dwStartPos, DWORD dwBytes, DWORD dwTime, int nMask, BOOL bSaveToFileOnBargin, DWORD dwRollbackTime)

int **SpyRecToFileA**(int nCic, WORD wDirection,LPSTR pszFileName, int nFormat, DWORD dwStartPos, DWORD dwBytes, DWORD dwTime, int nMask , LPRECTOMEM pfnCallbackA)

int **SpyRecToFileB**(int nCic, WORD wDirection,LPSTR pszFileName, int nFormat, DWORD dwStartPos, DWORD dwBytes, DWORD dwTime, int nMask,LPRECTOMEMB pfnCallbackB, PVOID pVoid)

int **SpyRecToFile**(int nCic, WORD wDirection, LPSTR pszFileName, int nFormat, DWORD dwStartPos, DWORD dwBytes, DWORD dwTime, int nMask)

Parameter Description:

<table>
<tr><td>ch</td><td>Channel Number</td></tr>
<tr><td>pszFileName
lpFileName</td><td>The name of the file storing the voice data(pszFileName is UTF8 formatted; lpFileName is UNICODE formatted)
If the file declared by the parameter pszFileName (lpFileName) doesn't exist, the driver automatically creates a new file based on the following rules:
✧ If pszFileName (lpFileName) has a file extension of '.wav', the driver will consider that the application wants to record a standard WAV file and create a standard WAV file;
✧ If the file extension is not '.wav', the driver will regard the file as a non-header file. All the data in the file are voice data and the file does not have a header.</td></tr>
<tr><td>nFormat</td><td>Encoding format of the voice data, range of value are listed below:
<table>
<tr><th>Value</th><th>CODEC</th><th>SsmRecToFile</th><th>SsmRecToFileA
SsmRecToFileB</th><th>SsmRecToFileEx</th><th>SpyRecToFile</th></tr>
<tr><td>-2</td><td>PCM16</td><td>☆</td><td>☆</td><td>—</td><td>☆</td></tr>
<tr><td>1</td><td>PCM8</td><td>☆</td><td>☆</td><td>—</td><td>☆</td></tr>
<tr><td>6</td><td>A-Law</td><td>√</td><td>√</td><td>√</td><td>√</td></tr>
<tr><td>7</td><td>μ-Law</td><td>√</td><td>√</td><td>√</td><td>√</td></tr>
<tr><td>17</td><td>IMA ADPCM</td><td>√</td><td>√</td><td>√</td><td>√</td></tr>
<tr><td>23</td><td>VOX</td><td>√</td><td>√</td><td>—</td><td>√</td></tr>
<tr><td>49</td><td>GSM</td><td>◇</td><td>◇</td><td>—</td><td>◇</td></tr>
<tr><td>85</td><td>MP3</td><td>◇</td><td>◇</td><td>—</td><td>◇</td></tr>
<tr><td>131</td><td>G.729A</td><td>√</td><td>√</td><td>—</td><td>√</td></tr>
</table>
Legend: ☆ Using software encoder, via the driver's encoding in software
√ Using hardware encoder. The board is required to have its own encoder. For more information, refer to 'SynCTI Supported CODECs' in Chapter 1.
★ Software encoding, via the external ACM program.
— Not supported
◇ Using hardware encoder if the board has one; otherwise, using software encoder via the external ACM program.
The actual encoding format for recording is determined by the following rules:
If the file declared by the parameter pszFileName (lpFileName) exists and it is a standard wav file, the encoding format will be the one designated in its file header and nFormat will be ignored.
If the existing file is not a standard wav file, or it is a newly created file, the encoding format is determined by nFormat.
Note: nFormat = -1 represents the default encoding format which is related to the configuration item DefaultRecordFormat in ShConfig.ini. If the prerecord feature is enabled, nFormat must be the same as the encoding format set in the prerecord feature, otherwise, this function will fail.</td></tr>
<tr><td>dwStartPos</td><td>The starting position where the driver writes the recording data into the file. If the target file has the file header of wav, dwStartPos is counted from the first data byte of the voice data area, and the length of the file header is not included.
✧ For existing files, if dwStartPos is less than or equals to the length of the voice data in the file, the data is written into the file starting from the position designated by this parameter; If dwStartPos is larger than the length of the voice data in the file, the recording data is written into the file starting from the end of the file.</td></tr>
</table>

	✧ For newly created files, this parameter will be ignored. If the file is a standard wav file, the recording data is written immediately following the file header; If not, the recording data is written into the file starting from the file header
dwBytes	Sets the length (bytes) of the voice data to be recorded. Once the byte length of the recorded data exceeds the set value of this parameter, the recording will terminate. Therefore, set this parameter with a large value (e.g. 0xffffffff) to ensure a continuous recording. Whether this parameter is valid or not is determined by nMask. **Note: If frame-structured encoding format (e.g. IMA ADPCM, GSM, MP3, G.729A etc.) is used, and the value of dwBytes is not integral times of the frame length, the driver will automatically adjust the value of dwBytes to be integral times of the frame length to ensure the integrity of the frame structure**
dwTime	Sets the total time length (ms) of the voice data to be recorded. If the time length of the recorded data exceeds the set value in this parameter, the recording will terminate. Set this parameter to be an extremely large value (e.g. 0xffffffff) can enable continuous recording. Whether this parameter is valid or not is determined by nMask. **Note: If frame-structured encoding format (e.g. IMA ADPCM, GSM, MP3, G.729A etc.) is used, and the value of dwTime is not integral times of the frame length, the driver will automatically adjust the value of dwTime to be integral times of the frame length to ensure the integrity of the frame structure**
nMask	Sets the way to terminate the recording. Range of value: 0: use the parameter dwBytes as the condition to stop the recording 1: use the parameter dwTime as the condition to stop the recording
pfnCallbackA pfnCallbackB	Pointer of callback function. If pfnCallbackA or pfnCallbackB is NULL, the feature of SsmRecToFileA or SsmRecToFileB is completely the same as the feature of SsmRecToFile, and the feature of SpyRecToFileA or SpyRecToFileB is completely the same as the feature of SpyRecToFile. If pfnCallbackA and pfnCallbackB are not NULL, when the recording data accumulated in the driver reaches 16384 bytes and are all written into the file at a time, the callback function in this parameter will be called to pass the voice data stream to the application simultaneously. The function SsmSetFlag (with the parameter F_RECTOFILEA_CALLBACKTIME) can be used to set the time interval of callback. The prototype of the callback function pfnCallbackA: void CALLBACK RecFileCallback(int **ch**, LPBYTE **lpData**, DWORD **dwDataLen**); The prototype of the callback function pfnCallbackB: void CALLBACK RecFileCallback (int **ch**, LPBYTE **lpData**, DWORD **dwDataLen**, PVOID **pVoid**); in the above prototypes, **ch**: Channel Number **lpData**: The address of the recording data buffer in the driver **dwDataLen**: The length (bytes) of the recording data **pVoid**: The pointer passed by the application upon invoking the function SsmRecToFileB or SpyRecToFileB
pVoid	Pointer of PVOID type. It is used by the application to pass the object pointer directly to the callback function pfnCallbackB.
bSaveToFileOnBargin	Whether to enable the feature of saving the recording data to the file upon the detection of Barge in. Range of value: TRUE: enables this feature FALSE: disables this feature If this parameter is set False, the parameter dwRollbackTime will be ignored. The features of the function SsmRecToFileEx and SsmRecToFile are completely the same
dwRollbackTime	The time length (ms) of the voice data to be written into the file before the generation of the Barge-in signal, with the maximum value of 2000ms. This parameter is valid only when the parameter bSaveToFileOnBargin is set TRUE. Because it takes certain time for Barge-in Detector to detect Barge-in, in order to ensure the integrity of the recorded voice data, this parameter needs to be set appropriately. It's recommended that this parameter should be slightly larger than the

	value set by the function SsmSetIsBargeInDtrmTime or the configuration item BargeInDtrmTime
nCic	The logical number of SpyCic. For more information, refer to 'DTP Series' in chapter 1
wDirection	Sets the recording signal source, range of value: 0: Only record the voice of the calling party. The driver automatically invokes the function SsmRecToFile on the calling party's channel bounded with nCic. Hence the related output messages will be returned on the calling party's channel; 1: Only record the voice of the called party. The driver automatically invokes the function SsmRecToFile on the called party's channel bounded with nCic. Hence the related output messages will be returned on the called party's channel; 2: Mixedly record both the calling and called parties. The driver automatically invokes the function SsmRecToFile on the called party's channel bounded with nCic. Hence the related output messages will be returned on the called party's channel

Return Value:

-1	Call failed, the failure reason can be obtained by the function SsmGetLastErrMsg
0	Successful

Function Description:

Starts the recording in File Mode on a designated channel.

In addition to having all the features of the function SsmRecToFile, SsmRecToFileA and SsmRecToFileB also can open the recording data stream saved in the internal buffer area of the driver to the application via a callback function. If the parameter pfnCallbackA (or pfnCallbackB) is NULL, the feature of the function SsmRecToFileA (or SsmRecToFileB) is completely the same as that of the function SsmRecToFile.

In addition to having all the features of the function SpyRecToFile, SpyRecToFileA and SpyRecToFileB also can open the recording data stream saved in the internal buffer area of the driver to the application via a callback function. If the parameter pfnCallbackA (or pfnCallbackB) is NULL, the feature of the function SpyRecToFileA (or SpyRecToFileB) is completely the same as that of the function SpyRecToFile.

After the function SsmRecToFileEx initiates the file recording, the driver only buffers the recording data in the internal recording buffer area instead of writing the data into the file. Only when Barge-in Detector detects Barge-in, will the driver start to write the recoding data saved in the internal buffer into the file. If there is no Barge-in detected until the current recording stops, no data will be written into the recording file.

After the application invokes the File Mode recording function, the driver will start a task of File Mode recording. The task will terminate if one of the following events happens:

- ✧ The file recording stops in a normal way, i.e. the length of the voice data recorded by the driver reaches the set value of dwBytes or dwTime;
- ✧ The driver fails to write the voice data into the file;
- ✧ The application calls SsmStopRecToFile;
- ✧ The driver detects DTMF or remote end hangup, for more information, refer to Setting Termination Condition in Chapter 1.

When the file recording is terminated, the driver will automatically close the file and throws out the event E_PROC_RecordEnd to the application. If the tail-truncation feature is enabled, the driver will automatically truncate a section of data at the file tail. For more information about the tail-truncation feature, refer to Setting Tail-Truncation Feature in Chapter 1.

During the execution of the File Mode recording task, the driver throws out the event of E_PROC_RecordFile to the application every 1000ms by default. Whether to throw out the event and the condition for throwing out the event

can be set by the function SsmSetEvent (with the parameter E_PROC_RecordFile). The function SsmChkRecToFile or SsmCheckRecord can also be used to query the completion status of the File Mode recording.

During the execution of the File Mode recording task, the application can:

- ✧ Invoke the function SsmPauseRecToFile to pause the File Mode recording;
- ✧ Invoke the function SsmStopRecToFile to stop the File Mode recording;
- ✧ Invoke the function SsmChkRecToFile or SsmCheckRecord to query the completion status of the file recording;
- ✧ Invoke the function SsmGetRecTime to obtain the duration of the File Mode recording;
- ✧ Invoke the function SsmGetDataBytesToRecord to obtain the unfinished byte length of the data to be recorded.

Note:

- For more information about choosing the recording signal source and setting the volume, refer to 'Setting Recording Signal Source' in Chapter 1;
- If the prerecord feature is enabled, the driver will first write the prerecord data saved in the internal buffer area into the file according to the prerecord parameter settings, and then starts normal file recording and writes the subsequent data into the file. For more information about the prerecord feature, refer to 'Setting Prerecord Feature' in Chapter 1;
- If the parameter nFormat is GSM or MP3, the configuration item GsmCodecEnable must be set with a value except 0 and the corresponding drivers of CODEC (ACM) must be installed in the Windows system;
- The function SpyRecToFile initiates the File Mode recording based on the circuit number and is only applicable to DTP Series boards. The events it throws out to the application are based on the calling or called party bound with nCic during the call progress. The function SpyGetCallInCh and SpyGetCallOutCh are respectively used to obtain the channel numbers of the calling and called parties of the current call. The function SpyStopRecToFile is used to stop the File Mode recording initiated by the function SpyRecToFile;
- Due to historical reasons, the coding value 65411 which ever represented G.729A is still valid in early versions and backward compatible. For G.729A recording in new versions, we suggest you use the coding value 131;
- This function is also supported in SynIPRecorder. The successful return of the function can only demonstrate that it has sent the command of starting recording to file to the Slaver. As to whether the command is successfully processed in the Slaver, check the value of dwParam in the event E_IPR_START_REC_CB.

Related Information:

Driver version	SsmRecToFileB requires SynCTI Ver. 4.7.2.0 or above; SpyRecToFileA and SpyRecToFileB require SynCTI Ver.5.3.1.3 or above; Other functions require SynCTI Ver. 2.0 or above.
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmStopRecToFile, SsmChkRecToFile, SsmGetRecTime, SsmPauseRecToFile, SsmRestartRecToFile, SsmGetDataBytesToRecord, SsmSetDTMFStopRecCharSet, SsmSetHangupStopRecFlag, SsmSetRecVolume, SsmSetRecMixer, SsmSetRecBack, SsmSetRecAGC

2.10.3.8 SsmStopRecToFile

Refer to SpyStopRecToFile

2.10.3.9 SpyStopRecToFile

Stops recording in File Mode.

Format:

int SsmStopRecToFile(int ch)

int SpyStopRecToFile(int nCic)

Parameter Description:

ch	Channel Number
nCic	Serial number of the monitored circuit

Return Value:

-1	Call failed, the failure reason can be obtained by the function SsmGetLastErrMsg
0	Successful

Function Description:

The function SsmStopRecToFile terminates the file recording initiated by SsmRecToFile, SsmRecToFileA, SsmRecToFileB or SsmRecToFileEx. The function SpyStopRecToFile stops the recording initiated by the function call of SpyRecToFile, SpyRecToFileA or SpyRecToFileB.

Note:

- After invoking this function, the driver will also throw out the event of E_PROC_RecordEnd;
- This function is also supported in SynIPRecorder. The successful return of the function can only demonstrate that it has sent the command of stopping recording to file to the Slaver. As to whether the command is successfully processed in the Slaver, check the value of dwParam in the event E_IPR_STOP_REC_CB.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function:

SsmRecToFile, SsmRecToFileA, SsmRecToFileB, SsmRecToFileEx, SsmSetDTMFStopRecCharSet, SsmSetHangupStopRecFlag, SsmSetRecVolume, SsmSetRecMixer, SsmSetRecBack

2.10.3.10 SsmPauseRecToFile

Suspends the File Mode recording.

Format:

int SsmPauseRecToFile(int ch)

Parameter Description:

ch	Channel Number

Return Value:

-1	Call failed, the failure reason can be obtained by the function SsmGetLastErrMsg
0	Successful

Function Description:

Suspends the File Mode recording initiated by the function SsmRecToFile, SsmRecToFileA, SsmRecToFileB or SsmRecToFileEx. Once the recording is suspended, it can only be restarted by the function SsmRestartRecToFile or terminated by the function SsmStopRecToFile.

Note:

- This function only supports the recording tasks initiated by SsmRecToFile, SsmRecToFileA, SsmRecToFileB, or SsmRecToFileEx;
- This function is also supported in SynIPRecorder. The successful return of the function can only demonstrate that it has sent the command of pausing recording to file to the Slaver. As to whether the command is successfully processed in the Slaver, check the value of dwParam in the event E_IPR_PAUSE_REC_CB.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmRestartRecToFile, SsmStopRecToFile

2.10.3.11 SsmRestartRecToFile

Restarts the suspended File Mode recording.

Format:

int SsmRestartRecToFile(int ch)

Parameter Description:

ch	Channel Number

Return Value:

-1	Call failed, the failure reason can be obtained by the function SsmGetLastErrMsg
0	Successful

Function Description:

Restarts the File Mode recording suspended by the function SsmPauseRecToFile.

Note:

- This function is also supported in SynIPRecorder. The successful return of the function can only demonstrate that it has sent the command of restarting recording to file to the Slaver. As to whether the command is successfully processed in the Slaver, check the value of dwParam in the event E_IPR_RESTART_REC_CB.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmPauseRecToFile

2.10.3.12 SsmChkRecToFile

Queries the completion status of the File Mode recording task.

Format:

int SsmChkRecToFile(int ch)

Parameter Description:

ch	Channel Number

Return Value:

-1	Call failed, the failure reason can be obtained by the function SsmGetLastErrMsg
0	The file recording stops normally
1	The file recording is under progress
2	The file recording is suspended

Function Description:

Queries the completion status of the File Mode recording task.

Note:

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmRecToFile, SsmRecToFileA, SsmRecToFileB, SsmRecToFileEx

2.10.3.13 SsmGetDataBytesToRecord

Obtains the number of unrecorded data bytes in the File Mode recording task, or obtains the number of recorded data bytes in the Pingpong Buffer Mode recording task.

Format:

int SsmGetDataBytesToRecord(int ch)

Parameter Description:

ch	Channel Number

Return Value:

-1	Call failed
≥0	Indicates the number of unrecorded data bytes in the File Mode recording task, or the number of recorded data bytes in the Pingpong Buffer Mode recording task

Function Description:

Obtains the number of unrecorded data bytes in the File Mode recording task, or obtains the number of recorded data bytes in the Pingpong Buffer Mode recording task.

Note:

- This function is not applicable to the GSM formatted Pingpong Buffer Mode recording task.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmGetRecTime

2.10.3.14 SsmRecStereoToFile

Starts stereophonic recording task in File Mode.

Format:

int WINAPI SsmRecStereoToFile(LPSTR pszFileName, DWORD ch1, DWORD ch2, int nFormat)

Parameter Description:

pszFileName	The name of the file storing the stereophonic recording data. It must have the file extension of '.wav'.
ch1	First channel
ch2	Second channel
nFormat	Encoding format of the stereophonic recording data, only A-Law and μ-Law are supported. The value 6 indicates A-Law and 7 indicates μ-Law.

Return Value:

-1	Call failed, the failure reason can be obtained by the function SsmGetLastErrMsg
0	Successful

Function Description:

Starts stereophonic recording task in File Mode.

Note:

Related Information:

Driver version	SynCTI Ver. 5.3.2.1 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmStopRecStereoToFile, SsmChkStereoToFile

2.10.3.15 SsmStopRecStereoToFile

Stops stereophonic recording task.

Format:

int WINAPI SsmStopRecStereoToFile(DWORD ch1, DWORD ch2)

Parameter Description:

ch1	First channel
ch2	Second channel

Return Value:

-1	Call failed, the failure reason can be obtained by the function SsmGetLastErrMsg
0	Successful

Function Description:

Stops stereophonic recording task.

Note:

Related Information:

Driver version	SynCTI Ver. 5.3.2.1 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmRecStereoToFile, SsmChkStereoToFile

2.10.3.16 SsmChkStereoToFile

Queries the completion status of the stereophonic recording task.

Format:

int WINAPI SsmChkStereoToFile(DWORD ch1, DWORD ch2)

Parameter Description:

ch1	First channel
ch2	Second channel

Return Value:

-1	Call failed, the failure reason can be obtained by the function SsmGetLastErrMsg
0	The stereophonic recording task stops normally
1	The stereophonic recording task is in progress

Function Description:

Queries the completion status of the stereophonic recording task.

Note:

Related Information:

Driver version	SynCTI Ver. 5.3.2.1 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmRecStereoToFile, SsmStopRecStereoToFile

2.10.3.17 SsmPauseRecToFileEx

Pauses the File Mode recording, only applicable to the IPR channels.

Format:

int SsmPauseRecToFileEx(int ch, BOOL bInsertSilence)

Parameter Description:

ch	Channel Number
bInsertSilence	Whether to fill in the time when the recording is being paused with the silence data.

Return Value:

-1	Call failed, the failure reason can be obtained by the function SsmGetLastErrMsg
0	Successful

Parameter Description:

Pauses the File Mode recording initiated by the function SsmRecToFile, SsmRecToFileA, or SsmRecToFileB. Once the recording is suspended, it can only be restarted by the function SsmRestartRecToFile or terminated by the function SsmStopRecToFile.

This function is only applicable to the IPR channels. The interaction of RTP streams will be stopped when the VoIP call goes into a hold state. To record a complete call process, the function SsmPauseRecToFileEx will be invoked to avoid the recording discontinuity caused by the RTP timeout or to avoid the DTMF data infiltrating into the recording file. And the parameter 'bInsertSilence' can be used to set a time for pausing the recording or to decide whether to fill in the time when the recording is being paused with the silence data, meeting every individual requirement.

Note:

- This function only supports the recording tasks initiated by SsmRecToFile, SsmRecToFileA, or SsmRecToFileB;
- This function is only supported in SynIPRecorder. The successful return of this function only indicates that it has sent the command of pausing the File Mode recording to the Slaver. To learn whether the command is successfully processed in the Slaver, check the value of dwParam in the event E_IPR_PAUSE_REC_CB.

Related Information:

Driver version	SynCTI Ver. 5.4.0.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmRestartRecToFile, SsmStopRecToFile

2.10.3.18 SsmRecToFileW

Refer to SpyRecToFile

2.10.3.19 SsmRecToFileAW

Refer to SpyRecToFile

2.10.3.20 SsmRecToFileBW

Refer to SpyRecToFile

2.10.3.21 SsmRecToFileExW

Refer to SpyRecToFile

2.10.4 Functions for Recording in Buffer Mode

2.10.4.1 SsmRecToMem

Refer to SpyRecToMem.

2.10.4.2 SpyRecToMem

Starts the recording in Buffer Mode.

Format:

int SsmRecToMem(int ch, int nFormat, LPBYTE pBuf, DWORD dwBufSize, DWORD dwStartOffset)

int SpyRecToMem(int nCic, WORD wDirection, int nFormat, LPBYTE pBuf, DWORD dwBufSize, DWORD dwStartOffset)

Parameter Description:

<table>
<tr><td>ch</td><td>Channel Number</td></tr>
<tr><td>nFormat</td><td>Encoding format of the recording data. Range of value:
<table>
<tr><th>Value</th><th>Encoding Format</th><th>Remarks</th></tr>
<tr><td>1</td><td>PCM8</td><td></td></tr>
<tr><td>-2</td><td>PCM16</td><td></td></tr>
<tr><td>6</td><td>A-Law</td><td></td></tr>
<tr><td>7</td><td>μ-Law</td><td></td></tr>
<tr><td>17</td><td>IMA ADPCM</td><td>The board is required to have the hardware encoding</td></tr>
</table></td></tr>
</table>

			capability or to be SynIPRecorder
	23	VOX	The board is required to have the hardware encoding capability
	49	GSM	
	85	MP3	The board is required to have the hardware encoding capability
	131	G.729A	The board is required to have the hardware encoding capability
	For more information about whether a channel supports the hardware encoder, refer to SynCTI Supported CODECs in Chapter 1		
pBuf	Pointer pointing to the first address of the buffer. The storage space of buffer area is allocated by the application.		
dwBufSize	The size (bytes) of the recording buffer area		
dwStartOffset	The offset of the starting position of recording in the recording buffer area. Range of value: 0~dwBufSize-1		
nCic	The logic number of SpyCic. For more information, refer to 'DTP Series' in Chapter 1		
wDirection	Sets the recording signal source, range of value: 0: Only record the calling party. The driver automatically invokes the function SsmRecToMem on the calling party's channel bounded with nCic. Hence the related output messages will be returned on the calling party's channel; 1: Only record the called party. The driver automatically invokes the function SsmRecToMem on the called party's channel bounded with nCic. Hence the related output messages will be returned on the called party's channel; 2: Mixedly record both the calling and called parties. The driver automatically invokes the function SsmRecToMem on the called party's channel bounded with nCic. The related output messages will be returned on the calling party's channel		

Return Value:

-1	Call failed, the failure reason can be obtained by the function SsmGetLastErrMsg
0	Successful

Function Description:

Starts the Buffer Mode recording task. SpyRecToMem initiates the Buffer Mode recording task based on the circuit number and it is only applicable to DTP Series boards.

After calling this function, the driver will start the task of Buffer Mode recording and writes the recording data into pBuf starting from the offset designated by the parameter dwStartOffset. Meanwhile, the driver maintains a recording pointer (offset) with an initial value equal to dwStartOffset, which is used to write data into the submitted buffer area. Once the buffer is full, the driver will automatically refill the buffer from the head.

The task will terminate if one of the following events happens:

- ✧ The application calls SsmStopRecToMem;
- ✧ The driver detects DTMF or remote end hangup. For more detailed information, refer to Setting Termination Condition in Chapter 1.

The function SsmCheckRecord can be used to query the completion status of the Buffer Mode recording.

During the execution of the Buffer Mode recording, once the preset condition is satisfied, the driver throws out the event of E_PROC_RecordMem to the application. The output conditions of the event E_PROC_RecordMem can be set by the function SsmSetEvent (with the parameter E_PROC_RecordMem). Below are three optional conditions:

- ✧ The driver finishes recording the voice data with designated time length;
- ✧ The recording pointer in the driver gets across the dwBufSize /2 position of the buffer area;
- ✧ The recording pointer in the driver reaches the end of the buffer.

The function SsmStopRecToMem can be used to stop the Buffer Mode recording. The function SsmGetRecOffset can be used to obtain the recording pointer in the driver.

Note:

- For more information about setting the volume and selecting the recording signal sources, refer to Setting Recording Signal Source in Chapter 1.
- The function SpyRecToMem initiates the Buffer Mode recording task based on the circuit number and is only applicable to DTP Series boards. It is a function based on the circuit number. The message it throws out to the application during the call is based on the calling or called party's channel that is bound with nCic. The functions SpyGetCallInCh and SpyGetCallOutCh are used respectively to obtain the channel numbers of the calling and called parties of the current call. The function SpyStopRecToMem is used to terminate the Buffer Mode recording initiated by the function SpyRecToMem.
- Due to historical reasons, the coding value 65411 which ever represented G.729A is still valid in early versions and backward compatible. For G.729A recording in new versions, we suggest you use the coding value 131.
- This function is also supported in SynIPRecorder. The successful return of the function can only demonstrate that it has sent the command of starting recording to memory to the Slaver. As to whether the command is successfully processed in the Slaver, check the value of dwParam in the event E_IPR_START_REC_CB.

Related Information:

Driver version	SsmRecToMem requires SynCTI Ver. 2.0 or above; SpyRecToMem requires SynCTI Ver.5.3.1.3 or above.
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmStopRecToMem, SsmGetRecOffset, SsmSetDTMFStopRecCharSet, SsmSetHangupStopRecFlag, SsmSetRecVolume, SsmSetRecMixer, SsmSetRecBack

2.10.4.3 SsmStopRecToMem

Refer to SpyStopRecToMem.

2.10.4.4 SpyStopRecToMem

Stops recording in Buffer Mode.

Format:

int SsmStopRecToMem(int ch)

int SpyStopRecToMem(int nCic)

Parameter Description:

ch	Channel Number
nCic	The logic number of SpyCic. For more information, refer to 'DTP Series' in Chapter 1

Return Value:

-1	Call failed, the failure reason can be obtained by the function SsmGetLastErrMsg
0	Successful

Function Description:

Stops recording in Buffer Mode. The function SpyStopRecToMem is used to terminate the Buffer Mode recording initiated by the function SpyRecToMem.

Note:

- When the recording is terminated, the driver will throw out the event of E_PROC_RecordEnd.
- This function is also supported in SynIPRecorder. The successful return of the function can only demonstrate that it has sent the command of stopping recording to memory to the Slaver. As to whether the command is successfully processed in the Slaver, check the value of dwParam in the event E_IPR_STOP_REC_CB.

Related Information:

Driver version	SsmStopRecToMem requires SynCTI Ver. 2.0 or above; SpyStopRecToMem requires SynCTI Ver.5.3.1.3 or above.
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmRecToMem

2.10.4.5 SsmGetRecOffset

Obtains the recording pointer in the driver during Buffer Mode recording.

Format:

int SsmGetRecOffset(int ch)

Parameter Description:

ch	Channel Number

Return Value:

-1	Call failed
≥0	Recording pointer in the driver, its value is the offset (bytes) to the first address of the buffer area submitted by the function SsmRecToMem

Function Description:

Obtains the recording pointer in the driver during Buffer Mode recording.

Note:

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmRecToMem

2.10.5 Functions for Recording in Pingpong Buffer Mode (CTI Series)

2.10.5.1 SsmRecordMemBlock

Submits a buffer to the driver and initiates the recording in Pingpong Buffer Mode.

Format:

int SsmRecordMemBlock(int ch, int nFormat, LPBYTE pBuf, DWORD dwBufSize, RECORDMEMBLOCKHANDLER pfnCallback, PVOID pV)

Parameter Description:

ch	Channel Number

<table>
<tr><td>nFormat</td><td>Encoding format of the recording data. Range of value:
<table>
<tr><th>Value</th><th>Encoding Format</th><th>Remarks</th></tr>
<tr><td>1</td><td>PCM8</td><td></td></tr>
<tr><td>-2</td><td>PCM16</td><td></td></tr>
<tr><td>6</td><td>A-Law</td><td></td></tr>
<tr><td>7</td><td>μ-Law</td><td></td></tr>
<tr><td>17</td><td>IMA ADPCM</td><td>The board is required to have the hardware encoding capability</td></tr>
<tr><td>23</td><td>VOX</td><td>The board is required to have the hardware encoding capability</td></tr>
<tr><td>49</td><td>GSM</td><td></td></tr>
<tr><td>85</td><td>MP3</td><td>The board is required to have the hardware encoding capability</td></tr>
<tr><td>131</td><td>G.729A</td><td>The board is required to have the hardware encoding capability</td></tr>
</table>
For more information about whether a channel supports the hardware encoder, refer to SynCTI Supported CODECs in Chapter 1</td></tr>
<tr><td>pBuf</td><td>Pointer pointing to the recording buffer</td></tr>
<tr><td>dwBufSize</td><td>The recording buffer size (bytes). The minimum value of dwBufSize is determined according to the following rules:
If the configuration item RecordAndPlayUseAsIP is set 0 (default), for A-law, μ-law, PCM8, PCM16, G729A, mp3, GC8, VOX and IMA ADPCM, dwBufSize should be no less than 768 bytes; for GSM encoding in hardware, dwBufSize should be no less than 260 bytes; for GSM encoding in software, dwBufSize should be no less than 2080 bytes.
If the configuration item RecordAndPlayUseAsIP is set 1, for A-law, μ-law, PCM8 and PCM16, dwBufSize should be no less than 192 bytes (recommended value: under Windows operating system, it should be no less than 192 bytes; under Linux, it should be no less than 384 bytes); for IMA ADPCM, VOX, G729A, mp3 and GC8, dwBufSize should be no less than 768 bytes; for GSM encoding in hardware, dwBufSize should be no less than 260 bytes; for GSM encoding in software, dwBufSize should be no less than 2080 bytes.</td></tr>
<tr><td>pfnCallback</td><td>Pointer of the callback function. Once the recording task is finished or terminated, the driver will automatically invoke the callback function set by this parameter. NULL means not to set the callback function. The prototype of the callback function is declared in shpa3api.h:
typedef BOOL (WINAPI * RECORDMEMBLOCKHANDLER)
(int ch, int nEndReason, PUCHAR pucBuf, DWORD dwStopOffset, PVOID pVoid);
See below for details.
ch: Channel Number;
nEndReason: The cause of termination, range of value:
1: The recording is terminated by the application.
2: The recording is terminated upon detection of DTMF.
3: The recording is terminated upon detection of remote end hangup.
4: The buffer recording is completed
pucBuf: The pointer pointing to the buffer where the recording task is finished.
dwStopOffset: The offset (bytes) of the driver's recording pointer in the buffer when the recording task is finished. If the return value of this parameter is 0xffffffff, it is invalid.
pVoid: The pointer passed by the application upon invoking this function.</td></tr>
<tr><td>pVoid</td><td>The pointer of PVOID type. It is commonly used by the application for transmitting its data structure to the callback function. This pointer is passed to the callback function directly</td></tr>
</table>

Return Value:

-1	Call failed, the failure reason can be obtained by the function SsmGetLastErrMsg
0	Successful

Function Description:

Submits a buffer to the driver and initiates the recording in Pingpong Buffer Mode. For more information about recording in Pingpng Buffer Mode, refer to Pingpong Buffer Mode in Chapter 1.

The task will terminate if one of the following events happens:

- ✧ The driver has finished recording in all the submitted buffer areas;
- ✧ The application calls the function SsmStopRecordMemBlock;
- ✧ The driver detects DTMF or remote end hangup. For more information, refer to Setting Termination Condition in Chapter 1.

The function SsmCheckRecord can be used to query the completion status of the recording in Pingpong Buffer Mode**.**

Note:

- For more information about setting the volume and selecting the recording signal sources, refer to Setting Recording Signal Source in Chapter 1.
- The application can invoke the function SsmStopRecordMemBlock at any moment to terminate the recording in Pingpong Buffer Mode. SsmStopRecordMemBlock cannot be invoked in the callback function.
- In the callback function, the time for the application to execute the submitted code should be minimized and not exceed the interruption time of the driver (8ms). Otherwise it may cause driver error.
- When the task of recording in Pingpong Buffer Mode stops, the driver will throw out the event of E_PROC_RecordEnd to the application.
- It is not allowed to submit more than two buffers to the driver at the same time.
- It is not allowed to use the timer to invoke the function SsmRecordMemBlock.
- Due to historical reasons, the coding value 65411 which ever represented G.729A is still valid in early versions and backward compatible. For G.729A recording in new versions, we suggest you use the coding value 131.

Sample Code:

```
......//initialize the buffers szRecBuf1, szRecBuf2
BOOL RecStart(int nCh)
{
if(SsmRecordMemBlock(nCh, 6, szRecBuf1, dwBufSize, RecordMemBlockHandler, NULL) != 0)//submit the buffer szRecBuf1 to the driver and start the recording in Pingpong Buffer Mode
{
SsmGetLastErrMsg(szErrMsg);
cout<<szErrMsg<<endl;
return FALSE;
}
if(SsmRecordMemBlock(nCh, 6, szRecBuf2, dwBufSize, RecordMemBlockHandler, NULL) != 0)//submit the buffer szRecBuf2 to the driver
{
SsmGetLastErrMsg(szErrMsg);
cout<<szErrMsg<<endl;
return FALSE;
}
return TRUE;
}
BOOL RecordMemBlockHandler(int ch, int nEndReason, PUCHAR pucBuf, DWORD dwStopOffset, PVOID
```

```
pV)
{
        int nResult = 0;
        if(pucBuf == szRecBuf1)
{
                .....//save the data in szRecBuf1
nResult = SsmRecordMemBlock(ch, 6, szRecBuf1, dwBufSize, RecordMemBlockHandler, pV);)//submit the buffer szRecBuf1 to the driver
}
else if(pucBuf == szRecBuf2)
{
                .....//save the data in szRecBuf2
nResult = SsmRecordMemBlock(ch, 6, szRecBuf2, dwBufSize, RecordMemBlockHandler, pV);// submit the buffer szRecBuf2 to the driver
}
return nResult;
}
```

Related Information:

Driver version	SynCTI Ver. 3.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmStopRecordMemBlock, SsmRecToMem

2.10.5.2 SsmStopRecordMemBlock

Stops recording in Pingpong Buffer Mode.

Format:

int SsmStopRecordMemBlock (int ch)

Parameter Description:

ch	Channel Number

Return Value:

-1	Call failed, the failure reason can be obtained by the function SsmGetLastErrMsg
0	Successful

Function Description:

Stops recording in Pingpong Buffer Mode.

Note:

- This function cannot be invoked in the callback function which is set by the function SsmRecordMemBlock;
- When the recording is terminated, the driver throws out the event E_PROC_RecordEnd.

Related Information:

Driver version	SynCTI Ver. 3.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmRecordMemBlock

2.11 TDM Functions

2.11.1 Functions for Two-way Connection (CTI Series)

2.11.1.1 Establishing Two-way Connection

2.11.1.1.1SsmTalkWith

Refer to SsmTalkWithEx

2.11.1.1.2SsmTalkWithEx

Establishes the two-way connection between two channels. The function SsmTalkWithEx is able to set the volume on the two voice exchanging channels.

Format:

int SsmTalkWith(int ch1,int ch2)

int SsmTalkWithEx(int ch1, int nVlm1, int ch2, int nVlm2)

Parameter Description:

ch1	Channel Number1
ch2	Channel Number2
nVlm1	The volume of ch1, range of value: -7~+6. The negative value means volume decreasing while the positive value means the volume increasing The default value is 0. This parameter multiplied by 3 equals to the dB value.
nVlm2	The volume of ch2, range of value: -7~+6. The negative value means volume decreasing while the positive value means the volume increasing The default value is 0. This parameter multiplied by 3 equals to the dB value.

Return Value:

-1	Call failed, the failure reason can be obtained by the function SsmGetLastErrMsg
0	Successful

Function Description:

Establishes the two-way connection between the designated channels ch1 and ch2, i.e. ch1 can hear the voice from ch2 and ch2 can also hear the voice from ch1. If the incoming call signal from ch1 or ch2 has not been put onto the TDM bus, the driver will automatically arrange a time slot and put the incoming call signal onto this time slot.

Note:

- This function is only applicable to SHT/SHD/SHN Series boards, the parameters nVlm1 and nVlm2 are used to set volume adjusters A4-1…A4-6 in 'Operation Principle of SHT Series' or 'Operation Principle of SHD Series' or 'Operation Principle of SHN Series' in Chapter 1. The gains of A4-1…A4-6 can also be set by the configuration item DefaultSpeakVolume, with the default value of 0DB;

- If the model name of the board doesn't include '-CT', ch1 and ch2 must be on the same board, otherwise, this function will fail;
- Invoke the function SsmStopTalkWith to tear down the two-way connection. Make sure that the operations of connect and disconnect are performed correspondingly in pairs.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmStopTalkWith, SsmListenTo, SsmListenToEx, SsmLinkFrom, SsmLinkFromEx

2.11.1.2 Disconnecting Two-way Connection

2.11.1.2.1SsmStopTalkWith

Tears down the two-way connection between two channels.

Format:

int SsmStopTalkWith(int ch1,int ch2)

Parameter Description:

ch1	Listener/talker channel number
ch2	Talker/listener channel number

Return Value:

-1	Call failed, the failure reason can be obtained by the function SsmGetLastErrMsg
0	Successful

Function Description:

Tears down the two-way connection between ch1 and ch2.

Note:

- This function is only applicable to SHT/SHD/SHN Series boards

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmTalkWith, SsmTalkWithEx

2.11.2 Functions for One-way Connection

2.11.2.1 Establishing One-way Connection

2.11.2.1.1SsmListenTo

Refer to SsmLinkFromAllCh

2.11.2.1.2SsmListenToEx

Refer to SsmLinkFromAllCh

2.11.2.1.3SsmLinkFrom

Refer to SsmLinkFromAllCh

2.11.2.1.4SsmLinkFromEx

Refer to SsmLinkFromAllCh

2.11.2.1.5SsmLinkFromAllCh

Establishes the one-way connection from the talker channel to the listener channel. The functions SsmListenTo and SsmListenToEx are used to establish the one-way connection between two channels, and one channel can listen to only one channel. The functions SsmLinkFrom and SsmLinkFromEx are used to establish the one-way connection between two channels, and one channel can listen to multiple channels. The functions SsmListenToEx and SsmLinkFromEx can also be used to set the volume of the one-way connection. The function SsmLinkFromAllCh can establish the one-way connection from one talker channel to multiple listener channels at one time.

Format:

int SsmListenTo(int nSourceCh, int nListeningCh)
int SsmListenToEx(int nSourceCh, int nVolume, int nListeningCh)
int SsmLinkFrom(int nSourceCh, int nListeningCh)
int SsmLinkFromEx(int nSourceCh, int nVolume, int nListeningCh)
int SsmLinkFromAllCh(int nSourceCh, int nVolume, int* pnListenerTable,int nListenerNum)

Parameter Description:

nSourceCh	Talker channel number
nListeningCh	Listener channel number **Note: If nListeningCh is set to be channel 0 for ATP or DST Series boards, the voice of the talker will be sent to the on-board speaker port**
nVolume	The volume of the talker, range of value: -7~+6. The negative value means volume decreasing while the positive value means the volume increasing The default value is 0. This parameter multiplied by 3 equals to the dB value. nVolume is used to set the volume of the incoming signals entering the off-bus mixer M2 on another channel from the TDM bus. The gains of A4-1…A4-6 can also be set by the configuration item DefaultSpeakVolume, with the default value of 0DB.
pnListenerTable	The pointer pointing to the table of listener channels, the storage space is allocated by the application. The listener channels must be stored consecutively
nListenerNum	The number of listener channels in pnListenerTable

Return Value:

-1	Call failed, the failure reason can be obtained by the function SsmGetLastErrMsg
0	Successful

Function Description:

Establishes the one-way connection from the talker channel to the listener channel, i.e. nListeningCh can hear the voice from nSourceCh , but nSourceCh is not able to hear the voice from nListeningCh.

In addition to having all the features of the function SsmListenTo, the function SsmListenToEx can also set the volume of connection. The functions SsmListenTo and SsmListenToEx can establish the one-way connection which enables a channel to listen to only one channel at one time. To disconnect the connection established by these two functions, invoke the function SsmStopListenTo, and make sure that the operations of connect and disconnect are performed correspondingly in pairs.

In addition to having all the features of the function SsmLinkFrom, the function SsmLinkFromEx can also set the volume of connection. To disconnect the one-way connection established by the function SsmLinkFrom and SsmLinkFromEx, invoke the function SsmStopLinkFrom, and make sure that the operations of connect and disconnect are performed correspondingly in pairs.

SsmLinkFromAllCh can establish the one-way connection with a talker channel and multiple listener channels at one time. The feature implemented by this function is the same as that implemented by invoking the function SsmLinkFromEx continuously on multiple listener channels. To disconnect the connection established by the function SsmLinkFromAllCh, invoke the function SsmUnLinkFromAllCh, and make sure the operations of connect and disconnect are performed correspondingly in pairs.

The functions SsmLinkFrom, SsmLinkFromEx and SsmLinkFromAllCh allow one channel to listen to multiple talker channels. The voices from the talker channels are sent to the conference mixer of the listener channel via the TDM bus, and then the mixed voice is sent as the outgoing call signal to the listener.

Note:

- For SHT/SHD/SHN Series boards, the off-bus mixer M2 has 6 input signal sources, $A_{4\text{-}1}$~$A_{4\text{-}6}$. Therefore, each channel can listen to up to 6 channels at one time, i.e. one channel can invoke the function SsmLinkFrom or SsmLinkFromEx up to 6 times. For more information, refer to 'Operation Principle of SHT Series', 'Operation Principle of SHD Series' or 'Operation Principle of SHN Series' in Chapter 1.

Related Information:

Driver version	SynCTI Ver. 3.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmStopListenTo, SsmStopLinkFrom, SsmUnLinkFromAllCh, SsmTalkWith, SsmTalkWithEx

Sample code:

This sample implements the following features:

- ✓ Channel 0 listens to channel 5;
- ✓ Channel 1 listens to channel 6 (the volume is increased by 6 units);
- ✓ Channel 2 listens to channel 9, channel 10, and channel 11 at the same time;
- ✓ Channel 3 listens to channel 9 (normal volume), channel 10 (the volume is decreased by 1 unit), and channel

11 (the volume is increased by 2 units) at the same time;

✓ The voice from channel 4 needs to be sent to channel 12, 13 and 14 (default volume) at the same time.

Code Implementation:

```
SsmListenTo( 5,0);                  // Establishes connection for the listener channel 0

SsmListenToEx( 6,6,1);              // Establishes connection for the listener channel 1

SsmLinkFrom( 9,2);                  // Establishes connection for the listener channel 2
SsmLinkFrom(10,2);
SsmLinkFrom(11,2);

SsmLinkFromEx( 9, 0,3);             // Establishes connection for the listener channel 3
SsmLinkFromEx(10,-1,3);
SsmLinkFromEx(11, 2,3);

int ListenerChList[3]={12,13,14};   // Establishes the one-way connections from the talker channel 4 to the listener
                                       channel 12,13 and 14

SsmLinkFromAllCh(4,0,ListenerChList,3);

  ......                            // Other operations of the application

SmStopListenTo( 5,0);               // Tears down the connection for the listener channel 0

SmStopListenTo( 6,1);               // Tears down the connection for the listener channel 0

SsmStopLinkFrom( 9,2);              // Tears down the connection for the listener channel 1
SsmStopLinkFrom(10,2);
SsmStopLinkFrom(11,2);

SsmStopLinkFrom( 9,3);              // Tears down the connection for the listener channel 2
SsmStopLinkFrom(10,3);
SsmStopLinkFrom(11,3);

SsmLinkFromAllCh(4,ListenerChList,3);// Tears down the one-way connection from channel 4 to listener channel 12, 13 and 14.
```

2.11.2.2 Disconnecting One-way Connection

2.11.2.2.1SsmStopListenTo

Refer to SsmUnLinkFromAllCh

2.11.2.2.2SsmStopLinkFrom

Refer to SsmUnLinkFromAllCh

2.11.2.2.3SsmUnLinkFromAllCh

Tears down the one-way connection between the talker channel and the listener channel. The function SsmStopListenTo is used tear down the one-way connection established by the function SsmListenTo or SsmListenToEx. The function SsmStopLinkFrom is used to tear down the one-way connection established by SsmLinkFrom or SsmLinkFromEx. The function SsmUnLinkFromAllCh is used to tear down the one-way connection established by SsmLinkFromAllCh.

Format:

int SsmStopListenTo(int nSourceCh,int nListeningCh)

int SsmStopLinkFrom(int nSourceCh,int nListeningCh)

int SsmUnLinkFromAllCh (int nSourceCh, int* pnListenerTable,int nListenerNum)

Parameter Description:

nSourceCh	Talker channel number
nListeningCh	Listener channel number
pnListenerTable	The pointer pointing to the table of listener channels, the storage space is allocated by the application. The listener channels must be stored consecutively
nListenerNum	The number of the listener channels in the table of pnListenerTable

Return Value:

-1	Call failed, the failure reason can be obtained by the function SsmGetLastErrMsg
0	Successful

Function Description:

Tears down the one-way connection between the talker channel and the listener channel.

Note:

- The operations of establishing and disconnecting the one-way connection must be performed correspondingly in pairs.

Related Information:

Driver version	SynCTI Ver. 3.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmListenTo, SsmListenToEx, SsmLinkFrom, SsmLinkFromEx, SsmLinkFromAllCh

2.11.3 Obtaining Bus Information

2.11.3.1 SsmGetChBusInfo

Obtains the relative bus information for Channel ch.

Format:

int WINAPI SsmGetChBusInfo(int ch, PBUS_OP* p)

Parameter Description:

ch	Channel Number
p	Bus structure pointer

Return Value:

-1	No channel is monitored by Channel ch
-2	This function is unsupported by Channel ch
>=0	The number of the channel monitored by Channel ch

Function Description:

Obtains the relative bus information for Channel ch.

Bus Structure

```
{       BOOL        bEnHwOpBus;                 Bus switch
        BOOL        bEnHwOpSetLinkFromVlm;      Volume control switch for bus connection
```

int	nST;	Default bit stream for bus connection
int	nTs;	Default time slot for bus connection
int	nToBusCh;	Total number of time slots used by default for bus connection
int	nPlayST;	Bit stream used by default for the recording module to put voices onto bus
int	nPlayTs;	Time slot connected by default for the recording module to put voices onto bus
int	nPlayToBusCh;	Total number of time slots used by default for the recording module to put voices onto bus
int	nSpeakerVlm;	Volume
int	nTotListener;	Total number of listeners listening to the current channel via bus connection
int	*pnListenerCh;	Number of the channel listening to the current channel via bus connection
int	nFromSpeaker;	Speaker channel number
int	nDefaultSpeakerVlm;	Volume for bus connection
int	nTotChS;	For internal use only
int	* pnChS;	For internal use only
int	nBindCh;	For internal use only
int	nToBusChForVox;}	For internal use only

This bus structure is defined by the driver itself and some of the parameters are senseless to the application layer. Therefore users may not pay much attention to it.

Note:

- This function only supports obtaining information about the bus connection between two channels, such as SsmListenTo, SsmLinkFrom or SsmTalkWith. It can't help obtain information of the bus connection among conference members.

Related Information:

Driver version	SynCTI Ver.3.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function:

2.11.4 Advanced Functions for Bus Operation (CTI Series)

2.11.4.1 Forcibly Disconnecting All Bus Connections

2.11.4.1.1SsmClearChBusLink

Tears down all bus connections on the channel and recovers the channel to the state upon the driver's initialization.

Format:

int SsmClearChBusLink(int nCh)

Parameter Description:

nCh	The monitoring channel number

Return Value:

-1	Call failed, the failure reason can be obtained by the function SsmGetLastErrMsg
0	Successful

Function Description:

Tears down all the TDM bus connections which are established on the channel nCh by invoking any of the functions below:

- SsmListenTo, SsmListenToEx, SsmLinkFrom, SsmLinkFromEx, SsmLinkFromAllCh
- SsmTalkWith, SsmTalkWithEx

Recovers the channel to the state upon the driver's initialization.

Note:

- If the channel has entered a conference room, its TDM bus connections will also be torn down forcibly.

Related Information:

Driver version	SynCTI Ver.4.5.6.2 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function:

2.11.4.2 One-way Connection from Channel to Bus Time Slot

2.11.4.2.1SsmLinkToBus

Changes the time slot which is set to receive the signals from the onto-bus mixer.

Format:

int SsmLinkToBus(int ch, int ts)

Parameter Description:

ch	Channel Number
ts	The number of the available common time slot on the TDM bus, range of value: N~4095, N is twice of the total amount of the channels in the system. For more information about the number of the available common time slot, refer to 'TDM Capability' in Chapter 1

Return Value:

-1	Call failed, the failure reason can be obtained by the function SsmGetLastErrMsg
0	Successful

Function Description:

During system initialization, the driver will put the incoming signal from Channel ch to a time slot (suppose it is ts0) on TDM bus. This function reconnects output signals to the time slot.

Invoke the function SsmUnLinkToBus to tear down the connection.

Note:

- The time slot ts must be an unused common time slot.
- This function must be used with SsmUnLinkToBus, otherwise, it may cause garbled signals.

- This function is a bottom function which is only applicable to the situation where Synway boards are interoperated with third party boards.
- The configuration item EnableCommonTimeSlot can be used to open all the common time slots on the TDM bus.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmUnLinkToBus, SsmLinkFromBus, SsmLinkFromBusEx

2.11.4.2.2SsmUnLinkToBus

Reuses the default time slot to receive the signals from the onto-bus mixer.

Format:

int SsmUnLinkToBus(int ch, int ts)

Parameter Description:

ch	Channel Number
ts	Time slot number

Return Value:

-1	Call failed, the failure reason can be obtained by the function SsmGetLastErrMsg
0	Successful

Function Description:

Tears down the one-way connection from the channel ch to the bus time slot ts, and reuses the default time slot to receive the signals from the onto-bus mixer.

Note:

- This function must be called along with SsmLinkToBus in pairs, otherwise, it may cause garbled signals.
- This function is a bottom function which is only applicable to the situation that Synway's boards interoperate with 3rd party's boards.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmLinkToBus

2.11.4.3 One-way Connection from Bus Time Slot to Channel

2.11.4.3.1SsmLinkFromBus

Refer to SsmLinkFromBusEx

2.11.4.3.2SsmLinkFromBusEx

Establishes the one-way connection from the bus time slot to the listener channel. Apart from having all the features of the function SsmLinkFromBus, SsmLinkFromBusEx can also set the volume.

Format:

int SsmLinkFromBus(int ts, int ch)

int SsmLinkFromBusEx(int ts, int ch, int nVolume)

Parameter Description:

ch	Channel Number
ts	Time slot number
nVolume	The volume of the voice data on the time slot ts entering channel ch, range of value: -6~+6, the negative value means volume decreasing, the positive value means the volume increasing , this parameter multiplied by 3 equals dB value

Return Value:

-1	Call failed, the failure reason can be obtained by the function SsmGetLastErrMsg
0	Successful

Function Description:

Establishes the one-way connection from the bus time slot ts to the listener channel ch.

Note:

- The function SsmLinkFromBus or SsmLinkFromBusEx must be called along with SsmUnLinkFromBus in pairs, otherwise, it may cause garbled signals.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmLinkToBus, SsmUnLinkFromBus

2.11.4.3.3SsmUnLinkFromBus

Tears down the one-way connection from the bus time slot to the listener channel.

Format:

int SsmUnLinkFromBus(int ts, int ch)

Parameter Description:

ch	Channel Number
ts	Time slot number

Return Value:

-1	Call failed, the failure reason can be obtained by the function SsmGetLastErrMsg

0	Successful

Function Description:

Tears dwon the one-way connection from the time slot ts to the listener channel ch.

Note:

- The function SsmLinkFromBus or SsmLinkFromBusEx must be called along with SsmUnLinkFromBus in pairs, otherwise, it may cause garbled signals.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmLinkFromBus, SsmLinkFromBusEx

2.11.4.3.4SsmListenToPlay

Enables a channel to monitor another recording channel.

Format:

int WINAPI SsmListenToPlay(int ch1, int vlm, int ch2)

Parameter Description:

ch1	The monitoring channel number
ch2	The monitored channel number. The channel is required to be a recording channel
vlm	The volume of voices played on the monitored channel

Return Value:

-1	Call failed, the failure reason can be obtained by the function SsmGetLastErrMsg
0	Successful

Function Description:

Enable Channel ch1 to monitor the recording channel Channel ch2.

Note:

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmUnListenToPlay

2.11.4.3.5SsmUnListenToPlay

Stops the monitoring of a recording channel.

Format:

int WINAPI SsmUnListenToPlay(int ch1, int ch2)

Parameter Description:

ch1	The monitoring channel number

ch2	The monitored channel number, and the channel is required to be a recording channel

Return Value:

-1	Call failed, the failure reason can be obtained by the function SsmGetLastErrMsg
0	Successful

Function Description:

Stops Channel ch1 from monitoring the recording channel Channel ch2.

Note:

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmListenToPlay

2.12 On-board Audio Power Amplifier Functions

2.12.1 SsmQueryOpPowerAmp

Queries if the channel is equipped with on-board audio amplifier circuit.

Format:

int SsmQueryOpPowerAmp(int ch)

Parameter Description:

ch	Channel Number

Return Value:

-1	Call failed
0	Not equipped with on-board audio amplifier circuit
1	Equipped with on-board audio amplifier circuit

Function Description:

Queries if the channel is equipped with on-board audio amplifier circuit. For detailed information about the on-board audio power amplifier, see the operation principles of the corresponding boards.

Note:

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmSetPowerAmpVlm, SetVolume

2.12.2 SsmSetPowerAmpVlm

Refer to SetVolume

2.12.3 SetVolume

Sets the gain of the on-board audio power amplifier. As SetVolume is designed for early versions, we suggest you

use the function SsmSetPowerAmpVlm instead.

Format:

void SsmSetPowerAmpVlm(int ch, int nGain)

void SetVolume(DWORD dwBoardId, DWORD nGain)

Parameter Description:

ch	Channel Number. Note: the internal number of Channel ch on the corresponding board must be 0, otherwise the function will fail
dwBoardId	Board ID
nGain	The gain of the power amplifier, range of value: 0~7, in which: 7: 0DB 6: -6DB 5: -12DB 4: -18DB 3: -24DB 2: -30DB 1: -36DB 0: Turned off The default value is 3(-24DB).

Return Value: None

Function Description:

Sets the gain of the on-board audio power amplifier. For detailed information about the on-board audio power amplifier, see the operation principles of the corresponding boards.

Note:

- Only Channel 0 on ATP, SHT or DST Series boards is equipped with the on-board audio power amplifier.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmQueryOpPowerAmp

2.13 On-board Speaker Functions

2.13.1 SsmSetLine0OutTo

Enables or disables the on-board speaker of the USB recording box/ voice box.

Format:

int SsmSetLine0OutTo(BOOL bEnable)

Parameter Description:

bEnable	=FALSE: sends the voice to the on-board speaker for playing; =TRUE: sends the voice to the on-board speaker output jack, to play the voice via an external speaker

Return Value:

-1	Call failed
0	Successful

Function Description:

Enables or disables the on-board speaker of the USB recording box/voice box. For more information, refer to 'Operation Principle of ATP Series' or 'Operation Principle of SHT Series' in Chapter 1.

Note:

- This function is only applicable to ATP Series USB recording box and SHT Series USB voice box.
- The configuration item USBLine0Output can implement the same features.

Related Information:

Driver version	SynCTI Ver. 4.7.1.7 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmSetPowerAmpVlm

2.14 Echo Canceller Functions (CTI Series)

2.14.1 Setting Operating Status of Echo Canceller

2.14.1.1 SsmQueryOpEchoCanceller

Queries if the channel supports echo cancellation.

Format:

int SsmQueryOpEchoCanceller(int ch)

Parameter Description:

ch	Channel Number

Return Value:

-1	Call failed
0	Not supported
1	Supported

Function Description:

Queries if the channel supports echo cancellation.

Note:

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function:

2.14.1.2 SsmSetEchoCanceller

Sets the operating mode of the echo canceller.

Format:

int SsmSetEchoCanceller(int ch, BOOL bEnable)

Parameter Description:

ch	Channel Number
bEnable	TRUE: Enable FALSE: Disable

Return Value:

-1	Call failed
0	Successful

Function Description:

Sets the operating mode of the echo canceller.

Note:

- The configuration item EnableEchoCancellor can implement the same feature. This feature is enabled by default.
- For more information, refer to Echo Canceller in Chapter 1.
- This function only supports SHD, SHT Series boards.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmGetEchoCancellerState

2.14.1.3 SsmGetEchoCancellerState

Obtains the operating status of the echo canceller.

Format:

int SsmGetEchoCancellerState(int ch)

Parameter Description:

ch	Channel number

Return Value:

-1	Call failed
0	Turned off
1	Turned on

Function Description:

Obtains the operating status of the echo canceller.

Note:

- This function only supports SHD Series and SHT Series boards.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmSetEchoCanceller

2.14.2 Setting Parameters of Echo Canceller

2.14.2.1 SsmSetEchoCancelDelaySize

Sets the filter delay coefficient of the echo canceller.

Format:

int SsmSetEchoCancelDelaySize(int ch,WORD wSize)

Parameter Description:

ch	Channel number
wSize	The filter delay coefficient, range of value: 0~31

Return Value:

-1	Call failed
0	Successful

Function Description:

Sets the filter delay coefficient of the echo canceller.

Note:

- This function only supports SHD Series and SHT Series boards;
- The configuration item EchoCancelDelaySize can implement the same features.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmGetEchoCancelDelaySize

2.14.2.2 SsmGetEchoCancelDelaySize

Obtains the filter delay coefficient of the echo canceller.

Format:

int SsmGetEchoCancelDelaySize(int ch)

Parameter Description:

ch	Channel Number

Return Value:

-1	Call failed
≥0	The filter delay coefficient

Function Description:

Obtains the filter delay coefficient of the echo canceller.

Note:

- This function only supports SHD Series and SHT Series boards.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmSetEchoCancelDelaySize

2.15 Teleconferencing Functions (CTI Series)

2.15.1 Initializing Conference Room

2.15.1.1 SsmCreateConfGroup

Creates a conference room.

Format:

int SsmCreateConfGroup(int nMaxMember, int nMaxSpeaker, int nMaxSpeaking, int nMaxSilenceTime)

Parameter Description:

nMaxMember	The maximum number of channels in a conference room, range of value: -1: Use the parameter specified in the configuration item ConfDefaultMaxGroupMember 3~N: N is the total number of channels in the application
nMaxSpeaker	Sets the maximum number of conference channels which have speaking rights. The channels in the speaking modes of organizer, chairman or dynamic speaker all have speaking rights. It's determined by the set value of the configuration item BackgroundVoicePriority whether the channel in the background music mode has the speaking right. Range of value: -1: Use the parameter designated by the configuration item ConfDefaultMaxGroupSpeaker ≥1: The total number of conference channels
nMaxSpeaking	The maximum number of speaking members allowed at a time, range of value: -1: Use the parameter designated by the configuration item ConfDefaultMaxGroupSpeaking; 1~6: The set value
nMaxSilenceTime	Sets the minimum duration for the channel to keep silent. Range of value: -1: Use the parameter designated by the configuration item ConfDefaultMaxSilenceTime ≥1: The set value (seconds) For more information, refer to 'Distributed Teleconferencing System'. Note: Now this parameter is invalid. Just set it to -1.

Return Value:

-1	Call failed, the failure reason can be obtained by the function SsmGetLastErrMsg
≥0	The ID number of the newly-created conference room

Function Description:

Creates a conference room.

Note: None

Related Information:

Driver version	SynCTI Ver. 3.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmFreeConfGroup, SsmJoinConfGroup, SsmGetConfGrpInfo, SsmGetConfGrpCfgInfo

2.15.1.2 SsmFreeConfGroup

Cancels a conference room.

Format:

int SsmFreeConfGroup(int nGrpId)

Parameter Description:

nGrpId	Conference room number

Return Value:

-1	Call failed, the failure reason can be obtained by the function SsmGetLastErrMsg
0	Successful

Function Description:

Cancels a conference room, and releases the resources it occupied.

Note:

- If there are still channels in the conference room, the driver will automatically invoke the function SsmExitConfGroup to put them out of the conference room.

Related Information:

Driver version	SynCTI Ver. 3.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmCreateConfGroup

2.15.2 Setting Conference Member's Behavior

2.15.2.1 SsmJoinConfGroup

Puts a channel into the conference room.

Format:

int SsmJoinConfGroup(int nGrpId, int ch, WORD wJoinMode, int nMixerVolume, BOOL bCreateAlways, BOOL bExitGrpAlways)

Parameter Description:

nGrpId	Conference room number
ch	Channel Number
wJoinMode	The speaking mode of the channel, range of value: 0: Organizer; 1: Dynamic Speaker; 2: Listener; 3: Background Music; Whether the channel in this mode can hear the voices from other channels is determined by the configuration item PlayVoiceIsListen. When PlayVoiceIsListen＝0, the channel can speak but not hear the voices from other channels, and it is usually used for playing music to a conference; when PlayVoiceIsListen＝1 (default), the channel can not only speak but also hear the voices from other channels. 4: Chairman; 5: Dynamic Speaker ONLY For more information about the speaking modes and conference scheduling, refer to 'Distributed Teleconferencing System'
nMixerVolume	The volume of the incoming call signal on the current channel entering the off-bus

	mixer M2 for other conference member, range of value: -6~+6. The positive value means volume increasing, while the negative value means volume decreasing. This parameter multiplied by 3 equals dB value. This volume is the speaking volume of this member heard by others in the conference.
bCreateAlways	When this function is called, if the conference room designated by the parameter nGrpId does not exist, this parameter indicates whether to automatically create a new conference room with the conference room number nGrpId by the driver. TRUE: yes FALSE: no, call failed
bExitGrpAlways	If this channel has already joined in another conference room, this parameter indicates whether to automatically put this channel out of that conference room by the driver TRUE: yes FALSE: no, call failed

Return Value:

-1	Call failed, the failure reason can be obtained by the function SsmGetLastErrMsg
0	Successful

Function Description:

Puts a channel into the conference room.

Note: none

Related Information:

Driver version	SynCTI Ver. 3.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmExitConfGroup, SsmSetListenVlmInConf

2.15.2.2 SsmExitConfGroup

Puts a the channel out of the conference room.

Format:

int SsmExitConfGroup(int ch, BOOL bFreeGrpAlways)

Parameter Description:

ch	Channel Number
bFreeGrpAlways	Sets whether to cancel the conference room automatically by the driver if the current channel is the only channel in it. TRUE: Cancel; FALSE: Not cancel

Return Value:

-1	Call failed, the failure reason can be obtained by the function SsmGetLastErrMsg
0	Successful

Function Description:

Puts a channel out of the conference room.

Related Information:

Driver version	SynCTI Ver. 3.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmJoinConfGroup

2.15.2.3 SsmSetListenVlmInConf

Sets the volume adjuster for the off-bus signals entering M2 (off-bus mixer).

Format:

int SsmSetListenVlmInConf(int ch, int nVlm)

Parameter Description:

ch	Channel number
nVlm	Volume of the off-bus signals, range of value: -7~6.

Return Value:

0	Successful
-1	Call failed

Function Description:

Sets the volume adjuster for the off-bus signals entering M2 (off-bus mixer), i.e. sets the volume adjuster A4-1…A4-6 in 'Operation Principle of SHT Series', 'Operation Principle of SHD Series' or 'Operation Principle of SHN Series' in Chapter 1. This function can only adjust the volume heard by the current channel, but not the volume heard by other members in the conference.

Note:

- This function must be invoked before the channel joins the conference room; otherwise, it will be invalid.

Related Information:

Driver version	SynCTI Ver. 3.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmJoinConfGroup

2.15.2.4 SsmSetContactInConf

Blocks a channel from hearing another channel in a teleconference.

Format:

int SsmSetContactInConf(int nGrpID, int chFrom, int chTo, BOOL bFlag)

Parameter Description:

nGrpID	Conference room number
chFrom	Number of the channel where voices come from
chTo	Number of the channel where voices go to
bFlag	TRUE: Channel chTo is blocked from hearing Channel chFrom in a teleconference; FALSE: Channel chTo is required to hear Channel chFrom in a teleconference.

Return Value:

-1	This function is unsupported
0	Call failed
1	Successful

Function Description:

Blocks a channel from hearing another channel in a teleconference.

Note:

- A channel can be blocked to only one channel . For example, when Channel A is blocked from hearing Channel B, it can't be blocked from hearing Channel C unless its blocking to Channel B is disabled first.

Related Information:

Driver version	SynCTI Ver. 3.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: None

2.15.3 Obtaining Conference Room Information

2.15.3.1 SsmGetConfCfgInfo

Obtains the configuration parameters of the distributed teleconferencing.

Format:

int SsmGetConfCfgInfo(PWORD pwMaxMember, PWORD pwMaxSpeaker, PWORD pwMaxSpeaking, PWORD pwMaxSilenceTime)

Parameter Description:

pwMaxMember	Returns the set value of the configuration item ConfDefaultMaxGroupMember
pwMaxSpeaker	Returns the set value of the configuration item ConfDefaultMaxGroupSpeaker
pwMaxSpeaking	Returns the set value of the configuration item ConfDefaultMaxGroupSpeaking
pwMaxSilenceTime	Returns the set value of the configuration item ConfDefaultMaxSilenceTime

Return Value:

-1	Call failed
≥0	Returns the set value of the configuration item ConfMaxGroup

Function Description:

Obtains the configuration parameters of the distributed teleconferencing.

Note: none

Related Information:

Driver version	SynCTI Ver. 3.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmGetConfGrpInfo

2.15.3.2 SsmGetTotalConfGroup

Obtains the actual number of the conference rooms.

Format:

int SsmGetTotalConfGroup()

Parameter Description: none

Return Value:

-1	Call failed
≥0	The actual number of the conference rooms

Function Description:

Obtains the actual number of the conference rooms.

Note: none

Related Information:

Driver version	SynCTI Ver. 3.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmGetConfGrpId

2.15.3.3 SsmGetConfGrpCfgInfo

Obtains the set values of the operating parameters of the conference room.

Format:

int SsmGetConfGrpCfgInfo(int nGrpId, PWORD pwMaxMember, PWORD pwMaxSpeaker, PWORD pwMaxSpeaking, PWORD pwMaxSilenceTime)

Parameter Description:

nGrpId	Conference room number
pwMaxMember	Returns the total number of the conference channels
pwMaxSpeaker	Returns the total number of the conference channels with speaking rights
pwMaxSpeaking	Returns the total number of conference channels which are permitted to speak at a time
pwMaxSilenceTime	Sets the minimum duration (s) for the channel to keep silent.

Return Value:

-1	Call failed
0	Successful

Function Description:

Obtains the set values of the operating parameters of the conference room which are configured by the application via the function SsmCreateConfGroup upon the creation of the conference room.

Note: none

Related Information:

Driver version	SynCTI Ver. 3.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmGetConfCfgInfo

2.15.3.4 SsmGetConfGrpInfo

Obtains the statistical information of the conference room.

Format:

int SsmGetConfGrpInfo(int nGrpId, PWORD pwTotalMember, PWORD pwTotalSpeaker, PWORD pwTotalSpeaking)

Parameter Description:

nGrpId	Conference room number
pwTotalMember	Returns the actual number of the channels joining in the conference
pwTotalSpeaker	Returns the actual number of the channels with speaking rights in the conference room
pwTotalSpeaking	Returns the actual number of the conference channels currently using the conference mixer

Return Value:

-1	Call failed

0	Successful

Function Description:

Obtains the statistical information of the conference room.

Note: none

Related Information:

Driver version	SynCTI Ver. 3.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmGetConfCfgInfo

2.15.3.5 SsmGetConfGrpId

Obtains the conference room number list for all the conference rooms.

Format:

int SsmGetConfGrpId(int* pnGrpId)

Parameter Description:

pnGrpId	Returns the pointer pointing to the array containing all the conference room numbers

Return Value:

-1	Call failed
0	Successful

Function Description:

Obtains the conference room number list for all the conference rooms.

Note: none

Related Information:

Driver version	SynCTI Ver. 3.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmValidateGrpId

2.15.3.6 SsmValidateGrpId

Queries if the conference room has been created.

Format:

int SsmValidateGrpId(int nGrpId)

Parameter Description:

nGrpId	Conference room number

Return Value:

1	The conference room is already created
0	Not created

Function Description:

Queries if the conference room has been created.

Note: None

Related Information:

Driver version	SynCTI Ver. 3.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmGetConfGrpId

2.15.4 Obtaining Conference Member Information

2.15.4.1 SsmGetConfGrpMmbrId

Obtains the number of each channel in the conference room. Note that the number herein is not a logical channel number in the whole system, but a number in the conference.

Format:

int SsmGetConfGrpMmbrId(int nGrpId, int* pnMmbrId)

Parameter Description:

nGrpId	Conference room number
pnMmbrId	The pointer pointing to the array containing the conference member IDs

Return Value:

-1	Call failed
0	Successful

Function Description:

Obtains the number of each channel in the conference room. Note that the number herein is not a logical channel number in the whole system, but a number in the conference.

Note: none

Related Information:

Driver version	SynCTI Ver. 3.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmGetConfGrpMmbrInfo

2.15.4.2 SsmGetConfGrpMmbrInfo

Obtains the detailed information about the conference channels in the conference room.

Format:

int SsmGetConfGrpMmbrInfo(int nGrpId, int nMmbrId, int* pnAppCh, PWORD pwJoinMode, PWORD pwIsSpeaking, PDWORD pdwSilenceTime)

Parameter Description:

nGrpId	Conference room number
nMmbrId	Conference channel number
pnAppCh	Returns the logical number corresponding to the conference channel number
pwJoinMode	Returns the speaking mode of the conference channel
pwIsSpeaking	Returns whether the conference channel is speaking
pdwSilenceTime	Returns the duration (s) for the conference channel to keep silent

Return Value:

-1	Call failed
0	Successful

Function Description:

Obtains the detailed information about the conference channels in the conference room according to the conference room number and the conference channel number.

Note: none

Related Information:

Driver version	SynCTI Ver. 3.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmGetConfChInfo

2.15.4.3 SsmGetConfChInfo

Queries if the channel has joined in a specific conference room.

Format:

int SsmGetConfChInfo(int ch, int* pnGrpId, int* pnMmbrId, PWORD pwJoinMode, PWORD pwIsSpeaking, PDWORD pdwSilenceTime)

Parameter Description:

ch	Channel Number
pnGrpId	Returns the conference room number
pnMmbrId	Returns the member ID of Channel ch in the conference room
pwJoinMode	Returns the speaking mode of Channel ch in the conference room
pwIsSpeaking	Returns the sign indicating whether Channel ch is speaking or not. pwIsSpeaking=1: Speaking; pwIsSpeaking=0: Not speaking.
pdwSilenceTime	Returns the duration (s) of Channel ch to keep silent

Return Value:

-1	Call failed
0	Successful

Function Description:

Queries if the channel has joined in a specific conference room.

Note: none

Related Information:

Driver version	SynCTI Ver. 3.0or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmGetConfGrpMmbrInfo

2.16 FSK Transceiver Functions (CTI Series)

2.16.1 FSK Transmitter

2.16.1.1 SsmSetFskPara

Sets the operating parameters of the FSK Transmitter.

Format:

int SsmSetFskPara(int nFreq0, int nFreq1, int nBaudrate, int nMdlAmp)

Parameter Description:

nFreq0	The carrier frequency (Hz) of bit 0, range of value: 300~3400. This parameter can also be set by the configuration item FreqBit0. The default value is 2200Hz
nFreq1	The carrier frequency (Hz) of bit 1, range of value: 300~3400. This parameter can also be set by the configuration item FreqBit1. The default value is 1200Hz
nBaudrate	Baud rate (bps), range of value: 300~3400. This parameter can also be set by the configuration item Baudrate. The default value is 1200bps
nMdlAmp	Modulation Amplitude, range of value: 0~255. This parameter can also be set by the configuration item MdlAmp. The default value is 128

Return Value:

-1	Call failed
0	Successful

Function Description:

Sets the operating parameters of the FSK Transmitter.

Note:

- If Synway boards are used at the reception end, normally no extra configuration is needed. Just use the default settings.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmGetFskPara

2.16.1.2 SsmGetFskPara

Obtains the operating parameters of the FSK Transmitter.

Format:

int SsmGetFskPara(int *nFreqBit0, int *nFreqBit1, int *nBaudrate, int *nMdlAmp)

Parameter Description:

nFreqBit0	Returns the carrier frequency (Hz) of bit 0
nFreqBit1	Returns the carrier frequency (Hz) of bit 1
nBaudrate	Returns the baud rate (bps) of the binary code stream
nMdlAmp	Returns the amplitude of the modulating signal

Return Value:

-1	Call failed

0	Successful

Function Description:

Obtains the operating parameters of the FSK Transmitter.

Note:

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmSetFskPara

2.16.1.3 SsmTransFskData

Converts the data to BFSK-formatted binary data stream.

Format:

int SsmTransFskData(unsigned char* pSrc, int nSrcLen,int nSyncLen,int nSyncOffLen,unsigned char* pStream)

Parameter Description:

pSrc	Pointer pointing to the buffer area storing the data
nSrcLen	The data length (bytes) in pSrc
nSyncLen	The bit number of the syn code, normally it is integral times of 8. It can be set according to your actual need and the recommended value is 96
nSyncOffLen	The bit number of the flag code, normally its integral times of 8. It can be set according to your actual need and the recommended value is 32
pStream	Pointer pointing to the buffer area storing the converted BFSK data stream. Its space is allocated by the application, and its size can be calculated by the following formula: (nSyncLen + nSyncOffLen + nSrcLen×10) / 8 + 1

Return Value:

-1	Call failed
≥0	The actual bit number in pStream

Function Description:

Converts the data to BFSK-formatted binary data stream. For more information about the format of the BFSK data stream, refer to 'FSK Transceiver' in Chapter 1.

Note:

- In some protocols the flag code is called 'synchronization end character', and it is together with the syn code called 'synchronization indicator string' which indicates the establishment of synchronization. In this function, we use nSyncLen to represent the syn code and nSyncOffLen to indicate the flag code.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmStartSendFSK

2.16.1.4 SsmStartSendFSK

Starts the FSK transmitter.

Format:

int SsmStartSendFSK(int ch, LPSTR pStream, DWORD dwMaxBit)

Parameter Description:

ch	Channel Number
pStream	The pointer pointing to the buffer area storing the BFSK data stream. For more information, refer to 'FSK Transceiver' in Chapter 1
dwMaxBit	The total number of valid bits in pStream, it could be of any length

Return Value:

-1	Call failed
0	Successful

Function Description:

Starts the FSK transmitter.

The operating parameters of the FSK transmitter can be configured by the function SsmSetFskPara or the configuration items FreqBit0, FreqBit1, Baudrate, and MdlAmp.

After the FSK transmitter is started, the application can invoke the function SsmStopSendFsk to stop it at anytime. When the driver has finished transmitting all the bits in pStream, it will throw out the event E_PROC_SendFSK to the application. The application can also invoke the function SsmCheckSendFsk to query the transmitting status of the FSK transmitter.

Note:

- You could neither start the voice playing task nor use the DTMF generator or the tone generator before the FSK transmitter finishes transmitting all the data.
- Before the FSK transmitter is started, in order to ensure the data integrity, the echo cancellation feature should be disabled.
- This function can be called only after the original data has been converted to BFSK data stream via the function call of SsmTransFskData.
- This function does not support ATP, DTP and SHN Series boards.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmCheckSendFsk, SsmStopSendFsk, SsmTransFskData

Sample code:

```
char Data[]='12345';                    //The string '12345' needs to be transmitted
unsigned char Stream[256];              // Stores the converted BFSK bit stream
int nMaxBit = SsmTransFskData( (unsigned char*)Data, 5,96, 32, Stream);
SsmStartSendFSK(ch, (PCHAR)Stream, nMaxBit);
```

2.16.1.5 SsmCheckSendFsk

Queries the transmitting status of the FSK transmitter.

Format:

int SsmCheckSendFsk(int ch)

Parameter Description:

ch	Channel Number

Return Value:

-1	Call failed
0	Finished
1	Unfinished

Function Description:

Queries the transmitting status of the FSK transmitter.

Note:

- If the application uses the event based programming mode, it's recommended to use the event E_PROC_SendFSK.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmStartSendFSK

2.16.1.6 SsmStopSendFsk

Stops the FSK transmitter.

Format:

int SsmStopSendFsk(int ch)

Parameter Description:

ch	Channel number

Return Value:

-1	Call failed
0	Successful

Function Description:

Stops the FSK transmitter.

Note:

- After invoking this function, the driver throws out the event E_PROC_SendFSK to the application.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmStartSendFSK

2.16.2 FSK Receiver

2.16.2.1 SsmStartRcvFSK

Starts the FSK receiver.

Format:

int SsmStartRcvFSK(int ch, WORD wTimeOut, WORD wMaxLen, UCHAR ucEndCode, WORD wEndCodeCount)

Parameter Description:

ch	Channel Number
wTimeOut	Timer interval (ms)
wMaxLen	The maximum length (bytes) of the FSK data, range of value: 1~1024
ucEndCode	The character indicating the end of the FSK data package
wEndCodeCount	The minimum times that ucEndCode continuously appears, range of value: 0~10

Return Value:

-1	Call failed, the failure reason can be obtained by the function SsmGetLastErrMsg
0	Successful

Function Description:

Starts the FSK receiver and begins to receive FSK data.

After the FSK receiver is started, the driver will start a timer and decide whether to stop receiving data based on the following rules:

- ✧ If the end character which is set by the parameter ucEndCode is detected in the FSK data, and the number of times it continuously appears is greater than or equal to wEndCodeCount, the driver will stop the task and throw out the event E_PROC_RcvFSK (dwParam is set to be 2) to the application;
- ✧ If the length (total byte number) of the FSK data is larger than or equal to wMaxLen, the current task of FSK data receiving will be stopped, and the driver will also throw out the event E_PROC_RcvFSK (dwParam is set to be 3) to the application;
- ✧ If the BFSK signal disappears on the line, and does not reappear in 100ms, the current task of FSK data receiving will be stopped, and the driver will throw out the event E_PROC_RcvFSK (dwParam is set to be 1) to the application.
- ✧ If the timer overflows, the current task of FSK data receiving will be stopped, and the driver will throw out the event E_PROC_RcvFSK (dwParam is set to be 1) to the application.

Note:

- The baud rate of the FSK receiver on Synway boards is 1200bps, the carrier frequency of bit 0 is 2200Hz, and the carrier frequency of bit 1 is 1200Hz. Neither of them can be changed via functions or configuration items.
- This function does not support ATP, DTP and SHN Series boards.

Related Information:

Driver version	SynCTI Ver. 3.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmGetRcvFSK, SsmStopRcvFSK, SsmStartRcvFSK_II

Sample code:

```
 ......
SsmStartRcvFSK (0, 10000, 260,   '#', 1);                    //Starts the FSK receiver

MESSAGE_INFO Event;
If ( SsmWaitForEvent (64, &Event) == 0 )                     //Waits the event
{
    switch(Event.EventCode)
    {
        case E_PROC_RcvFSK:                                  //Stops the FSK receiver
            if(Event.dwParam == 2)                           //Completes receiving FSK data ended with end character '#'
            {
                unsigned char DataBuf[260];
                SsmGetRcvFSK(Event.nReference, DataBuf);//Gets the data
                ......                                       //Processes the code
            }
            ......
```

```
            break;
            ......
        }
    }
```

2.16.2.2 SsmStartRcvFSK_II

Starts to receive FSK data.

Format:

int WINAPI SsmStartRcvFSK_II(int ch, WORD wTimeOut, WORD wMaxLen, PUCHAR pucMarkCodeBuf, UCHAR ucMarkCodeCount)

Parameter Description:

ch	Channel number
wTimeOut	Timer interval (ms)
wMaxLen	The maximum length (bytes) of the FSK data: 1~260
pucMarkCodeBuf	The buffer area storing flag codes, at most 10 different flag codes can be set
ucMarkCodeCount	The number of flag codes in pucMarkCodeBuf, range of value: 0~10. If this parameter is set to 0, the parameter pucMarkCodeBuf will be ignored

Return Value:

-1	Call failed. The failure reason can be obtained by the function SsmGetLastErrMsg
0	FSK receiver is started successfully

Function Description:

Starts the FSK receiver to begin the task of FSK data receiving.

This function supports two receiving modes: Length Mode and Frame Mode. Frame Mode means the FSK data to be transmitted has a fixed frame structure. The driver needs to align and extract the frame according to the frame flag code; Length Mode means the driver only examines the length of the data to be received. When the length of the received FSK data becomes larger than wmMaxLen, the task of data receiving is terminated. If ucMarkCodeCount is larger than 0, the driver will use Frame Mode as the first choice to receive data. Otherwise, it will use Length Mode where the parameter wMaxLen is valid.

The frame is composed of four fields: flag code, data length, data body and check byte.

Flag Code	Data Length	Data Body	Check Byte

Flag Code: The beginning of the frame, occupying 1 byte and used for frame alignment. Its value can be determined by the application itself;

Data Length: The length (bytes) of the data body, occupying 1 byte;

Data Body: The data to be transmitted, the maximum length can't exceed 255 bytes;

Check Byte: It occupies 1 byte. Note that the check byte is generated by the application, so the driver doesn't check the data integrity.

The execution rules of the FSK data receiver:

- ✧ Starts a timer and waits for the arrival of FSK data.
- ✧ Flag Code Mode: After the arrival of FSK data, the driver searches the flag code set by the parameter pucMarkCodeBuf in the first 10 bytes of the received data. If the flag code is found, it means the driver has finished the frame alignment successfully. After the driver has received a frame of data, the current task of data receiving is terminated and the driver throws out the event E_PROC_RcvFSK (dwParam is

set to 4) to the application. If the flag code is not found, the data receiving will turn to the length mode.

- ✧ Length Mode: When the FSK data length exceeds the set value of the parameter wMaxLen, the current task of data receiving will be stopped and the driver will throw out the event E_PROC_RcvFSK (dwParam is set to 3) to the application.
- ✧ If the BFSK signal disappears on the line and does not reappear within 100ms, the current task of FSK data receiving will be stopped and the driver will throw out the event E_PROC_RcvFSK (dwParam is set to 1 if the length of the received FSK data is less than wMaxLen, otherwise it is set to 3) to the application.
- ✧ If the timer overflows, the current task of FSK data receiving will be stopped and the driver will throw out the event E_PROC_RcvFSK (dwParam is set to 1) to the application.

Note:

- The baud rate of the FSK receiver on Synway boards is 1200bps, the carrier frequency of Bit 0 is 2200Hz and that of Bit 1 is 1200Hz. Neither of them can be changed by modifying functions or configuration items.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmGetRcvFSK, SsmStopRcvFSK, SsmStartRcvFSK

Sample code:

```
 ......
SsmStartRcvFSK_II (0, 10000, 1, '#', 1);                    //Start FSK receiver

MESSAGE_INFO Event;
If ( SsmWaitForEvent (64, &Event) == 0 )                    //Wait for an event to be triggered
{
    switch(Event.EventCode)
    {
        case E_PROC_RcvFSK:                                 //FSK receiver is terminated
            if(Event.dwParam == 4)                          //A frame of FSK data are received
            {
                unsigned char DataBuf[260];
                SsmGetRcvFSK(Event.nReference, DataBuf);    //Take the data
                ......                                      //Process the data
            }
            else
            {
                ......                                      //Process the failure in receiving
            }
        break;
        ......
    }
}
```

2.16.2.3 SsmStartRcvFSK_III

Starts to receive FSK data.

Format:

int WINAPI SsmStartRcvFSK_III(int ch,WORD wTimeOut,WORD wMaxLen,PUCHAR pucMarkCodeBuf, UCHAR ucMarkCodeCount)

Parameter Description:

ch	Channel number
wTimeOut	Timer interval (ms)

wMaxLen	The maximum length (bytes) of the FSK data, range of value: 1~1024
pucMarkCodeBuf	The buffer area storing flag codes, at most 10 different flag codes can be set
ucMarkCodeCount	The number of flag codes in pucMarkCodeBuf, range of value: 0~10. If this parameter is set to 0, the parameter pucMarkCodeBuf will be ignored; if this parameter is set to 5, the former 5 flag codes in pucMarkCodeBuf are valid.

Return Value:

-1	Call failed. The failure reason can be obtained by the function SsmGetLastErrMsg
0	FSK receiver is started successfully

Function Description:

Starts the FSK receiver to begin the task of FSK data receiving.

This function supports two receiving mode: Length Mode and Frame Mode. Frame Mode means the FSK data to be transmitted has a fixed frame structure. The driver needs to align and extract the frame according to the frame flag code; Length Mode means the driver only examines the length of the data to be received. When the length of the received FSK data becomes larger than wmMaxLen, the task of data receiving is terminated. If ucMarkCodeCount is larger than 0, the driver will use Frame Mode as the first choice to receive data. Otherwise, it will use Length Mode where the parameter wMaxLen is valid for receiving.

The frame is composed of four fields: flag code, data length, data body and check byte.

Flag Code	Data Length	Data Body	Check Byte

Flag Code: The beginning of the frame, occupying 1 byte and used for frame alignment. Its value can be determined by the application itself;

Data Length: The length (bytes) of the data body, occupying 2 bytes;

Data Body: The data to be transmitted, the maximum length can't exceed 1020 bytes;

Check Byte: It occupies 1 byte. Note that the check byte is generated by the application, so the driver doesn't check the data integrity.

The execution rules of the FSK data receiver:

- ✧ Starts a timer and waits for the arrival of FSK data.
- ✧ Flag Code Mode: After the arrival of FSK data, the driver searches the flag code set by the parameter pucMarkCodeBuf in the first 10 bytes of the received data. If the flag code is found, it means the driver has finished the frame alignment successfully. After the driver has received a frame of data, the current task of data receiving is terminated and the driver throws out the event E_PROC_RcvFSK (dwParam is set to 4) to the application. If the flag code is not found, the data receiving will turn to the length mode.
- ✧ Length Mode: When the FSK data length exceeds the set value of the parameter wMaxLen, the current task of data receiving will be stopped and the driver will throw out the event E_PROC_RcvFSK (dwParam is set to 3) to the application.
- ✧ If the BFSK signal disappears on the line and does not reappear within 100ms, the current task of FSK data receiving will be stopped and the driver will throw out the event E_PROC_RcvFSK (dwParam is set to 1 if the length of the received FSK data is less than wMaxLen, otherwise it is set to 3) to the application.
- ✧ If the timer overflows, the current task of FSK data receiving will be stopped and the driver will throw out the event E_PROC_RcvFSK (dwParam is set to 1) to the application.

Note:

- The baud rate of the FSK receiver on Synway boards is 1200bps, the carrier frequency of Bit 0 is 2200Hz and that of Bit 1 is 1200Hz. Neither of them can be changed by modifying functions or configuration items.
- The data length occupies 2 bytes in the frame structure: the first byte indicates the length of high bits and the second one represents the length of low bits.

  For example,

  sz[5] = {0xFF 0x0 0x3 0x1 0x2 0x3} is a correct FSK datum, in which

  sz[0] = 0xFF is the start flag;

  sz[1] = 0x0, sz[2] = 0x3 indicate the length is 3 bytes;

  0x1 0x2 0x3 at the end is the data body.

Related Information:

Driver version	SynCTI Ver. 5.0.1.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmGetRcvFSK, SsmStopRcvFSK, SsmStartRcvFSK, SsmStartRcvFSK_II

Sample code:

```
 ......
SsmStartRcvFSK_III (0, 10000, 1, '#', 1);                    //Start FSK receiver

MESSAGE_INFO Event;
If ( SsmWaitForEvent (64, &Event) == 0 )                     //Wait for an event to be triggered
{
    switch(Event.EventCode)
    {
        case E_PROC_RcvFSK:                                  //FSK receiver is terminated
            if(Event.dwParam == 4)                           //A frame of FSK data are received
            {
                unsigned char DataBuf[260];
                SsmGetRcvFSK(Event.nReference, DataBuf);     //Take the data
                ......                                       //Process the data
            }
            else
            {
                ......                                       //Process the failure in receiving
            }
        break;
        ......
    }
}
```

2.16.2.4 SsmCheckRcvFSK

Queries the receiving status of the FSK data

Format:

int SsmCheckRcvFSK(int ch)

Parameter Description:

ch	Channel Number

Return Value:

-1	Call failed
0	FSK receiver has not finished the current task of data receiving yet
1	Terminated because the timer overflows
2	Terminated due to the reception of the end character
3	Terminated due to the reception of the data with designated length

4	Terminated due to the reception of the data with designated format
5	Terminated because the application invokes the function SsmStopRcvFSK

Function Description:

Queries the receiving status of the FSK data. The receiving task could be initiated by the function SsmStartRcvFSK or SsmStartRcvFSK_II.

Note:

- If the application's programming mode is based on event, it's recommended to use the event E_PROC_RcvFSK.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmGetRcvFSK

2.16.2.5 SsmStopRcvFSK

Cancels the task of FSK data receiving.

Format:

int SsmStopRcvFSK(int ch)

Parameter Description:

ch	Channel number

Return Value:

-1	Call failed
0	Successful

Function Description:

Cancels the task of FSK data receiving. This function can cancel the data receiving task started by the function SsmStartRcvFSK or SsmStartRcvFSK_II.

Note:

- After the driver finishes executing this function, it will throw out the event E_PROC_RcvFSK to the application.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmGetRcvFSK, SsmCheckRcvFSK

2.16.2.6 SsmClearRcvFSKBuf

Clears the FSK receiving buffer area in the driver.

Format:

int SsmClearRcvFSKBuf(int ch)

Parameter Description:

ch	Channel Number

Return Value:

-1	Error occurs, the failure reason can be obtained by the function SsmGetLastErrMsg
0	Successful

Function Description:

Clears the FSK receiving buffer area in the driver.

Note:

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmGetRcvFSK

2.16.2.7 SsmGetRcvFSK

Gets the FSK data stored in the buffer area in the driver.

Format:

int SsmGetRcvFSK(int ch, PUCHAR pucBuf)

Parameter Description:

ch	Channel Number
pucBuf	Buffer area pointer, the storage space is allocated by the application, its size should be no less than 260 bytes

Return Value:

-1	Call failed
≥0	The length (bytes) of the received FSK data (excluding the end character)

Function Description:

Gets the FSK data stored in the buffer area in the driver.

Note:

- Each time when the application calls the function SsmStartRcvFSK or starts the task of data receiving, the driver will automatically clear the buffer area storing the FSK data in the driver.

Related Information:

Driver version	SynCTI Ver. 3.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmCheckRcvFSK, SsmClearRcvFSKBuf

2.17 Voltage Detector Functions

2.17.1 Ignoring Result of Voltage Detector

2.17.1.1 SsmSetIgnoreLineVoltage

Sets whether to ignore the result of the voltage detector on analog phone lines.

Format:

int SsmSetIgnoreLineVoltage(int ch, BOOL bIgnore)

Parameter Description:

ch	Channel Number
bIgnore	=FALSE: not to ignore =TRUE: ignore

Return Value:

-1	Operation failed
0	Operation is successful

Function Description:

Sets whether to ignore the result of the voltage detector on analog phone lines. If the result of the voltage detector is ignored, the recording channel will always keep 'off-hook' state. For more information, refer to 'Change in Analog Phone Line Voltage' in Chapter 1.

Note:

- This function is only applicable to recording channels (including analog trunk recording module and microphone module) of ATP Series boards;
- The parameter set by this function can also be set by the configuration item IgnoreLineVoltage, with the default value of 'not to ignore'.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmGetIgnoreLineVoltage

2.17.1.2 SsmGetIgnoreLineVoltage

Queries if the result of the line voltage detection of the channel is ignored.

Format:

int SsmGetIgnoreLineVoltage(int ch)

Parameter Description:

ch	Channel Number

Return Value:

-1	Call failed
0	Not to ignore
1	Ignore

Function Description:

Queries if the result of the line voltage detection of the channel is ignored.

Note:

- This function is only applicable to recording channels (including analog trunk recording module and microphone module) of ATP Series boards.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h

Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmSetIgnoreLineVoltage

2.17.2 Setting Threshold Voltage to Detect Pickup and Hangup Behaviors

2.17.2.1 SsmSetDtrmLineVoltage

Sets the threshold voltage to detect the pickup and hangup operations on the analog trunk recording channel.

Format:

int SsmSetDtrmLineVoltage(int ch, WORD wDtrmValtage)

Parameter Description:

ch	Channel Number
wDtrmValtage	Threshold voltage (Volt), range of value:5~48

Return Value:

-1	Call failed
0	Successful

Function Description:

Sets the threshold voltage to detect the pickup and hangup operations on the analog trunk recording channel.

For more details, refer to 'Change in Analog Phone Line Voltage' in Chapter 1.

Note:

- This function is only applicable to recording channels (including analog trunk recording module and microphone module) of ATP Series boards.
- The parameter set by this function can also be set by the configuration item IsHangupDtrmVoltage, with the default value of 26V.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmGetDtrmLineVoltage

2.17.2.2 SsmGetDtrmLineVoltage

Obtains the threshold voltage detecting the pickup/hangup operations on the analog trunk recording channel.

Format:

int SsmGetDtrmLineVoltage(int ch)

Parameter Description:

ch	Channel Number

Return Value:

-1	Call failed
≥0	Returns the threshold voltage (Volt) used to detect the pickup/hangup operations on the channel

Function Description:

Obtains the threshold voltage detecting the pickup/hangup operations on the analog trunk recording channel.

Note:

- This function is only applicable to recording channels (including analog trunk recording module and microphone module) of ATP Series boards.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmSetDtrmLineVoltage

2.17.3 Obtaining Result of Voltage Detector

2.17.3.1 SsmGetLineVoltage

Obtains the line voltage.

Format:

int SsmGetLineVoltage(int ch)

Parameter Description:

ch	Channel Number

Return Value:

-1	Call failed
≥0	Returns the line voltage (Volt)

Function Description:

Obtains the line voltage. It can also be obtained via the event E_CHG_LineVoltage. For more details, refer to Change in Analog Phone Line Voltage in Chapter 1.

Note:

- This function is only applicable to the recording channel of ATP Series boards and the analog trunk channel of SHT Series boards;
- It's normal that the voltage on the analog trunk may vary in the range of 1V-2V.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function:

2.17.4 Polarity Reversal Detector Functions

2.17.4.1 SsmQueryOpPolarRvrs

Queries if a specific channel supports the detection of polarity reversal.

Format:

int WINAPI SsmQueryOpPolarRvrs(int ch)

Parameter Description:

ch	Channel number

Return Value:

-1	Call failed
0	The specified channel does not support the detection of polarity reversal
1	The specified channel supports the detection of polarity reversal

Function Description:

Queries if a specific channel supports the detection of polarity reversal. For more information, refer to 'Change in Analog Phone Line Voltage' in Chapter 1.

Note:

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function:

2.17.4.2 SsmGetPolarRvrsCount

Obtains the count of the detected polarity reversals on the channel after pickup.

Format:

int SsmGetPolarRvrsCount(int ch)

Parameter Description:

ch	Channel Number

Return Value:

-1	Call failed
≥0	The count of polarity reversals

Function Description:

Obtains the count of the detected polarity reversals on the channel after pickup. For more details, refer to 'Change in Analog Phone Line Voltage' in Chapter 1.

Note:

- This function is only applicable to the channels featured with polarity reversal detection;
- The event E_CHG_PolarRvrsCount can implement the same feature.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Event:E_CHG_PolarRvrsCount

2.17.5 Ringing Current Detector Functions

2.17.5.1 SsmGetRingCount

Obtains the count of the detected ringing current signals.

Format:

int SsmGetRingCount(int ch)

Parameter Description:

ch	Channel Number

Return Value:

-1	Call failed
≥0	Count of rings

Function Description:

Obtains the count of the detected ringing current signals. For more details, refer to 'Ringing Current on Analog Phone Line' in Chapter 1.

Note:

- The event E_CHG_RingCount can implement the same features. It's recommended to use this event to increase the portability of the application;
- This function is only applicable to the analog trunk recording channel of ATP Series boards and the analog trunk channel of SHT Series boards.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmClearRingCount, SsmGetChState

2.17.5.2 SsmGetCallBackRingCount

Obtains the count of the detected ringback current signals.

Format:

int SsmGetCallBackRingCount(int ch)

Parameter Description:

ch	Channel Number

Return Value:

-1	Call failed
≥0	Count of rings

Function Description:

Obtains the count of the detected ringback current signals.

Note:

- The event E_CHG_CallBackRingCount can implement the same features. It's recommended to use this event to increase the portability of the application;
- This function is only applicable to the analog trunk recording channel of ATP Series boards and the analog trunk channel of SHT Series boards.

Related Information:

Driver version	SynCTI Ver. 5.3.0.4 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmClearRingCount, SsmGetChState

2.17.5.3 SsmClearRingCount

Clears the count of the ringing current signals stored in the driver.

Format:

int SsmClearRingCount (int ch)

Parameter Description:

ch	Channel Number

Return Value:

-1	Call failed
0	Successful

Function Description:

Clears the count of the ringing current signals stored in the driver. For more details, refer to 'Ringing Current on Analog Phone Line' in Chapter 1.

Note:

- This function is only applicable to the analog trunk recording channel of ATP Series boards and the analog trunk channel of SHT Series boards.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmGetRingCount

2.17.5.4 SsmGetRingFlag

Obtains the flag indicating the ringing status.

Format:

int SsmGetRingFlag (int ch)

Parameter Description:

ch	Channel number

Return Value:

0	Not ringing
1	Ringing

Function Description:

Obtains the flag indicating the ringing status. For more information, refer to 'Ringing Current on Analog Phone Line' in Chapter 1.

Note:

- This function is only applicable to the analog trunk recording channel of ATP Series boards and the analog trunk channel of SHT Series boards.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function:

2.18 Digital Trunk Functions

2.18.1 SsmGetMaxPcm

Obtains the total amount of the digital trunks in the application.

Format:

int SsmGetMaxPcm()

Parameter Description:

Return Value:

-1	Call failed
≥0	The total amount of the digital trunks in the system, which is set by the configuration item TotalPcm in the configuration section [PcmInfo]

Function Description:

Obtains the total amount of the digital trunks in the application.

Note:

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmGetPcmInfo

2.18.2 SsmGetPcmInfo

Obtains the operating parameters of the digital trunk interface unit (LIU).

Format:

int SsmGetPcmInfo(int nPcmNo, int* pnSSxMode, int* pnBoardId, int* pnBoardPcmNo, int* pnUsePcmTS16, PDWORD pdwRcvrMode, PDWORD pdwEnableAutoCall, PDWORD pdwAutoCallDirection)

Parameter Description:

nPcmNo	The logical number of the digital trunk, range of value: 0~N-1, N is the set value of the configuration item TotalPcm in the configuration section [PcmInfo], it can be obtained by the function SsmGetMaxPcm
pnBoardId	The ID of the board on which the digital trunk is located
pnBoardPcmNo	The internal number of the digital trunk on the board
pnSSxMode	Returns the signaling mode adopted by the PCM 0: ISDN(User side) 1: SS1 2: ISDN(Network side) 7: SS7
pnUsePcmTS16	Returns whether the 16th time slot of the PCM is used as voice channel. 1: Yes 0: No
PdwRcvrMode	Returns the operating mode of the Digital Trunk Interface Unit (LIU), it includes 32 bits, each bit is defined as follows: Bit1,Bit0: clock setting, range of value: 00: line-synchronization mode, master clock

	01: Free-run mode, master clock 10: slave clock 11: reserved Bit2: type of connection line: 0:120Ω twisted pair cable 1: 75Ω coaxial cable Bit3: whether the LIU needs to perform the jitter-cancellation to 16th time slot 0: not to perform the jitter-cancellation (default) 1: perform the jitter-cancellation for a duration of 14ms Bit4: sets the Alternate Digit Inversion (ADI) feature of the voice channel 0: turned off (default) 1: turned on Bit6,Bit5: sets the gain of the input-equalizer of LIU 00: automatically controlled by LIU (default) 01: 0 DB 10:12 DB 11: 6 DB Bit7: reserved Bit15~Bit8: sets the control word of LIU for jitter-cancellation Bit31~Bit16: reserved
pdwEnableAutoCall	Returns the auto-connection control word (32 bits) of the 32 time slots. The highest bit (bit31) corresponds to time slot 0; the lowest bit (bit0) corresponds to time slot 31. If the bit value equals 1, auto-connection is allowed; if the bit value equals 0, auto-connection is forbidden
pdwAutoCallDirection	Returns the auto-connection direction of the 32 time slots. Each bit corresponds to one time slot, the highest bit (bit31) corresponds to time slot 0, and the lowest bit (bit0) corresponds to time slot 31. If the bit value equals 0, it indicates that only incoming call is permitted, if the bit value equals 1, only outgoing call is allowed

Return Value:

-1	Call failed
0	Successful

Function Description:

Obtains the operating parameters of the digital trunk interface unit (LIU).

Note:

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmGetPcmLinkStatus

2.18.3 SsmGetPcmLinkStatus

Obtains the synchronization status of the digital trunk.

Format:

int SsmGetPcmLinkStatus(int nPcmNo, PWORD pwPcmLinkStatus)

Parameter Description:

nPcmNo	The logical number of the digital trunk, range of value: 0~N-1. N is the set value of the configuration item TotalPcm in the section of [PcmInfo], it can be obtained by the function SsmGetMaxPcm
pwPcmLinkStatus	Returns the synchronization status of the digital trunk. It includes 16 bits and bit 0 is the lowest valid bit. If the bit value is equal to 0, it indicates that the synchronization

	status is normal; If the bit value is equal to 1: For boards of the following models: SHD-30A-CT/PCI/SS1, SHD-30A-CT/PCI/ISDN, SHD-30A-CT/PCI/SS7, SHD-60A-CT/PCI/SS1, SHD-60A-CT/PCI/ISDN, SHD-60A-CT/PCI/SS7, SHD-120A-CT/PCI/SS1, SHD-120A-CT/PCI/ISDN, SHD-120A-CT/PCI/SS7, SHD-30B-CT/PCI/SS7/FAX, SHD-60B-CT/PCI/SS7/FAX, SHD-30A-CT/cPCI/SS7, SHD-60A-CT/cPCI/SS7, SHD-120A-CT/cPCI/SS7, SHD-30B-CT/cPCI/SS7/FAX, SHD-60B-CT/cPCI/SS7/FAX, SHD-30A-CT/PCI/FJ, SHD-60A-CT/PCI/FJ bit0=1: basic frame synchronization loss bit1=1: the duration of the basic frame synchronization loss exceeds 100ms bit2=1: multi-frame synchronization loss (SS1 only) bit3=1: CRC multi-frame synchronization loss bit4=1: remote alarm indication bit5=1: signal alarm indication bit6=1: all-ones alarm signal of time slot 16 (SS1 only) bit7=1: signal loss bit8=1: auxiliary alarm (SS1 only) bit9=1: alarm signal of multi-frame asychronization (SS1 only) Other bits: reserved, all remain 0 For boards of the following models: SHD-60B-CT/PCI/FJ, DTP-30C/PCI, DTP-30C/PCI+, DTP-60C/PCI, DTP-60C/PCI+, DTP-120C/PCI, DTP-120C/PCI+, DTP-30C/PCIe, DTP-30C/PCIe+, DTP-60C/PCIe, DTP-60C/PCIe+, DTP-120C/PCIe, DTP-120C/PCIe+, SHD-30D-CT/PCI, SHD-60D-CT/PCI, SHD-120D-CT/PCI, SHD-120D-CT/PCI/EC, SHD-120D-CT/PCI/CAS, SHD-240D-CT/PCI , SHD-240D-CT/PCI/EC, SHD-240D-CT/PCI/CAS, SHD-30E-CT/PCI, SHD-30E-CT/PCI/FAX, SHD-30E-CT/PCI/EC, SHD-60E-CT/PCI, SHD-60E-CT/PCI/FAX, SHD-60E-CT/PCI/EC, SHD-120E-CT/PCI, SHD-120E-CT/PCI/FAX, SHD-120E-CT/PCI/EC, SHD-240E-CT/PCI, SHD-240E-CT/PCI/FAX, SHD-240E-CT/PCI/EC, SHD-30E-CT/PCIe, SHD-30E-CT/PCIe/FAX, SHD-30E-CT/PCIe/EC, SHD-60E-CT/PCIe, SHD-60E-CT/PCIe/FAX, SHD-60E-CT/PCIe/EC, SHD-120E-CT/PCIe, SHD-120E-CT/PCIe/FAX, SHD-120E-CT/PCIe/EC, SHD-240E-CT/PCIe, SHD-240E-CT/PCIe/FAX, SHD-240E-CT/PCIe/EC, SHD-240E-CT/PCIe/VAR bit0=1: basic frame synchronization loss bit1=1: duration of the basic frame synchronization loss exceeds 100ms bit2=1: CAS re-synchronization (SS1and E1) bit3=1: CRC re-synchronization (E1 only) bit4=1: remote alarm indication bit5=1: signal alarm indication bit6=1: all-ones alarm signal of time slot 16 (SS1 only) bit7=1: signal loss bit9=1: MF alarm from the remote end (SS1 and E1) bit10=1: open circuit bit11=1: short circuit Other bits: reserved, all remain 0 For boards of the following models: SHD-480A-CT/cPCI, SHD-480S-CT/cPCI, SHD-240S-CT/cPCI, SHD-240A-CT/cPCI bit1=1: FAS synchronization active bit2=1: CAS MF synchronization active bit3=1: CRC4 MF synchronization active bit8=1: synchronization loss bit9=1: carrier loss bit10=1: remote alarm

	Other bits: reserved, all remain 0 For boards of the following models: SHD-30C-CT/PCI, SHD-60C-CT/PCI, SHD-30C-CT/PCI/FAX, SHD-60C-CT/PCI/FAX, SHD-30B-CT/PCI/FJ bit1=1: asynchronization time exceeds 120ms Other bits: reserved, all remain 0

Return Value:

-1	Call failed
0	Successful

Function Description:

Obtains the synchronization status of the digital trunk.

Note:

- For SS1, only some PBXes support multi-frame CRC check, so, it doesn't always mean the line synchronization fails when the CRC multi-frame synchronization loss occurs.
- For SS7, ISDN signaling, the 16th time slot is commonly used to transmit the Common Channel Signaling (CCS), having nothing to do with multi-frame or CRC multi-frame. Therefore, it can work normally once the synchronization status of the basic frame is normal (both bit0 and bit1 are 0). In addition, both SS7 and ISDN signaling are digital signaling with the capability of fault tolerance. It is allowed that the basic frame synchronization has jitters in a small range. Hence, bit0's changing from 0 to 1 is regarded as the alarm condition when the line synchronization status is used for alarming.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmGetPcmInfo

2.18.4 SsmPcmTsToCh

Queries the corresponding channel number according to the PCM logical number and the time slot number.

Format:

int SsmPcmTsToCh(int nLocalPcmNo, int nTs)

Parameter Description:

nLocalPcmNo	The logical number of the digital trunk. For more information about digital trunk, refer to 'Basic Concepts' in Chapter 1.
nTs	Time slot number, range of value : 1~31

Return Value:

-1	Call failed
≥0	The logical number of the corresponding channel

Function Description:

Queries the corresponding channel number according to the PCM logical number and time slot number.

Note:

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmChToPcmTs

2.18.5 SsmChToPcmTs

Queries the corresponding PCM logical number and time slot number according to the channel number.

Format:

int SsmChToPcmTs(int ch, int* pnLocalPcmNo, int* pnTs)

Parameter Description:

ch	Channel Number
pnLocalPcmNo	The logical number of the PCM on which Channel ch is located
pnTs	The number of the time slot on which Channel ch is located

Return Value:

-1	Call failed
≥0	Returns the value in the parameter pnLocalPcmNo

Function Description:

Queries the corresponding PCM logical number and time slot number according to the channel number.

Note:

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmPcmTsToCh

2.18.6 SsmSetPcmClockMode

Sets the clock mode of the digital trunk.

Format:

int SsmSetPcmClockMode(int nPcmNo, int nClockMode)

Parameter Description:

nPcmNo	The logical number of the digital trunk, range of value: 0~N-1 (N is the set value of the configuration item TotalPcm, it can be obtained by the function call SsmGetMaxPcm)
nClockMode	Clock mode, range of value: 0: master clock, line-synchronization mode (the board must be the master board) 1: master clock, free-run mode (the board must be the master board) 2: slave clock

Return Value:

-1	Call failed
0	Successful

Function Description:

Sets the clock mode of the digital trunk. For more information, refer to 'System Clock Configuration' in Chapter 1.

Note:

- This function is only applicable to SHD, DTP Series boards;
- The configuration item PcmClockMode can implement the same features.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h

Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmGetPcmInfo

2.18.7 SsmGetInboundLinkSet

During incoming call, obtains the information of the link set which is used by TUP/ISUP messages from the remote PBX.

Format:

int SsmGetInboundLinkSet(int ch, LPWORD pwLinkSetNo, LPSTR pszOpc, LPSTR pszDpc)

Parameter Description:

ch	Channel number
pwLinkSetNo	Link set number
pszOpc	OPC
pszDpc	DPC

Return Value:

1	Successful
-1	Call failed

Function Description:

During incoming call, obtains the information of the link set which is used by TUP/ISUP messages from the remote PBX, including the link set number, DPC and OPC.

Note:

- This function is only applicable to TUP/ISUP channels.

Related Information:

Driver version	SynCTI Ver. 4.7.1.2 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function:

2.18.8 SsmGetCbChStatus

Obtains the connection state of the large-capacity channel bank.

Format:

int WINAPI SsmGetCbChStatus(int ch, PWORD pwCBLinkStatus)

Parameter Description:

ch	Channel number
pwCBLinkStatus	Connection state of the channel bank

Return Value:

0	Call successful
-1	Call failed

Function Description:

Obtains the connection state of the large-capacity channel bank. pwCBLinkStatus=0 means the line is connected, while pwCBLinkStatus=1 means the line is disconnected.

Note:

- This function is only applicable to detecting the connection state of large-capacity channel bank.

- The event E_CHG_CbChStatus can implement the same feature.

Related Information:

Driver version	SynCTI Ver. 5.0.3.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function:

Related Event: E_CHG_CbChStatus

2.18.9 SsmSetPcmPowerDown

Sets the working mode of the digital trunk.

Format:

int WINAPI SsmSetPcmPowerDown(int nPcmNo, int nPowerMode)

Parameter Description:

nPcmNo	The logical number of the digital trunk, range of value: 0~N-1 (N is the set value of the configuration item TotalPcm, it can be obtained by the function call of SsmGetMaxPcm)
nPowerMode	Working mode, range of value: 0: normal working mode 1: idle mode (It is similar to the state that the PCM is unconnected)

Return Value:

-1	Call failed
0	Successful

Function Description:

Sets the working mode of the designated PCM. It can be set to normal working mode or idle mode.

Note:

- This function is only applicable to SHD Series D-type and E-type boards.

Related Information:

Driver version	SynCTI Ver. 5.3.0.1 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

2.19 Functions for SHT Series Boards (CTI Series)

2.19.1 Setting Operating Mode of Composite Module

2.19.1.1 SsmSetUnimoduleState

Sets whether to enable the direct connection between the analog trunk channel and station channel on the composite module.

Format:

int SsmSetUnimoduleState(int ch,int nState)

Parameter Description:

ch	Channel number of the analog trunk on the composite module
nState	=0: connect the analog trunk channel directly with the station channel

	=1: There is no connection between the analog trunk channel and the station channel, namely they are equivalent to two independently controlled channels

Return Value:

-1	Failed
0	Successful

Function Description:

Sets whether to enable the direct connection between the analog trunk channel and station channel on the composite module.

Note:

- The configuration item UnimoduleState can implement the same features.

Related Information:

Driver version	SynCTI Ver. 3.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function:

2.19.2 Functions for Analog Trunk Channel

2.19.2.1 Generating Flash Signal on Analog Phone Line

2.19.2.1.1SsmTxFlash

Generates a flash signal on the analog trunk channel (equivalent to a rapid clap on the hook switch).

Format:

int SsmTxFlash(int ch, WORD time)

Parameter Description:

ch	Channel Number
time	Sets the duration (ms) of the flash, range of value: 32~1000, usually it's set to 500ms

Return Value:

-1	Call failed. The failure reason can be acquired via the function SsmGetLastErrMsg
0	Successful

Function Description:

Generates a flash signal on the analog trunk channel (equivalent to a rapid clap on the hook switch).

When the application finishes generating a flash signal, it throws out the event of E_PROC_SendFlash to the application. The operating status can be obtained by the function of SsmChkTxFlash.

Note:

- This function supports the analog trunk channels on SHT Series boards and the SS1 LineSide protocol.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmChkTxFlash

2.19.2.1.2SsmSetTxFlashCharTime

Sets the duration of the flash signal generated on the analog trunk via the function call of SsmTxDTMF.

Format:

int SsmSetTxFlashCharTime(int ch,WORD time)

Parameter Description:

ch	Channel Number
Time	The duration (ms) of the flash signal, range of value: 32~1000, with the default value of 500

Return Value:

0	Successful
-1	Call failed. The failure reason can be acquired via the function SsmGetLastErrMsg

Function Description:

The function call of SsmTxDTMF with parameter of the character '!' can also generate a flash signal on the analog trunk. This function sets the duration of the flash signal generated by using the character '!'.

Note:

- The configuration item DefaultTxFlashTime can implement the same features, with the default value of 500ms.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmGetTxFlashCharTime, SsmTxDTMF

2.19.2.1.3SsmGetTxFlashCharTime

Obtains the duration of the flash signal generated on the analog trunk via the function call of SsmTxDTMF.

Format:

int SsmGetTxFlashCharTime(int ch)

Parameter Description:

ch	Channel Number

Return Value:

>0	The duration (ms) of the flash signal
-1	Call failed. The failure reason can be acquired via the function SsmGetLastErrMsg

Function Description:

Obtains the duration of the flash signal generated on the analog trunk by the function call of SsmTxDTMF.

Note: none

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h

Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmSetTxFlashCharTime

2.19.2.1.4SsmChkTxFlash

Queries if the operation of generating a flash signal on the analog trunk has been completed.

Format:

int SsmChkTxFlash(int ch)

Parameter Description:

ch	Channel Number

Return Value:

-1	Error occurs
0	The operation of generating a flash signal has been completed
1	The operation of generating a flash signal is under progress

Function Description:

Queries if the operation of generating a flash signal on the analog trunk has been completed.

Note:

- Only the analog trunk channel supports the generation of a flash signal.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmTxFlash

2.19.2.1.5SsmTxFlashEx

Generates a flash signal on the analog trunk (equivalent to a rapid clap on the hook switch) and sets the channel state after the flash.

Format:

int WINAPI SsmTxFlashEx(int ch, WORD time, int nChState, BOOL bIgnoreDlTn)

Parameter Description:

ch	Channel number
time	The duration (ms) of the flash signal, range of value: 32~1000, with the default value of 500.
nChState	The channel state after the flash, range of value: S_CALL_PICKUPED(1), S_CALL_TALKING(3)
bIgnoreDlTn	Sets whether to ignore the dial tone after the flash, range of value: FALSE (not to ignore), TRUE(ignore).

Return Value:

-1	Call failed. The failure reason can be acquired via the function SsmGetLastErrMsg.
0	Call successful

Function Description:

Generates on a specified channel a flash signal (equivalent to a rapid clap on the hook switch) within the duration set by the parameter 'time', and sets the channel state after the flash.

Note:

- This function only supports analog trunk channels on the SHT Series boards.

Related Information:

Driver version	SynCTI Ver. 4.8.0.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmTxFlash, SsmChkTxFlash

2.19.2.2 Forced Transition of Channel State

2.19.2.2.1SsmSetChState

Forces the analog trunk channel to transfer to the talking state.

Format:

int SsmSetChState (int ch, int nState)

Parameter Description:

ch	Channel Number
nState	The new state of the channel, it must be set to 3

Return Value:

-1	Call failed
0	Successful

Function Description:

When the analog trunk channel accidentally enters the pending state, this function can force the analog trunk channel to transfer to the talking state.

Note:

- This function is only applicable to the analog trunk channels of SHT Series boards. This function is used only in some special occasions, for example, when the application finds that the channel enters the pending state abnormally, and it wants to get the channel back to the talking state.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function:

2.19.2.3 Detecting Remote Pick-up during Outgoing Call

2.19.2.3.1SsmStartPickupAnalyze

Starts the 'Enhanced Remote Pickup Detector'.

Format:

int SsmStartPickupAnalyze(int ch)

Parameter Description:

ch	Channel number

Return Value:

-1	Failed
0	Successful

Function Description:

Starts the 'Enhanced Remote Pickup Detector'. For more information, refer to 'Enhanced Remote Pickup Detector' in Chapter 1.

Note:

- This function is only applicable to analog trunk channels of SHT Series boards.

Related Information:

Driver version	SynCTI Ver. 3.5.2.4 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmGetPickup

2.19.2.3.2SsmSetCalleeHookDetectP

Sets the operating parameters of the 'Enhanced Remote Pickup Detector' on the analog trunk channel.

Format:

int SsmSetCalleeHookDetectP(int ch, WORD wMulti, WORD wValidTime)

Parameter Description:

ch	Channel number
wMulti	Sensitivity, for more information, refer to the description of the configuration item HookEngyConfigMulti
wValidTime	Minimum duration, for more information, refer to the description of the configuration item HookValidEngyCnt

Return Value:

-1	Call failed
0	Successful

Function Description:

Sets the operating parameters of the 'Enhanced Remote Pickup Detector' on the analog trunk channel. For more information, refer to 'Enhanced Remote Pickup Detector' in Chapter 1.

Note:

- This function is only applicable to the analog trunk channels of SHT Series boards.

Related Information:

Driver version	SynCTI Ver. 3.5.2.4 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmStartPickupAnalyze

2.19.2.3.3SsmGetPickup

Queries the detected result of the 'Enhanced Remote Pickup Detector' on the analog trunk channel.

Format:

int SsmGetPickup (int ch)

Parameter Description:

ch	Channel number

Return Value:

-1	Failed
0	The called party didn't pick up
1	The called party has picked up

Function Description:

Queries the detected result of the 'Enhanced Remote Pickup Detector' on the analog trunk channel. For more information, refer to 'Enhanced Remote Pickup Detector' in Chapter 1.

Note:

- The same features can be implemented by the event of E_SYS_RemotePickup (recommended);
- This function is only applicable to the analog trunk channels of SHT Series boards.

Related Information:

Driver version	SynCTI Ver. 3.5.2.4 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmStartPickupAnalyze, SsmSetCalleeHookDetectP

2.19.2.4 Checking Pickup Status

2.19.2.4.1SsmCheckActualPickup

Queries if the operation of pickup has been completed.

Format:

int SsmCheckActualPickup(int ch)

Parameter Description:

ch	Channel Number

Return Value:

-1	Failed
0	The specified channel is in the hangup state
1	The specified channel is in the pickup state

Function Description:

Queries if the operation of pickup has been completed. For more information, refer to the description of the function SsmPickup.

Note:

- This function is only applicable to the analog trunk channels of SHT Series boards.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmPickup

2.19.2.5 Checking Analog-trunk-to-recording Channel

2.19.2.5.1SsmGetIsAnalogToRec

Checks whether a specific channel is a recording channel transformed from the analog trunk channel.

Format:

int WINAPI SsmGetIsAnalogToRec(int ch);

Parameter Description:

ch	Channel Number

Return Value:

-1	Failed
0	The specified channel is neither an analog trunk channel nor a recording channel transformed from the analog trunk channel
1	The specified channel is a recording channel transformed from the analog trunk channel

Function Description:

Checks whether a specific channel is a recording channel transformed from the analog trunk channel. For more information, refer to the configuration item SetAnalogChToRecCh.

Note:

- This function is only applicable to analog trunk channels of SHT Series boards.

Related Information:

Driver version	SynCTI Ver. 5.3.1.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Configuration Item: SetAnalogChToRecCh

2.19.3 Functions for Station Channel

2.19.3.1 Sending Ringing Signals to Phone

2.19.3.1.1SsmStartRing

Refer to SsmStartRingWithCIDStr

2.19.3.1.2SsmStartRingWithCIDStr

Sends ringing signals to the phone. SsmStartRing only generates the ringing signals, SsmStartRingWithCIDStr can also send the caller ID information to the phone.

Format:

int SsmStartRing(int ch)

int SsmStartRingWithCIDStr(int ch, LPSTR pCIDBuf, DWORD dwCIDLen, DWORD dwDelayTime)

Parameter Description:

ch	Channel Number
pCIDBuf	The first address of the buffer area storing the caller ID information. e.g. '0018657188861158'
dwCIDLen	The number of characters of the caller ID in pCIDBuf, range of value: 1~20
dwDelayTime	The time interval (ms) calculated starting from the negative edge of the first ringing signal to the emergence of the first FSK data on the line, range of value: 500~1500 Note: The FSK caller ID is transmitted between the 1st and 2nd ringing signal

Return Value:

-1	Call failed, the failure reason can be obtained by the function SsmGetLastErrMsg
0	Successful

Function Description:

Sends ringing signals to the phone.

The function SsmStopRing can be used to stop sending ringing signals to the phone at anytime.

Note:

- The function of SsmStartRing is applicable to both the station channel and magnet channel; the function of SsmStartRingWithCIDStr is only applicable to the station channel;
- During the execution of the function SsmStartRingWithCIDStr, none of the voice playing functions can be invoked.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmStopRing

2.19.3.1.3fPcm_ConvertFskCID

Converts the caller ID information to FSK audio signals which are fit for the transmission on the analog phone line.

Format:

DWORD fPcm_ConvertFskCID(LPSTR pFskStream, int nFskLen, LPSTR pszCIDNumber, LPSTR pszTime, LPSTR pszName, int nMode)

Parameter Description:

pFskStream	The first address of the buffer area storing the modulated FSK bit stream and the output parameter. The storage space is allocated by the application and should be no less than 500 bytes.
nFskLen	The size of the buffer area (bytes) pointed by pFskStream
pszCIDNumber	The first address of the buffer area storing the caller ID, the number of characters should be in the range of 4~14
pszTime	The first address of the buffer area storing the time information. Its length is 8 bytes and the format is 'MMDDhhmm'. MM is the month of the year; DD is the day of the month; hh is the hour of the day; mm is the minute of the hour. e.g., '12041224' denotes 12:24, 4th, December. If pszTime is NULL, the driver will use the system time as the time information
pszName	The first address of the buffer area storing the additional information, the length is 4~14 bytes, it can be null Note: most of the phones can't display the additional information
nMode	The frame format of the FSK caller ID, 0 denotes multi-frame, 1 denotes single-frame

Return Value:

0	Call failed
>0	The total bit number after modulation

Function Description:

Converts the caller ID information to FSK audio signals which are fit for the transmission on the analog phone line.

Note:

Related Information:

Driver version	SynCTI Ver.4.5.2.5 or above
Header	ShPcmApi.h
Library	ShPcmHandle.lib
DLL	ShPcmHandle.dll

Related Function:

2.19.3.1.4SsmStopRing

Stops sending ringing signals to magnet channels or station channels.

Format:

int SsmStopRing(int ch)

Parameter Description:

ch	Channel number

Return Value:

-1	Call failed, the failure reason can be obtained by the function SsmGetLastErrMsg

0	Successful

Function Description:

Stops sending ringing signals to magnet channels or station channels. The operation of ringing signal transmission can be initiated by the function SsmStartRing or SsmStartRingWithCIDStr.

Note:

- This function is only applicable to the station channels and magnet channels of SHT Series boards;
- After this function is invoked, the ringing signal counter in the driver will be cleared.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmStartRing, SsmStartRingWithCIDStr

2.19.3.1.5SsmCheckSendRing

Obtains the operating status of the ringing current generator of station channels and magnet channels.

Format:

int SsmCheckSendRing(int ch,int * pnCnt)

Parameter Description:

ch	Channel Number
pnCnt	Returns the value of the ringing signal counter in the driver

Return Value:

-1	Call failed
0	No ringing signal transmission on the channel
1	The channel is transmitting ringing signals

Function Description:

Obtains the operating status of the ringing current generator of station channels and magnet channels.

Note:

- This function is only applicable to the station channels and magnet channels of SHT Series boards.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmStartRing, SsmStartRingWithCIDStr

2.19.3.1.6SsmSetRingPeriod

Sets the durations of the ringing current generator at on and off states on station channels.

Format:

int WINAPI SsmSetRingPeriod(int ch,WORD wHigh,WORD wLow)

Parameter Description:

ch	Channel Number
wHigh	The duration (ms) of the ringing signal at on state, with the minimum value of 16ms
wLow	The duration (ms) of the ringing signal at off state, with the minimum value of 16ms

Return Value:

-1	Call failed
0	Call successful

Function Description:

Sets the durations of the ringing current generator at on and off states on station channels.

Note:

- This function is applicable to the station channels of SHT Series boards and large-capacity channel banks.

Related Information:

Driver version	SynCTI Ver. 5.1.0.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmStartRing, SsmStartRingWithCIDStr

2.19.3.2 Auto Dial Tone Sending upon Detection of Pickup

2.19.3.2.1SsmSetASDT

Sets whether to send the dial tone automatically by the driver upon the detection of pick-up operation on the phone connected with the station channel.

Format:

int SsmSetASDT(int ch, BOOL bEnAutoSendDialTone)

Parameter Description:

ch	Channel number
bEnAutoSendDialTone	=TRUE: Enable =FALSE: Disable

Return Value:

-1	Call failed, the failure reason can be obtained by the function SsmGetLastErrMsg
0	Successful

Function Description:

Sets whether to send the dial tone automatically by the driver upon the detection of pick-up operation on the phone connected with the station channel.

Note:

- This function is only applicable to station channels of SHT Series boards.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmGetASDT

2.19.3.2.2SsmGetASDT

Obtains the flag indicating whether the driver automatically sends the dial tone upon the detection of pick-up operation on the phone connected with the station channel.

Format:

int SsmGetASDT(int ch)

Parameter Description:

ch	Channel Number

Return Value:

-1	Call failed
0	'Disable' state
1	'Enable' state

Function Description:

Obtains the flag indicating whether the driver automatically sends the dial tone upon the detection of pick-up operation on the phone connected with the station channel.

Note:

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmSetASDT

2.19.3.3 Detecting Flash Signal

2.19.3.3.1SsmSetLocalFlashTime

Sets the maximum duration for judging the flash signal.

Format:

int SsmSetLocalFlashTime(int nFlashTime)

Parameter Description:

nFlashTime	Maximum duration (ms) of the flash signal, range of value: 32~2000

Return Value:

-1	Call failed, the failure reason can be obtained by the function SsmGetLastErrMsg
0	Successful

Function Description:

Sets the maximum duration for judging the flash signal. For more information, refer to 'Flash Signal Detection' in Chapter 1.

Note:

- This function is only applicable to station channels.

- The configuration item MaxLocalFlashTime can implement the same features, with the default value of 700ms.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function:

2.19.3.3.2SsmGetFlashCount

Obtains the value of the flash signal counter in the driver on the station channel or the recording channel.

Format:

int SsmGetFlashCount(int ch)

Parameter Description:

ch	Channel Number

Return Value:

-1	Call failed
≥0	The value of the flash signal counter

Function Description:

Obtains the value of the flash signal counter in the driver on the station channel or the recording channel.

Note:

- This function is only applicable to station channels and recording channels;
- When the station channel enters the pickup or hangup state, the driver will automatically reset the flash signal counter.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmClearFlashCount

2.19.3.3.3SsmClearFlashCount

Resets the flash signal counter on the designated station channel or recording channel.

Format:

int SsmClearFlashCount (int ch)

Parameter Description:

ch	Channel Number

Return Value:

-1	Call failed
0	Successful

Function Description:

Resets the flash signal counter on the designated station channel or recording channel.

Note:

- This function is only applicable to station channels and recording channels;
- When the station channel enters the pickup or hangup state, the driver will automatically reset the flash signal counter.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmGetFlashCount

2.19.3.4 Pickup/Hangup Detection

2.19.3.4.1SsmGetHookState

Obtains the on-/off-hook state of the station channel or the EM Control channel.

Format:

int SsmGetHookState(int ch)

Parameter Description:

ch	Channel Number

Return Value:

-1	Call failed
0	On-hook state
1	Off-hook state

Function Description:

Obtains the on-/off-hook state of the station channel or the EM Control channel. The same features can be implemented by the event of E_CHG_HookState.

Note:

- This function is only applicable to station channels and EM Control channels of SHT Series boards;
- When the station channel enters the ‘off-hook’ state or ‘on-hook’ state, the driver will automatically reset the flash signal counter.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Event: E_CHG_HookState

2.19.3.5 Generating Polarity Reversal Signals on Phone Line

2.19.3.5.1SsmSetPolarState

Sets the polarity of the feed voltage of the analog phone line connected with the station channel.

Format:

int SsmSetPolarState(int ch, int nPolar)

Parameter Description:

ch	Channel number
nPolar	The polarity of the channel to be set, range of value: 0 or 1

Return Value:

-1	Call failed
0	Successful

Function Description:

Sets the polarity of the feed voltage of the analog phone line connected with the station channel. The current voltage polarity of the channel can be obtained by the function SsmGetPolarState. For more information, refer to 'Generating Polarity Reversal Signal on Phone Line' in Chapter 1.

Note:

- This function is only valid when the configuration item UserSendPolar is set to 1.
- This function is only applicable to station channels of SHT Series boards.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmGetPolarState

Sample Code:

```
int v = SsmGetPolarState(ch);
SsmSetPolarState(ch, ~v);
```

2.19.3.5.2SsmGetPolarState

Obtains the polarity of the feed voltage of the analog phone line connected with the station channel.

Format:

int SsmGetPolarState(int ch)

Parameter Description:

ch	Channel number

Return Value:

-1	Call failed
0,1	Returns the current polarity

Function Description:

Obtains the polarity of the feed voltage of the analog phone line connected with the station channel.

Note:

- This function is only applicable to station channels of SHT Series boards.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmSetPolarState

2.19.4 Remote Pickup Detector Functions (SHT Series Only)

2.19.4.1 Functions for Ordinary Remote Pickup Detector

2.19.4.1.1 Setting Parameters

2.19.4.1.1.1 SsmSetVoiceOnDetermineTime

Sets the minimum duration of voice activities.

Format:

int SsmSetVoiceOnDetermineTime(int ch, WORD wIsVocDtrTime)

Parameter Description:

ch	Channel Number
wIsVocDtrTime	Minimum duration (ms), range of value: 16~1024

Return Value:

-1	Call failed, the failure reason can be obtained by the function SsmGetLastErrMsg
0	Successful

Function Description:

Sets the minimum duration of voice activities. Only when there is continuous voice activity appearing on the line and the duration of it is larger than the preset value of the parameter wIsVocDtrTime, will the driver judge that there is real voice activity. For more information, refer to 'Ordinary Remote Pickup Detector' in Chapter 1.

Note:

- The configuration item VoiceOnDetermineTime can implement the same features;
- This function is only applicable to the analog trunk channels of SHT Series boards.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmGetVoiceOnDetermineTime

2.19.4.1.2 Obtaining Parameters

2.19.4.1.2.1 SsmGetVoiceOnDetermineTime

Obtains the minimum duration of the voice activities.

Format:

int SsmGetVoiceOnDetermineTime(int ch, PWORD pwIsVocDtrTime)

Parameter Description:

ch	Channel Number
pwIsVocDtrTime	Returns the minimum duration (ms)

Return Value:

-1	Call failed, the failure reason can be obtained by the function SsmGetLastErrMsg
0	Successful

Function Description:

Obtains the minimum duration which judges whether there is voice signal on the line.

Note:

- This function is only applicable to the analog trunk channels of SHT Series boards.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmSetVoiceOnDetermineTime

2.20 Advanced Programming API for SHD Series (CTI Series)

2.20.1 Channel Blocking Control (CTI Series)

2.20.1.1 Functions for Blocking Local End

2.20.1.1.1 Blocking Single Channel

2.20.1.1.1.1 SsmBlockLocalCh

Blocks the local channel to prohibit the TUP and ISUP channel from making outgoing calls but still allow them to process incoming calls, or to prohibit the ISDN channel from making both incoming and outgoing calls.

Format:

int SsmBlockLocalCh(int ch)

Parameter Description:

ch	Channel number

Return Value:

-1	Call failed

0	Successful

Function Description:

Blocks the local channel to prohibit the TUP and ISUP channel from making outgoing calls but still allow them to process incoming calls, or to prohibit the ISDN channel from making both incoming and outgoing calls. For more information about channel blocking, refer to 'Channel Blocked and Unblocked' in chapter 1.

This function is applicable to TUP, ISUP and ISDN channels and has an impact on the corresponding state machine. For more detailed description about the TUP, ISUP and ISDN channel state machine, refer to TUP Channel State Machine, ISUP Channel State Machine and ISDN Channel State Machine in Chapter 1 (The event triggered by this function is named as 'Blocking' in the state machine).

Note:

- This function is only applicable to TUP, ISUP and ISDN channels;
- This function can be invoked at anytime. Note that once this function is invoked, the AutoDial task will fail if it is being executed on the TUP or ISUP channel, but won't fail if it is being executed on the ISDN channel.

Related Information:

Driver version	SynCTI Ver. 3.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmUnblockLocalCh, SsmBlockLocalPCM, SsmQueryLocalChBlockState, SsmBlockRemoteCh, SsmBlockRemotePCM

2.20.1.1.1.2 SsmUnblockLocalCh

Unblocks the local channel.

Format:

int SsmUnblockLocalCh(int ch)

Parameter Description:

ch	Channel number

Return Value:

-1	Call failed
0	Successful

Function Description:

This function unblocks the local channel blocked by the function call of SsmBlockLocalCh or SsmBlockLocalPCM.

Note:

- This function is only applicable to the TUP, ISUP and ISDN channels.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmBlockLocalCh, SsmUnblockLocalPCM, SsmQueryLocalChBlockState

2.20.1.1.1.3 SsmQueryLocalChBlockState

Obtains the blocking status (blocked/unblocked) of the local channel.

Format:

int SsmQueryLocalChBlockState(int ch, PDWORD pdwBlockState)

Parameter Description:

ch	Channel Number
pdwBlockState	Returns the blocking status of the local channel. The bit value equals to 1 means 'blocked', and the bit value equals to 0 means 'unblocked'. Below are the meanings of each bit:

Bit 0	Blocked because the application invokes the function SsmBlockLocalCh
Bit 1	Blocked because the remote circuit blocking message BLO is received
Bit 2	Blocked because the remote circuit group blocking message SGB is received
Bit 3	Blocked because the remote circuit group blocking message HGB is received
Bit 4	Blocked because the remote circuit group blocking message MGB is received
Bit 5	Blocked because the application invokes the function SsmBlockLocalPCM
Bit31~Bit6	Reserved

Return Value:

-1	Call failed
0	Successful

Function Description:

Obtains the blocking status (blocked/unblocked) of the local channel and the exact blocking reason

Note:

- This function is only applicable to the TUP channels.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmBlockLocalCh, SsmBlockLocalPCM

2.20.1.1.2 Blocking Whole Digital Trunk

2.20.1.1.2.1 SsmBlockLocalPCM

Blocks the local circuit group (the whole digital trunk).

Format:

int SsmBlockLocalPCM(int nLocalPcmNo)

Parameter Description:

nLocalPcmNo	PCM logical number

Return Value:

-1	Call failed

0	Successful

Function Description:

This function blocks the whole digital trunk. After the digital trunk is blocked, all the local channels included in it are forbidden to make outgoing calls but are still able to process incoming calls. Invoking this function is equivalent to calling the function of SsmBlockLocalCh on all the channels of this PCM simultaneously. For more information, refer to the description of the function SsmBlockLocalCh.

Note:

- This function is only applicable to the TUP, ISUP channels.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmUnblockLocalPCM, SsmQueryLocalPCMBlockState, SsmBlockLocalCh

2.20.1.1.2.2 SsmUnblockLocalPCM

Unblocks the whole local digital trunk.

Format:

int SsmUnblockLocalPCM(int nLocalPcmNo)

Parameter Description:

nLocalPcmNo	PCM logical number

Return Value:

-1	Call failed
0	Successful

Function Description:

This function unblocks the digital trunk blocked by the function call of SsmBlockLocalPCM. Invoking this function is equivalent to calling the function of SsmUnblockLocalCh on all the channels included in the digital trunk. Therefore for more information about this function, refer to the description of the function SsmUnblockLocalCh.

Note:

- This function is only applicable to the TUP, ISUP channels.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmBlockLocalPCM, SsmQueryLocalPCMBlockState, SsmUnblockLocalCh

2.20.1.1.2.3 SsmQueryLocalPCMBlockState

Obtains the local blocking state of the digital trunk.

Format:

int SsmQueryLocalPCMBlockState(int nLocalPcmNo, PDWORD pdwBlockState)

Parameter Description:

nLocalPcmNo	PCM logical number
pdwBlockState	Returns the blocking state of the local channel. The bit value equals to 1 means 'blocked' and the bit value equals to 0 means 'unblocked'. Below are the meanings of the bits: Bit1~Bit0: Reserved Bit 2: Blocked because of the reception of the circuit group blocking message SGB from the remote PBX. Bit 3: Blocked because of the reception of the circuit group blocking message HGB from the remote PBX. Bit 4: Blocked because of the reception of the circuit group blocking message MGB from the remote PBX. Bit 5: Blocked because the application invokes the function SsmBlockLocalPCM Bit31~Bit6: Reserved

Return Value:

-1	Failed
0	Successful

Function Description:

Obtains the local blocking state of the digital trunk

Note:

- This function is only applicable to the TUP channels.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmBlockLocalCh, SsmBlockLocalPCM

2.20.1.2 Functions for Blocking Remote End

2.20.1.2.1 Blocking Single Channel

2.20.1.2.1.1 SsmQueryOpBlockRemoteCh

Queries if the channel supports the blocking of the remote end.

Format:

int SsmQueryOpBlockRemoteCh(int ch)

Parameter Description:

ch	Channel number

Return Value:

-1	Call failed
0	Not support
1	Support

Function Description:

Queries if the channel supports the blocking of the remote end.

Note:

- Only the TUP channel and ISUP channel support to block the remote channel.

Related Information:

Driver version	SynCTI Ver. 3.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function:

2.20.1.2.1.2 SsmBlockRemoteCh

Blocks the remote channel.

Format:

int SsmBlockRemoteCh(int ch)

Parameter Description:

ch	Channel Number

Return Value:

-1	Call failed
0	Successful

Function Description:

Blocks the remote channel in order to forbid the remote PBX to call the local end, but still allow the local end to send the outgoing call towards the remote PBX. The remote channel means the circuit in the remote PBX which has the same CIC number as in the local channel. When the remote channel enters the blocked state, its outgoing call is forbidden, and the local channel is ensured to have no incoming call. The feature of blocking remote channel is normally used for system maintenance.

The following message processing procedure is used for blocking the remote channel:

(1) The driver sends the circuit blocking message BLO to the remote PBX;

(2) After the remote PBX receives the BLO message, it answers with the blocking acknowledgement message BLA;

(3) After receiving the BLA message from the remote PBX, the driver throws out the event of E_CHG_RemoteChBlock to the application. After the above steps are completed, the remote channel blocking is finished.

The function SsmGetRemoteChBlockStatus can be used to check whether the remote channel blocking is finished.

Note:

- Only TUP and ISUP channels support the remote channel blocking. Once this function is invoked, the TUP channel will stay IDLE, while the ISUP channel will stay LOCAL BLOCK but still can call out. Both TUP and ISUP channels can be found via the function SsmSearchIdleCallOutCh.

Related Information:

Driver version	SynCTI Ver. 3.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmUnblockRemoteCh, SsmGetRemoteChBlockStatus

2.20.1.2.1.3 SsmUnblockRemoteCh

Unblocks the remote channel.

Format:

int SsmUnblockRemoteCh(int ch)

Parameter Description:

ch	Channel number

Return Value:

-1	Call failed
0	Successful

Function Description:

Unblocks the remote channel in order to resume the capability of the remote PBX to start the call towards the local end. The following message processing procedure is used for unblocking the remote channel:

(1) The driver sends the circuit unblocking message UBL to the remote PBX;

(2) After the remote PBX receives the UBL message, it answers with the unblocking acknowledgement message UBA;

(3) After receiving the UBA message from the remote PBX, the driver throws out the event of E_CHG_RemoteChBlock to the application. After the above steps are completed, the remote channel unblocking is finished.

The function SsmGetRemoteChBlockStatus can be used to check whether the remote channel unblocking is finished.

Note:

- Only TUP channels and ISUP channels support the remote channel unblocking.

Related Information:

Driver version	SynCTI Ver. 3.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmBlockRemoteCh, SsmGetRemoteChBlockStatus, SsmBlockRemotePCM

2.20.1.2.1.4 SsmGetRemoteChBlockStatus

Queries if the channel on the remote PBX is blocked by the local end.

Format:

int SsmGetRemoteChBlockStatus(int ch)

Parameter Description:

ch	Channel number

Return Value:

-1	Call failed
0	The channel has not been blocked by the local end, or the blocking caused by the local end has been successfully unblocked
1	The remote end is blocked successfully
2	The circuit blocking message has been sent, waiting for the blocking acknowledgement message from the remote PBX
3	The circuit unblocking message has been sent, waiting for the unblocking

	acknowledgement message from the remote PBX

Function Description:

Queries if the channel on the remote PBX is blocked by the local end. The return value can be used to check the executing status of the function SsmBlockRemoteCh or SsmUnblockRemoteCh.

Note:

- Only TUP channels and ISUP channels support to check the remote channel blocking status.

Related Information:

Driver version	SynCTI Ver. 3.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmBlockRemoteCh, SsmUnblockRemoteCh

2.20.1.2.2 Blocking Whole Digital Trunk

2.20.1.2.2.1 SsmBlockRemotePCM

Blocks the remote circuit group.

Format:

int SsmBlockRemotePCM(int nLocalPcmNo, DWORD dwBlockMode)

Parameter Description:

nLocalPcmNo	PCM logical number
dwBlockMode	Choose the circuit group blocking message sent to the remote PBX. Range of value: TUP protocol: 1: send the maintenance oriented group blocking message MGB; 2: send the hardware failure oriented group blocking message HGB; 3: send the software generated group blocking message SGB. ISUP protocol: 0: send the maintenance oriented group blocking message CGB; 1: send the hardware failure oriented group blocking message CGB.

Return Value:

-1	Call failed
0	Successful

Function Description:

Use the circuit group blocking message to block the whole digital trunk of the remote PBX so that all the channels included in this PCM of the remote PBX are not able to initiate outgoing calls but only to process the incoming calls. Invoking this function call is equivalent to calling the function of SsmBlockRemoteCh on all the channels of this PCM simultaneously.

The following message processing procedure is used for blocking the remote circuit group:

(1) According to the settings of the parameter dwBlockMode, the driver sends the specified group blocking message towards the remote PBX;

(2) After the remote PBX receives the group blocking message, it will answer with the message of group blocking acknowledgement (for the TUP protocol, the corresponding blocking acknowledgement

messages to the group blocking messages MGB, HGB and SGB are MBA, HBA and SBA respectively; for the ISUP protocol, the corresponding blocking acknowledgement message to the group blocking message CGB is CGBA);

(3) After receiving the group blocking acknowledgement message, the driver throws out the event of E_CHG_RemotePCMBlock to the application. After the above steps are completed, the remote circuit group blocking is finished.

The function SsmGetRemotePCMBlockStatus can be used to check whether the remote circuit group blocking is finished.

Note:

- This function only supports the TUP and ISUP protocols. Once this function is invoked, the TUP channel will stay IDLE, while the ISUP channel will stay LOCAL BLOCK but still can call out. Both TUP and ISUP channels can be found via the function SsmSearchIdleCallOutCh;
- According to the TUP protocol and the ISUP protocol, the circuit group blocking message won't go into effect until it has been transmitted twice. Therefore the driver will send the circuit group blocking message to the remote PBX twice continuously, and the application need to invoke this function only once.

Related Information:

Driver version	SynCTI Ver. 3.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmUnblockRemotePCM, SsmGetRemotePCMBlockStatus

Related Event: E_CHG_RemotePCMBlock

2.20.1.2.2.2 SsmUnblockRemotePCM

Unblocks the remote circuit group.

Format:

int SsmUnblockRemotePCM(int nLocalPcmNo, DWORD dwUnblockMode)

Parameter Description:

nLocalPcmNo	PCM logic number
dwBlockMode	Selects the circuit group unblocking message sent to the remote PBX. Range of value: TUP protocol: 1: send MGU to unblock the circuit group blocked by the message of MGB; 2: send HGU to unblock the circuit group blocked by the message of HGB; 3: send SGU to unblock the circuit group blocked by the message of SGB. ISUP protocol: 0: send the maintenance oriented circuit group unblocking message CGU; 1: send the hardware failure oriented circuit group unblocking message CGU

Return Value:

-1	Call failed
0	Successful

Function Description:

Unblock the whole digital trunk of the remote PBX so that all the channels included in this PCM of the remote PBX are reenabled to initiate outgoing calls. Invoking this function call is equivalent to calling the function of

SsmUnBlockRemoteCh on all the channels of this digital trunk simultaneously.

The following message processing procedure is used for unblocking the remote circuit group:

(1) According to the settings of the parameter dwBlockMode, the driver sends the specified group unblocking message towards the remote PBX;

(2) After the remote PBX receives the group unblocking message, it will answer with the message of group unblocking acknowledgement message (for the TUP protocol, the corresponding unblocking acknowledgement messages for the group unblocking messages to MGU, HGU and SGU are MUA, HUA and SUA respectively; for the ISUP protocol, the corresponding unblocking acknowledgement messages for the group blocking messages to CGU is CGUA);

(3) After receiving the circuit group unblocking acknowledgement message, the driver throws out the event of E_CHG_RemotePCMBlock to the application. After the above steps are completed, the remote channel unblocking is finished.

The function SsmGetRemotePCMBlockStatus can be used to check whether the remote circuit group unblocking is finished.

Note:

- This function only supports the TUP protocol and ISUP protocol;
- According to the TUP protocol and ISUP protocol, the circuit group unblocking message won't go into effect until it has been transmitted twice. Therefore the driver will send the circuit group unblocking message to the remote PBX twice continuously, and the application need to invoke this function only once.

Related Information:

Driver version	SynCTI Ver. 3.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmBlockRemotePCM, SsmGetRemotePCMBlockStatus

Related Event: E_CHG_RemotePCMBlock

2.20.1.2.2.3 SsmGetRemotePCMBlockStatus

Obtains the blocking status of the circuit group.

Format:

int SsmGetRemotePCMBlockStatus(int nLocalPcmNo, DWORD dwBlockMode)

Parameter Description:

nLocalPcmNo	PCM logical number
dwBlockMode	Selects the circuit group blocking type to be enquired, the range of value: TUP protocol: 1: maintenance oriented circuit group blocking status; 2: hardware failure oriented circuit group blocking status; 3: circuit group blocking status generated by software. ISUP protocol: 0: maintenance oriented circuit group blocking status; 1: hardware failure oriented circuit group blocking status.

Return Value:

-1	Call failed
0	The remote circuit group is not blocked, or the remote circuit group is unblocked successfully
1	The remote circuit group is blocked successfully
2	The circuit group blocking message has been sent already, waiting for the acknowledgement message from the remote PBX
3	The circuit group unblocking message has been sent already, waiting for the acknowledgement message from the remote PBX

Function Description:

Obtains the blocking status of the circuit group. The return value can be used to check the executing status of the function SsmBlockRemotePCM or SsmUnblockRemotePCM.

Note:

- This function only supports the TUP protocol and ISUP protocol.

Related Information:

Driver version	SynCTI Ver. 3.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmBlockRemotePCM, SsmUnblockRemotePCM

2.20.2 SS7-MTP3 Based API

2.20.2.1 SsmSendSs7Msu

Sends a message (MSU) to the remote signaling point.

Format:

int SsmSendSs7Msu(WORD wMsuLength, PUCHAR pucMsuBuf)

Parameter Description:

wMsuLength	The length (bytes) of the transmitted MSU, with the maximum value of 273
pucMsuBuf	Pointer pointing to the first address of the buffer storing the MSU data. The first byte of pucMsuBuf (i.e. pucMsuBuf[0]) must be the SIO field

Return Value:

0	Successful
-1	Failed. Below are the possible failure reasons: ✧ Interruption of MTP3 Service ✧ The system doesn't support SS7 signaling ✧ The API of the system is not open yet

Function Description:

Sends a message (MSU) to the remote signaling point.

Note: none

Related Information:

Driver version	SynCTI Ver. 3.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmGetSs7Msu

2.20.2.2 SsmGetSs7Msu

Queries if there are MSU messages in the MSU receive buffer of the driver.

Format:

int SsmGetSs7Msu(PUCHAR* ppucMsuBuf)

Parameter Description:

ppucMsuBuf	Returns the pointer pointing to the first address of the buffer area in the driver storing MSU data. The first byte of the MSU message is SIO filed. (Note: the pointer of ppucMsgBuf should be declared before it's invoked, and its length can't be specified)

Return Value:

0	The MSU receive buffer in the driver is empty.
>0	The length (bytes) of the current MSU
-1	Failed. The failure reason may be: ✧ The system doesn't support SS7 signaling; ✧ The API of the system is not open yet

Function Description:

Queries if there are MSU messages in the MSU receive buffer of the driver. If there are, obtains the earliest received message.

Each time when the driver receives an MSU message, it will save the message to the MSU receive buffer of the driver, and then throw out the event of E_RCV_Ss7Msu to the application.

Note:

- The way to output SS7 MSU can be set by the configuration item GetMsuOnAutoHandle.
- When the MSU receive buffer is full, the newly received MSU messages will be discarded. Hence the application needs to obtain and process the messages as soon as possible.

Related Information:

Driver version	SynCTI Ver. 3.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmSendSs7Msu

Related Event:E_RCV_Ss7Msu

2.20.2.3 SsmGetMtp3State

Refer to SsmGetMtp3StateEx

2.20.2.4 SsmGetMtp3StateEx

Queries if the service between the original signaling point and the destination signaling point is enabled.

Format:

int SsmGetMtp3State()

int SsmGetMtp3StateEx(int nDpcNo)

Parameter Description:

nDpcNo	Destination signaling point number. The relation between the number and the code of the destination signaling point is specified in the SS7 server configuration program

Return Value:

1	The service is available
0	The service is interrupted
-1	Call failed

Function Description:

Queries if the service between the original signaling point and the destination signaling point is enabled. Only when the service is enabled can the application invoke the function of SsmSendSs7Msu to send message to the DPC. SsmGetMtp3StateEx supports the connection to multiple destination signaling points. SsmGetMtp3State is an old version function, and only applicable to connecting to a single destination signaling point, which is equivalent to calling the function SsmGetMtp3StateEx with the parameter nDpcNo set to 0.

Note:

- SS7 signaling has the auto resume feature. Therefore when the application uses this function as the alarm source for the SS7 signaling monitoring, it should not send the alarm signal until the service interruption is detected and keeps for some time.
- When the SS7 server detects that the link state of the destination signaling point is changed, it will save the service state of the signaling point and throw out the event of E_CHG_Mtp3State to the driver.

Related Information:

Driver version	SynCTI Ver. 3.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmGetMtp2Status

Related Event: E_CHG_Mtp3State

2.20.2.5 SsmGetMtp2Status

Obtains the operating status of the 64kbps signaling link in the SS7 signaling link set.

Format:

int SsmGetMtp2Status(int nLinkNum)

Parameter Description:

nLinkNum	The number of the signaling link which is specified in the SS7 server configuration program

Return Value:

0	The DSP software is uploaded but not yet started
1	Out of service
2	Initial alignment
3	Aligned ready
4	Aligned not ready
5	In service
6	Processor outage
-1	Call failed

Function Description:

This function obtains the operating status of the signaling link in SS7. The signaling link is valid only when the return value equals to 5 (in service).

Note:

- This function returns the information of the MTP2 layer and is normally used for monitoring and maintenance.

Related Information:

Driver version	SynCTI Ver. 3.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmGetMtp3State, SsmGetMtp3StateEx

2.20.2.6 SsmSendSs7MsuEx

Packs the message header according to the channel number (ch). The message content is configured by the application.

Format:

int WINAPI SsmSendSs7MsuEx(int ch, int nNewStep, WORD wMsuLength, PUCHAR pucMsuBuf)

Parameter Description:

ch	Channel number
nNewStep	Reserved
wMsuLength	Length of the message content, which is calculated from the message type but excluding the message header
pucMsuBuf	Length of the message

Return Value:

-1	Call failed: Wrong parameter
0	Call failed: Sending failure
1	Call successful

Function Description:

This function is used to pack the message header according to the channel number (ch). The message content is configured by the application.

Note: None

Related Information:

Driver version	SynCTI Ver. 5.1.0.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function:

2.20.2.7 SsmGetMaxSs7link

Obtains the total number of configured SS7 links in the system.

Format:

int WINAPI SsmGetMaxSs7link()

Parameter Description:

Return Value:

-1	Call failed
≥0	The total number of configured SS7 links in the system

Function Description:

Gets the total number of configured SS7 links in the system.

Note: None

Related Information:

Driver version	SynCTI Ver. 5.1.0.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function:

2.20.2.8 SsmGetSs7Mtp2Msu

Gets the MSU messages at MTP2 layer on a designated SS7 link.

Format:

int WINAPI SsmGetSs7Mtp2Msu(int ss7link, PUCHAR pucPara, PUCHAR* ppucMsuBuf)

Parameter Description:

ss7link	SS7 link number
pucPara	Reserved
ppucMsuBuf	Pointer pointing to the first address of the internal buffer storing the MSU message. The first byte of an MSU message is SIO field (Note that ppucMsgBuf should be declared before invoking).

Return Value:

-1	Call failed. The failure reason may be: 1) The system doesn't support SS7 signaling; 2) The API functions of the system are not opened yet.
0	The MSU receive buffer in the driver is empty.
>0	The length (bytes) of the current MSU message.

Function Description:

Queries if there are MSU messages in the MSU receive buffer in the MTP2 layer of the driver. If there are, takes out the first received message.

Note:

- The way to handle SS7 MTP2 MSU messages can be set by the configuration item AppHandleMtp2Msu;
- When the MSU receive buffer is full, the newly received MSU messages will be discarded. Hence the application needs to obtain and process the messages as soon as possible.

Related Information:

Driver version	SynCTI Ver. 5.1.0.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmSendSs7Mtp2Msu

2.20.2.9 SsmSendSs7Mtp2Msu

Sends the MSU messages at MTP2 layer on a designated SS7 link.

Format:

int WINAPI SsmSendSs7Mtp2Msu(int ss7link, WORD wMsuLength, PUCHAR pucMsuBuf)

Parameter Description:

ss7link	SS7 link number

wMsuLength	The length (bytes) of the MSU message, with the maximum value of 273
pucMsuBuf	The pointer pointing to the first address of the buffer storing MSU messages. The first byte of an MSU message is the SIO field.

Return Value:

-1	Call failed
0	Call successful

Function Description:

Sends the MSU messages at MTP2 layer on a designated SS7 link.

Note:

- The way to handle SS7 MTP2 MSU messages can be set by the configuration item AppHandleMtp2Msu.

Related Information:

Driver version	SynCTI Ver. 5.1.0.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmGetSs7Mtp2Msu

2.20.2.10 SsmSs7Mtp2CmdCtrl

Sends the MTP2 control command on a designated SS7 link.

Format:

int WINAPI SsmSs7Mtp2CmdCtrl(int ss7link, int l3_cmd, unsigned char *param, int len)

Parameter Description:

ss7link	SS7 link number

l3_cmd	Command at the MTP3 layer
	Range of value:
	enum
	{
	SS7_MTP2_START = 1,
	SS7_MTP2_STOP,
	SS7_MTP2_EMGCY,
	SS7_MTP2_EMGCY_CLRD,
	SS7_MTP2_RTV_BSNT,
	SS7_MTP2_RTVL_REQ
	};

param	Command parameter
len	Length of the command parameter

Return Value:

-1	Call failed
0	Call successful

Function Description:

Sends the MTP2 control command on a designated SS7 link.

Note:

- The feature of sending SS7 MTP2 command can be set by the configuration item AppHandleMtp2Msu.

Related Information:

Driver version	SynCTI Ver. 5.3.2.1 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmGetSs7Mtp2Msu

2.20.2.11 SsmGetDecodeSs7Msu

Obtains the calling/called party number, DPC, OPC and the original called party number from the MSU receive buffer of the driver.

Format:

int SsmGetDecodeSs7Msu(Ss7Msu *pDecodeSs7Msu)

Parameter Description:

pDecodeSs7Msu	The pointer pointing to the Ss7Msu structure

Return Value:

0	The IAM or IAI receive buffer in the driver is empty or the messages in it are all taken out
>0	The length (bytes) of the current IAM or IAI message
-1	Failed. The failure reason may be: ✧ The system doesn't support SS7 signaling; ✧ The API of the system is not open yet

Function Description:

Obtains the earliest received message which contains the calling/called party number, DPC, OPC etc. after an ISUP or TUP call is completed.

Each time when the driver receives an MSU message, it will save the message to the MSU receive buffer of the driver, and then throw out the event of E_RCV_DecodeSs7Msu to the application.

The structure of the returned Ss7Msu is as follows:

```
struct Ss7Msu
{
    UCHAR ucCallerNumber[50];//caller party number
    UCHAR ucCalledNumber[50];//called party number
    DWORD dwDPC[3];              //DPC
    DWORD dwOPC[3];              //OPC
    UCHAR ucOriginalCalledNum[50];//original called party number
    WORD wCicPcm;                //PCM value in CIC field
    WORD wCicTs;                 //Time Slot No. in CIC field
};
```

Note:

- The way to output SS7 MSU can be set by the configuration item GetMsuOnAutoHandle.
- When the MSU receive buffer is full, the newly received MSU messages will be discarded. Hence the application needs to obtain and process the messages as soon as possible.

Related Information:

Driver version	SynCTI Ver. 5.3.3.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Event: E_RCV_DecodeSs7Msu

2.20.3 Advanced Functions for ISDN

2.20.3.1 SsmGetIsdnMsu

Takes out an ISDN message from the buffer area in the driver.

Format:

int SsmGetIsdnMsu(int nPcmId, PUCHAR pucMsuBuf)

Parameter Description:

nPcmId	The logical number of the digital trunk
pucMsuBuf	Returns the ISDN message. The storage space of pucMsuBuf is allocated by the application, and its size should be no less than 256 bytes

Return Value:

0	No message in the buffer area
>0	The actual length (bytes) of the message in pucMsuBuf
-1	Call failed

Function Description:

Takes out an ISDN message from the buffer area in the driver.

Note:

- When the configuration item AutoHandleIsdn is set to 1, messages are processed by the driver; when it is set to 0, messages are processed by the application program.
- In real practice, this function call gets valid only when the configuration item AutoHandleIsdn is set to 0.
- In case a DTP board is used to monitor the ISDN line, E_RCV_IsdnSpyMsu will be thrown out upon the buffer area in the driver receiving an ISDN message.

Related Information:

Driver version	SynCTI Ver. 5.3.3.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmCheckIsdnMsu

2.20.3.2 SsmSendIsdnMsu

Sends an ISDN message.

Format:

int SsmSendIsdnMsu(int nPcmId, int nMsgLen, PUCHAR pucMsuBuf)

Parameter Description:

nPcmId	The logical number of the digital trunk
nMsgLen	The actual length (bytes) of the message in pucMsuBuf
pucMsuBuf	The buffer area for ISDN messages

Return Value:

0	Successful
-1	Call failed

Function Description:

Sends an ISDN message on TS16 of the designated digital trunk.

Note:

- When the configuration item AutoHandleIsdn is set to 1, messages are processed by the driver; when it is set to 0, messages are processed by the application program.
- In real practice, this function call gets valid only when the configuration item AutoHandleIsdn is set to 0.

Related Information:

Driver version	SynCTI Ver. 4.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmGetIsdnMsu

2.20.3.3 SsmCheckIsdnMsu

Queries if there are ISDN messages in the internal buffer area of the driver.

Format:

int SsmCheckIsdnMsu(int nPcmId)

Parameter Description:

nPcmId	The logical number of the digital trunk

Return Value:

-1	error
≥0	The number of ISDN messages

Function Description:

Queries if there are ISDN messages in the internal buffer area of the driver.

Note:

- When the configuration item AutoHandleIsdn is set to 1, messages are processed by the driver; when it is set to 0, messages are processed by the application program.
- In real practice, this function call gets valid only when the configuration item AutoHandleIsdn is set to 0.

Related Information:

Driver version	SynCTI Ver. 4.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmGetIsdnMsu

2.20.3.4 SsmISDNGetStatus

Obtains the operating status of the HDLC link in ISDN protocol.

Format:

int SsmISDNGetStatus(int nPcmNo, int *pL3Start, int *pL2DStatus, int *pL2D_L3Atom, int * pL3_L2DAtom, int *pRef_ind)

Parameter Description:

nPcmNo	The logical number of the local digital trunk
pL3Start	Returns the information that whether L3 in ISDN protocol is started. The return value of 1 means started, 0 means not started
pL2Dstatus	Returns the operating state of L2 in ISDN protocol. There are 8 states as follows: State 1 TEI unassigned State 2 Assign awaiting TEI State 3 Establish awaiting TEI State 4 TEI assigned State 5 Awaiting establishment State 6 Awaiting release State 7 Multiple frame established State 8 Timer recovery (See Q921 for detailed information)
pL2D_L3Atom	Returns the received original message from L2 to L3 in ISDN protocol. (See Q921 for detailed information)
pL3_L2DAtom	Returns the sent original message from L3 to L2 in ISDN protocol. (See Q921 for detailed information)
pRef_ind	Returns the internal indicator value in ISDN protocol (the fixed value is 0xfe)

Return Value:

0	Successful
-1	Call failed

Function Description:

Obtains the operating status of the HDLC link in ISDN protocol.

Note:

- This function is normally called in the response code of the event of E_CHG_ISDNStatus.

Related Information:

Driver version	SynCTI Ver. 3.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Event: E_CHG_ISDNStatus

2.20.3.5 SsmGetUserInfo

Obtains the content of the User-User message unit.

Format:

int WINAPI SsmGetUserInfo(int ch, PUCHAR pUUI)

Parameter Description:

ch	Channel number
pUUI	The pointer pointing to the buffer area storing content of User-User message units. The storage space is allocated by the application and should be no less than 131 bytes.

Return Value:

≥0	Call successful. Returns the content length of the User-User message unit.
-1	Call failed.

Function Description:

Obtains the content of the User-User message unit.

Note:

Related Information:

Driver version	SynCTI Ver. 5.1.0.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Event: SsmSetUserInfo

2.20.3.6 SsmSetUserInfo

Sets the content of the User-User message unit for the SETUP message.

Format:

int WINAPI SsmSetUserInfo(int ch, PUCHAR pUUI,WORD wLen)

Parameter Description:

ch	Channel Number
pUUI	Pointer pointing to the buffer storing the content of the User-User message unit (excluding message type and content length)
wLen	Length of the User-User message unit

Return Value:

0	Call successful.
-1	Call failed.

Function Description:

Sets the content of the User-User message unit for the SETUP message.

Note:

Related Information:

Driver version	SynCTI Ver. 5.3.1.4 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Event: SsmGetUserInfo

2.20.3.7 SsmISDNGetProgressMsg

Obtains the content of the PROGRESS message.

Format:

int WINAPI SsmISDNGetProgressMsg(int ch, BYTE* pbMsg)

Parameter Description:

ch	Channel number
pbMsg	The pointer pointing to the buffer area storing the PROGRESS message. The storage space, allocated by the application, should be no less than 300 bytes.

Return Value:

≥0	Call successful. Returns the length in byte of the PROGRESS message
-1	Call failed

Function Description:

Obtains the content of the PROGRESS message.

Note:

Related Information:

Driver version	SynCTI Ver. 5.2.0.1 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function:

2.20.3.8 SsmGetIsdnMsuEX

Takes out a designated ISDN message from the buffer area in the driver.

Format:

int WINAPI SsmGetIsdnMsuEx(int nPcmId,int nMsgType, PUCHAR pucMsuBuf, PWORD wMsgLen)

Parameter Description:

nPcmId	The logical number of the digital trunk
nMsgType	Designates the message type. Currently, only the 0x20 message is supported.
pucMsuBuf	Returns the ISDN message. The storage space of pucMsuBuf is allocated by the application, and its size should be no less than 256 bytes
wMsgLen	Returns the message length. The storage space of wMsgLen is allocated by the application, and its size should be no less than 1 byte

Return Value:

nCh	Returns the channel of the message
-1	Call failed or no message

Function Description:

Takes out a designated ISDN message from the buffer area in the driver.

Note:

- When the configuration item AutoHandleIsdn is set to 1, messages are processed by the driver; however, the 0x20 message can be obtained by this function.
- In real practice, this function call gets valid only when the configuration item AutoHandleIsdn is set to 1; the function SsmGetIsdnMsu will be invoked to obtain the message once the configuration item AutoHandleIsdn is set to 0.

Related Information:

Driver version	SynCTI Ver. 5.4.2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmCheckIsdnMsu, SsmGetIsdnMsu

2.20.3.9 SsmSendIsdnMsu

Sends an ISDN message when AutoHandleIsdn is set to 1.

Format:

int SsmSendIsdnMsuEx(int ch,int nMsgType, int nMsgLen, PUCHAR pucMsuBuf)

Parameter Description:

ch	The channel number
nMsgType	Designates the message type, reserved currently
nMsgLen	The actual length (bytes) of the message in pucMsuBuf
pucMsuBuf	The buffer area for ISDN messages, storing the contents after the message type

Return Value:

0	Successful
-1	Call failed

Function Description:

Sends an ISDN message on TS16 of the designated digital trunk.

Note:

- When the configuration item AutoHandleIsdn is set to 1, messages are processed by the driver; however, messages can be sent out via invoking this function.
- In real practice, this function call gets valid only when the configuration item AutoHandleIsdn is set to 1; the function SsmSendIsdnMsu will be invoked to send the message once the configuration item AutoHandleIsdn is set to 0.
- This function is used to send the ISDN message in which the protocol discriminator and the call reference in the first 4 bytes can be structured by the driver, and the application only stores the messages after the message type.

Related Information:

Driver version	SynCTI Ver. 5.4.2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmGetIsdnMsuEx

2.20.4 Advanced Functions for China SS1

2.20.4.1 Controlling ABCD Signaling Bits

2.20.4.1.1SsmGetCAS

Obtains the state of the ABCD signaling at the receiving end on the channel.

Format:

int SsmGetCAS(int ch)

Parameter Description:

ch	Channel number

Return Value:

-1	Call failed
≥0	The higher 4 bits of the return value are 0, the lower 4 bits are the received CAS signaling bits at the local end: bit3 bit2 bit1 bit0 A B C D

Function Description:

Obtains the current value of the ABCD signaling sent by the remote PBX.

Note:

- The ABCD signaling is transmitted on TS 16 of E1;
- The features of this function can also be implemented by the event of E_RCV_CAS. We recommend

you to use the event of E_RCV_CAS.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmSendCAS

2.20.4.1.2SsmSendCAS

Sets the state of the ABCD signaling at the transmitting end on the channel.

Format:

int SsmSendCAS(int ch, BYTE btCas)

Parameter Description:

ch	Channel number
btCas	The value of the CAS signaling to be transmitted, the higher 4 bits are 0, the lower 4 bits are the new state of the ABCD signaling bits: bit3 bit2 bit1 bit0 A B C D

Return Value:

-1	Call failed
0	Successful

Function Description:

Sets the state of the ABCD signaling at the local end.

Note:

- The ABCD signaling is transmitted on TS 16 of E1;
- If auto call connection is enabled on the channel, the call of this function will fail.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmGetSendingCAS, SsmGetCAS

2.20.4.1.3SsmGetSendingCAS

Obtains the state of the ABCD signaling at the transmitting end on the channel.

Format:

int SsmGetSendingCAS(int ch)

Parameter Description:

ch	Channel number

Return Value:

-1	Call failed
≥0	The lowest 4 bits of the return value are the ABCD signaling bits at the transmitting end: bit3 bit2 bit1 bit0 A B C D Other bits are reserved.

Function Description:

Obtains the state of the ABCD signaling at the transmitting end on the channel.

Note:

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmGetCAS, SsmSendCAS

2.20.4.1.4SsmSetSendCASFlag

Sets whether to allow the channel to send CAS signals.

Format:

int WINAPI SsmSetSendCASFlag(int ch, int nCASFlag)

Parameter Description:

ch	Channel number
nCASFlag	indicates whether the channel is allowed to send CAS signals. 0: not allowed 1: allowed

Return Value:

-1	Call failed
=0	Call successful

Function Description:

Sets whether to allow the channel to send CAS signals.

Note: None

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmGetSendCASFlag, SsmSendCAS

2.20.4.1.5SsmGetSendCASFlag

Obtains the flag indicating whether the channel is allowed to send CAS signals.

Format:

int WINAPI SsmGetSendCASFlag(int ch, int* pCASFlag)

Parameter Description:

ch	Channel number
pCASFlag	The pointer used to obtain the flag indicating whether the channel is allowed to send CAS signals

Return Value:

-1	Call failed
=0	Call successful

Function Description:

Obtains the flag indicating whether the channel is allowed to send CAS signals.

Note: None

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmGetCAS, SsmSetSendCASFlag

2.20.4.2 Controlling R2 Signal Transceiver

2.20.4.2.1SsmSetRxR2Mode

Sets the R2 signal receiver.

Format:

int SsmSetRxR2Mode(int ch, int nMode, BOOL bEnable)

Parameter Description:

ch	Channel Number
nMode	The operating mode of the R2 signal receiver: =0: Receives the backward R2 signal =1: Receives the forward R2 signal
bEnable	The operating status of the R2 signal receiver: =TRUE: Enable =FALSE: Disable

Return Value:

-1	Call failed
0	Successful

Function Description:

Sets the R2 signal receiver.

Note:

- If auto call connection is enabled on the channel, the call of this function will fail.
- The R2 signal transceiver and the DTMF detector can't work simultaneously. Hence when the R2 signal transceiver is used, the DTMF detector must be turned off. After the MFC process, you need to invoke the function SsmEnableRxDtmf to restart the DTMF detector.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib

DLL	shp_a3.dll

Related Function: SsmGetR2

2.20.4.2.2SsmGetR2

Obtains the detected result of the R2 signal receiver.

Format:

int SsmGetR2(int ch)

Parameter Description:

ch	Channel Number

Return Value:

-1	Call failed
0	R2 signal is not detected on the line
1~15	R2 signal is detected on the line. For the forward R2 signal, the range of the return value is: 1~15; For the backward R2 signal, the range of the return value is: 1~6

Function Description:

Obtains the detected result of the R2 signal receiver.

Note:

- If auto call connection is enabled on the channel, the call of this function will fail.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmSetRxR2Mode, SsmSendR2, SsmSendR2Ex

2.20.4.2.3SsmSendR2

Generates an R2 signal on the channel.

Format:

int SsmSendR2(int ch, int nMode, BYTE btR2)

Parameter Description:

ch	Channel Number
nMode	The operating mode of the R2 signal generator: =0: generates the backward R2 signal; =1: generates the forward R2 signal
btR2	The code value of the R2 signal: ● For the forward R2 signal, range of value: 1~15; ● For the backward R2 signal, range of value: 1~6

Return Value:

-1	Call failed
0	Successful

Function Description:

Generates an R2 signal on the channel which continues until the application calls the function SsmStopSendR2.

Note:

- If auto call connection is enabled on the channel, the call of this function will fail.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmStopSendR2, SsmGetSendingR2, SsmGetR2, SsmSendR2Ex

2.20.4.2.4SsmSendR2Ex

Generates an R2 signal of a specified time length on the channel.

Format:

int SsmSendR2Ex(int ch, int nMode, BYTE btR2, DWORD dwKeepTime)

Parameter Description:

ch	Channel Number
nMode	The operating mode of the R2 signal generator: =0: generates the backward R2 signal; =1: generates the forward R2 signal
btR2	The code value of the R2 signal: ● For the forward R2 signal, range of value: 1~15; ● For the backward R2 signal, range of value: 1~6
dwKeepTime	The R2 signal duration, calculated by millisecond(ms)

Return Value:

-1	Call failed
0	Successful

Function Description:

Generates an R2 signal of the specified time length dwKeepTime on the channel. The function SsmStopSendR2 can be called to stop the R2 signal.

Note:

- If auto call connection is enabled on the channel, the call of this function will fail.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmStopSendR2, SsmGetSendingR2， SsmGetR2， SsmSendR2

2.20.4.2.5SsmStopSendR2

Stops generating an R2 signal on the channel.

Format:

int SsmStopSendR2(int ch)

Parameter Description:

ch	Channel Number

Return Value:

-1	Call failed
0	Successful

Function Description:

Stops generating an R2 signal on the channel.

Note:

- If auto call connection is enabled on the channel, the call of this function will fail.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmSendR2, SsmSendR2Ex

2.20.4.2.6SsmGetSendingR2

Obtains the operating mode of the R2 signal generator on the channel.

Format:

int SsmGetSendingR2(int ch, int* pnMode, BYTE* pbtR2)

Parameter Description:

ch	Channel Number
pnMode	Returns the operating mode of the R2 signal generator: =0: generates the backward R2 signal; =1: generates the forward R2 signal
pbtR2	Returns the code value of the R2 signal: ● For the forward R2 signal, range of value: 1~15; ● For the backward R2 signal, range of value: 1~6

Return Value:

-1	Call failed
0	Successful

Function Description:

Obtains the operating mode of the R2 signal generator on the channel.

Note:

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmSendR2, SsmSendR2Ex

2.21 Functions for SHV Series Boards (CTI Series)

2.21.1 SsmGetMaxVCh

Obtains the total number of the voice-alteration channels in the application.

Format:

int SsmGetMaxVCh()

Parameter Description: none

Return Value:

-1	Call failed
≥0	The total number of the voice-alteration channels

Function Description:

Obtains the total number of the voice-alteration channels in the application.

Note:

Related Information:

Driver version	SynCTI Ver. 4.7.2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmGetMaxFreeVCh

2.21.2 SsmGetMaxFreeVCh

Obtains the total number of the free voice-alteration channels.

Format:

int SsmGetMaxFreeVCh()

Parameter Description: none

Return Value:

-1	Call failed
≥0	The total number of the free voice-alteration channels

Function Description:

Obtains the total number of the free voice-alteration channels.

Note:

Related Information:

Driver version	SynCTI Ver. 4.7.2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmGetMaxVCh

2.21.3 SsmBindVCh

Binds the voice channel to the voice-alteration channel.

Format:

int SsmBindVCh(int iCh)

Parameter Description: none

iCh	Voice channel number

Return Value:

0	Successful
-1	Call failed: iCh is out of range
-2	Call failed: iCh doesn't support bus exchange
-3	Call failed: There is no free voice-alteration channel
-4	Call failed: iCh is already bound to a voice-alteration channel
-5	Call failed: The driver is not loaded

Function Description:

Binds the voice channel to the voice-alteration channel.

When the driver invokes this function, it will automatically allocate a free voice-alteration channel and bind it to the vioce channel iCh. The incoming call signal on the channel iCh will first enter the voice-alteration channel and get through the process of voice-alteration, and then be sent to the bus.

Note:

- After the driver is successfully loaded, the application can invoke this function at anytime.

Related Information:

Driver version	SynCTI Ver. 4.7.2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmUnBindVCh, SsmSetVoiceEffect

Sample Code:

Example 1: There is two-way connection between channel 1 and channel 2, the voice from channel 1 needs to be altered, the voice from channel 2 doesn't need to be altered, below is the implementation code:

```
……                          // The code used to initiate the call
SsmBindVCh(1);              // Bind channel 1 to a voice-alteration channel
SsmSetVoiceEffect(1,180);   // Set the voice alteration effect
SsmTalkWith(1,2);           // Establish two-way connection between channel 1 and channel 2
……                          // Calling
SsmUnBindVCh(1);            // The call is completed, unbind channel 1 to the voice-alteration channel
SsmStopTalkWith(1,2);       // Disconnect the two-way connection
```

Example 2: Channel 1, channel 2 and channel 3 enter a conference room, Channel 1 and Channel 2 require the voice alteration. Below is the implementation code:

```
int GroupID = SsmCreateConfGroup(…);    // Create a conference room
……
SsmBindVCh(1);
SsmJoinConfGroup(GroupID,1,…);          //Channel 1 enters the conference room
SsmBindVCh(2);
SsmJoinConfGroup(GroupID,2,…);          //Channel 2 enters the conference room
SsmJoinConfGroup(GroupID,3,…);          //Channel 3 enters the conference room
……
```

2.21.4 SsmUnBindVCh

Unbinds the voice channel to the voice-alteration channel.

Format:

int SsmUnBindVCh(int iCh)

Parameter Description:

iCh	Voice channel number

Return Value:

0	Successful
-1	Call failed: iCh is out of range
-2	Call failed: iCh doesn't support bus exchange
-3	Call failed: iCh isn't bound to any voice alteration channel
-4	Call failed: the driver is not loaded yet

Function Description:

Unbinds the voice channel to the voice-alteration channel. When the voice-alteration channel is unbound, it will be automatically reclaimed and enter the idle queue.

Note:

- If the voice channel doesn't need voice alteration anymore, please unbind it to the voice alteration channel.

Related Information:

Driver version	SynCTI Ver. 4.7.2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmBindVCh

2.21.5 SsmSetVoiceEffect

Sets the voice-alteration effect.

Format:

int SsmSetVoiceEffect(int iCh, int iValue)

Parameter Description:

iCh	Voice channel number
iValue	Voice alteration effect, range of value: 70~220, 128 means the original voice (unchanged), greater than 128 means the female voice effect, less than 128 means the male voice effect

Return Value:

0	Successful
-1	Call failed: iCh is out of range
-2	Call failed: iCh isn't bound to any voice alteration channel
-3	Call failed: iValue is out of range
-4	Call failed: the driver is not loaded

Function Description:

Sets the voice-alteration effect.

Note:

- Only when the voice channel has been bound to a voice alteration channel, can this function be called;
- It's difficult to describe the voice alteration effect in details, set iValue based on the actual testing effect.

Related Information:

Driver version	SynCTI Ver. 4.7.2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmBindVCh, SsmGetVoiceEffect

2.21.6 SsmGetVoiceEffect

Obtains the voice-alteration effect.

Format:

int SsmGetVoiceEffect(int iCh)

Parameter Description:

iCh	Voice channel number

Return Value:

>0	Successful, returns the voice-alteration effect, refer to the descriptions of the function SsmSetVoiceEffect for the detailed values
-1	Call failed: iCh is out of range
-2	Call failed: iCh is not bound to any voice-alteration channel
-3	Call failed: the driver isn't loaded yet

Function Description:

Obtains the voice-alteration effect.

Note:

Related Information:

Driver version	SynCTI Ver. 4.7.2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmSetVoiceEffect

2.21.7 SsmSetVoiceEffectEx

Sets the voice-alteration parameters for 240E VAR boards.

Format:

int WINAPI SsmSetVoiceEffectEx(int iCh, int VarType, int VarParamA, int VarParamB, int VarParamC)

Parameter Description:

iCh	Voice channel number
VarType	Voice alteration type (0: No alteration; 1: Timbre alteration (including age and gender alteration); 2: Raucity effect)
VarParamA	Voice alteration parameter A (For timbre alteration, range of value: 20-180, 100 indicates no alteration; for raucity effect, range of value: 8-800, the larger value indicates the more obvious effect)
VarParamB	Voice alteration parameter B (For timbre alteration, range of value: 20-180, 100 indicates no alteration; for raucity effect, range of value (volume): 1-300, the larger value indicates the higher volume)
VarParamC	Voice alteration parameter C (For timbre alteration, range of value: 20-180, 100 indicates no alteration; for raucity effect, this parameter is invalid)

Return Value:

0	Successful

-1	Call failed: iCh is out of range
-2	Call failed: iCh isn't bound to any voice alteration channel
-3	Call failed: voice-alteration parameters are out of range
-4	Call failed: the driver is not loaded

Function Description:

Sets the voice-alteration parameters for 240E VAR boards.

Note:

- Only when the voice channel has been bound to a voice alteration channel can this function be called;
- It's difficult to describe the voice alteration effect in details, please set the corresponding parameters based on the actual testing effect. The voice alternation examples below are for your reference:
  From young man to young woman: VarParamA= 66; VarParamB = 100; VarParamC= 82;
  From young man to old woman: VarParamA= 72; VarParamB = 100; VarParamC= 88;
  From young man to old man: VarParamA= 140; VarParamB = 100; VarParamC= 118;
  From young man to child: VarParamA= 66; VarParamB = 100; VarParamC= 66;
  From young woman to young man: VarParamA= 140; VarParamB = 100; VarParamC= 118;
  From young woman to old woman: VarParamA= 120; VarParamB = 100; VarParamC= 108;
  From young woman to old man: VarParamA= 180; VarParamB = 100; VarParamC= 136;
  From young woman to child: VarParamA= 88; VarParamB = 100; VarParamC= 88;
  Make a young man's voice raucous: VarParamA= 405; VarParamB = 30;
  Make a young woman's voice raucous: VarParamA= 600; VarParamB = 10.

Related Information:

Driver version	SynCTI Ver. 5.3.2.3 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmBindVCh, SsmGetVoiceEffectEx

2.21.8 SsmGetVoiceEffectEx

Obtains the voice-alteration parameters for 240E VAR boards.

Format:

int WINAPI SsmGetVoiceEffectEx(int iCh, int* pVarType, int* pVarParamA, int* pVarParamB, int* pVarParamC)

Parameter Description:

iCh	Voice channel number
pVarType	Pointer storing the int variable indicating the voice-alteration type
pVarParamA	Pointer storing the int variable indicating the voice-alteration parameter A
pVarParamB	Pointer storing the int variable indicating the voice-alteration parameter B
pVarParamC	Pointer storing the int variable indicating the voice-alteration parameter C

Return Value:

>0	Successful, returns the voice-alteration effect, refer to the descriptions of the function SsmSetVoiceEffectEx for the detailed values
-1	Call failed: iCh is out of range or a parameter is a null pointer
-2	Call failed: iCh is not bound to any voice-alteration channel
-3	Call failed: the driver isn't loaded yet

Function Description:

Obtains the voice-alteration parameters for 240E VAR boards.

Note:

Related Information:

Driver version	SynCTI Ver. 5.3.2.3 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmSetVoiceEffectEx

2.22 Functions for SHN Series Boards (CTI Series)

2.22.1 Setting Special Programming Interfaces for SIP Channel

2.22.1.1 SsmIpGetSessionCodecType

Obtains the CODEC actually used by a designated channel in an ongoing call.

Format:

int SsmIpGetSessionCodecType (int ch)

Parameter Description:

ch	Channel number

Return Value:

-1	Call failed
6	A-Law is used by the designated channel in the ongoing call
7	μ-Law is used by the designated channel in the ongoing call
49	GSM is used by the designated channel in the ongoing call
131	G.729A is used by the designated channel in the ongoing call
4	G.723_1 is used by the designated channel in the ongoing call
9	G.722 is used by the designated channel in the ongoing call
96	AMR is used by the designated channel in the ongoing call
98	ILBC is used by the designated channel in the ongoing call

Function Description:

Obtains the CODEC actually used by a designated channel in an ongoing call.

Note:

- This function can be invoked only when the channel stays in the ‘Talking’ state.
- This function is called SsmIpGetUsingCodecType in SynCTI 5.1.0.0 and below versions.

Related Information:

Driver version	SynCTI Ver. 5.3.3.0 or above
Header	Shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

2.22.1.2 SsmIpSetForwardNum

Sets a forward number for a designated channel.

Format:

int SsmIpSetForwardNum(int ch, LPSTR pszForwardNum)

Parameter Description:

ch	Channel number
pszForwardNum	Forward number. Setting pszForwardNum to a null string will cancel the previous settings.

Return Value:

-1	Call failed
0	Call successful

Function Description:

Sets a forward number for a designated channel. Then any call to this channel will be forwarded unconditionally to the number set by this function.

Note:

- If a forward number has been set for a specified channel before, it can be cancelled by invoking this function again to set pszForwardNum to a null string.
- For the call forwarding operation over the VoIP board, it is not allowed to set the forward number to be the same as the called party number.

Related Information:

Driver version	SynCTI Ver. 5.0.0.0 or above
Header	Shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

2.22.1.3 SsmIpInitiateTransfer

Sets a transfer number for a designated channel.

Format:

int SsmIpInitiateTransfer(int ch, LPSTR pszTransferTo)

Parameter Description:

ch	Channel number
pszTransferTo	Transfer number

Return Value:

-1	Call failed
0	Call successful

Function Description:

Sets a transfer number for a designated channel. Invoking this function during an ongoing call will transfer this call to the preset number.

Note:

- This function can be invoked only when the channel stays in the ‘Talking’ state.

Related Information:

Driver version	SynCTI Ver. 5.0.0.0 or above
Header	Shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

2.22.1.4 SsmIpGetMessageField

Obtains a series of field values in messages related to the call establishment of a designated channel.

Format:

int SsmIpGetMessageField(int ch, int type, LPSTR szBuffer, int *pSize)

Parameter Description:

ch	Channel number
Type	Field type to be obtained
SzBuffer	Buffer used to store the obtained field content
PSize	A pointer pointing to the value which indicates the size of the user-provided buffer. If this function call is successful, the value it points to is the size of valid data obtained; if this function call is failed, the value it points to is the buffer size actually needed.

Return Value:

-1	Call failed
0	Call successful

Function Description:

Obtains a series of field values in messages related to the call establishment of a designated channel.

The values of the parameter 'Type' are as follows.

- TYPE_MESSAGE_CONTACT

  Obtains the content of the 'Contact' field in the 'INVITE' or 'REINVITE' message. It only works for the called party. If there are multiple 'Contact' fields in the message body, the value it returns will be shown in the following format.

Contact0	\0	Contact1	\0	…	Contactn	\0	\0

- TYPE_MESSAGE_SDP

  Obtains the content of the 'SDP' field in the 'INVITE' or '200OK' message from the remote end. This parameter works for both calling and called parties. For a called party, it obtains the content of the 'SDP' field in the remote 'INVITE' message; for a calling party, it obtains the content of the 'SDP' field in the remote '200OK' message.

Note:

- To obtain the content of the 'Contact' field, this function is valid only when the channel stays in the 'Ringing', 'Talking' or 'Pending' state.
- To obtain the content of the 'SDP' field, for a called party, this function is valid only when the channel stays in the 'Ringing', 'Talking' or 'Pending' state; for a calling party, this function is valid only when the channel stays in the 'Talking' or 'Pending' state.
- Obtaining the content of the 'SDP' field requires SynCTI Ver. 5.3.1.4 or above.

Related Information:

Driver version	SynCTI Ver. 5.0.0.0 or above
Header	Shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

2.22.1.5 SsmIPGetMsgFieldStr

Obtains a series of field values from those messages related to the call establishment of a designated channel.

Format:

int SsmIPGetMsgFieldStr(int ch, int nMsgType, LPSTR szParaType, LPSTR szBuf, int nBufSize)

Parameter Description:

ch	Channel number	
nMsgType	Value	Description
	0	obtains the fields in invite message

	1	obtains the fields in 480 message
	2	obtains the fields in 183 message
	3	obtains the fields in 181 message
szParaType	Field name	
szBuf	Buffer used to store the obtained field content	
nBufSize	The size of szBuf buffer	

Return Value:

-1	Call failed
>=0	Call successful, denoting the length of valid data obtained by szBuf buffer

Function Description:

Obtains a series of field values from those messages related to the call establishment of a designated channel.

Note:

- If you want to obtain the fields in invite message, it is required to invoke this function when the channel is in ringing or talking state;
- If you want to obtain the fields in the 480 message, invoke this function when the channel is pending.
- If you want to obtain the fields in the 183 or 181 message, invoke this function when the channel is in the state of S_CALL_VOIP_SESSION_PROCEEDING;
- Currently, this function only supports obtaining the fields in the invite, 480, 183 or 181 message.

Example of C Language:

```
char szParaType[128]="History-Info";
char szBuf[128];
int nBufSize = sizeof(szBuf);
if(SsmGetChState(ch) == S_CALL_VOIP_SESSION_PROCEEDING)
SsmIPGetMsgFieldStr(ch,3,szParaType,szBuf,nBufSize);//3 obtains the fields in 181 message
```

Related Information:

Driver version	SynCTI Ver. 5.3.3.0 or above
Header	Shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

2.22.1.6 SsmSipMsgSetHeader

Add a field to the INVITE message for outgoing calls through an IP channel.

Format:

int WINAPI SsmSipMsgSetHeader(char* h_Name,char* h_Value)

Parameter Description:

h_Name	Field name
h_Value	Field content

Return Value:

-1	Call failed
0	Successful

Function Description:

Add a field to the INVITE message for outgoing calls through an IP channel.

Note:

- This function only supports adding non-critical fields.
- The parameters set in this function are only valid to the current call.

- The configuration items SipMsgHeaderName and SipMsgHeaderValue can implement the same feature.

Related Information:

Driver version	SynCTI Ver. 5.3.2.1 or above
Header	Shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

2.22.1.7 SsmSipMsgSetHeaderA

Add a field to the INVITE/REFER/180/183 message for outgoing calls through a designated IP channel.

Format:

int WINAPI SsmSipMsgSetHeaderA(int nCh, char* h_Name, char* h_Value)

Parameter Description:

nCh	Channel number
h_Name	Field name
h_Value	Field content

Return Value:

-1	Call failed
0	Successful

Function Description:

Add a field to the INVITE/REFER/180/183 message for outgoing calls through a designated IP channel.

Note:

- This function only supports adding non-critical fields.
- This function can be invoked repeatedly to add multiple fields.
- To delete all fields on a designated channel, you can invoke this function with the parameters h_Name and h_Value set to NULL.

Related Information:

Driver version	SynCTI Ver. 5.4.4.0 or above
Header	Shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

2.22.1.8 SsmSipStackRegister

Sends the Register request to SIP Server from SynSIP protocol stack.

Format:

int WINAPI SsmSipStackRegister(LPCSTR szRegSrvAddr, LPCSTR szOutBoundAddr, LPCSTR szDisplayName, LPCSTR szUserName, LPCSTR szAuthUserName, LPCSTR szPasswd, LPCSTR szRealm, int nExpires)

Parameter Description:

szRegSrvAddr	IP address of SIP Server
szOutBoundAddr	Address of the bound SIP Server
szDisplayName	Displayed username
szUserName	Username registered for the corresponding board
szAuthUserName	Authentication username (valid only when the registration on SIP Server needs to be authenticated)
szPasswd	Password registered for the corresponding board (valid only when the registration on SIP Server needs to be authenticated)

szRealm	Alias of SIP Server
nExpires	Expiration of the registration

Return Value:

-1	Call failed
>0	Register ID

Function Description:

Sends the Register request to SIP Server from SynSIP protocol stack.

Note:

- The register ID returned after the successful call of this function should be maintained by users themselves. This parameter is necessary for querying the register information and binding the channel with register information.
- The channel must be unbound from register information before you invoke this function with nExpires=0 to cancel the register.
- The result (registration successful or not) cannot be acquired from the returned value of this function call, but can be achieved by invoking SsmSipGetRegInfo.

Related Information:

Driver version	SynCTI Ver. 5.3.3.0 or above
Header	Shpa3api.h
Library	shp_a3.lib, SynSip.lib
DLL	shp_a3.dll, SynSip.dll

Related Function: SsmSipStackUnRegister, SsmSipStackRemoveRegister, SsmSipGetRegInfo

2.22.1.9 SsmSipBindChWithRegInfo

Binds a designated channel with register information.

Format:

int WINAPI SsmSipBindChWithRegInfo(int nChID, int nRegID)

Parameter Description:

nChID	Channel number
nRegId	Register ID

Return Value:

-1	Binding failed
0	Binding successful

Function Description:

Binds a designated channel with register information.

Note:

- This function can only be invoked when the channel stays in the ‘idle’ state. Otherwise, invoking this function will fail.
- If the channel has been already bound with register information, invoking this function will fail.

Related Information:

Driver version	SynCTI Ver. 5.3.2.6 or above
Header	Shpa3api.h
Library	shp_a3.lib, SynSip.lib
DLL	shp_a3.dll, SynSip.dll

Related Function: SipUnBindChWithRegInfo

2.22.1.10 SsmSipUnBindChWithOutRegInfo

Unbinds a designated channel from register information.

Format:

int SsmSipUnBindChWithOutRegInfo(int nChID);

Parameter Description:

nChID	Channel number

Return Value:

-1	Cancellation failed
0	Cancellation successful

Function Description:

Unbinds a designated channel from register information.

Note:

- This function can only be invoked when the channel stays in the 'idle', 'register failed' or 'registering' state. Otherwise, invoking this function will fail.

Related Information:

Driver version	SynCTI Ver. 5.4.2.0 or above
Header	Shpa3api.h
Library	shp_a3.lib, SynSip.lib
DLL	shp_a3.dll, SynSip.dll

Related Function: SsmSipBindChWithRegInfo

2.22.1.11 SsmSipStackUnRegister

Cancel the register from SynSIP protocol stack to the SIP Server.

Format:

int WINAPI SsmSipStackUnRegister(int nRegID)

Parameter Description:

nRegID	Register ID

Return Value:

-1	Call failed
0	Successful

Function Description:

Cancel the register from SynSIP protocol stack to the SIP Server.

Note:

- Invoking the function SsmSipStackRegister with nExpire=0 can implement the same feature as this function. Only if the channel is unbound from register information and the register state is 'register successful' can this function be invoked.

Related Information:

Driver version	SynCTI Ver. 5.3.2.6 or above
Header	Shpa3api.h
Library	shp_a3.lib, SynSip.lib
DLL	shp_a3.dll, SynSip.dll

Related Function: SsmSipStackRegister

2.22.1.12 SsmSipStackRemoveRegister

Removes the register from SynSIP protocol stack.

Format:

int WINAPI SsmSipStackRemoveRegister(int nRegID)

Parameter Description:

nRegId	Register ID

Return Value:

-1	Call failed
0	Successful

Function Description:

Removes the register from SynSIP protocol stack.

Note:

- Only if the channel is unbound from the register information to be removed and the register state is not 'register successful' can this function be invoked.

Related Information:

Driver version	SynCTI Ver. 5.3.2.6 or above
Header	Shpa3api.h
Library	shp_a3.lib, SynSip.lib
DLL	shp_a3.dll, SynSip.dll

Related Function: SsmSipStackRegister

2.22.1.13 SsmSipGetRegInfo

Queries the register information.

Format:

int WINAPI SsmSipGetRegInfo(int nRegID, RegInfo *pstRegInfo)

Parameter Description:

nRegID	Register ID
pstRegInfo	Pointer for register information structure

Return Value:

-1	Call failed or no corresponding register information is found
0	Call successful and the corresponding register information is obtained

Function Description:

Queries the register information.

The structure of the returned Reginfo is as follows:

```
typedef struct tag_RegInfo
{

    int nRegID; // Registration ID
    enRegState enRegS; // Registration status (0: registration initialization, 1: registering, 2: registration successful, 3: registration failed)
    int nRetryCount; // Registration count
    BOOL bIsUpdateReg; // Registration successful or not
    int nExpires; // Registration interval
```

```
int nOriginalRegID; //Original registration ID

int nTotalBindNum; // Total amount of bound channels
int sBindChList[MAX_SIPIP_BINDCHNUM]; // List of bound channels
char szServerAddress[MAX_SIPIP_ADDR_LENGTH]; // Registrar IP address
char szOutBoundAddr[MAX_SIPIP_ADDR_LENGTH]; // Outbound proxy IP address of the registration
 char szAuthName[MAX_SIPIP_ADDR_LENGTH]; // Authentication username
char szUserName[MAX_SIPIP_ADDR_LENGTH]; // Registration username
char szPassword[MAX_SIPIP_ADDR_LENGTH]; // Password
char szRealm[MAX_SIPIP_ADDR_LENGTH]; // Alias of the SIP server
char szLocalIdentity[MAX_SIPIP_ADDR_LENGTH]; // Local SIP From URL
char szDisplayName[MAX_SIPIP_ADDR_LENGTH]; // Displayed username
```

}RegInfo;

Related Information:

Driver version	SynCTI Ver. 5.3.3.0 or above
Header	Shpa3api.h
Library	shp_a3.lib, SynSip.lib
DLL	shp_a3.dll, SynSip.dll

Related Function: SsmSipStackRegister

2.22.1.14 SsmSipSetTxUserName

Sets the local UserName for an outgoing call.

Format:

int SsmSipSetTxUserName(int ch, LPSTR pszUserName)

Parameter Description:

ch	Channel number
pszUserName	pszUserName sets the 'UserName' part in a complete SIP URI, with the maximum size of 50. The format of a complete SIP URI is "displayname" <sip:user@host:port>.

Return Value:

-1	Call failed
0	The size of UserName is 0, invalid
>0	Size of UserName

Function Description:

Sets the local UserName for an outgoing call in case of non registration.

You are allowed to modify the registered UserName after adding a configuration item 'EnableSetUsername=1' to Section [SIP] in the configuration file.

Note:

- Usually this function is valid only in case of non registration;
- In case of registration, for SynCTI Ver. 5.3.1.3 and the above versions, this function is valid after the configuration item 'EnableSetUsername=1' is added to Section [SIP] in the configuration file.
- This function is valid only when it is set before starting a call.

Related Information:

Driver version	SynCTI Ver. 5.1.0.0 or above
Header	Shpa3api.h
Library	shp_a3.lib

DLL	shp_a3.dll

2.22.1.15 SsmSetIpFlag

Sets some parameters for VoIP boards.

Format:

int SsmSetIpFlag(int ch,int Type,LPSTR pszBuffer)

Parameter Description:

ch	Channel number
Type	The type to be modified
pszBuffer	Modified value

Return Value:

-1	Call failed
0	The size of pszBuffer is 0, invalid
>0	Size of pszBuffer

Function Description:

Sets some parameters for VoIP boards.

Note:

- When this function is used to set outgoing calls with the parameter Type = 1, the IP address in the 'from' field of Invite message is valid only in case of non registration.
- This function is invalid if the value of Type is not equal to 1.

Related Information:

Driver version	SynCTI Ver. 5.1.0.0 or above
Header	Shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

2.22.1.16 SsmSipGetReferStatus

Gets the call state of the third party after the call has been transferred.

Format:

int SsmSipGetReferStatus(int ch)

Parameter Description:

ch	Channel number

Return Value:

-1	Fail to get the call state.
0	The call has not been transferred
40	The channel is in the state of 'UNUSABLE'.
1	The third party which the call transferred to is in the state of 'Trying'. That is, the subscription is pending.
2	In the state of 180 RingBack
3	In the state of 200 OK or 3XX
4	In the state of 4XX, 5XX or 6XX which indicates the failure

Function Description:

Gets the call state of the third party when the board invokes the function SsmIpInitiateTransfer to transfer the call to the third party.

Note:

- This function is valid only after the call has been transferred by invoking the function SsmIpInitiateTransfer.

Related Information:

Driver version	SynCTI Ver. 5.1.0.0 or above
Header	Shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

2.22.1.17 SsmSipSendRequest

Refer to SsmSipSendRequestA.

2.22.1.18 SsmSipSendRequestA

Sends the Notify or Option message in the state of calling.

Format:

int WINAPI SsmSIPSendRequest(int ch, LPSTR SipMessageType, char SipMessageNewHeaders[SipMessageHeadersNum][SipMessageHeadersLen], int SipMessageNewHeadersNum, LPSTR SipMessageBody)

int WINAPI SsmSIPSendRequestA(int nCh, LPSTR pSipMessageType, char **pSipMessageNewHeaders, int nSipMessageNewHeadersNum, LPSTR pSipMessageBody)

Parameter Description:

ch nCh	Channel number
SipMessageType pSipMessageType	Message type, i.e. Notify or Option
SipMessageNewHeaders pSipMessageNewHeaders	Content of the message header
SipMessageNewHeadersNum nSipMessageNewHeadersNum	Number of the message headers
SipMessageBody pSipMessageBody	Message body

Return Value:

-1	Call failed
0	Successful

Function Description:

Sends the Notify or Option message in the state of calling.

Note:

- Sending Notify message

  The Notify message is composed of two parts: message header and message body. There are two methods to store the message header in the SynSip layer. One is to store it in common fields, such as “from”, “to” and “contact”, which are usually unnecessary to be modified and will be added automatically when the message is sent. For example, the field “Content-Type” will be by default included in the Notify message when it is sent. At present, this function only supports the modification of three fields “Content-Type”, “to” and “request-uri”. The other method is to store the message header in special fields like “P-Preferred-Identity” and “Event” which are all put in the same header field list and share one Synsip

interface.

- Sending Option message

  The Option message only has the message header. The way to insert and modify header fields is the same as that for the Notify message.

Related Information:

Driver version	SynCTI Ver. 5.3.2.2 or above for SsmSIPSendRequest SynCTI Ver. 5.3.2.6 or above for SsmSIPSendRequestA
Header	Shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

2.22.1.19 SsmSetHangupReason

Sets the hangup reason for SIP channels in the Ringing state.

Format:

int WINAPI SsmSetHangupReason(int ch, int nReason)

Parameter Description:

ch	Channel number
nReason	4xx, 5xx, 6xx SIP message

Return Value:

-1	Failed
0	Successful

Function Description:

Sets the hangup reason for SIP channels in the Ringing state.

Note:

Related Information:

Driver version	SynCTI Ver. 5.3.2.2 or above
Header	Shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

2.22.1.20 SsmSipGetBoardRegStatus

Queries the registration status of the board.

Format:

int SsmSipGetBoardRegStatus(int nBId, LPSTR pszRegFailInfo)

Parameter Description:

nBId	The ID number of the board whose registration status is to be queried
pszRegFailInfo	The information returned upon registration failure

Return Value:

-1	Registration failed or this board is not a VoIP board
≥1	Registration successful

Function Description:

Gets the registration status returned upon last Register request of the board. The parameter pszRegFailInfo returns the failure reason of registration, and no additional information will appear in case of a successful registration. If this parameter is set to NULL, it will not return the failure reason of registration.

Note:

- In SIP protocol, no signaling message can query the registration status in real time; therefore, this function can only obtain the registration status returned upon last Register request.
- To improve the validity of the registration status obtained by this function, you may shorten the period of the registration expiration. For example, if the registration expiration is 300 (s), the worst situation is that the function gets the registration status 5 minutes ago.

Related Information:

Driver version	SynCTI Ver. 5.1.0.0 or above
Header	Shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

2.22.1.21 SsmSipGetChRegStatus

Queries the registration status of the channel.

Format:

int SsmSipGetChRegStatus(int nChId, LPSTR pszRegFailInfo)

Parameter Description:

nChId	The ID number of the channel whose registration status is to be required
pszRegFailInfo	The information returned upon registration failure

Return Value:

-1	Registration failed or this channel is not an IP channel
≥1	Registration successful

Function Description:

Gets the registration status returned upon last Register request of the channel. The parameter pszRegFailInfo returns the failure reason of registration, and no additional information will appear in case of a successful registration. If this parameter is set to NULL, it will not return the failure reason of registration.

Note:

- In SIP protocol, no signaling messages can query the registration status in real time; therefore this function can only obtain the registration status returned upon last Register request.
- To improve the validity of the registration status obtained by this function, you may shorten the period of the registration expiration. For example, if the registration expiration is 300 (s), the worst situation is that the function gets the registration status 5 minutes ago.

Related Information:

Driver version	SynCTI Ver. 5.1.0.0 or above
Header	Shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

2.22.1.22 SsmSetRcvRegisterCallBack

Sets the callback function upon the reception of registration message.

Format:

int WINAPI SsmSetRcvRegisterCallBack(RCVREGISTER RcvRegisterCallBack)

Parameter Description:

RcvRegisterCallBack	Sets the callback function upon the reception of registration message. **Format:** int WINAPI RcvRegisterCallBack(char* SipBuf, int SipSize, char* SipBufResponse); in the above format, **SipBuf:** Original data; **SipSize:** Data length; **SipBufResponse:** Content of the reply which is sent after receiving the registration message; **Return Value:** Length of the reply which is sent after receiving the registration message.

Return Value:

-1	Call failed
0	Successful

Function Description:

Sets the callback function upon the reception of registration message.

The parameter RcvRegisterCallBack is used to obtain the original data of the registration message and return its response. The driver will send the content of the parameter SipBufResponse to the initiator of the registration.

Note:

- If the callback function is empty, the protocol stack will return directly and print the error log.

Related Information:

Driver version	SynCTI Ver. 5.3.2.1 or above
Header	Shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

2.22.1.23 SsmIpGetBoardMacAddress

Obtains the MAC address for VoIP boards.

Format:

int SsmIpGetBoardMacAddress(int nBId, PUCHAR pucMacAddrBuff)

Parameter Description:

nBId	The ID number of the VoIP board whose MAC address needs to be obtained
pucMacAddrBuff	The buffer to store the obtained MAC address

Return Value:

-1	Call failed or this is not a B-type VoIP board
6	The length of the MAC address

Function Description:

Obtains the MAC address for the B-type VoIP boards.

The parameter pucMacAddrBuff is used to store the obtained MAC address. If this function call is successful, it will return the length of the MAC address which is 6 bytes of unsigned char type; if this function call is failed, it will return -1.

Note:

- ✧ The VoIP board must be B-type or above; A-type boards cannot obtain the MAC address.
- ✧ This function cannot be invoked before a successful startup of the board.

Related Information:

Driver version	SynCTI Ver. 5.3.1.3 or above

Header	Shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

2.22.1.24 SsmSipRegister

Send the Register requests from SIP channels to the SIP Server.

Format:

int SsmSipRegister(int nRegMode, int nParam1, int nParam2, LPCSTR szDisplayName, LPCSTR szUserName, LPCSTR szPasswd, LPCSTR szAuthUserName, LPCSTR szRegSrvAddr, LPCSTR szRealm, LPCSTR szOutboundProxy, int nExpires)

Parameter Description:

nRegMode	Register Mode. Range of value: nRegMode=0, nRegMode=3
nParam1	The start channel number
nParam2	The end channel number
szDisplayName	Displayed name
szUserName	Registration username
szPasswd	Registration password
szAuthUserName	Authentication username (It is necessary only when the SIP Server requires authentication for registration)
szRegSrvAddr	Registrar IP address (proxy IP address)
szRealm	Registrar alias
szOutboundProxy	Outbound proxy IP address of the registration
nExpires	Registry validity period. The default value is 3600s. Once it is overdue, a new registration will be initiated.

Return Value:

-1	Call failed
0	Call successful

Function Description:

Send the Register requests from SIP channels to the SIP Server.

Note:

- The failure of the function call indicates that either the register parameters or the channel state does not meet the requirements which results in the SynSipStack not sending the register message. However, it does not mean the failure of the registration.
- Invoking this function on an SIP channel in a state none of 'IDLE', 'UNUSABLE' and 'registration failed' will return -1.
- If the SIP channel has been registered, only the Registration update message (actually involving the change of the parameter 'Expires' only) can be sent. That is,
  (1) Expires=0 indicates the cancellation of the existing registration. All parameters except Expires should keep the same as what they were when registered.
  (2) Expires≠0 indicates the resending of the registration message. All parameters except Expires should keep the same as what they were when registered.
  (3) After the registration is cancelled, 'Identity' is updated to 'sip:Username@IPAdress'. Also the IP address can be used to make a call.
- If nRegMode=3, invoking the function will add the authentication information but not initiate a registration;
- If nRegMode=4, invoking the function will remove the authentication information but not initiate a

registration.

Related Information:

Driver version	SynCTI Ver. 5.3.2.6 or above
Header	Shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

2.22.1.25 SsmSipSetConnectionInforOfSDP

Sets the IP address in the Connectioninformation field in the SDP message body of the Invite or 200 message for SIP and informs the remote end to send the RTP data to this IP address. Usually, it is an IP address of the LAN gateway (The port number is allocated by the driver automatically and does not need to be set).

Format:

int WINAPI SsmSipSetConnectionInforOfSDP(char * pConnectionInfo)

Parameter Description:

pConnectionInfo	IP address in the filed ConnectionInformation

Return Value:

-1	Call failed
0	Call successful

Function Description:

Sets the IP address in the Connectioninformation field in the SDP message body of the Invite or 200 message for SIP and informs the remote end to send the RTP data to this IP address. Usually, it is an IP address of the LAN gateway (The port number is allocated by the driver automatically and does not need to be set).

Note:

- The configuration item MapIP can implement the same feature;
- If the parameter pConnectionInfo is set to NULL, this function is invalid and the driver will fill in the IP address in the ConnectionInformation field automatically.

Related Information:

Driver version	SynCTI Ver. 5.3.2.3 or above
Header	Shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

2.22.1.26 SsmSipSetMsgFieldParameter

Sets the content of a designated field in a SIP message.

Format:

int WINAPI SsmSipSetMsgFieldParameter(int nCh, int nMsgType, DWORD dwParam, LPCSTR h_Name, LPCSTR h_Value)

Parameter Description:

nCh	Channel number
nMsgType	Message type, 0 means REFER, 1 means INVITE
dwPara	Reserved parameter, with the default value of 0
h_Name	Field name
h_Value	Field content

Return Value:

-1	Call failed
0	Call successful

Function Description:

Sets the content of a designated field in a SIP message.

Note:

- Currently, this function only supports setting the url of the 'Request', 'From' and 'To' fields in the Refer or INVITE message.
- Each channel can invoke this function at most 10 times to modify the SIP messages.

Related Information:

Driver version	The Refer message requires SynCTI Ver. 5.3.2.5 or above, and the INVITE message requires SynCTI Ver. 5.4.0.0 or above
Header	Shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

2.22.1.27 SsmSipChHold

Sends the Call Hold or Call UnHold request from SIP channels to the remote end.

Format:

int WINAPI SsmSipChHold(int ch, BOOL bHold)

Parameter Description:

ch	Channel number
bHold	TRUE：send the Call Hold request FALSE：send the Call UnHold request

Return Value：

-1	Call failed
0	Call successful

Function Description:

Sends the Call Hold or Call UnHold request from SIP channels to the remote end.

Note:

- This function can be invoked only when the channel stays in the 'Talking' state.

Related Information:

Driver version	SynCTI Ver. 5.3.2.7 or above
Header	Shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

2.22.1.28 SsmSipChTransfer

Sends the Call Transfer request from SIP channels to the remote end during an ongoing call.

Format:

int WINAPI SsmSipChTransfer(int ch, int nReferToCh)

Parameter Description:

ch	Channel number
nReferToCh	The local channel number that the call transferring target corresponds to

Return Value：

-1	Call failed
0	Call successful

Function Description:

Sends the Call Transfer request from SIP channels to the remote end during an ongoing call. Suppose the remote device A is talking with the channel designated by the parameter ch, and meanwhile the channel designated by the parameter nReferToCh is talking with the device C. You can call this function to transfer the call from A to C and finally achieve the talking between A and C.

Note:

- This function can be invoked only when the channel designated by ch is in the 'Holding' state and the channel designated by nReferToCh stays in the 'Talking' state.
- During the call between the channel designated by nReferToCh and the device C, only if the former is the calling party will the call transfer succeeds.
- As a successful call transfer requires the channel designated by ch to be in the call-holding state, this function must be called after SsmSipChHold.

Related Information:

Driver version	SynCTI Ver. 5.3.2.7 or above
Header	Shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

2.22.1.29 SsmSipSetConnectionInforOfSDPEx

Sets the IP address in the Connectioninformation field in the SDP message body of the Invite or 200 message for SIP and informs the remote end to send the RTP data to this IP address. Usually, it is an IP address of the LAN gateway. (The port number is allocated by the driver automatically and does not need to be set.)

Format:

int WINAPI SsmSipSetConnectionInforOfSDPEx(int nCh, char * pConnectionInfo)

Parameter Description:

nCh	Channel number
pConnectionInfo	IP address in the filed ConnectionInformation

Return Value:

-1	Call failed
0	Call successful

Function Description:

Sets the IP address in the Connectioninformation field in the SDP message body of the Invite or 200 message for SIP and informs the remote end to send the RTP data to this IP address. Usually, it is an IP address of the LAN gateway. (The port number is allocated by the driver automatically and does not need to be set.)

Note:

- If the parameter pConnectionInfo is set to NULL, this function is invalid and the driver will fill in the IP address in the ConnectionInformation field automatically.

Related Information:

Driver version	SynCTI Ver. 5.3.3.0 or above
Header	Shpa3api.h

Library	shp_a3.lib
DLL	shp_a3.dll

2.22.1.30 SsmSipSetContactSection

Designate the host and port of the 'Contact' field in the invite, 180 and 200 messages which are sent out from the SIP channels.

Format:

int WINAPI SsmSipSetContactSection(int nCh, char *szContactSection)

Parameter Description:

ch	Channel number
szContactSection	IP address and port, in the format of ipAddr.port

Return Value:

-1	Call failed
0	Call successful

Function Description:

Designate the host and port of the 'Contact' field in the invite, 180 and 200 messages which are sent out from the SIP channels.

Note:

- If szContactSection is NULL, the host and port of the 'Contact' field will be the IP and port for board signaling.

Related Information:

Driver version	SynCTI Ver. 5.3.3.0 or above
Header	Shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

2.22.1.31 SsmSipChEnableRtpStun

Sets whether to enable the RTP traversal feature on those SIP channels within the range from channel nChFrom to channel nChTo.

Format:

int WINAPI SsmSipChEnableRtpStun(int nChFrom, int nChTo, BOOL bRtpStun)

Parameter Description:

nChFrom	The start channel number
nChTo	The end channel number
bRtpStun	Indicates whether to enable the RTP traversal feature

Return Value:

-1	Call failed
0	Call successful

Function Description:

Sets whether to enable the RTP traversal feature on those SIP channels within the range from channel nChFrom to channel nChTo.

Note: None

Related Information:

Driver version	SynCTI Ver. 5.3.3.0 or above
Header	Shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

2.22.1.32 SsmSipOutCallSendOptions

Sends the Option message to the SIP server from the SynSIP protocol stack.

Format:

int WINAPI SsmSipOutCallSendOptions(char *pszSipServerIP)

Parameter Description:

pszSipServerIP	IP address of the SIP server, containing the port.

Return Value:

-1	Call failed
0	Call successful

Function Description:

Sends the Option message to the SIP server from the SynSIP protocol stack.

Note: None

Related Information:

Driver version	SynCTI Ver. 5.3.3.2 or above
Header	Shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

2.22.1.33 SsmSipChEnableSrtp

Srtp encryption for SIP channels.

Format:

int WINAPI SsmSipChEnableSrtp (intnChFrom, intnChTo, BOOL bSrtp)

Parameter Description:

nChFrom	Start channel number
nChTo	End channel number
bSrtp	Srtp encryption, TRUE: Enable; FALSE: Disable

Return Value:

-1	Failed
0	Successful

Function Description:

Enables/disables Srtp encryption for SIP channels.

Note: None

Related Information:

Driver version	SynCTI Ver. 5.4.3.0 or above
Header	Shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

2.22.1.34 SsmSipSetMultiNetIP

Sets the IP addresses used for outgoing calls.

Format:

int WINAPI SsmSipSetMultiNetIP (int nCh, char *szSipSetMultiNetIP)

Parameter Description:

nCh	Channel Number
szSipSetMultiNetIP	IP Address

Return Value:

-1	Failed
0	Successful

Function Description:

Sets the IP addresses used for outgoing calls.

Note:

- The dual NIC feature must be enabled before using this function. That is configuration item SipMutiIpOn must be set to 1.

Related Information:

Driver version	SynCTI Ver. 5.4.0.0 or above
Header	Shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

2.22.1.35 SsmSipOutCallSendNotify

Sets the contents of the Notify message sent out by the SIP protocol stack.

Format:

int WINAPI SsmSipOutCallSendNotify (LPSTR pszFrom,LPSTR pszTo,LPSTR *pszHeader,int nHeaderNum,LPSTR pszBody,int nTransType)

Parameter Description:

pszFrom	Designates the contents of the From field in the Notify message
pszTo	Designate the contents of the To field in the Notify message as well as the destination address of the Notify message
pszHeader	Array pointer of the message header
nHeaderNum	Numbers of the message header
pszBody	Designates the content of the Notify message
nTransType	Transfer protocol type, 0: UDP, 1: TCP

Return Value:

-1	Call failed
0	Call successful

Function Description:

Sets the contents of the Notify message sent out by the SIP protocol stack.

Related Information:

Driver version	SynCTI Ver. 5.4.1.0 or above
Header	Shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

2.22.1.36 SsmSipSetTransportProtocol

Designates the transfer protocol for the SIP message.

Format:

int WINAPI SsmSipSetTransportProtocol(int nCh,int nTransType)

Parameter Description:

nCh	Channel Number
nTransType	Transfer protocol type, 1: TCP, 0: UDP

Return Value:

-1	Failed
0	Successful

Function Description:

Designates the transfer protocol for the SIP message.

Note: None

Related Information:

Driver version	SynCTI Ver. 5.4.1.0 or above
Header	Shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

2.22.1.37 SsmSipChHoldA

Sends the Call Hold or Call UnHold request from SIP channels to the remote end.

Format:

int WINAPI SsmSipChHoldA(int ch)

Parameter Description:

ch	Channel Number

Return Value:

-1	Failed
0	Successful

Function Description:

Sends the Call Hold or Call UnHold request from SIP channels to the remote end.

Note:

- This function can be invoked only when the channel stays in the ‘Talking’ state.

Related Information:

Driver version	SynCTI Ver. 5.4.2.0 or above
Header	Shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

2.22.1.38 SsmSipChTransferA

Sends the Call Transfer request from SIP channels to the remote end during an ongoing call.

Format:

int WINAPI SsmSipChTransferA(int ch)

Parameter Description:

ch	Channel Number

Return Value:

-1	Failed
0	Successful

Function Description:

Sends the Call Transfer request from SIP channels to the remote end during an ongoing call.

Note:

- The channel designated by ch must be in 'Talking' state;
- As a successful call transfer requires the channel designated by ch to be in the call-holding state, this function must be called after SsmSipChHoldA.

Related Information:

Driver version	SynCTI Ver. 5.4.2.0 or above
Header	Shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

2.22.2 Setting Special API functions for VoIP Resource Boards

2.22.2.1 SsmLockMediaCh

Locks the media channel. If a media channel is idle, locks it and obtains available resources on the channel.

Format:

int SsmLockMediaCh(int ch)

Parameter Description:

ch	Media channel number

Return Value:

-1	Call failed, the failure reason can be obtained by the function SsmGetLastErrMsg
> 0	Call successful

Function Description:

If a media channel is idle, locks it and obtains available resources on the channel. Only the local RTP address and port can be obtained.

Note:

- This channel must be idle. It will go into the state S_IP_MEDIA_LOCK (with the corresponding value of 160) after this function is called successfully. This function is only applicable to VoIP resource boards.

Related Information:

Driver version	SynCTI Ver. 5.2.0.0 or above
Header	Shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

2.22.2.2 SsmGetMediaChParam

Obtains the media parameters. If the media channel is unlocked, obtains the current media parameters; if it is locked, obtains the locked parameters.

Format:

int SsmGetMediaChParam(int ch, struct MediaParam *mParam)

Parameter Description:

ch	Media channel number
mParam	The pointer pointing to the MediaParam structure

Return Value:

-1	Call failed, the failure reason can be obtained by the function SsmGetLastErrMsg
0	Call successful

Function Description:

Obtains media parameters of the specified media channel. If the media channel is locked, only the local IP address and port can be obtained, and other parameters will remain empty or 0.

If the media channel is unlocked, obtains the following parameters: local IP address, local port, remote IP address, remote port, codec, transmit and receive modes.

If the media channel is neither locked nor unlocked, the return value will be -1.

Note:

- This function is only applicable to VoIP resource boards.

Related Information:

Driver version	SynCTI Ver. 5.2.0.0 or above
Header	Shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

2.22.2.3 SsmOpenMediaCh

Opens the media channel.

Format:

int SsmOpenMediaCh(int ch, struct MediaParam *mParam)

Parameter Description:

ch	Media channel number
mParam	The pointer pointing to the MediaParam structure

Return Value:

-1	Call failed, the failure reason can be obtained by the function SsmGetLastErrMsg
0	Call successful

Function Description:

Creates an RTP Session on the specified media channel to transmit and receive voice data. Parameters include transmit and receive modes, local RTP port and address, remote RTP port and address, load type and etc. Meanwhile this function sets the channel state to S_IP_MEDIA_OPEN.

Note:

- This function is only applicable to VoIP resource boards.

Related Information:

Driver version	SynCTI Ver. 5.2.0.0 or above
Header	Shpa3api.h
Library	shp_a3.lib

DLL	shp_a3.dll

2.22.2.4 SsmCloseMediaCh

Closes the media channel.

Format:

int SsmCloseMediaCh(int ch)

Parameter Description:

ch	Media channel number

Return Value:

-1	Call failed, the failure reason can be obtained by the function SsmGetLastErrMsg
0	Call successful

Function Description:

Closes the media channel. That is, sets the channel state to be locked.

Note:

- This function can be invoked successfully only to those opened channels. It closes a voice channel but does not release it. This function is only applicable to VoIP resource boards.

Related Information:

Driver version	SynCTI Ver. 5.2.0.0 or above
Header	Shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

2.22.2.5 SsmUnlockMediaCh

Releases the media channel. When the media channel is locked or open, release it and put it into the idle state.

Format:

int SsmUnlockMediaCh(int ch)

Parameter Description:

ch	Media channel number

Return Value:

-1	Call failed, the failure reason can be obtained by the function SsmGetLastErrMsg
0	Call successful

Function Description:

If the media channel is not in the idle state, release the locked resources on this channel and put it into the idle state.

If the media channel is in the idle state, the return value will be -1.

Note:

- This function is only applicable to VoIP resource boards.

Related Information:

Driver version	SynCTI Ver. 5.2.0.0 or above
Header	Shpa3api.h
Library	shp_a3.lib

DLL	shp_a3.dll

2.22.2.6 SsmUpdateMediaCh

Updates the media channel.

Format:

int SsmUpdateMediaCh (int ch, struct MediaParam *mParam)

Parameter Description:

ch	Media channel number
mParam	The pointer pointing to the MediaParam structure

Return Value:

-1	Call failed, the failure reason can be obtained by the function SsmGetLastErrMsg
0	Call successful

Function Description:

Updates the RTP Session on the specified media channel. Parameters include transmit and receive modes, local RTP port and address, remote RTP port and address, load type and etc.

Note:

- This function is valid only when the media channel is opened. It is only applicable to VoIP resource boards.

Related Information:

Driver version	SynCTI Ver. 5.2.0.0 or above
Header	Shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

2.22.2.7 SsmCheckMediaChRTPTimeOut

Sets the timeout value for a specified media channel to receive RTP data from the remote end or checks whether the RTP data reception is overtime.

Format:

int WINAPI SsmCheckMediaChRTPTimeOut(int ch, int nCheckTime)

Parameter Description:

ch	Channel number
nCheckTime	Timeout value, calculated by s.

Return Value:

-1	Call failed
0	Call successful (for the first time this function being invoked upon driver startup) or RTP data reception is not overtime
1	RTP data reception is overtime

Function Description:

For the first time being invoked upon driver startup, this function is used to set the timeout value for a specified media channel to receive RTP data from the remote end. For subsequent calls of this function, it is used to check whether the RTP data reception is overtime.

Note:

- This function can be invoked multiple times, but it sets the timeout value for a specified media channel to receive RTP data from the remote end only at the first time being invoked upon driver startup. For subsequent calls of this function, the parameter nCheckTime will be ignored.

Related Information:

Driver version	SynCTI Ver. 5.3.2.6 or above
Header	Shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

2.22.3 Setting Special API functions for NAT Traversal

2.22.3.1 SsmIPGetStunPublicIP

Performs NAT detection to obtain the IP address and the port of the public network.

Format:

int SsmIPGetStunPublicIP(int nBid,WORD nLocalPort,int nStunServerPort,LPSTR pstrStunServer,StunAddress4 *stunServerAdd,int flag)

Parameter Description:

nBid	Board ID
nLocalPort	Local port
nStunServerPort	Port of Stun Server
pstrStunServer	IP address or domain name of Stun Server
stunServerAdd	Returned IP address and port of public network
flag	Steps of detection

Return Value:

-1	Call failed, the failure reason can be obtained by the function SsmGetLastErrMsg
≥0	Detected NAT type. 0: unknown; 1: no NAT; 2: ConeNat; 3: RestrictedNat; 4: PortRestrictedNat; 5: Symmetric NAT; 6: Symmetric NAT with firewall; 7: can't detect over (fail to send detect message); 8: fail to detect (No reply from the stun server)

Function Description:

Performs NAT detection on VoIP boards or VoIP resource boards, using the Stun protocol. If flag=0, it detects the NAT types; if flag=1, it performs the NAT detection of the IP address and the port of the public network. The former detection costs more time.

Note:

- To perform RTP traversal on the SHN series VoIP boards, set the corresponding configuration item to enable the RTP traversal feature and invoke this function. In the subsequent calls, the RTP stun will be detected automatically and this function needn't be invoked again.
- The stun traversal includes the signaling traversal and the RTP traversal which should be performed respectively. To enable the signaling traversal, add two configuration items EnableSIPStun and SIPStunServerIP to the SIP section; to enable the RTP traversal, add two configuration items EnableRTPStun and StunServerIP to the BoardID section.
- If the function returns -1, 0, 1, 5, 6, 7 or 8, (i.e. fail to obtain the stun type), both the calling and called parties of the board will prompt error.

Related Information:

Driver version	SynCTI Ver. 5.2.0.0 or above
Header	Shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

2.22.4 Setting Special API functions for SHN B-type/C-type Boards

2.22.4.1 SsmCheckBoardIcmp

Enables or disables the ICMP feature of the build-in network interface for B-type/C-type VoIP boards.

Format:

int SsmCheckBoardIcmp(int BoardID, char *sDestAddr, BOOL bRunIcmp)

Parameter Description:

BoardID	Board ID
sDestAddr	Destination address
bRunIcmp	ICMP feature enabling signal. bRunIcmp=1: Enable; bRunIcmp=0: Disable

Return Value:

-1	Call failed
0	Call successful

Function Description:

Enables or disables the ICMP feature of the build-in network interface for B-type/C-type VoIP boards.

Note:

- The driver will throw out the event E_BOARD_ICMP_CHANGE to the application once the ICMP result (ICMP normal or ICMP abnormal) changes. The default ICMP result is abnormal.

Related Information:

Driver version	SynCTI Ver. 5.4.4.0 or above
Header	Shpa3api.h
Library	shp_a3.lib, Synsip.lib
DLL	shp_a3.dll, Synsip.dll, M537.dll
BIN	Shn537.bin

2.22.4.2 SsmSipStackAddWhiteOrBlackList

Adds the white or black list.

Format:

int WINAPI SsmSipStackAddWhiteOrBlackList(char *pszSipMsgIp,intnWhiteOrBlack)

Parameter Description:

pszSipMsgIp	IP address
nWhiteOrBlack	0: Black list; 1: White list

Return Value:

-1	Call failed
0	Call successful

Function Description:

Adds the white or black list.

Note:

- Sets neither black list nor white list, all the IPs can receive the SIP message;

- Sets white list only and only the IPs in it can receive the SIP message;
- Sets black list only and the IPs excluding those in it can receive the SIP message;
- Sets both black list and white list, the IPs excluding those in the black list can receive the SIP message;
- Each black or white list can accommodate up to 256 IPs.
- The IPs in the black list cannot be the same as those in the white list; otherwise, the function call may probably fail..

Related Information:

Driver version	SynCTI Ver. 5.4.3.0 or above
Header	Shpa3api.h
Library	shp_a3.lib, Synsip.lib
DLL	shp_a3.dll, Synsip.dll

2.22.4.3 SsmSipStackRemoveWhiteOrBlackList

Removes IP addresses from the white or black list.

Format:

int WINAPI SsmSipStackRemoveWhiteOrBlackList(char *pszSipMsgIp,intnWhiteOrBlack)

Parameter Description:

pszSipMsgIp	IP address
nWhiteOrBlack	0: Black list; 1: White list

Return Value:

-1	Call failed
0	Call successful

Function Description:

Removes IP addresses from the white or black list.

Note: None

Related Information:

Driver version	SynCTI Ver. 5.4.3.0 or above
Header	Shpa3api.h
Library	shp_a3.lib, Synsip.lib
DLL	shp_a3.dll, Synsip.dll

2.22.5 Setting Functions for VoIP Board Registration Server

2.22.5.1 SsmAutoDialAgent

Submits the AutoDial task to the driver to start an outgoing call.

Format:

int SsmAutoDialAgent(int ch, LPSTR szCallerPhoNum, LPSTR szCalleePhoNum)

Parameter Description:

ch	Channel Number
szCallerPhoNum	The pointer pointing to the buffer storing the calling party number
szCalleePhoNum	The pointer pointing to the buffer storing the called party number

Return Value:

-1	Failed. The failure reason can be acquired from the function SsmGetLastErrMsg.
0	Successful.

Function Description:

Submits the AutoDial task to the driver to start an outgoing call.

After the AutoDial task is started, the channel state machine automatically initiates an outgoing call connection. For related information, refer to the channel state transfer introductions.

IP Channel

For an IP channel, this function will trigger the driver to use a message specified by the protocol to start an outbound call. To be exact, the INVITE message is used for SIP.

Note:

- This function is only applicable to SIP channels;
- The CalleeID must be the username which has been registered to the IP board in advance;
- The CallerID can be registered or unregistered; The format is "displayname"<sip:user@host:port> if it is unregistered; otherwise, fill in the registered number directly.

Related Information:

Driver version	SynCTI Ver. 5.4.1.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmAutoDial

2.22.5.2 SsmSipRegResponse

Sends the registration authentication information to the driver.

Format:

int SsmSipRegResponse(LPCSTR pszUserName, RegResp *pstRegResp)

Parameter Description:

pszUserName	Registered username
pstRegResp	The pointer pointing to the first address of the buffer area storing the RegResp structure of the authentication information which will be sent to the driver.

Return Value:

-1	Failed.
0	Successful.

Function Description:

The application will invoke this function to send the authentication information of the user to be authenticated to the driver upon receiving the event E_REG_REQUEST.

The RegResp structure is as follows:

```
typedef struct tag_RegResp
{
    int    nAuth;// Authentication parameter, value of 0 or 1
     char szPassword[MAX_SIPIP_ADDR_LENGTH];// User password
     char szUsername[MAX_SIPIP_ADDR_LENGTH];
}RegResp;
```

Related Information:

Driver version	SynCTI Ver. 5.4.1.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Event: E_REG_REQUEST

2.22.5.3 SsmSipGetUserInfoByIndex

Obtains the information of successfully registered users via the registration index.

Format:

int SsmSipGetUserInfoByIndex(int nRegIndex,Reg_Info *pstReg_Info)

Parameter Description:

nRegIndex	Registration Index
pstReg_Info	The pointer pointing to the first address of the buffer area storing the Reg_Info structure of the successfully registered authentication information.

Return Value:

-1	Failed.
0	Successful.

Function Description:

The application can invoke this function according to the event E_REG_REGSTATUS to obtain the user information that corresponds to the registration index.

The Reg_Info structure is as follows:

```
typedef struct tag_Reg_Info
{
    int     nRegIndex;// Index of the registered user
    int     nRegState;// Registration status: 0 failed; 1 successful
    int     nT0State;// State controller of the timer T0
    char    szTelNo[MAX_SIPIP_ADDR_LENGTH];// Registration username
    char    szClientAddress[MAX_SIPIP_ADDR_LENGTH];// IP address of the registered client
    char    szClientPort[MAX_SIPIP_ADDR_LENGTH];// Port number of the registered terminal
    char    szUserID[MAX_SIPIP_ADDR_LENGTH];// Authentication username
    int     nExpires;
    int     nRegCountForAuthT0;// Counter of the timer T0
    int     nRegCountForAuthT1;// Counter of the timer T1
    char    szNonce[MAX_SIPIP_ADDR_LENGTH];

    int    nAuth;// Authentication parameter, value of 0 or 1
    char szPassword[MAX_SIPIP_ADDR_LENGTH];// User password
}Reg_Info;
```

Related Information:

Driver version	SynCTI Ver. 5.4.1.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Event: E_REG_REGSTATUS

2.22.5.4 SsmGetCallNum

Obtains the information of the calling/called party number for the transferred incoming call.

Format:

int SsmGetCallNum(int ch,LPSTR szCallerNum,LPSTR szCalleeNum)

Parameter Description:

ch	Channel Number
szCallerNum	The string pointer pointing to the buffer storing basic information of the original calling party number of the transferred call. The storage space is allocated by the application.
szCalleeNum	The string pointer pointing to the buffer storing basic information of the original called party number of the transferred call. The storage space is allocated by the application.

Return Value:

-1	Failed.
0	Successful.

Function Description:

The application can invoke this function according to the event E_RCV_REFER to obtain the calling/called party information of the corresponding channel.

Related Information:

Driver version	SynCTI Ver. 5.4.1.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Event: E_RCV_REFER

2.22.5.5 SsmSipStackSetUserAgent

Sets the user agent name of the SHN series boards or SynHMP.

Format:

int WINAPI SsmSipStackSetUserAgent(const char * pszUserAgent)

Parameter Description:

pszUserAgent	Name of the user agent

Return Value:

-1	Failed.
0	Successful.

Function Description:

Sets the user agent name of the SHN series boards or SynHMP.

Note:

- The configuration item UserAgent can implement the same feature.

Related Information:

Driver version	SynCTI Ver. 5.4.3.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

2.23 Functions for ATP Series Recording Boards (REC Series)

2.23.1 Input Gain Control of Microphone Channel

2.23.1.1 SsmQueryOpMicGain

Queries if the channel supports setting the gain of the input signal.

Format:

int SsmQueryOpMicGain(int ch)

Parameter Description:

ch	Channel number

Return Value:

-1	Call failed
0	Not support
1	Support

Function Description:

Queries if the channel supports setting the gain of the input signal.

Note:

- Only analog recording channels of ATP Series boards (including analog trunk channel and microphone recording channel) support this feature.

Related Information:

Driver version	SynCTI Ver. 2.1or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmSetMicGain

2.23.1.2 SsmSetMicGain

Sets the gain of the input signal on the analog recording channel.

Format:

int SsmSetMicGain(int ch, int nMicGain)

Parameter Description:

ch	Channel Number
nGain	Gain value: 0: normal gain 1: increase 20DB

Return Value:

-1	Call failed
0	Successful

Function Description:

Sets the gain of the input signal on the analog recording channel.

Note:

- The configuration item MicGain can implement the same features.

- This function is only applicable to analog recording channels of ATP Series boards (excluding ATP-24A series boards).

Related Information:

Driver version	SynCTI Ver. 2.1or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmGetMicGain

2.23.1.3 SsmGetMicGain

Obtains the preset gain value of the input signal on the analog recording channel.

Format:

int SsmGetMicGain(int ch)

Parameter Description:

ch	Channel Number

Return Value:

-1	Call failed
0	Normal gain
1	High gain (20DB)

Function Description:

Obtains the preset gain value of the input signal on the analog recording channel.

Note:

- This function is only applicable to analog recording channels of ATP Series boards.

Related Information:

Driver version	SynCTI Ver. 2.1 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmSetMicGain

2.23.2 MF Detector Functions

2.23.2.1 SsmEnableRxMF

Enables or disables the MF detector.

Format:

int SsmEnableRxMF(int ch, BOOL bRun)

Parameter Description:

ch	Channel Number
bRun	The control switch determining whether to enable the MF detector TRUE: start FALSE: stop

Return Value:

-1	Failed. The failure reason can be acquired via the function SsmGetLastErrMsg
0	Successful

Function Description:

Enables or disables the MF detector.

Note:

- The MF detector is disabled by default. Only if you want to control the switch of the MF detector manually does this function need to be invoked.
- This function is only applicable to ATP Series.
- The configuration item AlwaysEnableRxMF can implement the same feature.
- This function is always valid once it is invoked.

Related Information:

Driver version	SynCTI Ver. 5.3.2.3 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function:

2.23.2.2 SsmClearRxMFBuf

Clears up the MF-reception buffer area in the driver.

Format:

int SsmClearRxMFBuf(int ch)

Parameter Description:

ch	Channel number

Return Value:

-1	Failed. The failure reason can be acquired via the function SsmGetLastErrMsg.
0	Successful

Function Description:

Clears up the MF-reception buffer area in the driver.

Note:

- When the channel goes into the state of idle, the driver automatically clears the buffer area of the MF detector.

Related Information:

Driver Version	SynCTI Ver 5.3.1.1 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function:

2.23.2.3 SsmGetMFStr

Obtains the MF characters saved in the MF-reception buffer area. The characters are in ASCII.

Format:

int SsmGetMFStr(int ch, LPSTR pszMF)

Parameter Description:

ch	Channel number
pszMF	The pointer pointing to the buffer storing the received MF strings. The storage space is allocated by the application

Return Value:

-1	Failed
≥0	The number of digits in the MF detector

Function Description:

Obtains the MF characters saved in the MF-reception buffer area. The characters are in ASCII.

Note:

- While executing this function, the driver will not clear the MF-reception buffer area.

Related Information:

Driver Version	SynCTI Ver 5.3.1.1 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function:

2.24 Functions for DTP Series Boards (REC Series)

2.24.1 State Machine Programming Mode Based Functions

2.24.1.1 Obtaining Basic Information

2.24.1.1.1SpyGetMaxCic

Obtains the total number of the monitored circuits.

Format:

int SpyGetMaxCic()

Parameter Description: none

Return Value:

-1	Call failed
≥0	The total number of SpyCic

Function Description:

Obtains the total number of the monitored circuits, i.e. the set value of the configuration item TotalAppSpyCIC in the section [AppSpyCICTable]. For more information about SpyCic, refer to 'DTP Series' in Chapter 1.

Note:

- This function is only applicable to DTP Series.

Related Information:

Driver version	SynCTI Ver. 3.0 or above
Header	shpa3api.h
Library	shp_a3.lib

DLL	shp_a3.dll

Related Function: SpyChToCic

2.24.1.1.2 SpyChToCic

Searches the corresponding logical number of the SpyCic according to the channel logical number.

Format:

int SpyChToCic(int ch)

Parameter Description:

ch	Channel logical number

Return Value:

-1	Failed
≥0	The logical number of the corresponding SpyCic

Function Description:

Searches the corresponding logical number of the SpyCic according to the channel logical number. For more information about the SpyCic logical number, refer to the 'DTP Series' in Chapter 1.

Note:

- This function is only applicable to DTP Series.

Related Information:

Driver version	SynCTI Ver. 3.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SpyGetMaxCic

2.24.1.2 Obtaining Call Progress Information

2.24.1.2.1 SpyGetState

Obtains the current state of the SpyCic based state machine.

Format:

int SpyGetState(int nCic)

Parameter Description:

nCic	The number of the monitored circuit

Return Value:

Return value equals to -1 means Call failed. The meanings of the other return values are listed below:

Value	MACRO in shpa3api.h	State Description
0	S_SPY_STANDBY	Idle
2	S_SPY_RINGING	Ringing
3	S_SPY_TALKING	Connected (talking). After the SpyCic is transferred to the state of S_SPY_TALKING, the application can do the followings: ✧ Call SpyRecToFile to start recording; ✧ Call SpyGetCalleeId to get the callee ID; ✧ Call SpyGetCallerId to get the caller ID;

		✧ Call the channel based functions, using the channel number of the calling party or called party as the parameter, to get other related information
11	S_CALL_UNAVAILABLE	Channel unavailable
105	S_SPY_RCVPHONUM	Receive callee ID
110	S_SPY_SS1RESET	Circuit reset
111	S_SPY_SS1WAITBWDACK	SS1: waiting for the backward acknowledgement
112	S_SPY_SS1WAITKB	SS1: waiting for the KB signal
Others	Reserved	

Function Description:

Obtains the current state of the SpyCic based state machine.

Note:

- We suggest you use the event of E_CHG_SpyState in stead of this function;
- This function is only applicable to DTP Series.

Related Information:

Driver version	SynCTI Ver. 3.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SpyGetCalleeId, SpyGetCallerId, SpyGetCallOutCh, SpyGetCallInCh

2.24.1.3 Obtaining Call Direction

2.24.1.3.1SpyGetCallInCh

Refer to SpyGetCallOutCh

2.24.1.3.2SpyGetCallOutCh

Obtains the logical number of the called party channel (SpyGetCallOutCh) or the calling party channel (SpyGetCallInCh) in the current call.

Format:

int SpyGetCallInCh(int nCic)

int SpyGetCallOutCh(int nCic)

Parameter Description:

nCic	The logical number of the SpyCic. For more information about SpyCic, refer to 'DTP Series' in Chapter 1.

Return Value:

-1	Call failed
≥0	SpyGetCallInCh: returns the logical number of the calling party channel; SpyGetCallOutCh: returns the logical number of the called party channel

Function Description:

Obtains the logical number of the called party channel (SpyGetCallOutCh) or calling party channel (SpyGetCallInCh) in the current call. For more information about the called party's channel, refer to 'DTP Series' .

Note:

- Only when the state of the SpyCic has been transferred to the state of S_SPY_RINGING or S_SPY_TALKING, calling the function of SpyGetCallOutCh or SpyGetCallInCh can get valid information.
- This function is only applicable to DTP Series.

Related Information:

Driver version	SynCTI Ver. 3.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SpyGetCallerId, SpyGetCalleeId

2.24.1.4 Obtaining Calling and Called Party Numbers

2.24.1.4.1SpyGetCallerId

Refer to SpyGetCalleeId

2.24.1.4.2SpyGetCalleeId

Obtains the calling party number (SpyGetCallerId) or the called party number (SpyGetCalleeId) in the current call.

Format:

int SpyGetCallerId(int nCic, char *pszCallingPartyNumber)

int SpyGetCalleeId(int nCic, char * pszCalledPartyNumber)

Parameter Description:

nCic	The logical number of SpyCic
pszCallingPartyNumber	The pointer pointing to the buffer area storing the calling party number. The storage space is allocated by the application, and it should be no less than 50 characters
pszCalledPartyNumber	The pointer pointing to the buffer area storing the called party number. The storage space is allocated by the application, and it should be no less than 50 characters

Return Value:

-1	Call failed
≥0	The length of the phone number

Function Description:

After the call is established, the function of SpyGetCallerId obtains the calling party number of the current call, and the function of SpyGetCalleeId obtains the called party number of the current call.

Note:

- Only when the state of the SpyCic has been transferred to S_SPY_RINGING or S_SPY_TALKING, calling the function of SpyGetCallerId or SpyGetCalleeId can get the complete information;
- This function is only applicable to DTP Series.

Related Information:

Driver version	SynCTI Ver. 3.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SpyGetCallOutCh, SpyGetCallInCh

2.24.1.5 Obtaining Hangup Information

2.24.1.5.1SpyGetHangupInfo

Obtains the current hangup information based on SpyCic.

Format:

DWORD WINAPI SpyGetHangupInfo(int nCic)

Parameter Description:

nCic	Monitored circuit number

Return Value:

-1	Call failed or no hangup information
Others	The 16 higher bits represent the monitored PCM number of SpyCic which sends the disconnect message. The 16 lower bits imply the hangup information of SpyCic. To be exact: 0: Called party hangs up first 1: Calling party hangs up first

Function Description:

Obtains the current hangup information based on SpyCic.

Note:

- We suggest you use the event E_CHG_SpyHangupInfo instead of this function.
- This function is only applicable to ISDN monitoring for DTP Series.

Related Information:

Driver version	SynCTI Ver. 5.1.1.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SpyGetCalleeId, SpyGetCallerId, SpyGetCallOutCh, SpyGetCallInCh

2.24.1.6 Obtaining Calling and Called Party Number Types

2.24.1.6.1SpyGetCallerType

Refer to SpyGetCalleeType

2.24.1.6.2SpyGetCalleeType

Obtains the type of the calling party number (SpyGetCallerType) or the called party number (SpyGetCalleeType) of the current call.

Format:

int WINAPI SpyGetCallerType(int nCic)

int WINAPI SpyGetCalleeType(int nCic)

Parameter Description:

nCic	The logical number of SpyCic

Return Value:

-1	Call failed
≥0	The type of calling/called party number

Function Description:

After the call is established, the function SpyGetCallerType obtains the type of the calling party number of the current call, while the function SpyGetCalleeType obtains the type of the called party number of the current call.

Note:

- Only when the state of the SpyCic has been transferred to S_SPY_RINGING or S_SPY_TALKING, can you call the function SpyGetCallerType or SpyGetCalleeType to get complete information;
- This function is only applicable to ISDN monitoring for DTP Series.

Related Information:

Driver version	SynCTI Ver. 5.1.0.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SpyGetCallOutCh, SpyGetCallInCh

2.24.1.7 Obtaining Digital Trunk State

2.24.1.7.1 SpyGetLinkStatus

Obtains the operating status of the monitored signaling link.

Format:

int SpyGetLinkStatus(int nSpyPcmNo,UCHAR ucFlag);

Parameter Description:

nSpyPcmNo	The logical number of the monitored PCM, the range of value is determined by the configuration item of TotalSpyPcm
ucFlag	0: Choose the outbound digital trunk 1: Choose the inbound digital trunk

Return Value:

-1	Call failed
0	Error (disconnected)
1	Normal (connected)

Function Description:

Obtains the operating status of the monitored signaling link.

Note:

- This function is only applicable to DTP Series.

Related Information:

Driver version	SynCTI Ver. 3.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function:

2.24.1.8 Obtaining Original Signaling Messages from SS7 Call State Machine

2.24.1.8.1SsmGetSs7SpyMsu

Obtains the original signaling messages synchronously output by the call state machine of SS7 signaling.

Format:

int SsmGetSs7SpyMsu (PUCHAR* ppucMsuBuf)

Parameter Description:

ppucMsuBuf	The pointer pointing to the buffer area storing original ISUP/TUP messages. The storage space is allocated by the application, and it can't be less than 273 bytes

Return Value:

0	No messages
>0	Returns the length (bytes) of the message
-1	Call failed

Function Description:

Obtains the original signaling messages synchronously output by the call state machine of SS7 signaling. The way of taking out the messages is 'First in, first out", i.e. always return the earliest received message in the buffer. Every time when the driver receives an MSU message, it will put the message into its internal buffer storing MSU messages and then throw out the event E_RCV_Ss7SpyMsu to the application.

Note:

- Only when the configuration item bOpenSpySS7Msu is set to 1 will the driver copy the received ISUP/TUP messages to the internal buffer area and throw out the event E_RCV_Ss7SpyMsu to the application.

Related Information:

Driver version	SynCTI Ver. 4.7.2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: none

Related Event: E_RCV_Ss7SpyMsu

2.24.1.9 Obtaining Connection Number

2.24.1.9.1SpyGetConId

Obtains the connection number of the current call.

Format:

int WINAPI SpyGetConId (int nCic, char *pcCid)

Parameter Description:

nCic	The logical number of SpyCic
pcCid	The pointer pointing to the buffer area storing connection numbers. The storage space, no less than 50 characters, is allocated by the application.

Return Value:

≥0	Returns the length of the number
-1	Call failed.

Function Description:

After the call is established, obtains the connection number of this call.

Note:

- If the state of SpyCic has been transferred to S_SPY_TALKING, only by invoking the function SpyGetConId can you get the complete information.
- This function is only applicable to ISDN monitoring for DTP Series.

Related Information:

Driver version	SynCTI Ver. 5.1.0.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: none

2.24.1.10 Obtaining Original Message of L2 in ISDN Protocol

2.24.1.10.1 SsmGetIsdnL2SpyMsu

Obtains an original monitoring message of L2 in ISND protocol from the internal buffer area of the driver.

Format:

int WINAPI SsmGetIsdnL2SpyMsu(int nPcmId, int nBufLen,PUCHAR pucMsuBuf);

Parameter Description:

nPcmId	The logical number of the digital trunk
nBufLen	The size of pucMsuBuf provided by the application.
pucMsuBuf	Returns the original message of L2 in ISDN protocol. The storage space of pucMsuBuf is allocated by the application,

Return Value:

0	No message in the buffer area
>0	The actual length (bytes) of the message in pucMsuBuf
-1	Call failed

Function Description:

Obtains an original monitoring message of L2 in ISND protocol from the internal buffer area of the driver.

Note:

- In real practice, this function call gets valid only when the configuration item bOpenSpyIsdnL2 is set to 1.

Related Information:

Driver version	SynCTI Ver. 5.3.3.1 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Event: E_RCV_IsdnL2SpyMsu

2.25 Functions for DST Series Recording Boards (REC Series)

2.25.1 SsmGetDstChSNRofUplink

Obtains the signal-to-noise ratio of uplink signals on a specified channel of the DST board.

Format:

int SsmGetDstChSNRofUplink(int ch)

Parameter Description:

ch	Channel number

Return Value:

-1	Failed
≥0	Value of the signal-to-noise ratio

Function Description:

Obtains the signal-to-noise ratio of uplink signals on a specified channel of the DST board, to monitor the signal quality on the line.

Note:

- This function only supports DST Series B-type boards.

Related Information:

Driver version	SynCTI Ver. 5.1.0.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmGetDstChSNRofDownlink

2.25.2 SsmGetDstChSNRofDownlink

Obtains the signal-to-noise ratio of downlink signals on a specified channel of the DST board.

Format:

int SsmGetDstChSNRofDownlink(int ch)

Parameter Description:

ch	Channel number

Return Value:

-1	Failed
≥0	Value of the signal-to-noise ratio

Function Description:

Obtains the signal-to-noise ratio of downlink signals on a specified channel of the DST board, to monitor the signal quality on the line.

Note:

- This function only supports DST Series B-type boards.

Related Information:

Driver version	SynCTI Ver. 5.1.0.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmGetDstChSNRofUplink

2.25.3 SsmGetDstChVoltageState

Obtains the signal level on a specified channel of the DST board.

Format:

int SsmGetDstChVoltageState (int ch)

Parameter Description:

ch	Channel number

Return Value:

-1	Failed
0	Normal
3	Abnormal

Function Description:

Obtains the signal level on a specified channel of the DST board.

Note:

- This function is only supported for DST Series B-type boards.

Related Information:

Driver version	SynCTI Ver. 5.2.0.1 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function:

2.26 Functions for PCM1280E Recording Board (REC Series)

2.26.1 SsmGetPcm32LineState

Obtains the line synchronization status and the signal state.

Format:

int WINAPI SsmGetPcm32LineState(int ch)

Parameter Description:

ch	Channel number

Return Value:

-1	Call failed
≥0	Only the lower 16 bits are used to represent the line synchronization status and the signal state, among which bit0-bit7 indicate the line synchronization status and bit8-bit15 indicate the signal state. See below for details. a) Value range for line synchronization status: 1) 0 - frame synchronization normal 2) 1 - disconnected 3) 2 - reset state 4) 3 - frame synchronization abnormal b) Value range for signal state: 1) 0 - data normal

	2) 1 - disconnected or no data 3) 2 - reset state 4) 3 - data abnormal

Function Description:

Obtains the line synchronization status and the signal state.

Note:

- This function is only supported for the PCM1280E board.

Related Information:

Driver version	SynCTI Ver. 5.3.1.2 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function:

2.27 Functions for IPR Series Products (REC Series)

2.27.1 Functions for SynIPRecorder

2.27.1.1 SsmIPRSetRecVolume

Sets the recording volume for primary and secondary.

Format:

int SsmIPRSetRecVolume(int nChId, int nPrimaryVlm, int nSecondaryVlm)

Parameter Description:

nChId	Channel number
nPrimaryVlm	Recording volume for primary, range of value is -7~+6. -7 implies the volume adjuster is stopped. Other values represent the volume gains (the volume value multiplies 3 being the decibel value (db)), those greater than 0 indicating the increase in volume and those less than 0 reflecting the decrease in volume
nSecondaryVlm	Recording volume for secondary, range of value is -7~+6. -7 implies the volume adjuster is stopped. Other values represent the volume gains (the volume value multiplies 3 being the decibel value (db)), those greater than 0 indicating the increase in volume and those less than 0 reflecting the decrease in volume

Return Value:

-1	Call failed
≥0	Call successful

Function Description:

Sets the recording volume for primary and secondary.

Note:

- This function only supports SynIPRecorder.

Related Information:

Driver version	SynCTI Ver.5.3.2.0
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function:

2.27.1.2 SsmIPRSetMixerType

Sets the mixer type for the recording.

Format:

int SsmIPRSetMixerType(int nChId, int nFlag)

Parameter Description:

nChId	Channel number
nFlag	0: record the Primary only; 1: record the Secondary only; 2: record both parties

Return Value:

-1	Call failed
≥0	Call successful

Function Description:

Sets the mixer type for the recording.

Note:

- This function only supports SynIPRecorder.

Related Information:

Driver version	SynCTI Ver.5.3.2.0
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmIPRGetMixerType

2.27.1.3 SsmIPRGetMixerType

Obtains the channel mixer type.

Format:

int SsmIPRGetMixerType(int nChId)

Parameter Description:

nChId	Channel number

Return Value:

-1	Call failed
0	Record the Primary only
1	Record the Secondary only
2	Record both parties

Function Description:

Obtains the channel mixer type.

Note:

- This function only supports SynIPRecorder.

Related Information:

Driver version	SynCTI Ver.5.3.2.0
Header	shpa3api.h
Library	shp_a3.lib

DLL	shp_a3.dll

Related Function: SsmIPRSetMixerType

2.27.1.4 SsmIPRGetRecSlaverCount

Obtains the count of recording Slavers.

Format:

int SsmIPRGetRecSlaverCount(int nBId)

Parameter Description:

nBId	Board ID

Return Value:

<0	Call failed
>=0	Call successful, the return value is the count of the current recording Slavers.

Function Description:

Obtains the count of recording Slavers.

Note:

- This function only supports SynIPRecorder.

Related Information:

Driver version	SynCTI Ver.5.3.2.0
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmIPRGetRecSlaverList, SsmIPRGetRecSlaverInfo

2.27.1.5 SsmIPRGetRecSlaverList

Obtains the information about a specified number of recording Slavers.

Format:

int SsmIPRGetRecSlaverList(int nBId, int nRecSlaverNum, int* nReturnSlaverNum, PIPR_SLAVERADDR pIPR_SlaverAddr)

Parameter Description:

nBId	Board ID
nRecSlaverNum	The number of recording Slavers required to get information
nReturnSlaverNum	The number of valid Slavers obtained
pIPR_SlaverAddr	As the initial address of structure array of PIPR_SLAVERADDR, pIPR_SlaverAddr is used to return the address information of each Slaver.

Return Value:

<0	Call failed
>=0	Call successful

Function Description:

Obtains the information about a specified number of recording Slavers.

Note:

- This function only supports SynIPRecorder.

Related Information:

Driver version	SynCTI Ver.5.3.2.0

Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmIPRGetRecSlaverCount, SsmIPRGetRecSlaverInfo

2.27.1.6 SsmIPRGetRecSlaverInfo

Obtains the detailed information of a certain recording Slaver.

Format:

int SsmIPRGetRecSlaverInfo(int nBId, int nRecSlaverId, BOOL *bStarted, int *nTotalResources, int *nUsedResources, int *nThreadPairs, PCOMPUTER_INFO pcomputerinfo)

Parameter Description:

nBId	Board ID
nRecSlaverId	ID of the recording Slaver required to get information
bStarted	Whether the Slaver is started, that is whether it is in the working state
nTotalResources	Total number of the recording resources in the recording Slaver
nUsedResources	Number of used recording resources in the recording Slaver
nThreadPairs	Number of thread pairs in the recording Slaver
pcomputerinfo	Pointer pointing to the structure COMPUTER_INFO, where stores the hardware information of the machine the Slaver is in.

Return Value:

<0	Call failed
>=0	Call successful

Function Description:

Obtains the detailed information of a certain recording Slaver.

Note:

- This function only supports SynIPRecorder.

Related Information:

Driver version	SynCTI Ver.5.3.2.0
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmIPRGetRecSlaverCount, SsmIPRGetRecSlaverList

2.27.1.7 SsmIPRStartRecSlaver

Starts the recording Slaver properly to enter the working state.

Format:

int SsmIPRStartRecSlaver(int nBId, int nRecSlaverId, int* nTotalResources, int *nThreadPairs)

Parameter Description:

nBId	Board ID
nRecSlaverId	ID of the recording Slaver required to get information
nTotalResources	Total number of the recording resources in the recording Slaver
nThreadPairs	Number of thread pairs in the recording Slaver, evaluated according to the number of CPUs. The maximum value is 64 and the recommended value is the same as the number of CPUs

Return Value:

<0	Call failed

>=0	Call successful

Function Description:

Starts the recording Slaver properly to enter the working state.

Note:

- This function only supports SynIPRecorder;
- The successful return of the function can only demonstrate that it has sent the command Start Record Slaver to the Slaver. As to whether the command is successfully processed in the Slaver, check the value of dwParam in the event E_IPR_START_SLAVER_CB.

Related Information:

Driver version	SynCTI Ver.5.3.2.0
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmIPRCloseRecSlaver

2.27.1.8 SsmIPRCloseRecSlaver

Closes the recording Slaver to exit the working state.

Format:

int SsmIPRCloseRecSlaver(int nBId, int nRecSlaverId)

Parameter Description:

nBId	Board ID
nRecSlaverId	ID of the recording Slaver required to be closed

Return Value:

<0	Call failed
>=0	Call successful

Function Description:

Closes the recording Slaver to exit the working state.

Note:

- This function only supports SynIPRecorder;
- The successful return of the function can only demonstrate that it has sent the command Close Record Slaver to the Slaver. As to whether the command is successfully processed in the Slaver, check the value of dwParam in the event E_IPR_CLOSE_SLAVER_CB.

Related Information:

Driver version	SynCTI Ver.5.3.2.0
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmIPRStartRecSlaver

2.27.1.9 SsmIPRActiveSession

Activates the corresponding ports of a specified recording Slaver to receive RTP data.

Format:

int SsmIPRActiveSession(int nCh, int nRecSlaverId, DWORD dwSessionId,LPSTR szPriAddr, int nPriPort, int*

pnPriRcvPort,int nPriCodec,LPSTR szSecAddr, int nSecPort, int *pnSecRcvPort, int nSecCodec)

Parameter Description:

nChId	Channel ID
nRecSlaverId	Recording Slaver ID
dwSessionId	SesseionID of the session that is bound to nChId
szPriAddr	IP address of RTP in the primary end
nPriPort	RTP port in the primary end
pnPriRcvPort	The port used to receive the RTP data in the primary end
nPriCodec	CODEC in the primary end. The supported payload values are listed below: CODEC: Payload μ-Law: 0 A-Law: 8 G723: 4 G729: 18 GSM: 3 G722: 9
szSecAddr	IP address of RTP in the secondary end
nSecPort	RTP port in the secondary end
pnSecRcvPort	The port used to receive the RTP data in the secondary end
nSecCodec	CODEC in the secondary end. The supported payload values are listed below: CODEC: Payload μ-Law: 0 A-Law: 8 G723: 4 G729: 18 GSM: 3 G722: 9

Return Value:

<0	Call failed
>=0	Call successful

Function Description:

Activates the corresponding ports of a specified recording Slaver to receive RTP data.

Note:

- This function only supports SynIPRecorder;
- The successful return of the function can only demonstrate that it has sent the command ActiveSession to the Slaver. As to whether the command is successfully processed in the Slaver, check the value of dwParam in the event E_IPR_ACTIVE_SESSION_CB.
- The parameter dwSessionId, differentiating the recording Sessions of IPRecorder channel, does not need to be the same as the SessionID generated by IPAnalyzer channel.
- The parameters szPriAddr and szSecAddr are not used at present, just set them to “127.0.0.1”.
- The parameters nPriPort and nSecPort are not used at present, you can enter any valid port value (i.e. within the range of 1~65535).
- The parameters nPriCodec and nSecCodec are the payload format supported by IPRecorder. They do not need to be the same as the actual payload format in calls because IPRecorder supports self-adaptive payload format.

Related Information:

Driver version	SynCTI Ver.5.3.2.7
Header	shpa3api.h
Library	shp_a3.lib

DLL	shp_a3.dll

Related Function: SsmIPRDeActiveSession

2.27.1.10 SsmIPRDeActiveSession

Closes the Session receiving port of the corresponding recording Slaver.

Format:

int SsmIPRDeActiveSession(int nCh)

Parameter Description:

nChId	Channel ID

Return Value:

<0	Call failed
>=0	Call successful

Function Description:

Closes the Session receiving port of the corresponding recording Slaver.

Note:

- This function only supports SynIPRecorder;
- The successful return of the function can only demonstrate that it has sent the command DeActiveSession to the Slaver. As to whether the command is successfully processed in the Slaver, check the value of dwParam in the event E_IPR_DEACTIVE_SESSION_CB.

Related Information:

Driver version	SynCTI Ver.5.3.2.0
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmIPRActiveSession

2.27.1.11 SsmIPRActiveAndRecToFile

Activates the corresponding ports of a specified recording Slaver to receive RTP data and starts recording.

Format:

int SsmIPRActiveAndRecToFile(int nCh, int nRecSlaverId, int dwSessionId, int nCodec, int* pnPriRcvPort, int* pnSecRcvPort, LPCSTR pszFileName, int nFormat, DWORD dwStartPos, DWORD dwBytes, DWORD dwTime, int nMask)

Parameter Description:

nChId	Channel ID
nRecSlaverId	Recording Slaver ID
dwSessionId	SesseionID of the session that is bound to nChId
nCodec	CODEC for incoming and outgoing calls in the Session (RTP payload). The supported payload values are listed below: **CODEC** / **Payload** μ-Law / 0 A-Law / 8 G723 / 4 G729 / 18 GSM / 3 G722 / 9
pnPriRcvPort	The port used to receive the RTP data in the primary

pnSecRcvPort	The port used to receive the RTP data in the secondary
pszFileName	See the corresponding parameter in the function SsmRecToFile
nFormat	See the corresponding parameter in the function SsmRecToFile
dwStartPos	See the corresponding parameter in the function SsmRecToFile
dwBytes	See the corresponding parameter in the function SsmRecToFile
dwTime	See the corresponding parameter in the function SsmRecToFile
nMask	See the corresponding parameter in the function SsmRecToFile

Return Value:

<0	Call failed
>=0	Call successful

Function Description:

Activates the corresponding ports of a specified recording Slaver to receive RTP data and starts recording.

Note:

- This function only supports SynIPRecorder;
- The successful return of the function can only demonstrate that it has sent the command ActiveSessionAndRecToFile to the Slaver. As to whether the command is successfully processed in the Slaver, check the value of dwParam in the event E_IPR_ACTIVE_AND_REC_CB.

Related Information:

Driver version	SynCTI Ver.5.3.2.7
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmIPRDeActiveAndStopRecToFile

2.27.1.12 SsmIPRDeActiveAndStopRecToFile

Closes the Session receiving port corresponding to the recording Slaver and stops recording.

Format:

int SsmIPRDeActiveAndStopRecToFile (int nCh)

Parameter Description:

nChId	Channel ID

Return Value:

<0	Call failed
>=0	Call successful

Function Description:

Closes the Session receiving port corresponding to the recording Slaver and stops recording.

Note:

- This function only supports SynIPRecorder;
- The successful return of the function can only demonstrate that it has sent the command DeActiveSessionAndStopRecToFile to the Slaver. As to whether the command is successfully processed in the Slaver, check the value of dwParam in the event E_IPR_DEACTIVE_AND_STOPREC_CB.

Related Information:

Driver version	SynCTI Ver.5.3.2.0
Header	shpa3api.h

Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmIPRActiveAndRecToFile

2.27.1.13 SsmGetUSBKeySerial

Obtains the serial number of a SynIPRecorder or SynIPAnalyzer board.

Format:

DWORD WINAPI SsmGetUSBKeySerial(int nBId)

Parameter Description:

nBId	Board Id

Return Value:

0xffffffff	The Board Id does not correspond to a SynIPRecorder or SynIPAnalyzer board
>0	The serial number of the board

Function Description:

Obtains the serial number of a SynIPRecorder or SynIPAnalyzer board.

Note:

- This function supports SynIPRecorder and SynIPAnalyzer.

Related Information:

Driver version	SynCTI Ver5.3.2.0
Header	Shpa3api.h
Library	Shp_a3.lib
DLL	Shp_a3.dll

2.27.1.14 SsmIsBoardIPR

Judges whether the designated board is SynIPRecorder or SynIPAnalyzer.

Format:

BOOL WINAPI SsmIsBoardIPR(int nBId)

Parameter Description:

nBId	Board Id

Return Value:

FALSE	The board designated by nBId is neither SynIPRecorder nor SynIPAnalyzer
TRUE	The board designated by nBId is either SynIPRecorder or SynIPAnalyzer

Function Description:

Judges whether the designated board is SynIPRecorder or SynIPAnalyzer.

Note:

- This function supports SynIPRecorder and SynIPAnalyzer.

Related Information:

Driver version	SynCTI Ver5.3.2.0
Header	Shpa3api.h
Library	Shp_a3.lib
DLL	Shp_a3.dll

2.27.1.15 SsmIPRConnectToSlaver

Connects the Master to the designated Slaver in passive connection mode.

Format:

int WINAPI SsmIPRConnectToSlaver(char* szSvrAddr, int nSvrPort)

Parameter Description:

szSvrAddr	IP address of the Slaver to be connected with
nSvrPort	Port of the Slaver to be connected with

Return Value:

-1	Connection failed or call failed, the failure reason can be obtained by the function SsmGetLastErrMsg
0	Call successful

Function Description:

Connects the Master to the designated Slaver in passive connection mode.

Note:

- This function only supports SynIPRecorder.
- For the information about setting passive connection mode, refer to the configuration item ActiveConnect.

Related Information:

Driver version	SynCTI Ver5.3.2.0
Header	Shpa3api.h
Library	Shp_a3.lib
DLL	Shp_a3.dll

2.27.2 Functions for SynIPAnalyzer

2.27.2.1 SsmIPRAddProtocol

Newly supports a monitoring protocol.

Format:

int SsmIPRAddProtocol(int nBId, int nPtlId, PIPR_MONITOR_CFGS pParams, DWORD dwLen)

Parameter Description:

nBId	Board ID
nPtlId	Protocol type ID
pParams	Structure parameter of the monitoring protocol PIPR_MONITOR_CFGS
dwLen	Data length of the parameter pParam

Return Value:

<0	Call failed
>=0	Call successful

Function Description:

Newly supports a monitoring protocol.

IPR_MONITOR_CFGS Structure

```
typedef struct
{
union
{
```

```
    IPR_SCCP_CFGS CISCO; // CISCO SCCP protocol
    IPR_SIP_CFGS SIP;       // SIP protocol
    IPR_H323_CFGS Avaya_H323; // Avaya H323 protocol
    IPR_SHORTEL_MGCP_CFGS Shortel_MGCP; // Shortel MGCP protocol
    IPR_H323_CFGS H323;// H323 protocol
    IPR_PANASONIC_MGCP_CFGS Panasonic_MGCP;// Panasonic MGCP protocol
    IPR_TOSHIBA_MEGACO_CFGS Toshiba_MEGACO; // Toshiba MEGACO protocol
    IPR_SIEMENS_H323_CFGS Siemens_H323;        // Siemens H323 parameters
    IPR_ALCATEL_CFGS ALCATEL;                  // Alcatel parameters
    IPR_MITEL_CFGS MITEL;                      // Mitel parameter
    IPR_LG_NORTEL_CFGS LG_Nortel;              // LG Nortel parameters
    IPR_MITEL_CFGS SAMSUNG;                    // Samsung parameters
    IPR_PANASONIC_MGCP_CFGS Tadicom_MGCP; // Tadicom MGCP protocol
    IPR_TADICOM_MGCP_CFGS Tadicom_MGCP;        // Tadicom MGCP parameters
    IPR_ZENITEL_CFGS ZENITEL;                  // Zenitel parameters
    IPR_NORTEL_UNISTIM_CFGS Nortel_UNIStim;    // Nortel UNIStim parameters

};
}IPR_MONITOR_CFGS, *PIPR_MONITOR_CFGS;
```

For detailed information about the above parameters, refer to shpa3api.h.

Note:

- This function only supports SynIPAnalyzer.

Related Information:

Driver version	SynCTI Ver.5.3.3.0
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmIPRRmvProtocol, SsmIPRGetProtocol

2.27.2.2 SsmIPRRmvProtocol

Removes the support of a monitoring protocol.

Format:

int SsmIPRRmvProtocol(int nBId, int nPtlId)

Parameter Description:

nBId	Board ID
nPtlId	Protocol type ID

Return Value:

<0	Call failed
>=0	Call successful

Function Description:

Removes the support of a monitoring protocol.

Note:

- This function only supports SynIPAnalyzer.

Related Information:

Driver version	SynCTI Ver.5.3.2.0
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmIPRAddProtocol, SsmIPRGetProtocol

2.27.2.3 SsmIPRGetProtocol

Obtains the parameters of a monitoring protocol.

Format:

int SsmIPRGetProtocol(int nPtlId, PIPR_MONITOR_CFGS pParams, DWORD* dwLen)

Parameter Description:

nPtlId	Protocol type ID
pParams	Structure parameter PIPR_MONITOR_CFGS of the monitoring protocol
dwLen	Data length of the parameter pParam

Return Value:

<0	Call failed
>=0	Call successful

Function Description:

Obtains the parameters of a monitoring protocol.

Note:

- This function only supports SynIPAnalyzer.

Related Information:

Driver version	SynCTI Ver.5.3.2.0
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmIPRAddProtocol, SsmIPRRmvProtocol

2.27.2.4 SsmIPRGetStationCount

Obtains the count of the current active Stations.

Format:

int SsmIPRGetStationCount(int nBId)

Parameter Description:

nBId	Board ID

Return Value:

<0	Call failed
>=0	Call successful, the return value is the count of the current active Stations

Function Description:

Obtains the count of the current active Stations.

Note:

- This function only supports SynIPAnalyzer.

Related Information:

Driver version	SynCTI Ver.5.3.2.0
Header	shpa3api.h

Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmIPRGetStationList, SsmIPRGetStationInfo

2.27.2.5 SsmIPRGetStationList

Obtains the information about a specified number of Stations.

Format:

int SsmIPRGetStationList(int nBId, int nStationNum, PSTATION_LIST pStationList)

Parameter Description:

nBId	Board ID
nStationNum	The number of Stations required to get information
pStationList	Pointer pointing to PSTATION_LIST which stores the Station information

Return Value:

<0	Call failed
>=0	Call successful

Function Description:

Obtains the information about a specified number of Stations.

Note:

- This function only supports SynIPAnalyzer.

Related Information:

Driver version	SynCTI Ver.5.3.2.0
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmIPRGetStationCount, SsmIPRGetStationInfo

2.27.2.6 SsmIPRGetStationInfo

Obtains the detailed information of a certain Station.

Format:

int SsmIPRGetStationInfo(int nBId, int nStationId, int* nPtlId, int* nTransType, int* nPort, LPSTR szIP, LPSTR szMAC)

Parameter Description:

nBId	Board ID
nStationId	StationId of the station required to get information
nPtlId	ID of the call control protocol
nTransType	Type of the transfer protocol
nPort	Signaling port
szIP	Signaling IP, set to NULL if not necessary
szMAC	MAC address corresponding to the signaling IP, set to NULL if not necessary

Return Value:

<0	Call failed
>=0	Call successful

Function Description:

Obtains the detailed information of a certain Station.

Note:

- This function only supports SynIPAnalyzer.

Related Information:

Driver version	SynCTI Ver.5.3.2.0
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmIPRGetStationCount, SsmIPRGetStationList, SsmIPRGetStationInfoEx

2.27.2.7 SsmIPRGetStationInfoEx

Obtains all the information of a certain Station.

Format:

int SsmIPRGetStationInfoEx(int nBId, int nStationId, PStationInfoEx pSInfoEx)

Parameter Description:

nBId	Board ID
nStationId	Station Id of the station required to get information
pSInfoEx	Station information structure

Return Value:

<0	Call failed
>=0	Call successful

Function Description:

Obtains all the information of a certain Station.

Note:

- This function only supports SynIPAnalyzer.
- The memory of pSInfoEx is required to be managed manually.

Related Information:

Driver version	SynCTI Ver.5.3.2.0
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmIPRGetStationCount, SsmIPRGetStationList, SsmIPRGetStationInfo

2.27.2.8 SsmIPRSendSession

Sends a specified Session to a specified recording server.

Format:

int SsmIPRSendSession(int nChId, LPSTR szPriSlaverAddr, int nPriSlaverPort, LPSTR szSecSlaverAddr, int nSecSlaverPort)

Parameter Description:

nChId	Channel ID
szPriSlaverAddr	IP address of the recording Slaver for receiving RTP data in the primary
nPriSlaverPort	Port address of the recording Slaver for receiving RTP data in the primary
szSecSlaverAddr	IP address of the recording Slaver for receiving RTP data in the secondary
nSecSlaverPort	Port address of the recording Slaver for receiving RTP data in the secondary

Return Value:

<0	Call failed
>=0	Call successful

Function Description:

Sends a specified Session to a specified recording server.

Note:

- This function only supports SynIPAnalyzer.

Related Information:

Driver version	SynCTI Ver.5.3.2.0
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmIPRStopSendSession, SsmIPRGetSessionInfo

2.27.2.9 SsmIPRStopSendSession

Stops sending the Session.

Format:

int SsmIPRStopSendSession(int nChId)

Parameter Description:

nChId	Channel ID

Return Value:

<0	Call failed
>=0	Call successful

Function Description:

Stops sending the Session.

Note:

- This function only supports SynIPAnalyzer.
- After the driver throws out the event E_RCV_IPR_MEDIA_SESSION_STOPED, it will automatically stop transmitting RTP. Invoking this function at that time will fail therefore.

Related Information:

Driver version	SynCTI Ver.5.3.2.0
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmIPRSendSession, SsmIPRGetSessionInfo

2.27.2.10 SsmIPRGetSessionInfo

Obtains the related information of a specified Session.

Format:

int SsmIPRGetSessionInfo(int nChId, pIPR_SessionInfo pSInfo)

Parameter Description:

nChId	Channel ID
pSInfo	The obtained Session information, stored in pIPR_SessionInfo

Return Value:

<0	Call failed
>=0	Call successful

Function Description:

Obtains the related information of a specified Session.

Note:

- This function only supports SynIPAnalyzer.

Related Information:

Driver version	SynCTI Ver.5.3.2.0
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmIPRSendSession, SsmIPRStopSendSession

2.27.2.11 SsmIPRGetMonitorType

Obtains the current monitoring type (complete monitoring or monitoring based on Station).

Format:

int WINAPI SsmIPRGetMonitorType(int nBId)

Parameter Description:

nBId	Board ID

Return Value:

-1	Call failed
0	Complete monitoring
1	Monitoring based on Station

Function Description:

Obtains the current monitoring type (complete monitoring or monitoring based on Station).

Note:

- This function only supports SynIPAnalyzer.

Related Information:

Driver version	SynCTI Ver.5.3.2.0
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmIPRSetMonitorType

2.27.2.12 SsmIPRSetMonitorType

Sets the monitoring type (Monitor All or Monitor By Station).

Format:

int SsmIPRSetMonitorType(int nBId, nMType)

Parameter Description:

nBId	Board ID
nMType	Monitoring type, 0: Monitor All; 1: Monitor By Station

Return Value:

<0	Call failed
>=0	Call successful

Function Description:

Sets the monitoring type (Monitor All or Monitor By Station).

Note:

- This function only supports SynIPAnalyzer.
- When this function is invoked to modify the monitoring type, the driver will delete all the session and Station information and stop the corresponding transmission. If the original monitoring type is 'Monitor By Station', the driver will clear the monitoring list too.
- To set 'Monitor By Station' as the original monitoring type, we suggest you do it via the configuration item MonitorType instead of this function; or the driver may detect the session and Station before you invoke this function and may probably waste extra time and resources to deal with the monitored session.

Related Information:

Driver version	SynCTI Ver.5.3.2.0
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmIPRGetMonitorType

2.27.2.13 SsmIPRAddStationToMap

Adds the Station to be monitored to the map list.

Format:

int SsmIPRAddStationToMap(int nBId, int nStationId, LPSTR szAddr, int nPort)

Parameter Description:

nBId	Board ID
nStationId	User defined Station Id
szAddr	IP address or MAC address of the Station to be monitored
nPort	Port of the Station required to be monitored

Return Value:

-1	Call failed
=0	Call successful

Function Description:

Adds the Station to be monitored to the map list.

Note:

- This function only supports SynIPAnalyzer. If the parameter nPort is less than 0, szAddr indicates the MAC address of the Station to be monitored; otherwise szAddr indicates the IP address of the Station to be monitored. If nPort is set to 0, it indicates that any port on the IP address designated by szAddr can be monitored; if nPort is larger than 0, it indicates the port to be monitored. The format of the MAC address is "aa:bb:cc:dd:ee:ff".

Related Information:

Driver version	SynCTI Ver.5.3.2.0
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmIPRRmvStationFromMap, SsmIPRRmvStationFromMapEx, SsmIPRAddStationToMapEx

2.27.2.14 SsmIPRAddStationToMapEx

Adds a Station to the monitoring list.

Format:

int SsmIPRAddStationToMapEx(int nBId, int nStationId, LPSTR szAddr, int nPort, LPSTR szName, LPVOID lpReserve)

Parameter Description:

nBId	Board ID
nStationId	Station Id of the Station to be added to the monitoring list
szAddr	IP or MAC address of the Station to be monitored
nPort	Port of the Station to be monitored
szName	Username to be monitored
lpReserve	Reserved

Return Value:

-1	Call failed
=0	Call successful

Function Description:

Adds a Station to the monitoring list.

Note:

- The parameters szAddr and szName cannot be set to null at the same time. When szAddr is not null, szName will be ignored and this function is completely the same as SsmIPRAddStationToMap. When szName is not null, it indicates the username to be monitored.

Related Information:

Driver version	SynCTI Ver.5.3.2.0
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmIPRRmvStationFromMap, SsmIPRRmvStationFromMapEx, SsmIPRAddStationToMap

2.27.2.15 SsmIPRRmvStationFromMap

Removes a Station from the monitoring list.

Format:

int SsmIPRRmvStationFromMap(int nBId, LPSTR szAddr, int nPort)

Parameter Description:

nBId	Board ID
szAddr	IP address or MAC address of the Station to be removed from the monitoring list
nPort	Port of the Station to be removed from the monitoring list

Return Value:

-1	Call failed
=0	Call successful

Function Description:

Removes a Station from the monitoring list.

Note:

- This function only supports SynIPAnalyzer. If the parameter nPort is less than 0, szAddr indicates the MAC address of the Station to be removed form the monitoring list, otherwise szAddr means the IP

address of the Station to be removed form the monitoring list. In case of removing the MAC address of the Station from the monitoring list, the parameter nPort in this function needn't be the same as that in the function SsmIPRRmvStationFromMap. The format of the MAC address is "aa:bb:cc:dd:ee:ff".

- This function only supports the 'delayed' delete mode, i.e. if the Station to be removed from the monitoring list is still in the conversation, this function will fail. Once the conversation ends, the designated Station will be removed automatically. If the Station is required to be removed at once, refer to the function SsmIPRRmvStationFromMapEx.

Related Information:

Driver version	SynCTI Ver.5.3.2.0
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmIPRAddStationToMap, SsmIPRRmvStationFromMapEx

2.27.2.16 SsmIPRRmvStationFromMapEx

Removes a designated Station from the monitoring list.

Format:

int SsmIPRRmvStationFromMapEx(int nBId, int nStationId,bool bDelAtOnce)

Parameter Description:

nBId	Board ID
nStationId	The ID of the Station to be removed from the monitoring list
bDelAtOnce	Whether to remove the Station at once

Return Value:

<0	Call failed
=0	Call successful in the 'delayed' delete mode. The designated Station will be removed after the conversation ends.
=1	Call successful

Function Description:

Removes a designated Station from the monitoring list.

Note:

- This function only supports SynIPAnalyzer.
- This function supports two delete modes: the 'delayed' delete mode and the 'instant' delete mode. Provided the Station to be removed from the monitoring list is still in the conversation, in the 'delayed' delete mode, it will be removed after the conversation ends; in the 'instant' delete mode, the monitoring of the conversation and the transfer of RTP will be stopped immediately, and the designated Station will be removed at once.
- The return value of 0 only appears in the 'delayed' delete mode.

Related Information:

Driver version	SynCTI Ver.5.3.2.0
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmIPRAddStationToMap, SsmIPRRmvStationFromMap

2.27.2.17 SsmIPRChkForward

Checks whether the corresponding channel is forwarding the RTP data.

Format:

int SsmIPRChkFoward(int nChId)

Parameter Description:

nChId	Channel number

Return Value:

0	The channel is not forwarding the RTP data
1	The channel is forwarding the RTP data
<0	Call failed

Function Description:

Checks whether the corresponding channel is forwarding the RTP data.

Note:

- This function only supports SynIPAnalyzer.

Related Information:

Driver version	SynCTI Ver.5.3.2.0
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function:

2.27.2.18 SsmIPRGetCallInfo

Obtains the information of a designated call.

Format:

int WINAPI SsmIPRGetCallInfo(int nBId,int nCallRef,pIPR_SIP_CALL_INFOEX pCallInfo)

Parameter Description:

nBId	Board ID
nCallRef	Index of the call required to get information
pCallInfo	User provided buffer area which is used for storing the obtained call information

Return Value:

-1	Call failed
=0	Call successful

Function Description:

Obtains the information of a designated call.

Note:

- This function only supports SynIPAnalyzer.

Related Information:

Driver version	SynCTI Ver.5.3.2.1
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function:

2.27.2.19 SsmIPRGetMessageField

Obtains the message of a field in the designated call.

Format:

int WINAPI SsmIPRGetMessageField(int nBId, int nCallRef, LPSTR pStrFieldName, LPSTR pStrContentBuf, int* pSize)

Parameter Description:

nBId	Board ID
nCallRef	Index of the call required to get information
pStrFieldName	Name of the field required to get message, such as CALL-ID
pStrContentBuf	Message reception buffer
pSize	As an input parameter, pSize indicates the length of pStrContentBuf; as an output parameter, pSize indicates the actual length of the field message

Return Value:

-1	Call failed
=0	Call successful

Function Description:

Obtains the message of a field in the designated call. This function has the capability to obtain the whole original SIP message and will output it if pStrFieldName="SIP RAW MSG".

Note:

- This function only supports SynIPAnalyzer.

Related Information:

Driver version	SynCTI Ver.5.3.2.7
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function:

2.27.3 Structures for IPR Series

2.27.3.1 IPR_Addr

The structure storing the IP address and port

```
typedef union tagIPR_Addr
{
    __int64    nLLaddr;   // Combination of address and port

    struct
    {
        union
        {
            struct { unsigned char s_b1,s_b2,s_b3,s_b4; } S_un_b;
            unsigned long   S_addr;
        };// IP address
        unsigned long   usPort;     // Port
    };
}IPR_Addr,*pIPR_Addr;
```

2.27.3.2 IPR_SLAVERADDR

The structure storing the address information of the Slaver

```
typedef struct tagIPR_SLAVERADDR
{
    int nRecSlaverID;    // SlaverID
    IPR_Addr ipAddr;     // IP address and port in the Slaver
}IPR_SLAVERADDR, *PIPR_SLAVERADDR;
```

2.27.3.3 IPR_SessionInfo

For Session information, refer to IPR_SessionInfo in Chapter 1.

2.27.3.4 COMPUTER_INFO

It stores the computer information.

```
#define MAX_NET_ADAPTERS      6     // Storage limit of the network adapter
typedef struct tagCOMPUTER_INFO
{
    char szOSVersion[128];          // Operating system
    int nCPUNO;                     // number of CPUs
    char szCPUInfo[256];            // CPU information
    ULONG ulPhysicalMemory;         // Physical memory in MB (1K=1024)
    ULONG ulHardDisk;               // HD capacity in GB (1K=1000)
    int nNetAdapterNO;              // Number of network adapters
    char szMAC[MAX_NET_ADAPTERS][30];// MAC address
    char szIPAddr[MAX_NET_ADAPTERS][30];// IP address
    char szMask[MAX_NET_ADAPTERS][30];      // Subnet mask
    char szGateway[MAX_NET_ADAPTERS][30];   // Gateway
}COMPUTER_INFO, *PCOMPUTER_INFO;
```

2.27.3.5 IPR_MONITOR_CFGS

This structure is used by SsmIPRAddProtocol in SynIPAnalyzer to configure the monitoring protocol.

```
{
    union
    {
     IPR_SCCP_CFGS CISCO;                  // CISCO SCCP protocol parameter
     IPR_SIP_CFGS SIP;                     // SIP protocol parameter
    };
}IPR_MONITOR_CFGS, *PIPR_MONITOR_CFGS;
```

2.27.3.6 STATION_LIST

A list storing the terminal information

```
typedef struct tagStationList
{
    int nStationId;             //Station Id
```

```
    UCHAR ucCallCtrlId;     // Signaling protocol ID
}STATION_LIST,*PSTATION_LIST;
```

2.27.3.7 IPR_CALL_INFO

Call information output with call state events

```
#define IPR_MAX_CALL_ID_SIZE 120    // Maximum length (bytes) of CallID
typedef struct tagIPR_CALL_INFO
{
    ULONG               CallRef ;               // Call index, increasing
    ULONG               CallSource ;            // Call direction
    ULONG               Cause ;                 // Cause
    char szCallerId[IPR_MAX_CALL_ID_SIZE];      // CallerID
    char szCalledId[IPR_MAX_CALL_ID_SIZE];      // CalleeID
    char szReferredBy[IPR_MAX_CALL_ID_SIZE];    // Where the call is transferred from
    char szReferTo[IPR_MAX_CALL_ID_SIZE];       // Where the call is transferred to
}IPR_CALL_INFO, *PIPR_CALL_INFO;
```

2.27.3.8 IPR_CALL_INFOEX

Detailed call information.

```
typedef struct tagIPR_CALL_INFOEX
{
    int nCallRef;           // Call index
    int nStationId;         // Primary StationId
    int nStationId2;        // Secondary StationId (i.e SIP p2p)

    unsigned int ulPrimaryIP;   // Primary IP
    unsigned int ulSecondaryIP;// Secondary IP

    unsigned short usPrimaryPort; // Primary Port, little endian
    unsigned short usSecondaryPort;// Secondary Port, little endian
    ULONG               CallSource;         // Call direction
    ULONG               Cause;              // Reason
    char szCallerId[IPR_MAX_CALL_ID_SIZE];  // CallerID
    char szCalledId[IPR_MAX_CALL_ID_SIZE];  // CalledID
    char szReferredBy[IPR_MAX_CALL_ID_SIZE];// Where the call is transferred from
    char szReferTo[IPR_MAX_CALL_ID_SIZE];   // Where the call is transferred to

    char szSrcMAC[6];               // Primary Mac address
    char szDstMAC[6];               // Secondary Mac address
}IPR_SIP_CALL_INFOEX, *pIPR_SIP_CALL_INFOEX;
```

2.27.3.9 StationInfoEx

The structure used by SsmIPRGetStationInfoEx in SynIPAnalyzer.

```
typedef struct tagStationInfoEx
```

```
{
    int nStationId;    //Station Id
    char szCallName[IPR_MAX_CALL_ID_SIZE]; //Extension number or username
    LPVOID lpReserve;          //Reserved
    IPR_Addr CallCtrlAddr;     //Call control address + port
    USHORT usSecPort;          //Another control port, especially for H323
    UCHAR ucCallCtrlPtl;       //Call control protocol
    UCHAR ucTransPtl;          //Transfer protocol
    UCHAR ucMacAddr[6];        //MAC address
}StationInfoEx,*PStationInfoEx;
```

2.28 CODEC Transcoding Functions

2.28.1 fPcm_Mp3ConvertALaw

Refer to fPcm_Mp3ConvertULaw

2.28.2 fPcm_Mp3ConvertULaw

fPcm_Mp3ConvertALaw converts IMP3 formatted files recorded by the Synway board to be A-law formatted files, fPcm_Mp3ConvertULaw converts MP3 formatted files to be μ-law formatted files.

Format:

int fPcm_Mp3ConvertALaw (char*szSourecFile, char*szTargetFile)

int fPcm_Mp3ConvertULaw (char*szSourecFile, char*szTargetFile)

Parameter Description:

szSoureFile	The name of the MP3-formatted source file recorded by Synway boards, it may include the full path of the file. The file format could be either the non-header file (i.e. without file header) or the standard wav file. It's through analyzing the source file of szSourceFile to determine whether it has the wav file header instead of checking the file extension name that the driver decides if the source file is standard or not. If it's not a standard wav file, it will be regarded as a non-header file
SzTargetFile	The target file name, it may include the full path of the file. If the target file doesn't exist, the driver will automatically create it; If the target file exists already, it will be overlaid. If the extension name of the target file is wav, the converted file format is the standard wav; otherwise, the format is plain

Return Value:

1	Successful
-1	Call failed. The failure reason can be obtained by the function fPcm_GetLastErrMsg

Function Description:

fPcm_Mp3ConvertALaw converts the MP3 formatted file to be A-law formatted file, fPcm_Mp3ConvertULaw converts the MP3 formatted file to be μ-law formatted file. The source file must be MP3-formatted non-header files or standard wav files recorded by Synway boards.

Note:

- Because this function is provided by ShPcmHandle.dll, it can be invoked before SsmStartCti is called successfully;

- When this function is called, the ACM decoding engine of MP3 will be used.

Related Information:

Driver version	SynCTI Ver. 4.0 or above
Header	ShPcmApi.h
Library	ShPcmHandle.lib
DLL	ShPcmHandle.dll

Related Function:

2.28.3 fPcm_AdpcmToAlaw

Refer to Pcm_AdpcmToMp3

2.28.4 fPcm_AdpcmToGsm

Refer to fPcm_AdpcmToMp3

2.28.5 fPcm_AdpcmToMp3

Converts the IMA ADPCM formatted file recorded by Synway boards to be A-law formatted file (by the function of fPcm_AdpcmToAlaw), GSM 6.10 formatted file (by the function of fPcm_AdpcmToGsm), or MP3 formatted file (fPcm_AdpcmToMp3).

Format:

int fPcm_AdpcmToAlaw(char*szSoureFile, char* szTargetFile)

int fPcm_AdpcmToGsm(char*szSoureFile,char*szTargetFile)

int fPcm_AdpcmToMp3(char*szSoureFile, char* szTargetFile)

Parameter Description:

szSoureFile	The name of the IMA ADPCM formatted source file recorded by Synway boards, it may include the full path of the file. The file format could be either the non-header file (i.e. without file header) or the standard wav file. It's through analyzing the source file of szSourceFile to determine whether it has the wav file header instead of checking the file extension name that the driver decides if the source file is standard or not. If it's not a standard wav file, it will be regarded as a non-header file
szTargetFile	The name of A-law formatted target file, it may include the full path of the file. If the target file doesn't exist, the driver will automatically create it; If the target file exists already, it will be overlaid. If the extension name of the target file is wav, the converted file format is the standard wav; otherwise, the format is plain

Return Value:

1	Successful
-1	Call failed. The failure reason can be obtained by the function fPcm_GetLastErrMsg.

Function Description:

The function of fPcm_AdpcmToAlaw converts the IMA ADPCM formatted file recorded by Synway boards to be A-law formatted file, the function of fPcm_AdpcmToGsm converts the IMA ADPCM formatted file to be GSM 6.10 formatted file, the function of fPcm_AdpcmToMp3 converts the IMA ADPCM formatted file to be Mp3 formatted file.

Note:

- Because this function is provided by ShPcmHandle.dll, it can be invoked before SsmStartCti is called successfully;
- To invoke the above functions, the file macmcvt.dll must be used.
- To invoke the function fPcm_AdpcmToGsm, the ACM encoding engine of GSM must be used; to invoke the function fPcm_AdpcmToMp3, the ACM encoding engine of MP3 must be used.

Related Information:

Driver version	fPcm_AdpcmToAlaw requires SynCTI Ver. 4.0 or above; fPcm_AdpcmToGsm requires SynCTI Ver. 4.7.2.0 or above; fPcm_AdpcmToMp3 requires SynCTI Ver. 4.7.3.1 or above
Header	ShPcmApi.h
Library	ShPcmHandle.lib
DLL	ShPcmHandle.dll

Related Function: fPcm_MemAdpcmToALAW, fPcm_MemAdpcmToULAW

2.28.6 fPcm_AlawToUlaw

Refer to fPcm_UlawToAlaw

2.28.7 fPcm_UlawToAlaw

Converts A-Law formatted files recorded by the Synway board to be μ-Law formatted files (fPcm_ AlawToUlaw), or converts μ-Law formatted files to be A-Law formatted files (fPcm_ UlawToAlaw).

Format:

int WINAPI fPcm_AlawToUlaw(char*szSourceFile, char*szTargetFile)

int WINAPI fPcm_UlawToAlaw(char*szSourceFile, char*szTargetFile)

Parameter Description:

szSoureFile	The name of the A-Law or μ-Law formatted source file recorded by Synway boards, it may include the full path of the file. The file format could be either the non-header file (i.e. without file header) or the standard wav file. It's through analyzing the source file of szSourceFile to determine whether it has the wav file header instead of checking the file extension name that the driver decides if the source file is standard or not. If it's not a standard wav file, it will be regarded as a non-header file.
szTargetFile	The target file name, it may include the full path of the file. If the target file doesn't exist, the driver will automatically create it; If the target file exists already, it will be overlaid. If the extension name of the target file is wav, the converted file format is the standard wav; otherwise, the format is plain.

Return Value:

1	Call successful
-1	Call failed. The failure reason may be obtained by the function fPcm_GetLastErrMsg

Function Description:

fPcm_ AlawToUlaw converts A-Law formatted files recorded by the Synway board to be μ-Law formatted files; fPcm_ UlawToAlaw converts μ-Law formatted files to be A-Law formatted files.

Note:

- Because this function is provided by ShPcmHandle.dll, it can be invoked before the call of SsmStartCti.

Related Information:

Driver version	SynCTI Ver. 5.0.0.0 or above
Header	ShPcmApi.h
Library	ShPcmHandle.lib
DLL	ShPcmHandle.dll

Related Function:

2.28.8 fPcm_MemAdpcmToALAW

Refer to fPcm_MemAdpcmToULAW

2.28.9 fPcm_MemAdpcmToULAW

Converts the IMA ADPCM formatted data which is stored in the buffer area and recorded by Synway boards to be A-law or μ-law formatted data. The function of fPcm_MemAdpcmToALAW converts IMA ADPCM to be A-law, The function of fPcm_MemAdpcmToULAW converts IMA ADPCM to be μ-law.

Format:

DWORD fPcm_MemAdpcmToALAW(char * pSource, DWORD dwSourceSize, char * pTarget, DWORD dwTargetSize)

DWORD fPcm_MemAdpcmToULAW(char * pSource, DWORD dwSourceSize, char * pTarget, DWORD dwTargetSize)

Parameter Description:

pSource	The pointer pointing to the first address of the input buffer area
dwSourceSize	The size (bytes) of the input buffer. Because the length of the IMA ADPCM formatted frame is 256 bytes, this parameter must be integral times of 256 bytes
pTarget	The pointer pointing to the first address of the output buffer area storing the converted data; the storage space is allocated by the application
dwTargetSize	The size of the output buffer area. Because 505 bytes data is generated after each IMA ADPCM formatted frame is converted to be A-law or μ-law formatted data, the size of the output buffer area should be no less than dwSourceSize / 256 * 505 + 1

Return Value:

0	Conversion failed, the failure reason can be obtained via the function call of fPcm_GetLastErrMsg
>0	The byte number of the voice data after conversion

Function Description:

Converts the IMA ADPCM formatted data which is stored in the buffer area and recorded by Synway boards to be A-law or μ-law formatted data.

Note:

- Because this function is provided by ShPcmHandle.dll, it can be invoked before SsmStartCti is called successfully.

Related Information:

Driver version	SynCTI Ver. 4.0 or above
Header	ShPcmApi.h
Library	ShPcmHandle.lib
DLL	ShPcmHandle.dll

Related Function: fPcm_AdpcmToAlaw

2.28.10 fPcm_ALawConvertPcm16

Refer to fPcm_ALawConvertMp3

2.28.11 fPcm_ALawConvertPcm16_16K

Refer to fPcm_ALawConvertMp3

2.28.12 fPcm_ALawConvertPcm8

Refer to fPcm_ALawConvertMp3

2.28.13 fPCM_AlawConvertGC8

Refer to fPcm_ALawConvertMp3

2.28.14 fPcm_ALawConvertMp3

Converts the A-law formatted file to the 16-bit PCM formatted file (by fPcm_ALawConvertPcm16), the 16-bit 16KHz PCM formatted file (by fPcm_ALawConvertPcm16_16K), the 8-bit unsigned PCM formatted file (by fPcm_ALawConvertPcm8), the G.729A formatted file (by fPCM_AlawConvertGC8), or the MP3 formatted file (by fPcm_ALawConvertMp3).

Format:

int fPcm_ALawConvertPcm16(char*fnALaw, char* szTargetFile)

int WINAPI fPcm_ALawConvertPcm16_16K(char* fnALaw, char*szTargetFile)

int fPcm_AlawConvertPcm8(char* fnALaw, char* szTargetFile)

int fPCM_AlawConvertGC8(char * fnALaw,char*szTargetFile)

int fPcm_ALawConvertMp3(char*szSourceFile, char*szTargetFile)

Parameter Description:

fnALaw	The name of the A-law formatted source file, it may include the full path of the file. The file format could be either the non-header file (i.e. without file header) or the standard wav file. It's through analyzing the source file of szSourceFile to determine whether it has the wav file header instead of checking the file extension name that the driver decides if the source file is standard or not. If it's not a standard wav file, it will be regarded as a non-header file
szTargetFile	The target file name, it may include the full path of the file. If the target file doesn't exist, the driver will automatically create it; If the target file exists already, it will be overlaid. If the extension name of the target file is wav, the converted file format is the standard wav; otherwise, the format is plain

Return Value:

1	Call of fPcm_ALawConvertPcm16, fPcm_ALawConvertPcm16_16K, fPcm_AlawConvertPcm8 or fPcm_ALawConvertMp3 successful
0	Call of fPCM_AlawConvertGC8 successful
-1	Call of fPcm_ALawConvertPcm16, fPcm_ALawConvertPcm16_16K, fPcm_AlawConvertPcm8, fPCM_AlawConvertGC8 or fPcm_ALawConvertMp3 failed, the failure reason can be obtained via the function call of fPcm_GetLastErrMsg

Function Description:

fPcm_ALawConvertPcm16 converts the A-law formatted file to the 16-bit PCM formatted file; fPcm_ALawConvertPcm16_16K converts the A-law formatted file to the 16-bit 16KHz PCM formatted file; fPcm_ALawConvertPcm8 converts the A-law formatted file to the unsigned 8-bit PCM formatted file; fPCM_AlawConvertGC8 converts the A-law formatted file to the G.729A formatted file; fPcm_ALawConvertMp3 converts the A-law formatted file to the MP3 formatted file provided the MP3 recording engine has been installed.

Note:

- Because this function is provided by ShPcmHandle.dll, it can be invoked before SsmStartCti is called successfully.
- fPcm_ALawConvertPcm16, fPcm_ALawConvertPcm16_16K, fPcm_AlawConvertPcm8 or fPcm_ALawConvertMp3 returns only 1 or -1.
- fPCM_AlawConvertGC8 returns only 0 or -1.
- To invoke this function, the file macmcvt.dll must be used.
- To invoke the function fPCM_AlawConvertGC8, the ACM encoding engine of G729A must be used; to

invoke the function fPcm_ALawConvertMp3, the ACM encoding engine of MP3 must be used.

Related Information:

Driver version	fPcm_ALawConvertPcm16, fPcm_ALawConvertPcm8 require SynCTI Ver. 4.0 or above; fPCM_AlawConvertGC8 requires SynCTI Ver. 4.7.1.8 or above; fPcm_ALawConvertMp3 requires SynCTI Ver. 4.8.0.0 or above; fPcm_ALawConvertPcm16_16K requires SynCTI Ver. 5.2.0.1 or above.
Header	ShPcmApi.h
Library	ShPcmHandle.lib
DLL	ShPcmHandle.dll

Related Function: fPcm_Pcm16ConvertALaw, fPcm_GC8Convert, fPcm_MemAlawToPcm8, fPcm_MemAlawToPcm16

2.28.15 fPcm_ULawConvertMp3

Converts the μ-Law formatted file to the MP3 formatted file.

Format:

Int WINAPI fPcm_ULawConvertMp3(char*szSourceFile, char*szTargetFile)

Parameter Description:

szSoureFile	The name of the μ-Law formatted source file which is recorded by Synway boards. It may include the full path of the file. The file could be either a non-header file (i.e. without file header) or a standard wav file. Instead of checking the file extension name, the driver judges if the source file is standard or not by analyzing szSourceFile to find whether it has the wav file header. If it's not a standard wav file, it will be regarded as a non-header file.
szTargetFile	The target file name. It may include the full path of the file. If the target file doesn't exist, the driver will automatically create it; if the target file exists already, it will be overlaid. If the extension name of the target file is wav, the converted file is a standard wav file; otherwise, it will be a non-header file.

Return Value:

1	Successful
-1	Call failed. The failure reason can be obtained via the function call of fPcm_GetLastErrMsg

Function Description:

Converts the μ-Law formatted file recorded by Synway boards to the MP3 formatted file.

Note:

- Because this function is provided by ShPcmHandle.dll, it can be invoked before the call of SsmStartCti.
- To invoke this function, the file macmcvt.dll must be used.
- To invoke this function, the ACM encoding engine of MP3 must be used.

Related Information:

Driver version	SynCTI Ver. 5.0.0.0 or above.
Header	ShPcmApi.h
Library	ShPcmHandle.lib
DLL	ShPcmHandle.dll

Related Function: fPcm_ALawConvertMp3

2.28.16 fPcm_GetDtmfAndMskFromMp3

Parses DTMF and MSK structure information from an MP3 formatted file.

Format:

int fPcm_GetDtmfAndMskFromMp3(LPCSTR szSourceFile, char* pszDtmf, int* pcchDtmf, pMSK_INFO pMsk, int*

pcchMsk)

Parameter Description:

szSoureFile	The name of the MP3 (16K) formatted source file which is recorded by Synway boards. It may include the full path of the file. The file could be either a non-header file (i.e. without file header) or a standard wav file. Instead of checking the file extension name, the driver judges if the source file is standard or not by analyzing szSourceFile to find whether it has the wav file header. If it's not a standard wav file, it will be regarded as a non-header file.
pszDtmf	Pointer pointing to the first address of DTMF buffer area storing the DTMF characters parsed from szSoureFile. It can be NULL.
pcchDtmf	Size of DTMF buffer area. If pcchDtmf is NULL, the function will not parse DTMF from szSourceFile file. If neither pcchDtmf nor pszDtmf is NULL, the function will parse all DTMF characters from szSourceFile file. If pcchDtmf is not NULL and pszDtmf is NULL, the function will only get the number of DTMF characters in szSourceFile. *pcchDtmf returns the number of DTMF characters. If *pcchDtmf is smaller than the size of the DTMF buffer area, it indicates the exact number of DTMF characters parsed from szSourceFile. If *pcchDtmf equals to the size of the DTMF buffer area, it may indicate that the size of DTMF buffer area is just or not enough.
pMsk	Pointer pointing to the first address of MSK buffer area storing the MSK structure parsed from szSourceFile. It can be NULL.
pcchMsk	Size of MSK buffer area. If pcchMSK is NULL, the function will not parse the MSK structure from szSourceFile file. If neither pcchMSK nor pMSK is NULL, the function will parse all MSK structures from szSourceFile file. If pcchMSK is not NULL and pMSK is NULL, the function will only get the number of MSK structures in szSourceFile. *pcchMSK returns the number of MSK structures. If *pcchMSK is smaller than the size of the MSK buffer area, it indicates the exact number of MSK structures parsed from szSourceFile. If *pcchDtmf equals to the size of the MSK buffer area, it may indicate that the size of MSK buffer area is just or not enough.

Return Value:

1	Call successful
-1	Call failed, the failure reason can be obtained by the function fPcm_GetLastErrMsg

Function Description:

Parses DTMF and MSK structure from an MP3 formatted file.

The MSK structure is as follows:

```
typedef struct tagMSK_INFO
{
    unsigned char op;
    unsigned char arg;
    unsigned short unitID;
    unsigned char extra0;
    unsigned char extra1;
    unsigned char extra2;
    unsigned char extra3;
    BOOL bIsPacketDouble;
}MSK_INFO, *pMSK_INFO;
```

If bIsPacketDouble=TRUE, it indicates that the former seven variables are all parsed from the structure; if bIsPacketDouble=FALSE, it indicates that only the variables op, arg and unitID are parsed.

Note:

- Because this function is provided by ShPcmHandle.dll, it can be invoked before SsmStartCti.

- To invoke this function, the ACM decoding engine of MP3 must be used.

Related Information:

Driver version	SynCTI Ver. 5.3.2.1 or above;
Header	ShPcmApi.h
Library	ShPcmHandle.lib
DLL	ShPcmHandle.dll

Related Function:

2.28.17 fPcm_ALawToVox

Converts the A-Law formatted file to the vox formatted file.

Format:

int WINAPI fPcm_ALawToVox (char *szSourceFile, char *szTargetFile)

Parameter Description:

szSoureFile	The name of the A-Law formatted source file which is recorded by Synway boards. It may include the full path of the file. The file could be either a non-header file (i.e. without file header) or a standard wav file. Instead of checking the file extension name, the driver judges if the source file is standard or not by analyzing szSourceFile to find whether it has the wav file header. If it's not a standard wav file, it will be regarded as a non-header file.
szTargetFile	The target file name. It may include the full path of the file. If the target file doesn't exist, the driver will automatically create it; if the target file exists already, it will be overlaid.

Return Value:

0	Successful
-1	Call failed. The failure reason can be obtained via the function call of fPcm_GetLastErrMsg

Function Description:

Converts the A-Law formatted file recorded by Synway boards to the vox formatted file.

Note:

- Because this function is provided by ShPcmHandle.dll, it can be invoked before the call of SsmStartCti.

Related Information:

Driver version	SynCTI Ver. 5.1.1.0 or above.
Header	ShPcmApi.h
Library	ShPcmHandle.lib
DLL	ShPcmHandle.dll

Related Function:

2.28.18 fPcm_MemAlawToPcm8

Refer to fPcm_MemAlawToPcm16

2.28.19 fPcm_MemAlawToPcm16

Converts the A-law formatted data stored in the buffer area to be unsigned 8-bit PCM formatted data (by the function fPcm_MemAlawToPcm8) or 16 bit PCM formatted data (by the function fPcm_MemAlawToPcm16).

Format:

DWORD fPcm_MemAlawToPcm8 (char*pSrcBuf,DWORD dwSrcSize,char *pDstBuf,DWORD dwDstSize8)

DWORD fPcm_MemAlawToPcm16(char* pSrcBuf,DWORD dwSrcSize,char * pDstBuf,DWORD dwDstSize16)

Parameter Description:

pSrcBuf	The pointer pointing to the buffer area storing source data, it stores A-law formatted voice data
dwSrcSize	The size (bytes) of pSrcBuf
pDstBuf	The pointer pointing to the buffer area storing the converted data, the storage space is allocated by the application
dwDstSize8 dwDstSize16	The size of pDstBuf (bytes). dwDstSize8 must be larger than or equal to dwSrcSize, dwDstSize16 must be larger than or equal to two times of dwSrcSize

Return Value:

0	Conversion failed. The failure reason can be obtained by the function fPcm_GetLastErrMsg
>0	The number of the converted bytes

Function Description:

fPcm_MemAlawToPcm8 converts the A-law format to be unsigned 8-bit PCM format, fPcm_MemAlawToPcm16 converts the A-law format to be 16 bit PCM format.

Note:

- Because this function is provided by ShPcmHandle.dll, it can be invoked before SsmStartCti is called successfully.

Related Information:

Driver version	SynCTI Ver. 4.7.2.0 or above
Header	ShPcmApi.h
Library	ShPcmHandle.lib
DLL	ShPcmHandle.dll

Related Function: fPcm_ALawConvertPcm16, fPcm_ALawConvertPcm8, fPcm_MemPcm8ToAlaw

2.28.20 fPcm_MemAlawToPcm16_16K

Converts the A-law formatted data stored in the buffer area to be 16-bit 16KHz PCM formatted data.

Format:

Int WINAPI fPcm_MemAlawToPcm16_16K(char * szSource, int nSourceLen, char * szTarget, int nTargetLen)

Parameter Description:

szSource	The pointer pointing to the buffer area storing source data, it stores A-law formatted voice data
nSourceLen	The size (bytes) of szSource
szTarget	The pointer pointing to the buffer area storing the converted data, the storage space is allocated by the application
nTargetLen	The size of szTarget (bytes). nTargetLen must be larger than or equal to four times of nSourceLen.

Return Value:

0	Conversion failed. The failure reason can be obtained by the function fPcm_GetLastErrMsg
>0	The number of the converted bytes

Function Description:

fPcm_MemAlawToPcm16_16K converts the A-law format to be the 16-bit 16KHz PCM format.

Note:

- Because this function is provided by ShPcmHandle.dll, it can be invoked before SsmStartCti is called successfully.

Related Information:

Driver version	SynCTI Ver. 5.2.0.1 or above
Header	ShPcmApi.h
Library	ShPcmHandle.lib
DLL	ShPcmHandle.dll

Related Function: fPcm_ALawConvertPcm16, fPcm_ALawConvertPcm8, fPcm_MemPcm8ToAlaw

2.28.21 fPcm_MemGSMToPcm16

Converts the GSM formatted data stored in the buffer to the 16-bit PCM formatted data.

Format:

DWORD WINAPI fPcm_MemGSMToPcm16(char *pSource, DWORD dwSourceSize, char *pTarget, DWORD dwTargetSize)

Parameter Description:

pSource	The pointer that points to the input buffer area storing GSM formatted voice data
dwSourceSize	The size of the input buffer area (bytes). This parameter should be in the range of 2600≤dwSourceSize＜3250 and is as suggested set to the integral times of 65.
pTarget	The pointer that points to the target buffer area storing 16-bit PCM formatted voice data.
dwTargetSize	The size of the target buffer area. It must be greater than or equal to 10 times of dwSourceSize.

Return Value:

0	Conversion failed
>0	The number of converted bytes

Function Description:

Converts the GSM formatted file to the 16-bit PCM formatted file.

Note:

- Because this function is provided by ShPcmHandle.dll, it can be invoked before the call of SsmStartCti.
- To invoke this function, the file macmcvt.dll must be used.
- To invoke this function, the ACM decoding engine of GSM must be used.

Related Information:

Driver version	SynCTI Ver. 4.8.0.0 or above
Header	ShPcmApi.h
Library	ShPcmHandle.lib
DLL	ShPcmHandle.dll

Related Function: fPcm_ALawConvertPcm16, fPcm_ALawConvertPcm8, fPcm_MemPcm8ToAlaw, fPcm_MemGSMToPcm8

2.28.22 fPcm_MemGSMToPcm8

Converts the GSM formatted data stored in the buffer to the unsigned 8-bit PCM formatted data.

Format:

DWORD WINAPI fPcm_MemGSMToPcm8(char *pSource, DWORD dwSourceSize, char *pTarget, DWORD dwTargetSize)

Parameter Description:

pSource	The pointer that points to the input buffer area storing GSM formatted voice data
dwSourceSize	The size of the input buffer area (bytes). This parameter should be in the range of 2600≤dwSourceSize＜3250 and is as suggested set to the integral times of 65
pTarget	The pointer that points to the target buffer area storing 8 Bit PCM formatted voice data
dwTargetSize	The size of the target buffer area. It must be greater than or equal to 5 times of

	dwSourceSize

Return Value:

0	Conversion failed
>0	The number of converted bytes

Function Description:

Converts the GSM formatted data stored in the buffer to the unsigned 8-bit PCM formatted data.

Note:

- Because this function is provided by ShPcmHandle.dll, it can be invoked before the call of SsmStartCti.
- To invoke this function, the file macmcvt.dll must be used.
- To invoke this function, the ACM decoding engine of GSM must be used.

Related Information:

Driver version	SynCTI Ver. 5.0.3.0 or above
Header	ShPcmApi.h
Library	ShPcmHandle.lib
DLL	ShPcmHandle.dll

Related Function: fPcm_ALawConvertPcm16, fPcm_ALawConvertPcm8, fPcm_MemPcm8ToAlaw, fPcm_MemGSMToPcm16

2.28.23 fPcm_MemGSMToUlaw

Converts the GSM formatted data stored in the buffer to μ-law formatted data.

Format:

DWORD WINAPI fPcm_MemGSMToUlaw (char *pSource, DWORD dwSourceSize, char *pTarget, DWORD dwTargetSize)

Parameter Description:

pSource	The pointer that points to the input buffer area storing GSM formatted voice data
dwSourceSize	The size of the input buffer area (bytes). This parameter should be in the range of 2600≤dwSourceSize＜3250 and is as suggested set to the integral times of 65
pTarget	The pointer that points to the target buffer area storing μ-law formatted voice data
dwTargetSize	The size of the target buffer area. It must be greater than or equal to 10 times of dwSourceSize

Return Value:

0	Conversion failed
>0	The number of converted bytes

Function Description:

Converts the GSM formatted data stored in the buffer to μ-law formatted data.

Note:

- Because this function is provided by ShPcmHandle.dll, it can be invoked before the call of SsmStartCti.
- To invoke this function, the file macmcvt.dll must be used.
- To invoke this function, the ACM decoding engine of GSM must be used.

Related Information:

Driver version	SynCTI Ver. 5.0.3.0 or above
Header	ShPcmApi.h
Library	ShPcmHandle.lib
DLL	ShPcmHandle.dll

Related Function: fPcm_ALawConvertPcm16, fPcm_ALawConvertPcm8, fPcm_MemPcm8ToAlaw,

fPcm_MemGSMToPcm16

2.28.24 fPcm_Pcm16ConvertALaw

Converts the 16-bit PCM formatted file to be A-law formatted file.

Format:

int fPcm_Pcm16ConvertALaw (char*fnPCM16, char* szTargetFile)

Parameter Description:

fnPCM16	The name of the 16-bit PCM formatted source file, it may include the full path of the file. The file format could be either the non-header file (i.e. without file header) or the standard wav file. It's through analyzing the source file to determine whether it has the wav file header instead of checking the file extension name that the driver decides if the source file is standard or not. If it's not a standard wav file, it will be regarded as a non-header file
szTargetFile	The target file name, it may include the full path of the file. If the target file doesn't exist, the driver will automatically create it; If the target file exists already, it will be overlaid. If the extension name of the target file is wav, the converted file format is the standard wav; otherwise, the format is plain

Return Value:

1	Successful
-1	Call failed. The failure reason can be obtained via the function fPcm_GetLastErrMsg

Function Description:

Converts the 16-bit PCM formatted file to be the A-law formatted file.

Note:

- Because this function is provided by ShPcmHandle.dll, it can be invoked before SsmStartCti is called successfully.

Related Information:

Driver version	SynCTI Ver. 4.7.1.7 or above
Header	ShPcmApi.h
Library	ShPcmHandle.lib
DLL	ShPcmHandle.dll

Related Function:

2.28.25 fPcm_Pcm16_16KconvertALaw

Converts the 16-bit 16KHz PCM formatted file to be A-law formatted file.

Format:

int fPcm_Pcm16_16KConvertALaw (char*fnPCM16, char* szTargetFile)

Parameter Description:

fnPCM16	The name of the 16-bit 16KHz PCM formatted source file, it may include the full path of the file. The file format could be either the non-header file (i.e. without file header) or the standard wav file. The driver, by analyzing the source file to determine whether it has the wav file header, but not by checking the file extension name, decides if the source file is standard or not. If it's not a standard wav file, it will be regarded as a non-header file.
szTargetFile	The target file name, it may include the full path of the file. If the target file doesn't exist, the driver will automatically create it; If the target file exists already, it will be overlaid. If the extension name of the target file is wav, the converted file format is the standard wav; otherwise, the format is plain.

Return Value:

1	Successful
-1	Call failed. The failure reason can be obtained via the function fPcm_GetLastErrMsg.

Function Description:

Converts the 16-bit 16KHz PCM formatted file to be the A-law formatted file.

Note:

- Because this function is provided by ShPcmHandle.dll, it can be invoked before SsmStartCti is called successfully.

Related Information:

Driver version	SynCTI Ver. 5.2.0.1 or above
Header	ShPcmApi.h
Library	ShPcmHandle.lib
DLL	ShPcmHandle.dll

Related Function:

2.28.26 fPcm_MemPcm8ToAlaw

Converts the unsigned 8-bit PCM formatted data stored in the buffer area to be the A-law formatted data.

Format:

int fPcm_MemPcm8ToAlaw(char * szSource, int nSourceLen, char * szTarget, int nTargetLen)

Parameter Description:

szSource	The pointer pointing to the first address of the input buffer area, it stores 8-bit PCM formatted voice data
nSourceLen	The size (bytes) of szSource
szTarget	The pointer pointing to the buffer area storing the converted data, the storage space is allocated by the application
nTargetLen	The size of szTarget, must be larger than or equal to dwSrcSize

Return Value:

0	Conversion failed. The failure reason can be obtained by the function fPcm_GetLastErrMsg
>0	The number of the converted bytes

Function Description:

Converts the unsigned 8-bit PCM formatted data stored in the buffer area to be the A-law formatted data.

Note:

- Because this function is provided by ShPcmHandle.dll, it can be invoked before SsmStartCti is called successfully.

Related Information:

Driver version	SynCTI Ver. 4.7.2.0 or above
Header	ShPcmApi.h
Library	ShPcmHandle.lib
DLL	ShPcmHandle.dll

Related Function: fPcm_MemAlawToPcm8, fPcm_ALawConvertPcm8

2.28.27 fPcm_MemAlawToUlaw

Refer to fPcm_MemUlawtoAlaw

2.28.28 fPcm_MemUlawtoAlaw

Converts the A-law formatted data stored in the buffer area to be μ-Law formatted data (fPcm_MemAlawToUlaw), or converts the μ-Law formatted data stored in the buffer area to be A-law formatted data (fPcm_MemUlawtoAlaw).

Format:

int WINAPI fPcm_MemAlawToUlaw(char * szSource, int nSourceLen, char * szTarget, int nTargetLen)

int WINAPI fPcm_MemUlawtoAlaw(char * szSource, int nSourceLen, char * szTarget, int nTargetLen)

Parameter Description:

szSource	The pointer pointing to the first address of the input buffer area, it stores A-Law or μ-Law formatted voice data
nSourceLen	The size (bytes) of szSource
szTarget	The pointer pointing to the buffer area storing the converted data, the storage space is allocated by the application
nTargetLen	The size of szTarget, must be larger than or equal to nSourceLen

Return Value:

0	Conversion failed. The failure reason can be obtained by the function fPcm_GetLastErrMsg
>0	The number of the converted bytes

Function Description:

fPcm_MemAlawToUlaw converts the A-law formatted data stored in the buffer area to be μ-Law formatted data; fPcm_MemUlawtoAlaw converts the μ-Law formatted data stored in the buffer area to be A-law formatted data.

Note:

- Because this function is provided by ShPcmHandle.dll, it can be invoked before the call of SsmStartCti.

Related Information:

Driver version	SynCTI Ver. 5.0.0.0 or above
Header	ShPcmApi.h
Library	ShPcmHandle.lib
DLL	ShPcmHandle.dll

Related Function:

2.28.29 fPcm_MemPcm16ToAlaw

Converts the unsigned 16-bit PCM formatted data stored in the buffer area to be the A-law formatted data.

Format:

DWORD fPcm_MemPcm16ToAlaw(char *pSrcBuf, DWORD dwSrcSize,char *pDstBuf, DWORD dwDstSize)

Parameter Description:

szSource	The pointer pointing to the first address of the input buffer area, it stores 16-bit PCM formatted voice data
nSourceLen	The size (bytes) of nSourceLen
szTarget	The pointer pointing to the buffer area storing the converted data, the storage space is allocated by the application
nTargetLen	The size of szTarget, must be larger than or equal to dwSrcSize

Return Value:

0	Conversion failed. The failure reason can be obtained by the function fPcm_GetLastErrMsg
>0	The number of the converted bytes

Function Description:

Converts the unsigned 16-bit PCM formatted data stored in the buffer area to be the A-law formatted data.

Note:

- Because this function is provided by ShPcmHandle.dll, it can be invoked before SsmStartCti is called successfully.

Related Information:

Driver version	SynCTI Ver. 4.7.2.0 or above
Header	ShPcmApi.h
Library	ShPcmHandle.lib
DLL	ShPcmHandle.dll

Related Function: fPcm_MemAlawToPcm16, fPcm_ALawConvertPcm16

2.28.30 fPcm_MemPcm16_16KtoAlaw

Converts the unsigned 16-bit 16KHz PCM formatted data stored in the buffer area to be the A-law formatted data.

Format:

DWORD WINAPI fPcm_MemPcm16_16KToAlaw(char * pSource, DWORD dwSourceSize, char * pTarget, DWORD dwTargetSize)

Parameter Description:

pSource	The pointer pointing to the first address of the input buffer area, it stores 16-bit PCM formatted voice data
dwSourceSize	The size (bytes) of pSource
pTarget	The pointer pointing to the buffer area storing the converted data, the storage space is allocated by the application
dwTargetSize	The size of pTarget, must be larger than or equal to 1/4 of dwSourceSize

Return Value:

0	Conversion failed. The failure reason can be obtained by the function fPcm_GetLastErrMsg
>0	The number of the converted bytes

Function Description:

Converts the unsigned 16-bit 16KHz PCM formatted data stored in the buffer area to be the A-law formatted data.

Note:

- Because this function is provided by ShPcmHandle.dll, it can be invoked before SsmStartCti is called successfully.

Related Information:

Driver version	SynCTI Ver. 5.2.0.1 or above
Header	ShPcmApi.h
Library	ShPcmHandle.lib
DLL	ShPcmHandle.dll

Related Function: fPcm_MemAlawToPcm16, fPcm_ALawConvertPcm16

2.28.31 fPcm_GC8Convert

Refer to fPcm_G729AConvert

2.28.32 fPcm_Vox6KTo8K

Refer to fPcm_Vox8KTo6K

2.28.33 fPcm_Vox8KTo6K

Implements the conversion between 6K sampling rate and 8K sampling rate of the VOX formatted voice file.

Format:

int fPcm_Vox6kTo8k(char *sz6kHzFile,char *sz8kHzFile)

int fPcm_Vox8KTo6K (char *sz8kHzFile,char *sz6kHzFile)

Parameter Description:

sz6kHzFile	The voice file with the sampling rate 6kHz and Vox encoding format, without file header. If the full path is not included, the driver will look for the file under the current directory
sz8kHzFile	The voice file with the sampling rate 8kHz and Vox encoding format, without file header. If the full path is not included, the driver will look for the file under the current directory

Return Value:

1	Successful
-1	Call failed. The failure reason can be obtained via the function fPcm_GetLastErrMsg

Function Description:

fPcm_Vox6kTo8k converts the Vox file with 6kHz sampling rate to be the Vox file with 8kHz sampling rate; fPcm_Vox8KTo6K converts the Vox file with 8kHz sampling rate to be the Vox file with 6kHz sampling rate.

Note:

- Because this function is provided by ShPcmHandle.dll, it can be invoked before SsmStartCti is called successfully.

Related Information:

Driver version	SynCTI Ver. 4.7.1.7 or above
Header	ShPcmApi.h
Library	ShPcmHandle.lib
DLL	ShPcmHandle.dll

Related Function:

2.28.34 fPcm_ULawConvertGSM

Converts the μ-Law formatted file to the GSM formatted file.

Format:

int WINAPI fPcm_ULawConvertGSM(char*szSourceFile, char*szTargetFile)

Parameter Description:

szSourceFile	The name of the μ-Law formatted source file, allowed to include the full path of the file. The file format could be either the non-header file or the standard wav file. The driver decides whether the source file is a standard wav file or not by analyzing if it has a wav file header, rather than by checking the file extension name. If it is not a standard wav file, it will be regarded as a non-header file
szTargetFile	The target file name, allowed to include the full path of the file. If the target file doesn't exist, the driver will automatically create it; If the target file exists already, it will be overlaid. If the extension name of the target file is wav, the converted file is a standard wav file; otherwise, it is a non-header file.

Return Value:

1	Call successful
-1	Call failed. The failure reason can be obtained via the function fPcm_GetLastErrMsg

Function Description:

fPcm_ULawConvertGSM is used to convert the μ-Law formatted file to the GSM formatted file.

Note:

- Because this function is provided by ShPcmHandle.dll, it can be invoked before the call of SsmStartCti.
- To invoke this function, the file macmcvt.dll must be used.
- To invoke this function, the ACM encoding engine of GSM must be used.

Related Information:

Driver version	SynCTI Ver. 4.8.0.0 or above
Header	ShPcmApi.h
Library	ShPcmHandle.lib
DLL	ShPcmHandle.dll, macmcvt.dll

Related Function: fPcm_Pcm16ConvertALaw, fPcm_GC8Convert, fPcm_MemAlawToPcm8, fPcm_MemAlawToPcm16

2.28.35 fPCM_Pcm16ConvertG729A

Converts the 16-bit PCM formatted file to the G.729A formatted file.

Format:

Int WINAPI fPCM_Pcm16ConvertG729A(char*szSoureFile, char* szTargetFile)

Parameter Description:

szSoureFile	The name of the 16-bit PCM formatted source file which is recorded by Synway boards. It may include the full path of the file. The file could be either a non-header file (i.e. without file header) or a standard wav file. Instead of checking the file extension name, the driver judges if the source file is standard or not by analyzing szSourceFile to find whether it has the wav file header. If it's not a standard wav file, it will be regarded as a non-header file.
szTargetFile	The target file name. It may include the full path of the file. If the target file doesn't exist, the driver will automatically create it; if the target file exists already, it will be overlaid. If the extension name of the target file is wav, the converted file is a standard wav file; otherwise, it will be a non-header file.

Return Value:

0	Successful
-1	Call failed. The failure reason can be obtained via the function call of fPcm_GetLastErrMsg

Function Description:

Converts the 16-bit PCM formatted file recorded by Synway boards to the G.729A formatted file.

Note:

- Because this function is provided by ShPcmHandle.dll, it can be invoked before the call of SsmStartCti.
- To invoke this function, the files macmcvt.dll and g729a.dll must be used.
- To invoke this function, the ACM encoding engine of G729A must be used.

Related Information:

Driver version	SynCTI Ver. 5.0.0.0 or above.
Header	ShPcmApi.h
Library	ShPcmHandle.lib
DLL	ShPcmHandle.dll

Related Function: fPcm_Pcm16ConvertALaw

2.28.36 fPcm_Pcm8ConvertGSM

Converts the 8-bit PCM formatted file to the GSM formatted file.

Format:

Int WINAPI fPcm_Pcm8ConvertGSM(char*szSoureFile, char* szTargetFile)

Parameter Description:

szSoureFile	The name of the 8-bit PCM formatted source file which is recorded by Synway boards. It may include the full path of the file. The file could be either a non-header file (i.e. without file header) or a standard wav file. Instead of checking the file extension name, the driver judges if the source file is standard or not by analyzing szSourceFile to find whether it has the wav file header. If it's not a standard wav file, it will be regarded as a non-header file.
szTargetFile	The target file name. It may include the full path of the file. If the target file doesn't exist, the driver will automatically create it; if the target file exists already, it will be overlaid. If the extension name of the target file is wav, the converted file is a standard wav file; otherwise, it will be a non-header file.

Return Value:

0	Successful
-1	Call failed. The failure reason can be obtained via the function call of fPcm_GetLastErrMsg

Function Description:

Converts the 8-bit PCM formatted file to the GSM formatted file.

Note:

- Because this function is provided by ShPcmHandle.dll, it can be invoked before the call of SsmStartCti.
- To invoke this function, the file macmcvt.dll must be used.
- To invoke this function, the ACM encoding engine of GSM must be used.

Related Information:

Driver version	SynCTI Ver. 5.1.0.0 or above.
Header	ShPcm.h
Library	ShPcmHandle.lib
DLL	ShPcmHandle.dll

Related Function:

2.28.37 fPcm_GSMToMp3

Converts the GSM formatted file to the MP3 formatted file.

Format:

Int WINAPI fPcm_GSMToMp3 (char*szSoureFile, char* szTargetFile)

Parameter Description:

szSoureFile	The name of the GSM formatted source file which is recorded by Synway boards. It may include the full path of the file. The file could be either a non-header file (i.e. without file header) or a standard wav file. Instead of checking the file extension name, the driver judges if the source file is standard or not by analyzing szSourceFile to find whether it has the wav file header. If it's not a standard wav file, it will be regarded as a non-header file.
szTargetFile	The target file name. It may include the full path of the file. If the target file doesn't exist, the driver will automatically create it; if the target file exists already, it will be overlaid. If the extension name of the target file is wav, the converted file is a standard wav file; if the extension name of the target file is mp3, the converted file is an mp3 file; otherwise, it will be a non-header file.

Return Value:

1	Successful

-1	Call failed. The failure reason can be obtained via the function call of fPcm_GetLastErrMsg

Function Description:

Converts the GSM formatted file recorded by Synway boards to the MP3 formatted file.

Note:

- Because this function is provided by ShPcmHandle.dll, it can be invoked before the call of SsmStartCti.
- To invoke this function, the file macmcvt.dll must be used.
- To invoke this function, the ACM decoding engine of GSM and the ACM encoding engine of MP3 must be used.

Related Information:

Driver version	SynCTI Ver. 5.0.0.0 or above.
Header	ShPcmApi.h
Library	ShPcmHandle.lib
DLL	ShPcmHandle.dll

Related Function:

2.28.38 fPcm_G729AConvert

Converts the G.792A formatted file to a voice file of other encoding format.

Format:

Int WINAPI fPcm_G729AConvert(char*szSoureFile, char* szTargetFile, int nTargetFormat)

Parameter Description:

szSoureFile	The name of the G.792A formatted source file, it may include the full path of the file. The file format could be either the non-header file (i.e. without file header) or the standard wav file. It's through analyzing the source file to determine whether it has the wav file header instead of checking the file extension name that the driver decides if the source file is standard or not. If it's not a standard wav file, it will be regarded as a non-header file
szTargetFile	The target file name, it may include the full path of the file. If the target file doesn't exist, the driver will automatically create it; If the target file exists already, it will be overlaid. If the extension name of the target file is wav, the converted file format is the standard wav; otherwise, the format is plain
nTargetFormat	The encoding format of the target file: =10001: 16 Bit PCM =6: A-law

Return Value:

1	Successful
-1	Call failed. The failure reason can be obtained by the function fPcm_GetLastErrMsg.

Function Description:

Converts the G.792A formatted file to a voice file of other encoding format.

Note:

- Because this function is provided by ShPcmHandle.dll, it can be invoked before SsmStartCti is called successfully.
- To invoke this function, the files macmcvt.dll and g729a.dll must be used.
- To invoke this function, the ACM decoding engine of G729A must be used.

Related Information:

Driver version	SynCTI Ver. 4.0 or above

Header	ShPcmApi.h
Library	ShPcmHandle.lib
DLL	ShPcmHandle.dll

Related Function: fPCM_AlawConvertGC8

2.28.39 fPCM_MemULawToG729A

Converts the μ-Law data saved in the buffer to G.729A format.

Format:

Int WINAPI fPCM_MemULawToG729A (char* szSoure, int nSourceLen, char* szTarget, int nTargetLen)

Parameter Description:

szSource	The pointer pointing to the buffer area that stores μ-Law formatted source data
nSourceLen	The size (bytes) of szSource
szTarget	The pointer pointing to the buffer area that stores the converted data, the storage space is allocated by the application. According to the scale between the sizes of μ-Law and G.729A format data, szTarget must be greater than nSourceLen/8.
nTargetLen	The size (bytes) of data actually returned by szTarget

Return Value:

>0	The number of converted bytes
0	Conversion failed. The failure reason can be obtained by the function fPcm_GetLastErrMsg

Function Description:

Converts the μ-Law data saved in the buffer to G.729A format.

Note:

- Because this function is provided by ShPcmHandle.dll, it can be invoked before the successful call of SsmStartCti.
- To invoke this function, the files macmcvt.dll and g729a.dll must be used.
- To invoke this function, the ACM encoding engine of G729A must be used.

Related Information:

Driver version	SynCTI Ver. 5.1.1.0 or above
Header	ShPcmApi.h
Library	ShPcmHandle.lib
DLL	ShPcmHandle.dll

Related Function:

2.28.40 fPcm_MemMp3ToPcm16

Converts the MP3 formatted data stored in the buffer area to be the 16-bit PCM formatted data.

Format:

Int WINAPI fPCM_MemMp3ToPcm16 (char* szSoure, int nSourceLen, char* szTarget, int nTargetLen)

Parameter Description:

szSoure	The pointer pointing to the buffer area that stores the MP3 formatted source data
nSourceLen	The size (bytes) of szSource
szTarget	The pointer pointing to the buffer area that stores the converted data, the storage space is allocated by the application. According to the scale between the size of the MP3 data and that of the 16 Bit PCM formatted data, szTarget must be greater than nSourceLen*16.
nTargetLen	The actual size (bytes) of the data that are returned by szTarget

Return Value:

>0	The number of converted bytes
0	Conversion failed. The failure reason can be obtained by the function fPcm_GetLastErrMsg

Function Description:

Converts the MP3 formatted data stored in the buffer area to be the 16-bit PCM formatted data.

Note:

- Because this function is provided by ShPcmHandle.dll, it can be invoked before SsmStartCti.
- To invoke this function, the file macmcvt.dll must be used.
- To invoke this function, the ACM decoding engine of MP3 must be used.
- You should give a complete piece of voice data but not some divided segments for conversion; otherwise the converted data may be incomplete. That's because the MP3 data in the buffer must be integral times of the frame to ensure the successful call of this function.

Related Information:

Driver version	SynCTI Ver. 5.1.1.0 or above
Header	ShPcm.h
Library	ShPcmHandle.lib
DLL	ShPcmHandle.dll

Related Function:

2.28.41 fPcm_MemG729AToPcm8

Refer to fPcm_MemG729AToPcm16

2.28.42 fPcm_MemG729AToPcm16

Converts the G.729A data saved in the buffer area to the PCM16 or PCM8 format.

Format:

DWORD WINAPI fPcm_MemG729AToPcm16(char *pSource, DWORD dwSourceSize, char *pTarget, DWORD dwTargetSize);

DWORD WINAPI fPcm_MemG729AToPcm8(char *pSource, DWORD dwSourceSize, char *pTarget, DWORD dwTargetSize);

Parameter Description:

pSource	The pointer pointing to the buffer area that stores G.729A formatted source data
dwSourceSize	The size (bytes) of pSource
pTarget	The pointer pointing to the buffer area that stores the converted data, the storage space is allocated by the application. According to the scale between the sizes of G.729A and PCM16 (or PCM8) formatted data, pTarget must be 16 times as dwSourceSize for the PCM16 formatted data and 8 times as dwSourceSize for the PCM8 formatted data.
dwTargetSize	The size (bytes) of data actually returned by pTarget

Return Value:

>0	The number of the converted bytes
0	Conversion failed. The failure reason can be obtained by the function fPcm_GetLastErrMsg

Note:

- Because this function is provided by ShPcmHandle.dll, it can be invoked before the successful call of SsmStartCti.

- To invoke the above functions, the files macmcvt.dll and g729a.dll must be used.
- To invoke the function fPcm_MemG729AToPcm16 or fPcm_MemG729AToPcm8, the ACM decoding engine of G729A must be used.

Related Information:

Driver version	SynCTI Ver. 5.3.1.1 or above
Header	ShPcmApi.h
Library	ShPcmHandle.lib
DLL	ShPcmHandle.dll

Related Function:

2.28.43 fPcm_GetLastErrMsg

Obtains the call failure reason of the functions for voice encoding format conversion.

Format:

int fPcm_GetLastErrMsg(char*szMsg)

Parameter Description:

szMsg	The pointer storing error messages, the storage space is allocated by the application

Return Value:

1	Successful
-1	Call failed

Function Description:

Obtains the call failure reason of the functions for voice encoding format conversion.

Note:

- Because this function is provided by ShPcmHandle.dll, it can be invoked before SsmStartCti is called successfully.

Related Information:

Driver version	SynCTI Ver. 4.0 or above
Header	ShPcmApi.h
Library	ShPcmHandle.lib
DLL	ShPcmHandle.dll

Related Function:

2.28.44 fPcm_Close

Releases all resources used by ShPcmHandle.dll.

Format:

void fPcm_Close()

Parameter Description: None

Return Value: None

Function Description:

Releases all resources used by ShPcmHandle.dll.

Note:

- Please make sure to call this function to release used resources after invoking any of the transcoding functions in ShPcmHandle.dll.

Related Information:

Driver version	SynCTI Ver. 4.0 or above
Header	ShPcmApi.h
Library	ShPcmHandle.lib
DLL	ShPcmHandle.dll

Related Function:

2.28.45 fPcm_NotchFilter_ULAW

Filters the specific frequency from a μ-Law formatted voice file which is recorded by the Synway board using the notch filtering algorithm.

Format:

int WINAPI fPcm_NotchFilter_ULAW(LPCSTR szSourceFile, LPCSTR szTargetFile, UINT nFrequence, int nField, UINT nGain)

Parameter Description:

szSoureFile	The name of the μ-Law source file recorded by Synway boards, which may include the full path of the file. The file format could be either the non-header file (i.e. without file header) or the standard wav file. It's through analyzing the source file of szSourceFile to determine whether it has the wav file header instead of checking the file extension name that the driver decides if the source file is standard or not. If it's not a standard wav file, it will be regarded as a non-header file.
SzTargetFile	The target file name, which may include the full path of the file. If the target file doesn't exist, the driver will automatically create it; If the target file exists already, it will be overlaid. If the extension name of the target file is wav, the converted file format is the standard wav; otherwise, the format is plain.
nFrequence	The specific frequency to be notch filtered.
nField	The threshold value range for the notch filter frequency which is calculated as nFrequence±nField.
nGain	The energy gain. It is used only to reduce the voice energy (nGain >= 1). A larger value indicates a more obvious adjustment.

Return Value:

1	Call successful
-1	Call failed. The failure reason can be obtained by the function fPcm_GetLastErrMsg

Function Description:

Filters the specific frequency from a μ-Law formatted voice file which is recorded by the Synway board using the notch filtering algorithm.

Note:

- Because this function is provided by ShPcmHandle.dll, it can be invoked before SsmStartCti.

Related Information:

Driver version	SynCTI Ver. 5.3.2.7 or above
Header	ShPcmApi.h
Library	ShPcmHandle.lib
DLL	ShPcmHandle.dll

Related Function:

2.28.46 fPcm_ ALawToAdpcm

Converts the A-law formatted file recorded by Synway boards to be IMA ADPCM formatted file.

Format:

in WINAPI fPcm_ALawToAdpcm(LPCSTR szSourceFile, LPCSTR szTargetFile)

Parameter Description:

szSoureFile	The name of the A-law formatted source file recorded by Synway boards, it may include the full path of the file. The file format could be either the non-header file (i.e. without file header) or the standard wav file. It's through analyzing the source file of szSourceFile to determine whether it has the wav file header instead of checking the file extension name that the driver decides if the source file is standard or not. If it's not a standard wav file, it will be regarded as a non-header file
szTargetFile	The target file name, it may include the full path of the file. If the target file doesn't exist, the driver will automatically create it; If the target file exists already, it will be overlaid. If the extension name of the target file is wav, the converted file format is the standard wav; otherwise, the format is plain

Return Value:

0	Successful
-1	Call failed. The failure reason can be obtained by the function fPcm_GetLastErrMsg.

Function Description:

Converts the A-law formatted file recorded by Synway boards to be IMA ADPCM formatted file.

Note:

- Because this function is provided by ShPcmHandle.dll, it can be invoked before SsmStartCti.

Related Information:

Driver version	SynCTI Ver. 5.3.3.1 or above
Header	ShPcmApi.h
Library	ShPcmHandle.lib
DLL	ShPcmHandle.dll

Related Function:

2.28.47 fPcm_InitEx

Initializes the memory and other resources in the driver which are required for converting voice data.

Format:

int fPcm_InitEx(int nMaxCh)

Parameter Description:

nMaxCh	The number of channels used to initialize macmcvt.dll.

Return Value:

0	Call failed. The failure reason can be obtained by the function fPcm_GetLastErrMsg.
1	Successful

Function Description:

Initializes the memory and other resources in the driver which are required for converting voice data.

Note:

- ShPcmHandle.dll will use the default number of channels 100 to initialize macmcvt.dll when the transcoding functions in it are invoked. In such case, this function is not necessary to be called. However, if 100 channels are insufficient for the application to use, this function can be invoked before the transcoding functions in ShPcmHandle.dll to set a number more than 100 for the initialization of macmcvt.dll.

- If ShPcmHandle.dll and SHP_A3.dll are invoked at the same time, SHP_A3.dll will use the total number of channels in the application to initialize macmcvt.dll while calling the function SsmStartCti, In such case, this function is not necessary to be called.
- The application must invoke the function fPcm_Close to release relative resources if it will no longer invoke the functions in ShPcmHandle.dll.

Related Information:

Driver version	SynCTI Ver. 5.3.3.2 or above
Header	ShPcmApi.h
Library	ShPcmHandle.lib
DLL	ShPcmHandle.dll

Related Function:

2.28.48 fPcm_MemPcm16ToGSMEx

Converts the 16-bit PCM formatted data stored in the buffer area to be the GSM formatted data.

Format:

DWORD fPcm_MemPcm16ToGSMEx(int nCh,char *pSource, DWORD dwSourceSize, char *pTarget, DWORD dwTargetSize)

Parameter Description:

nCh	The channel number used for converting. Be sure that data of the same 16-bit PCM source use the same nCh, and data of different 16-bit PCM sources use different nChs.
pSource	The pointer pointing to the buffer area that stores the 16-bit PCM formatted data
dwSourceSize	The size (bytes) of pSource
pTarget	The pointer pointing to the buffer area that stores the converted data, the storage space is allocated by the application.
dwTargetSize	The size (bytes) of pTarget

Return Value:

0	Call failed. The failure reason can be obtained by the function fPcm_GetLastErrMsg.
>0	Successful

Function Description:

Converts the 16-bit PCM formatted data stored in the buffer area to be the GSM formatted data.

Note:

- Because this function is provided by ShPcmHandle.dll, it can be invoked before SsmStartCti.

Related Information:

Driver version	SynCTI Ver. 5.3.3.2 or above
Header	ShPcmApi.h
Library	ShPcmHandle.lib
DLL	ShPcmHandle.dll

Related Function:

2.29 Faxing Functions (CTI Series)

2.29.1 Setting Faxing Parameters

2.29.1.1 SsmFaxSetChSpeed

Sets the maximum rate used to send or receive fax on a designated fax channel.

Format:

int SsmFaxSetChSpeed(int ch, int speed)

Parameter Description:

ch	Fax channel number
speed	Maximum rate (bps). Optional value: 4800bps, 9600bps, 14400bps, 33600bps; Default value: 9600bps

Return Value:

-1	Call failed
0	Successful

Function Description:

Sets the maximum rate used to send or receive fax on a designated fax channel.

Note:

- The parameter set by this function is only effective to the designated fax channel;
- When the set rate exceeds the maximum rate supported by the board, the latter becomes the upper limit to the fax rate.

Related Information:

Driver version	SynCTI Ver. 5.3.2.1 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmFaxGetSpeed, SsmFaxSetMaxSpeed

2.29.1.2 SsmFaxSetMaxSpeed

Sets the maximum rate used to send or receive fax.

Format:

void SsmFaxSetMaxSpeed(int speed)

Parameter Description:

speed	Maximum rate (bps). Optional value: 4800bps, 9600bps, 14400bps, 33600bps; Default value: 9600bps

Return Value: none

Function Description:

Sets the maximum rate used to send or receive fax.

Note:

- The parameters set by this function are effective to all the fax channels;
- When the set rate exceeds the maximum rate supported by the board, the latter becomes the upper limit

to the fax rate.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmFaxGetSpeed, SsmFaxSetChSpeed

2.29.1.3 SsmFaxSetID

Sets the local fax identification code.

Format:

int SsmFaxSetID(int ch,char *szID)

Parameter Description:

ch	Fax channel number
szID	Pointer pointing to the string of the ASCII formatted local fax identification code. Valid length: less than 20 characters

Return Value:

-1	Call failed
0	Successful

Function Description:

Sets the local fax identification code. The identification code could be either the local phone number or some other ASCII formatted string. This local identification code will be sent to the remote fax machine during faxing.

Note:

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmFaxGetID

2.29.2 Transmitting Fax

2.29.2.1 SsmFaxStartSend

Refer to SsmFaxStartSendEx

2.29.2.2 SsmFaxStartSendEx

Starts the transmission of a single fax file. SsmFaxStartSend is used to transmit the entire file. SsmFaxStartSendEx is used to transmit a range of designated pages.

Format:

int SsmFaxStartSend(int ch,char *filename)

int SsmFaxStartSendEx(int ch, char *filename, int nStartPage, int nEndPage)

Parameter Description:

ch	Fax channel number
filename	Name of the file in the tiff format. The file extension name could be '.tif'or '.tiff'. For

	more information about file formats supported by fax channels, refer to Fax in Chapter 1
nStartPage	Designated starting page number of the fax file. If 'filename' includes N pages, the range of value is: 1~N
nEndPage	Designated ending page number of the fax file. If 'filename' includes N pages, the range of value is: 1~N: the number of the designated ending page; ≥N: transmit until page number N (included); -1: transmit until the last page (i.e. the page numbered N). **Note: nEndPage must be larger than or equal to nStartPage**

Return Value:

0	Successful
-1	Call failed. The failure reason may be that the channel is not free or failure in opening the file

Function Description:

Starts the transmission of a single fax file.

After the fax transmission is started, the driver begins to exchange the handshake message with the remote fax machine. If the handshake is successful, the fax file will be sent to the remote fax machine.

Each time when the driver finishes the transmission of one page, it throws out the event of E_CHG_FaxPages to the application. When the driver finishes the fax transmission, it throws out the event of E_PROC_FaxEnd to the application. During the fax transmission, the application can call the function of SsmFaxCheckEnd to check whether the transmission has been finished successfully, or call the function of SsmFaxStop to terminate the transmission.

Note:

- Only when the incoming or outgoing call has been established on the analog trunk channel or the digital trunk channel, can the fax transmission be started.
- The fax channel only works in simplex mode, i.e. it can't transmit and receive fax simultaneously.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmFaxCheckEnd, SsmFaxStop, SsmFaxSendMultiFile, SsmFaxSendMultiFileEx, SsmFaxAppendSend

2.29.2.3 SsmFaxSendMultiFile

Refer to SsmFaxSendMultiFileEx

2.29.2.4 SsmFaxSendMultiFileEx

Starts transmission of multiple fax files, SsmFaxSendMultiFileEx can specify the range of pages to be transmitted.

Format:

int SsmFaxSendMultiFile(int ch, char * szPath, char * szFile)

int SsmFaxSendMultiFileEx(int ch, FAX_FILE_SCT *pFileNameList, int nNum)

Parameter Description:

ch	Fax channel number

szPath	The path of the fax file szFile. For example, if the fax file is stored under the directory of FaxFile in drive C, for C/C++, szPath can be written as: 'C:\\FaxFile\\'
szFile	List of fax file names, the file extension can be '.tif' or '.tiff', the adjacent file names are separated by ';' For example, there are two fax files to be transmitted, namely 'A.tif' and 'B.tif', szFile should be written as 'A.tif;B.tif'. **Note: the fax files in szFile must have the same formats**

pFileNameList	The fax file list. The struct of FAX_FILE_SCT is declared as:
	typedef struct tagFAX_FILE_SCT{
	char szFileName[256]; //The file name including the full path
	int nStartPage; //The start page, range of value: 1~N
	int nEndPage; //=1~N: The designated end page; =-1: the last page
	int nReserve1; //Reserved
	int nReserve2; //Reserved
	}FAX_FILE_SCT, *PFAX_FILE_SCT;
	In the above struct, N is the total number of pages in the fax file. For example, nStartPage=1 and nEndPage=1 means that only the first page is sent; nStart=1 and nEndPage=2 means that the first and second pages are sent; if nStart=1 and nEndPage=-1, the entire file is transmitted.
	Note: nEndPage must be larger than or equal to nStartPage

nNum	The number of the fax files

Return Value:

0	Successful
-1	Call failed, the failure reason can be obtained by the function SsmGetLastErrMsg

Function Description:

Starts transmission of multiple fax files.

After this function is called, the driver starts to exchange the handshake message with the remote fax machine, and then transmits the files designated in szFile based on the sequence from left to right to the remote fax machine, or sends the fax files starting from the head of the list of pFileNameList until the last file is transmitted.

Each time when the driver finishes the transmission or reception of one page, it throws out the event of E_CHG_FaxPages to the application. When the driver finishes the fax transmission, it throws out the event of E_PROC_FaxEnd to the application. During the transmission, the application can call the function of SsmFaxCheckEnd to check whether the transmission has been finished successfully, or call the function of SsmFaxStop to terminate the transmission.

Note:

- Only when the incoming or outgoing call has been established on the analog trunk channel or the digital trunk channel, can the fax transmission be started.
- The fax channel only works in simplex mode, i.e. it can't transmit and receive fax simultaneously.
- The punctuation symbols such as colon, pause, period, etc are illegal in the file path string. When multiple fax files are transmitted simultaneously, it's required that the fax files have the same format.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmFaxCheckEnd, SsmFaxStop, SsmFaxStartSend, SsmFaxStartSendEx

2.29.2.5 SsmFaxAppendSend

Appends a single file to send.

Format:

int SsmFaxAppendSend (int ch,char *filename)

Parameter Description:

ch	Fax channel number
filename	Fax file name. The file name extension can be '.tif' or '.tiff'. For more information about the file formats supported by fax channels, refer to Fax in Chapter 1

Return Value:

0	Successful, the driver has added the designated fax file into the fax transmission pool
-1	Call failed, the failure reason may be that the channel is free or failure occurs in opening the file

Function Description:

Appends a single file to send. It's applicable to the fax transmission triggered by the function call of SsmFaxSendMultiFile, SsmFaxStartSend or SsmFaxStartSendEx.

After this function is called, the driver adds the appended fax file into the fax transmission pool. When the current fax file has been transmitted, the fax files in the transmission pool will be sent immediately.

Note:

- This function can be called only when the fax transmission has been started.
- During the fax transmission, if the transmission is stopped by the client or the transmission failed, the appended fax file will be discarded.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmFaxStartSend, SsmFaxStartSendEx, SsmFaxSendMultiFile, SsmFaxCheckEnd, SsmFaxStop

2.29.3 Receiving Fax

2.29.3.1 SsmFaxStartReceive

Starts a fax reception.

Format:

int SsmFaxStartReceive(int ch,char *pszFileName)

Parameter Description:

ch	Fax channel number
pszFileName	The name of the Tiff formatted file storing fax data; it must use '.tif' or '.tiff' as the file extension. If the file exists already, its original content will be overwritten

Return Value:

0	Successful
-1	Call failed, the failure reason may be that the channel is not free or failure occurs in opening the file

Function Description:

Starts a fax reception.

After this function is called, the driver starts to exchange the handshake message with the remote fax machine. If

the handshake is successful, it starts to receive the fax and stores the fax into the file designated by the parameter pszFileName.

Each time when the driver finishes the reception of one page, it throws out the event of E_CHG_FaxPages to the application. When the driver finishes the reception of all the fax pages, it throws out the event of E_PROC_FaxEnd to the application. During the fax reception, the application may call the function of SsmFaxStop to terminate the fax reception.

Note:

- The fax channel only works in simplex mode, i.e. it can't transmit and receive fax simultaneously.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmFaxCheckEnd, SsmFaxStop

2.29.4 Stopping Faxing

2.29.4.1 SsmFaxStop

Stops the current faxing compulsorily.

Format:

int SsmFaxStop(int ch)

Parameter Description:

ch	Fax channel number

Return Value:

-1	Call failed
0	Successful

Function Description:

Stops the current faxing (transmission/reception) compulsorily.

This function not only terminates the fax transmission triggered by the function call of SsmFaxSendMultiFile, SsmFaxSendMultiFileEx, SsmFaxStartSend and SsmFaxStartSendEx but also terminates the fax reception triggered by the function call of SsmFaxStartReceive.

After this function call has completed, the driver will throw out the event of E_PROC_FaxEnd to the application. The execution status of this function can also be obtained via the function call of SsmFaxCheckEnd.

Note:

- This function is an asynchronous function; its execution status can be obtained via the function call of SsmFaxCheckEnd.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmFaxStartSend, SsmFaxStartSendEx, SsmFaxSendMultiFile, SsmFaxSendMultiFileEx, SsmFaxAppendSend, SsmFaxStartReceive, SsmFaxCheckEnd

2.29.5 Obtaining Faxing Information

2.29.5.1 SsmFaxGetSpeed

Obtains the current fax transmission/reception rate.

Format:

int SsmFaxGetSpeed(int ch)

Parameter Description:

ch	Fax channel number

Return Value:

-1	Not supported by this function
>0	Obtains the actual fax rate, whose possible values are 24, 48, 72, 96, 120, 144, 168, 192, 216, 240, 264, 288, 312, 336, respectively representing 2400, 4800, 7200, 9600, 12000, 14400, 16800, 19200, 21600, 24000, 26400, 28800, 31200, 33600bps

Function Description:

Obtains the current fax transmission/reception rate.

Note:

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmFaxSetMaxSpeed, SsmFaxSetChSpeed

2.29.5.2 SsmFaxCheckEnd

Obtains the execution status of the fax transmission or reception.

Format:

int SsmFaxCheckEnd(int ch)

Parameter Description:

ch	Fax channel number

Return Value:

0	Faxing has not been finished
1	Faxing finishes, and the fax channel enters the 'idle' state
2	Driver error occurs during the latest fax transmission or reception, or the faxing has been terminated by the function call of the SsmFaxStop and the channel has returned to the idle state. If this function is called again, it will return 1; For the SHF-4D series board, if the channel no longer starts a faxing task, the return value will be 2; if the channel starts a faxing task again, the return value will be 0
3	Fax data transmission or reception has been completed, and the negotiation process of disconnection is undergoing
-1	Call failed

Function Description:

Obtains the execution status of the fax transmission or reception.

Note:

- This function is applicable to both fax transmission and fax reception;

- If the return value is 2, the faxing has not been finished, but the fax data transmission or reception may have been finished;
- If the application uses the event mode for programming, we suggest you use the event of E_PROC_FaxEnd instead.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmFaxStartSend, SsmFaxStartSendEx, SsmFaxSendMultiFile, SsmFaxStop, SsmFaxSendMultiFileEx, SsmFaxStartReceive, SsmFaxAppendSend

2.29.5.3 SsmFaxGetDcnTag

When the fax reception via a fax board is successfully completed, judge if the remote fax machine has ever been compelled to stop.

Format:

int WINAPI SsmFaxGetDcnTag(int ch)

Parameter Description:

ch	Fax channel number

Return Value:

-1	This function is unsupported by the channel.
0	The faxing process is complete.
1	If the faxing is terminated by receiving a DCN message, it may result from the compulsive stop of the fax machine.

Function Description:

When the fax reception via a fax board is successfully completed, judge if the remote fax machine has ever been compelled to stop.

Note:

- This function is only applicable to the fax reception.
- This function can be used to judge if the remote fax machine has ever been stopped compulsively, and it should be invoked after SsmFaxCheckEnd returns 1.
- If the application program uses the event mode for programming, we suggest you use the event E_PROC_FaxDcnTag instead.

Related Information:

Driver version	SynCTI Ver. 5.0.2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmFaxCheckEnd

2.29.5.4 SsmFaxGetChStateMsg

Obtains the fax channel state.

Format:

int SsmFaxGetChStateMsg(int ch ,char *buf)

Parameter Description:

ch	Fax channel number
buf	Returns the string indicating the channel state. The storage space of buf must be allocated by the application and greater than 100 characters

Return Value:

-1	Call failed
0	Successful

Function Description:

Obtains the fax channel state.

Note:

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function:

2.29.5.5 SsmFaxGetPages

Obtains the number of sent/received pages during faxing.

Format:

int SsmFaxGetPages(int ch)

Parameter Description:

ch	Fax channel number

Return Value:

-1	Call failed
>0	The number of the finished fax pages

Function Description:

Obtains the number of sent/received pages during faxing.

Note:

- If, after the fax transmission/reception is finished, the return value of this function is 0, it means the current fax transmission/reception failed;
- If the application uses the event mode for programming, we suggest you use the event of E_CHG_FaxPages instead.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function:

2.29.5.6 SsmFaxGetFailReason

Obtains the faxing failure reason for the fax channel.

Format:

int SsmFaxGetFailReason(int ch)

Parameter Description:

ch	Fax channel number

Return Value:

-1	Call failed or error information of 'No Fax'
0	Fail to send Dis
1	No dcs information received (the remote end is not a fax machine or the 'Start' button on the fax machine is not pressed)
2	dis information received during the fax reception; the remote end is also receiving the fax, which is a wrong operation of the remote end
3	No training packet detected during the fax reception
4	Always fail to pass the training during the fax
5	Fail to send FTT information
6	Fail to send cfr
7	No carrier detected
8	No subsequent frames detected after carrier reception
9	Fail to send MCF
10	Fail to send PPR
11	Fail to send CTR
12	Fail to send RTN
13	Fail to send ERR
14	Unidentifiable frames received
15	Received page data not good
16	Data unavailable
17	Data error in the pages sent by the remote end
20	No Dis information received by the fax sender (the remote end is not a fax machine or the 'Start' button on the fax machine is not pressed, or the the fax board fails to detect Dis information)
21	No response to train is received from the fax receiver
22	Fail to send fax data
23	Fail to send frames after fax transmission
24	No response from the fax receiver detected after the transmission of a page of data
25	DSP running out of track
26	Fail to write data into file
27	Fax resending refused by the remote end
28	Faxing stopped on the application layer
29	Repeatedly fail to send RR
30	RR T5 times out
31	Disturbed at the beginning of fax reception
32	Wrong DCS
33	Wrong DIS
34	Connection error between fax channel and voice channel, or invoking SsmStopTalkWith too early

Function Description:

Obtains the faxing failure reason for the fax channel.

Note:

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h

Library	shp_a3.lib
DLL	shp_a3.dll

Related Function:

2.29.5.7 SsmFaxGetID

Obtains the identification code of the remote fax machine.

Format:

int SsmFaxGetID(int ch,char *myid)

Parameter Description:

ch	Fax channel number
myid	The pointer pointing to the buffer area storing the identification code, its storage space is allocated by the application. The length of the buffer area can't be less than 20 bytes

Return Value:

-1	Call failed
0	Successful

Function Description:

Obtains the identification code of the remote fax machine.

Note:

- This function can only be called after the handshake has been completed; otherwise, the correct identification code is not able to be obtained.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function: SsmFaxSetID

2.29.5.8 SsmFaxGetMode

Obtains the operating status and supported modes of the fax channel—transmitting or receiving, fine mode or common mode, ECM mode or data stream mode.

Format:

int SsmFaxGetMode(int ch, int * pnDir, int *pnResMode, int * pnTransMode)

Parameter Description:

ch	The number of the fax channel
pnDir	Used to store transmitting or receiving indicator =0: indicates the fax channel is receiving data =1: indicates the fax channel is transmitting data
pnResMode	Used to store the fax data mode indicator =0: indicates the common mode =1: indicates the fine mode
pnTransMode	Used to store the transmitting mode indicator for the fax channel =0: indicates the data stream mode =1: indicates the ECM mode

Return Value:

-1	The fax channel does not support this operation
0	The fax channel is idle and necessary parameters are unavailable by this function
1	Function call is successful

Function Description:

Obtains the operating status and supported modes of the fax channel—transmitting or receiving, fine mode or common mode, ECM mode or data stream mode.

Note:

- Only after the handshake phase is completed can this function be invoked; otherwise it is impossible to obtain the correct indicator.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function:

2.29.5.9 SsmFaxGetAllBytes

During fax transmission, obtains the total number of bytes on the fax page.

Format:

int SsmFaxGetAllBytes(int ch)

Parameter Description:

ch	Fax channel number

Return Value:

-1	Call failed
≥0	The total number of bytes on the current fax page

Function Description:

During fax transmission, obtains the total number of bytes on the fax page.

Note:

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function:

2.29.5.10 SsmFaxGetSendBytes

Obtains the number of sent bytes on the current page during fax transmission.

Format:

int SsmFaxGetSendBytes(int ch)

Parameter Description:

ch	Fax channel number

Return Value:

-1	Call failed
≥0	The number of sent bytes on the current page

Function Description:

Obtains the number of sent bytes on the current page during fax transmission.

Note:

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function:

2.29.5.11 SsmFaxGetRcvBytes

Obtains the number of received bytes on the current page during fax reception.

Format:

int SsmFaxGetRcvBytes(int ch)

Parameter Description:

ch	Fax channel number

Return Value:

-1	Call failed
≥0	The number of received bytes on the current page

Function Description:

Obtains the number of received bytes on the current page during fax reception.

Note:

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function:

2.29.5.12 SsmFaxGetCodeMode

Obtains the fax CODEC – MH, MR, MMR.

Format:

int SsmFaxGetCodecMode(int ch, DWORD * dwReserver)

Parameter Description:

ch	Fax channel number
dwReserver	Reserved

Return Value:

0	MH
1	MR
2	MMR

Function Description:

Obtains the fax CODEC.

Note:

- This function can only be invoked after the handshake has been completed, otherwise, the correct identification code will fail to be obtained.

Related Information:

Driver version	SynCTI Ver. 5.3.1.5 or above
Header	shpa3api.h
Library	shp_a3.lib
DLL	shp_a3.dll

Related Function:

2.29.6 Relative Operations on .tif File

2.29.6.1 fBmp_ValidateFaxFile

Judges whether the .tif file conforms to the fax format of Synway boards.

Format:

int fBmp_ValidateFaxFile(char *szFile)

Parameter Description:

szFile	The string storing the fax file

Return Value:

0	Successful, fax transmission supports the .tif file
-1	Fax transmission doesn't support the .tif file, more detailed reason can be obtained via the function fBmp_GetErrMsg
-2	Failure in opening the file, more detailed reason can be obtained via the function fBmp_GetErrMsg

Function Description:

Judges whether the .tif file conforms to the fax format of Synway boards.

Note:

Related Information:

Driver version	SynCTI Ver. 4.7.3.1 or above
Header	BmpApi.h
Library	BmpUtil.lib
DLL	BmpUtil.dll

Related Function:

2.29.6.2 fBmp_SetHeaderFormat

Sets the property of the page header of .tif formatted fax file.

Format:

int fBmp_SetHeaderFormat(int nRow, int nFromX, int nFromY, char szFrom, int nSubX, int nSubY, char*szSubject, int nToX, int nToY, char*szTo, int nTimeX, int nTimeY, char*szTime)

Parameter Description:

nRow	Number of rows of the page header, range of value: 1~3, with the default value of 1
nFromX	Starting position of the field of From, with the default value of 5
nFromY	The row where the field 'From' locates, range of value: 1~3, with the default value of 1,

	indicating the first row
szFrom	Prompting field of the field 'From', by default it is 'From:', and the maximum length is 20 bytes
nSubX	Starting position of the field 'Subject', with the default value of 270
nSubY	The row where the field 'Subject' locates, range of value: 1~3, with the default value of 1, indicating the first row
szSubject	Prompting field of the field 'Subject', by default it's 'Sub', with the maximum length of 20 bytes
nToX	Starting position of the field 'To', with the default value of 440
nToY	The row where the field 'To' locates, range of value: 1~3, with the default value of 1, indicating the first row
szTo	Prompting field of the field 'To', by default it is 'To:', with the maximum length of 20 bytes
nTimeX	Starting position of the field 'Time', with the default value of 660
nTimeY	The row where the field 'Time' locates, range of value: 1~3, with the default value of 1, indicating the first row
szTime	Prompting field of the field 'Time', by default it is '', with the maximum length of 20 bytes

Return Value:

-1	Call failed, the failure reason can be obtained via the function fBmp_GetErrMsg
1	Successful

Function Description:

Sets the property of the page header generated by fBmp_AddTxtToTif.

Note:

- If 'Subject' needs to be put in the second row, set nRow =2, nSubY=2.
- The horizontal range is 0~864. Adjust the positions of the data fields and keep them in the range.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	BmpApi.h
Library	BmpUtil.lib
DLL	BmpUtil.dll

Related Function: fBmp_AddTxtToTif, fBmp_AddTxtToTif_Big, fBmp_UniteTif

2.29.6.3 fBmp_AddTxtToTif

Adds a page header to the .tif file (only international encoding is supported).

Format:

int fBmp_AddTxtToTif(char szTifName, char* szFaxFrom, char*szFaxTo, char *szFaxSubbject, char * szDataTime , char *szTargetFile, DWORD dwReserve)

Parameter Description:

szTifName	The .tif file required to add with a page header
szFaxFrom	The information about the fax transmitting end, the maximum length is 20 bytes, each Chinese character occupies 2 bytes
szFaxTo	The information about the fax receiving end, the maximum length is 20 bytes, each Chinese character occupies 2 bytes
szFaxSubject	The fax subject, the maximum length is 10 bytes, each Chinese character occupies 2 bytes
szDataTime	Time information, the maximum length is 20 bytes
szTargetFile	The target tif file with page headers, if it has the same name as szTifName, the original file will be overwritten
dwReserve	When it is set to 1, the above parameters have no limit in length;

	If the bit2 of dwReserve is set to 1, the higher 16 bits of it denote the page on which the page header will be added; If the bit3 of dwReserve is set to 1, the page header will automatically code the pagination for the current page.

Return Value:

-1	Call failed, the failure reason can be obtained via the function fBmp_GetErrMsg
1	Successful

Function Description:

Adds a page header to the .tif file and supports the page header to automatically code the pagination (format: P.%d/%d) for the current page. To be exact, this function adds a page header to the file szTifName and outputs the result to szTargetFile. The page header includes 16 rows and each row has 1728 pixels.

Note:

- If szTifName and szTargetFile are the same, the original file will be overwritten;
- The data on the top of the file from left to right are: szFaxFrom, szSubject, szFaxTo, szDataTime.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	BmpApi.h
Library	BmpUtil.lib
DLL	BmpUtil.dll

Related Function: fBmp_SetHeaderFormat, fBmp_AddTxtToTif_Big, fBmp_CutTifHeader

2.29.6.4 fBmp_AddTxtToTif_Big

Adds a page header to the .tif file. Apart from supporting the international code, it also supports Big-5 traditional Chinese characters code.

Format:

int fBmp_AddTxtToTif_Big(char*szTifName, char*szFaxFrom, char*szFaxTo, char*szFaxSubject, Char *szDataTime, char *szTargetFile, DWORD dwReserve)

Parameter Description:

szTifName	The .tif file required to add with a page header
szFaxFrom	The information about the fax transmitting end, the maximum length is 20 bytes, each Chinese character occupies 2 bytes
szFaxTo	The information about the fax receiving end, the maximum length is 20 bytes, each Chinese character occupies 2 bytes
szFaxSubject	The fax subject, the maximum length is 10 bytes, each Chinese character occupies 2 bytes
szDataTime	Time information, the maximum length is 20 bytes
szTargetFile	The target tif file with the page header, if it has the same name as szTifName, the original file will be overwritten
dwReserve	If the bit2 of dwReserve is set to 1, the higher 16 bits of it denote the page on which the page header will be added; If the bit3 of dwReserve is set to 1, the page header will automatically code the pagination for the current page

Return Value:

-1	Call failed, the failure reason can be obtained via the function fBmp_GetErrMsg
1	Successful

Function Description:

Adds a page header to the .tif file. It also supports Big-5 traditional Chinese characters code.

This function adds a page header to the file szTifName and outputs the result to szTargetFile. The page header includes 16 rows and each row has1728 pixels.

Note:

- If szTifName and szTargetFile are the same, the original file will be overwritten;
- The data on the top of the file from left to right are: szFaxFrom, szSubject, szFaxTo, szDataTime;
- The function supports the fonts supported by the operating system.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	BmpApi.h
Library	BmpUtil.lib
DLL	BmpUtil.dll

Related Function: fBmp_SetHeaderFormat, fBmp_AddTxtToTif, fBmp_CutTifHeader

2.29.6.5 fBmp_UniteTif

Integrates two .tif files.

Format:

int fBmp_UniteTif(char*szHeadTif, char * szSourceTif, char *szTargetTif, DWORD dwReserve)

Parameter Description:

szHeadTif	The smaller .tif file added to the top of another file. It can only be a single page file
szSourceTif	The .tif file to which the smaller file will be added. It can be a file with multiple pages
szTargetTif	The generated .tif file
dwReserve	Reserved

Return Value:

-1	Call failed, the failure reason may be obtained via the function fBmp_GetErrMsg
1	Successful

Function Description:

Integrates two .tif files, i.e. adds the file szHeadTif to the file szSourecTif to create szTargetTif.

Note:

- If szSourceTif and szTargetTif are the same, the original file will be overwritten;
- If the second bit of dwReserve is set to 1, the higher 16 bits denote the page on which the page header will be added. If this page is not specified, all the pages of the file SzSourceTif will be added with szHeadTif.

Related Information:

Driver version	SynCTI Ver. 2.0 or above
Header	BmpApi.h
Library	BmpUtil.lib
DLL	BmpUtil.dll

Related Function: fBmp_SetHeaderFormat

2.29.6.6 fBmp_CutTifHeader

Cuts the file header of the .tif file.

Format:

int fBmp_CutTifHeader(char *Source, char*szTarget, int nHeight, int nPgeNo, DWORD dwReserve)

Parameter Description:

szSource	The .tif source file, whose file header needs to be cut
szTarget	Generated target file. If the file exists, it will overwrite the original file
nHeight	The height of the part to be cut
nPageNo	The designated page in the .tif file whose file header needs to be cut
dwReserve	Reserved

Return Value:

-1	Call failed, the failure reason can be obtained via the function fBmp_GetErrMsg
1	Successful

Function Description:

Cuts the file header of the .tif file, i.e. cuts a set part from the top of the .tif file.

Note:

- If szSourceTif and szTargetFile are the same, the original file szSourceTif will be overwritten;
- Normally nHeight is in the range of 16~40. For a single row of the file header, its height is approximately 20;
- nPageNo is the designated page. If it's set to -1, the file headers of all the pages will be cut.

Related Information:

Driver version	SynCTI Ver. 4.7.1.5 or above
Header	BmpApi.h
Library	BmpUtil.lib
DLL	BmpUtil.dll

Related Function: fBmp_AddTxtToTif_Big, fBmp_AddTxtToTif

2.29.6.7 fBmp_GetFileAllPage

Outputs the total page number of the fax file.

Format:

int fBmp_GetFileAllPage(char *filename)

Parameter Description:

filename	The string storing the fax file

Return Value:

-1	Failed, the failure reason may be failure in opening the file or incorrect file format
>=0	Successful, outputs the total page number of the file

Function Description:

Outputs the total page number of the fax file of filename.

Note:

Related Information:

Driver version	SynCTI Ver. 4.4.0.6 or above
Header	BmpApi.h
Library	BmpUtil.lib
DLL	BmpUtil.dll

Related Function:

2.29.6.8 fBmp_SetHeaderFormatA

Sets the properties of the page header of a .tif formatted fax file to determine the pagination position, the prefix, etc.

Format：

int WINAPI fBmp_SetHeaderFormatA(int nRow, int nFromX, int nFromY, LPCSTR szFrom, int nSubX, int nSubY, LPCSTR szSubject, int nToX, int nToY, LPCSTR szTo, int nTimeX, int nTimeY, LPCSTR szTime, int nPageX, int nPageY, LPCTSTR szPage)

Parameter Description：

nRow	Number of rows of the page header, range of value: 1~3, with the default value of 1
nFromX	Starting position of the field of From, with the default value of 5
nFromY	The row where the field 'From' locates, range of value: 1~3, with the default value of 1, indicating the first row
szFrom	Prompting field of the field 'From', by default it is 'From:', and the maximum length is 20 bytes
nSubX	Starting position of the field 'Subject', with the default value of 270
nSubY	The row where the field 'Subject' locates, range of value: 1~3, with the default value of 1, indicating the first row
szSubject	Prompting field of the field 'Subject', by default it's 'Sub', with the maximum length of 20 bytes
nToX	Starting position of the field 'To', with the default value of 440
nToY	The row where the field 'To' locates, range of value: 1~3, with the default value of 1, indicating the first row
szTo	Prompting field of the field 'To', by default it is 'To:', with the maximum length of 20 bytes
nTimeX	Starting position of the field 'Time', with the default value of 660
nTimeY	The row where the field 'Time' locates, range of value: 1~3, with the default value of 1, indicating the first row
szTime	Prompting field of the field 'Time', by default it is '', with the maximum length of 20 bytes
nPageX	Starting horizontal position of the page coding , with the default value of 780
nPageY	Starting vertical position of the page coding , with the default value of 1
szPage	Prefix hint field of the page coding, with the default value of NULL

Return Value：

-1	Call failed, the failure reason can be obtained via the function fBmp_GetErrMsg
1	Successful

Function Description:

Sets the properties of the page header generated by fBmp_AddTxtToTif to determine the pagination position, the prefix, etc.

Note:

- If the field 'Subject' needs to be put in the second row, set nRow =2, nSubY=2.
- The horizontal range is 0~864. Adjust the position of each data segment to keep it in the range.

Related Information:

Driver version	SynCTI Ver. 5.3.2.7 or above
Header	BmpApi.h
Library	BmpUtil.lib
DLL	BmpUtil.dll

Related Function: fBmp_AddTxtToTif, fBmp_AddTxtToTif_Big, fBmp_UniteTif

3 SynCTI Driver Configuration

3.1 System Configuration File ShConfig.ini

When Synway boards are being used, correct and necessary configurations must be implemented on the driver in order to make the application operate properly.

The configuration information of SynCTI driver is saved in the INI file adopting the text format. After the SynCTI driver is successfully installed, the installation program will generate a sample configuration file namely ShConfig.ini under the installation directory. The developers may not only straightly edit or modify the configuration file by using text editing software, but may also be able to change the configuration by using the system configuration tools provided by Synway.

In the configuration file, the configuration information is grouped with sections (Section); each section includes one or multiple items (Item). The name of the section appears in a pair of square brackets, each of the continuous text rows is a configuration item. The effective range of one section starts from the second row of the current section and ends with one row above the next section. Each configuration item has the format of 'x=y', the name of the configuration item is at the left side; the content of the configuration is at the right side.

3.1.1 Essential Configuration Items

3.1.1.1 Setting Total Number of Boards

3.1.1.1.1 TotalBoards

Configuration Item	**TotalBoards**
Section	[**SystemConfig**]
Format	TotalBoards=N
Value Range	N equals to the total number of Synway's boards installed in the System.
Feature	Sets the total number of Synway's boards installed in the System.

3.1.1.1.2 WhoSupplySysClock

Configuration Item	**WhoSupplySysClock**
Section	[**SystemConfig**]
Written Format	WhoSupplySysClock=n
Value Range	n=-1: Set all the Synway's boards to be slave boards; n=N: Set all the Synway's boards to be master boards; 0≤n≤N-1: Set the board with logical number n to be the master board, all the rest of the boards are slave boards. Note: N is the set value of the configuration item TotalBoards
Description	Sets the master board of the Synway's boards in the application system. For more information, refer to 'System Clock Configuration'.

3.1.1.2 Setting Channel Number

3.1.1.2.1 TotalAppCh

Configuration Item	**TotalAppCh**
Section	[**AppChToBoardChTable**]
Written Format	TotalAppCh=M
Value Range	M is the total channel number of all the boards in the System.
Description	Sets the total channel number in the System.

3.1.1.2.2 AppCh

Configuration Item	**AppCh**
Section	[**AppChToBoardChTable**]
Written Format	Format 1:AppCh[i_s]=k,j_s…j_e Format 2:AppCh[i]=k,j In the above formulas, i: The channel number used by the application, it normally starts with 0 i_s: The start channel number used by the application k:The BoardID of the board on which a mapping needs to be created j: The internal channel number on the board j_s:The internal start channel number on the board j_e:The internal end channel number on the board
Value Range	
Description	Creates the mapping between the channel number of the application and the channel number on the board. Format 1 is used for batched mapping creation; format 2 is used for one by one mapping creation.

3.1.1.3 Setting Board Information

3.1.1.3.1 BoardModel

Configuration Item	**BoardModel**
Section	[**BoardId=x**]
Written Format	BoardModel=s
Value Range	s is the string denoting the board model. For the available models, refer to Board Classification.
Description	Sets the board model.
Example	BoardModel=SHT-16B-CT/PCI

3.1.1.3.2 BoardSerialNumber

Configuration Item	**BoardSerialNumber**
Section	[**BoardId=x**]
Written Format	BoardSerialNumber=N
Value Range	The serial number of the board.
Description	Sets the board serial number, for the detailed value, refer to the related hardware manual.
Example	BoardSerialNumber=11

3.1.1.4 SHD/DTP Series

3.1.1.4.1 Setting Board Reset Feature

3.1.1.4.1.1 ResetBoardOnClose

Configuration Item	**ResetBoardOnClose**
Section	[**BoardId=x**]
Written Format	ResetBoardOnClose =n
Value Range	n=0:Not to reset; n=1:Reset (default).
Description	Sets whether to reset the board when the board is being closed.

3.1.1.4.2 Setting Parameters of Digital Trunk

3.1.1.4.2.1 PcmNumber

Refer to **CRC-4**

3.1.1.4.2.2 PcmSSx

Refer to **CRC-4**

3.1.1.4.2.3 PcmClockMode

Refer to **CRC-4**

3.1.1.4.2.4 PcmLinkType

Refer to **CRC-4**

3.1.1.4.2.5 IsdnAutoBuildLink

Refer to **CRC-4**

3.1.1.4.2.6 CRC-4

Configuration Item	**PcmNumber** **PcmSSx** **PcmClockMode** **PcmLinkType** **IsdnAutoBuildLink** **CRC-4**
Section	[**BoardId=x**]
Written Format	PcmNumber=M PcmSSx[i]=n PcmClockMode[i]=m PcmLinkType[i]=k IsdnAutoBuildLink[i]=b CRC-4[i]=c **//Note:** Requires the version of SynCTI Ver. 4.7.1.5 or above

Value Range	M: The total number of the digital trunks on the board, it's related with the board model, for more information, refer to 'SHD Series' in Chapter 1;
	i: The physical number of the digital trunk on the board, for more information, refer to related manuals. Range of value: 0≤i≤M;
	n: Choose the signaling protocol. Range of value:
	n=0: ISDN protocol (User side);
	n=1: SS1 signaling (SS1);
	n=2: ISDN protocol (Network side);
	n=3: DASS2 protocol;
	n=4: DPNSS protocol;
	n=7: SS7 signaling (SS7)
	m: Set the clock mode of the current digital trunk, below are the details:
	m=0: Master clock, line-synchronization (the board must be a master board)
	m=1: Master clock, free-run (the board must be a master board)
	m=2: Slave clock.
	k: Choose the type of the communication cable. Range of value:
	k=0:120 ohm twisted pair cable;
	k=1:75 ohm coaxial cable.
	b: Set whether to enable the feature of automatic link building for the ISDN signaling.
	b=0: Disabled (default);
	b=1: Enabled (If you enable this feature, you should set the configuration items of UserSendEstablish and NetSendEstablish in the [ISDN] section both to 0).
	Note: When TEI value is in the range of 64 to 126, the automatic link building feature can be enabled only by the user side but not the network side. So in such situation, this configuration item should be set with b=0 for the network side.
	c: The control switch for CRC-4 verification:
	c=0: Disabled;
	c=1: Enabled (default).

Description	PcmNumber sets the total number of the digital trunks on the board; PcmSSx sets the signaling type of the digital trunk; PcmClockMode sets the operating mode of the clock, for more information, refer to 'System Clock Configuration' in chapter 1; PcmLinkType sets the communication type of the cable; CRC-4 sets the CRC-4 verification feature of the digital trunk on the board.
Note	● If the set value of PcmNumber is larger than 0, for the configuration items PcmSSx, PcmClockMode and PcmLinkType, starting from i=0, the configuration needs to be repeated M times, i.e. each physical digital trunk needs to be configured; ● These configuration items are only applicable to SHD/DTP Series. ● Currently, DASS2 and DPNSS protocols are only supported by DTP Series.

Example	It's supposed that there are two pieces of SHD-60A-CT/PCI/SS7 boards installed in the system.
	Board 0 uses SS7 signaling, connects with 120 ohm twisted pair cable. Digital Trunk 0 on Board 0 provides the master clock to the application using line-synchronization mode; Board 0 uses ISDN protocol (user side), connects with 75 ohm twisted pair cable. Below are the configuration items:
	[**BoardId=x**]
	……
	PcmNumber=4
	
	PcmSSx[0]=7 //SS7 signaling
	PcmSSx[1]=7
	PcmSSx[2]=0 //ISDN user side
	PcmSSx[3]=0
	
	PcmClockMode[0]=0 //Master clock, line-synchronization
	PcmClockMode[1]=2 //Must be set as slave clock
	PcmClockMode[2]=2
	PcmClockMode[3]=2
	
	PcmLinkType[0]=0 //120 ohm twisted pair cable
	PcmLinkType[1]=0
	PcmLinkType[2]=1 //75 ohm coaxial cable
	PcmLinkType[3]=1

3.1.1.4.2.7 Loopback

Configuration Item	**loopback**
Section	[**BoardId=x**]
Written Format	loopback[pcm]=c
Value Range	pcm: The on-board physical number for corresponding digital trunk. Refer to relative manuals for more information. Range of value: 0≤i≤M. c: Controls the loopback feature of digital trunks: =0x1: Transmitted data loop back from the framer to the receive end (LOOP_FRAME); =0x2: Transmitted data loop back from LIU transmitter to LIU receiver (LOOP_LOCAL); =0x4: Analog signals loop back over LIU (LOOP_ANALOG, only for D-type and E-type boards); =0x10: Transmitted data loop back to LIU transmitter after being decoded in LIU receiver (LOOP_REMOTE); =0x20: Payload data loop back to the transmitter end (LOOP_PAYLOAD).
Description	Sets the loopback feature of trunks, used for diagnoses or debugging.
Note	● This configuration item requires SynCTI Ver. 5.1.0.0 or above. ● It is only applicable to SHD Series.
Example	[**BoardId=x**] **……** Loopback[0]=0x20 **……**

3.1.1.4.2.8 SpySyncAndCCS

Configuration Item	**SpySyncAndCCS**
Section	**[BoardId=x]**
Written Format	SpySyncAndCCS=n

Value Range	n=0: Disable the sync & signaling messages acquisition mode; n=1: Enable the sync & signaling messages acquisition mode. Don't enable this switch in normal operation.
Description	This switch shall be enabled only when the line synchronization or the signaling delivery is in an abnormal state. In such situation, set SpySyncAndCCS=1 to enable the sync& signaling messages acquisition mode and use the test program 'TEST.EXE' contained in the driver to record sync and signaling messages. That is, connect the time slots which deliver sync and signaling messages to corresponding channels which previously transported voice data, and select 'A-law' to start recording. To be exact, record the raw data on the time slot for synchronization via the PCM channel 0, record the raw data on the time slot for signaling reception via the PCM channel 1 and record the raw data on the time slot for signaling transmission via the PCM channel 2. Record all these data to .PCM files.
Note	● For ISDN lines, enable the feature of CRC check via the configuration items UserCrcMode and NetCrcMode before using this feature; ● For DTP series C-type boards, if you want to record the sync messages on T1 line, set the configuration item DefaultVoiceFormat to 6 and use A-Law to record speech files; if you want to record the signaling messages on T1/E1 lines, set the configuration item DefaultVoiceFormat to 7 and use μ-Law to record speech files; ● For SS1 signaling, as signaling data are transmitted via a special interface chip, the raw data cannot be recorded; ● Make sure you use this feature under the direction of Synway technical support engineers. Then provide the recorded files to our developers for further analysis and diagnosis; ● Requires the version of SynCTI 5.3.0.3 or above; ● This configuration item is only applicable to some of the SHD/DTP series boards. See below for detailed information.

Below is a list of board models that support this feature.

Board Model	Acquisition of raw data on the time slot for synchronization	Acquisition of raw data on the time slot for signaling reception	Acquisition of raw data on the time slot for signaling transmission
SHD-30A-CT/PCI/SS1	√	√	×
SHD-30A-CT/PCI/ISDN	√	√	×
SHD-30A-CT/PCI/SS7	√	√	×
SHD-30A-CT/PCI/FJ	√	√	-
SHD-60A-CT/PCI/SS1	√	√	×
SHD-60A-CT/PCI/ISDN	√	√	×
SHD-60A-CT/PCI/SS7	√	√	×
SHD-60A-CT/PCI/FJ	√	√	-
SHD-120A-CT/PCI/SS1	√	√	×
SHD-120A-CT/PCI/ISDN	√	√	×
SHD-120A-CT/PCI/SS7	√	√	×
SHD-30B-CT/PCI/FJ	√	√	-
SHD-60B-CT/PCI/FJ	√	√	-
SHD-30C-CT/PCI	√	√	×
SHD-30C-CT/PCI/FAX	√	√	×
SHD-60C-CT/PCI	√	√	×
SHD-60C-CT/PCI/FAX	√	√	×
SHD-30D-CT/PCI	√	√	√
SHD-60D-CT/PCI	√	√	√

SHD-120D-CT/PCI	√	√	√
SHD-120D-CT/PCI/EC	√	√	√
SHD-240D-CT/PCI	√	√	√
SHD-240D-CT/PCI/EC	√	√	√
SHD-30E-CT/PCI(SSW)	×	√	×
SHD-30E-CT/PCI/EC(SSW)	×	√	×
SHD-30E-CT/PCI/FAX(SSW)	×	√	×
SHD-60E-CT/PCI(SSW)	×	√	×
SHD-60E-CT/PCI/EC(SSW)	×	√	×
SHD-60E-CT/PCI/FAX(SSW)	×	√	×
SHD-120E-CT/PCI(SSW)	√	√	√
SHD-120E-CT/PCI/EC(SSW)	√	√	√
SHD-120E-CT/PCI/FAX(SSW)	√	√	√
SHD-240E-CT/PCI(SSW)	√	√	√
SHD-240E-CT/PCI/EC(SSW)	√	√	√
SHD-240E-CT/PCI/FAX(SSW)	√	√	√
SHD-30E-CT/PCIe	×	√	×
SHD-30E-CT/PCIe/EC	×	√	×
SHD-30E-CT/PCIe/FAX	×	√	×
SHD-60E-CT/PCIe	×	√	×
SHD-60E-CT/PCIe/EC	×	√	×
SHD-60E-CT/PCIe/FAX	×	√	×
SHD-120E-CT/PCIe	√	√	√
SHD-120E-CT/PCIe/EC	√	√	√
SHD-120E-CT/PCIe/FAX	√	√	√
SHD-240E-CT/PCIe	√	√	√
SHD-240E-CT/PCIe/EC	√	√	√
SHD-240E-CT/PCIe/FAX	√	√	√
SHD-240E-CT/PCIe/VAR	√	√	√
DTP-30C/PCI	√	√	-
DTP-30C/PCI+	√	√	-
DTP-60C/PCI	√	√	-
DTP-60C/PCI+	√	√	-
DTP-120C/PCI	√	√	-
DTP-120C/PCI+	√	√	-
DTP-30C/PCIe	√	√	-
DTP-30C/PCIe+	√	√	-
DTP-60C/PCIe	√	√	-
DTP-60C/PCIe+	√	√	-
DTP-120C/PCIe	√	√	-
DTP-120C/PCIe+	√	√	-

Legend: √: Supported ×: Unsupported -: Invalid

3.1.1.4.2.9 framing

Refer to syncc.

3.1.1.4.2.10 coding

Refer to syncc.

3.1.1.4.2.11 syncc

Configuration Item	**framing** **coding** **syncc**
Section	**[BoardId=x]**
Written Format	framing[pcm]=c coding[pcm]=d sync[pcm]=e
Value Range	pcm: The physical number of the digital trunk on the board. For more information, refer to related manuals. Range of value: 0≤i≤M. c: Set the frame format for T1 trunk: 2: ESF framing 3: D4 framing 8: SLC-96 framing d: Set the encoding format for T1 trunk: 2: B8ZS coding 5: AMI coding e: Set the syncc: In D4 Framing Mode: 0 = search for Ft pattern, then search for Fs pattern 1 = cross couple Ft and Fs pattern In ESF Framing Mode: 0 = search for FPS pattern only 1 = search for FPS and verify with CRC-6
Description	Sets the frame format and frame encoding mode for T1 trunk.
Note	● It requires SynCTI Ver. 5.3.1.3 or above; ● This configuration item is only applicable to T1 trunk.

3.1.1.4.3 Setting Logical PCM Number

3.1.1.4.3.1 TotalPcm

Refer to Pcm

3.1.1.4.3.2 Pcm

Configuration Item	**TotalPcm** **Pcm**
Section	**[PcmInfo]**

Written Format	TotalPcm=M Pcm[m]=k,j
Value Range	M: The total number of the digital trunks in the application. It must be less than or equal to the total physical PCM number in the system; m: The logical number of the digital trunk in the application. It must start from 0, the range of value: 0≤m<M (M is the set value of the configuration item TotalPcm) k: The logical number of the board. Range of value: ✧ 0≤k<N: N is the set value of the configuration item TotalBoards. The logical number of the digital trunk in the application is bound with the actual physical digital trunk. ✧ k=-1: The logical number of the digital trunk in the application is bound with the virtual circuit. The binding is only applicable to SS7 TUP and ISUP protocols, it requires the driver version to be 4.7.1.0 or above. For more information about the virtual circuit, refer to 'Virtual Circuit Programming Interface on SS7 Server' in chapter 1. j: If k≠-1, j denotes the physical number of the digital trunk, it must be the number of physically existing PCM; If k=-1, the corresponding digital trunk of j maybe virtual.
Description	TotalPcm sets the total number of the digital trunks in the application; Pcm establishes the mapping relation between the logical number and physical number of the digital trunk .
Note	This configuration item is only applicable to the SHD and DTP Series.

Example	It's supposed that there are two pieces of SHD-60A-CT/PCI/ISDN boards installed in the system, the below configuration items illustrate how to establish the mapping relation between the logical number and physical number of the digital trunk:
	[PcmInfo]
	TotalPcm=4
	Pcm[0]=0,0 //Set Digital Trunk 0 on Board 0 to be Digital Trunk 0 in the application
	Pcm[1]=0,1 //Set Digital Trunk 1 on Board 0 to be Digital Trunk 1 in the application
	Pcm[2]=1,0 //Set Digital Trunk 0 on Board 1 to be Digital Trunk 2 in the application
	Pcm[3]=1,1 //Set Digital Trunk 1 on Board 1 to be Digital Trunk 3 in the application

3.1.1.4.3.3 2PcmIn1Line

Configuration Item	**2PcmIn1Line**
Section	**[BoardId=x]**
Written Format	2PcmIn1Line=m
Value Range	m=0 (default): We recommend you use the outlet board RSD081 (standard 8-pin RJ48C) for the UMCT intelligent switch, as in such situation the interfaces are arranged in the same order as the PCM numbers. For the outlet board RSD082 (4-pin RJ48C/M), the first jack corresponds to PCM0 and PCM4, the second corresponds to PCM1 and PCM5, the third corresponds to PCM2 and PCM6, and the fourth corresponds to PCM3 and PCM7. m=1: We recommend you use the outlet board RSD082 (4-pin RJ48C/M) for the UMCT intelligent switch. In such situation, the first jack corresponds to PCM0 and PCM1, the second corresponds to PCM2 and PCM3, the third corresponds to PCM4 and PCM5, and the fourth corresponds to PCM6 and PCM7. For the outlet board RSD081 (standard 8-pin RJ48C), the 8 jacks in order correspond to PCM0, PCM2, PCM4, PCM6, PCM1, PCM3, PCM5 and PCM7.
Description	As on the UMCT intelligent switch the mainboard is separated from the outlet board and the switch itself can not distinguish RSD081 from RSD08, this configuration item helps to make the PCM arrangement more in accordance with the common use habit.
Note	This configuration item is applicable to the 240-port E-type digital boards used on the UMCT intelligent switch.

3.1.1.4.4 Setting SpyPCM Number (DTP Series)

3.1.1.4.4.1 TotalSpyPcm

Refer to SpyPcm

3.1.1.4.4.2 SpyPcm

Configuration Item	**TotalSpyPcm** **SpyPcm**
Section	**[SpyPcm]**
Written Format	TotalSpyPcm=M SpyPcm[m]=Pcm[j],Pcm[k]
Value Range	M: The total number of digital trunks to be monitored in the application; m: The number of SpyPcm, starts from 0, 0≤m≤M; j,k: The logical number of the digital trunk, it's set by the configuration item Pcm in the section of **[PcmInfo]**. For more information, refer to 'Digital Trunk' in chapter 1. It's recommended to use the connecting input ports of two digital trunks on the board which have continuous logical number to monitor one SpyPCM, hence, under most conditions, k=j+1. Note: j and K must be located on the same board.
Description	TotalSpyPcm sets the total number of the digital trunks in the application system to be monitored; SpyPcm determines which two on-board digital trunks should be connected with circuits to process the monitored digital trunk. That is, bind SpyPCM m to the circuits on Digital Trunk j and Digital Trunk k.
Note	This configuration item is only applicable to DTP Series

Example	It's supposed that only one board of SHD-60A-CT/PCI/FJ is installed in the system, the board is used to record the 60 voice channels (30x2) and two ISDN PRI signaling channels on the two E1 trunks (75 ohm coaxial cable). The following configuration is needed to implement the monitoring:
	
	[**BoardId=x**]
	……
	PcmNumber=4
	
	PcmSSx[0]=0 //ISDN user side
	PcmSSx[1]=0
	PcmSSx[2]=0
	PcmSSx[3]=0
	
	PcmClockMode[0]=0 //Master clock, line-synchronization
	PcmClockMode[1]=2 // It must be set to be slave clock
	PcmClockMode[2]=2
	PcmClockMode[3]=2
	
	PcmLinkType[0]=1 //75 ohm coaxial cable
	PcmLinkType[1]=1
	PcmLinkType[2]=1
	PcmLinkType[3]=1
	
	[PcmInfo]
	
	TotalPcm=4
	
	Pcm[0]=0,0 // Set Digital Trunk 0 on Board 0 to be Digital Trunk 0 in the application
	Pcm[1]=0,1 // Set Digital Trunk 1 on Board 0 to be Digital Trunk 1 in the application
	Pcm[2]=0,2 // Set Digital Trunk 2 on Board 0 to be Digital Trunk 2 in the application
	Pcm[3]=0,3 // Set Digital Trunk 3 on Board 0 to be Digital Trunk 3 in the application

	[SpyPcm]
	TotalSpyPcm=2
	SpyPcm[0]=Pcm[0],Pcm[1] //Using PCM 0 and PCM 1 to record the information on SpyPCM 0
	SpyPcm[1]= Pcm[2],Pcm[3] //Using PCM 2 and PCM 3 to record the information on SpyPCM 1
	[AppSpyCICTable]
	TotalAppSpyCIC=60
	AppSpyCIC[0]=SpyPcm[0],0..29 // Set the physical position for the SpyCIC which has a logical number in the range of 0..29
	AppSpyCIC[30]=SpyPcm[1],0..29 // Set the physical position for the SpyCIC which has a logical number in the range of 30~59

3.1.1.4.4.3 SpyT1TransE1Line

Configuration Item	**SpyT1TransE1Line**
Section	**[SpyPcm]**
Written Format	SpyT1TransE1Line[k]=b
Value Range	k: SpyPcm number; b=0: The monitored line is the common E1 line; b=1: The monitored line is the line transformed by a T1-to-E1 device.
Description	Sets whether the channels are numbered by continuous codes (E1:1-30 or T1:1-23) or discontinuous codes (E1:1-15, 17-31).
Note	● This configuration item is only applicable to monitoring of ISDN lines; ● This configuration item is only applicable to DTP Series.

3.1.1.4.4.4 SpyDefaultGetCallerIdType

Configuration Item	**SpyDefaultGetCallerIdType**
Section	**[SpyPcm]**
Written Format	SpyDefaultGetCallerIdType=b
Value Range	b=0: Obtain the second callerId; b=1: Obtain the callerId provided by the ISDN User Side; b=2: Obtain the callerId provided by the ISDN Net Side; b=3: Obtain the first callerId.
Description	Obtains the designated callerId contained in the signaling.
Note	● This configuration item is only applicable to the monitoring of ISDN lines; ● This configuration item is only applicable to DTP Series.

3.1.1.4.5 Setting SpyCIC Number (DTP Series)

3.1.1.4.5.1 TotalAppSpyCIC

Refer to AppSpyCIC

3.1.1.4.5.2 AppSpyCIC

Configuration Item	**TotalAppSpyCIC** **AppSpyCIC**
Section	**[AppSpyCICTable]**
Written Format	TotalAppSpyCIC=N Format 1: AppSpyCIC[n]=SpyPcm[m],k Format 2: AppSpyCIC[n]=SpyPcm[m],k_s..k_e

Value Range	N: The total number of SpyCIC; n: The logical number of SpyCIC, numbered from 0, 0≤n<N; m: The logical number of SpyPCM, for more information, refer to the configuration item TotalSpyPcm or SpyPcm; k: The circuit number on SpyPcm, 0≤k≤29. For SS7 signaling, the value of k is the CIC field in the message of TUP or ISUP; For ISDN PRI signaling, the actual relation between the value of k and PCM is: k_s: The starting circuit number on SpyPcm, 0≤k_s≤29; k_e: The ending circuit number on SpyPcm, 0≤k_e≤29
Description	TotalAppSpyCIC sets the total number of the monitored circuits; AppSpyCIC establishes the mapping between the logical number of the SpyCIC in the application and one actual circuit on the SpyPCM. For more information, refer to 'DTP Series' in Chapter 1
Note	● Format 1 is used for one-by-one setting, format 2 is used for batched setting which is equivalent to the continuous k_e-k_s+1 times of setting with format 1 starting from k_s; ● This configuration item is only applicable to DTP Series
Example	Refer to the example in the configuration item TotalSpyPcm

3.1.1.4.6 Essential Configuration Items for SS7 (DTP Series)

3.1.1.4.6.1 TotalSpyLinkSet

Refer to SpyIsupCICPcm

3.1.1.4.6.2 SpyLinkSet

Refer to SpyIsupCICPcm

3.1.1.4.6.3 TotalSpyLinkPcm

Refer to SpyIsupCICPcm

3.1.1.4.6.4 SpyLinkPcm

Refer to SpyIsupCICPcm

3.1.1.4.6.5 SpyCICPcm

Refer to SpyIsupCICPcm

3.1.1.4.6.6 SpySpCodeLen

Refer to SpyIsupCICPcm

3.1.1.4.6.7 SpyIsupCICPcm

Configuration Item	**TotalSpyLinkSet** **SpyLinkSet** **TotalSpyLinkPcm** **SpyLinkPcm** **SpyCICPcm** **SpySpCodeLen** **SpyIsupCICPcm**
Section	**[SS7Spy]**
Written Format	**Configuration 1: applicable to single-link group:** TotalSpyLinkPcm=M SpyLinkPcm[m]=SpyPcm[x] SpyCICPcm[k]=SpyPcm[y] SpyIsupCICPcm[k]=SpyPcm[y] SpySpCodeLen=j **Configuration 2: applicable to single-link group or multi-link group:** TotalSpyLinkSet=N Format 1: SpyLinkSet[n]=SpyPcm[x] Format 2: SpyLinkSet[n]=SpyPcm[x]+SpyPcm[y] SpyLinkSet[n].SpyCICPcm[k]=SpyPcm[m] SpyLinkSet[n].SpyIsupCICPcm[k]=SpyPcm[m] SpySpCodeLen=j **Note:** Format 1 is applicable to a link set which contains one signaling link only; Format 2 is applicable to a link set which has two signaling links. **Note:** Configuration 1 has a higher priority than Configuration 2.
Value Range	N: The total number of signaling link sets, 1≤N≤48, must start from n=0, continuously configure SpyLinkSet N times; n: The logical number of the link set, 0≤n<N; x, y, m: The logical number of SpyPcm, 0≤x<M, 0≤y<M, 0≤m<M; k: The digital trunk number assigned to SpyPcm by the Central Office, i.e. the value of the PCM parameter in the CIC field of TUP or ISUP messages, it can be obtained from the Central office; j : The value is either 24 or 14. It is used to set the encoding standard for signaling points: 24bit is the National Signaling Point Code (NSPC) format, 14bit is the International Signaling Point Code (ISPC) format. Note: M is the set value of the configuration item TotalSpyPcm. TotalSpyLinkPcm has the same value as TotalSpyPcm.
Description	✧ TotalSpyLinkSet sets the total number of the SS7 signaling link sets; ✧ SpyLinkSet sets the physical position of the signaling link in the designated Link Set n; ✧ TotalSpyLinkPcm sets the total number of digital trunks to be monitored in the application program; ✧ SpyLinkPcm sets the physical position of a signaling link in the specified link set; ✧ SpyCICPcm and SpyIsupCICPcm establish the mapping relation between the number of the digital trunk in the CIC field of TUP or ISUP messages and the logical number of SpyPCM in Link Set n, SpyCICPcm is only applicable to the TUP protocol, SpyIsupCICPcm is only applicable to the ISUP protocol.
Note	● SpyLinkSet[n].SpyCICPcm[k] is only applicable to the TUP protocol; ● SpyLinkSet[n].SpyIsupCICPcm[k] is only applicable to the ISUP protocol; ● This configuration item is only applicable to the SS7 signaling of DTP Series; ● This configuration item requires the SynCTI driver of version 4.5.8.0 or above.
Example	Refer to 'Application and Configuration Instance for DTP Series' in Chapter 1.

3.1.1.4.7 Essential Configuration Items for SS7 (SHD Series)

3.1.1.4.7.1 Setting Digital Trunk Using ISUP Protocol

3.1.1.4.7.1.1 TotalIsupPcm

Refer to IsupPcm

3.1.1.4.7.1.2 IsupPcm

Configuration Item	**TotalIsupPcm** **IsupPcm**
Section	**[SS7]**
Written Format	TotalIsupPcm=M IsupPcm[m]=LocalPcm[k]
Value Range	M: The total number of digital trunks using the ISUP protocol, 0≤M<N. N is the set value of the configuration item. If M is not 0, the configuration item IsupPcm must be set; m: The logical number of the digital trunk using ISUP protocol, numbered from 0. Range of value: 0≤m<M k: The logical number of the digital trunk in the application, range of value: 0≤k<N. The logical number of the digital trunk is set by the configuration item TotalPcm and Pcm.
Description	TotalIsupPcm sets the total number of the digital trunks using the ISUP protocol in the application; IsupPcm sets the digital trunk using ISUP protocol in the local PC
Note	● If the ISUP protocol is not used in the system, this configuration item doesn't need to be set; ● If TotalIsupPcm is set to be 0, the configuration item IsupPcm doesn' t need to be set, and the configuration item AutoHandleIsup will be ignored; ● This configuration item is only applicable to SHD Series

3.1.1.4.8 Essential Configuration Items for ISDN (SHD Series)

3.1.1.4.8.1 Setting Operating Mode

3.1.1.4.8.1.1 UseISDNMode

Configuration Item	**UseISDNMode**
Section	**[ISDN]**
Written Format	UseISDNMode=m

Value Range	m=0: User side (default);
	m=1: Network side;
	m=2: User side + Network side.
	Note: Under this mode, all the digital trunks must have the same mode on one single board, i.e. either all of them are user side mode, or all of them are network side mode, but different boards may support different modes

Description	Sets the operating mode when ISDN protocol is used. For more information, refer to 'Network-side Mode and User-side Mode' in Chapter 1

3.1.1.4.8.2 Setting Digital Trunk Using ISDN Signaling (User Side)

3.1.1.4.8.2.1 TotalUserLinker

Refer to UserTEIValue

3.1.1.4.8.2.2 UserPcmLink

Refer to UserTEIValue

3.1.1.4.8.2.3 UserSendEstablish

Refer to UserTEIValue

3.1.1.4.8.2.4 UserTEIValue

Configuration Item	**TotalUserLinker** **UserPcmLink** **UserSendEstablish** **UserTEIValue**
Section	**[ISDN]**
Written Format	TotalUserLinker=N UserPcmLink[n]=LocalPCM[k] UserSendEstablish[n]=b UserTEIValue[n]=m
Value Range	N: The total number of digital trunks at user side; n: The logical number of each digital trunk at user side, numbered from 0; k: The logical number of the digital trunk at local end. For more information, refer to the description of the configuration item TotalPcm and Pcm; b: Determines whether this end sends the message of automatic link building for the ISDN signaling or not. 0: Not to send; 1: Send (default). m: TEI value at user side. Range of value: 0≤m≤63: Fixed TEI; m>63: Dynamic TEI. The default value of m is 0. **Note: Provided the TEI value at the network side (NetTEIValue) is x, if 0≤m≤63, x must be equal to m, otherwise the link between the local and network sides can't be established; if m>63, x may not equal m and can be any value within the range of 0≤x≤126. For more information about the TEI value, refer to 'TEI value' in Chapter 1**
Description	TotalUserLinker sets the total number of local digital trunks which use the ISDN signaling and operate at the user side; UserPcmLink establishes the mapping relation between the number of each digital trunk at user side and its logical number; UserTEIValue sets the TEI value of the digital trunk at user side
Note	● If N>0, both UserPcmLink and UserTEIValue must be set N times; ● This configuration item is only applicable to the mode of user side.

3.1.1.4.8.3 Setting Digital Trunk Using ISDN Signaling (Network Side)

3.1.1.4.8.3.1 TotalNetLinker

Refer to NetTEIValue

3.1.1.4.8.3.2 NetPcmLink

Refer to NetTEIValue

3.1.1.4.8.3.3 NetSendEstablish

Refer to NetTEIValue

3.1.1.4.8.3.4 NetTEIValue

Configuration Item	**TotalNetLinker** **NetPcmLink** **NetSendEstablish** **NetTEIValue**
Section	**[ISDN]**
Written Format	TotalNetLinker=N NetPcmLink[n]=LocalPCM[k] NetSendEstablish[n]=b NetTEIValue[n]=x
Value Range	N: The total number of the digital trunks at network side in the system; n: The logical number of the digital trunk at network side, numbered from 0; k: The logical number of the digital trunk at local end. For more information, refer to the description of the configuration item TotalPcm and Pcm; b: Sets whether to send the message of automatic link building at the local end. 0: Not to send (default); 1: Send. x: TEI value at network side. Range of value: Provided the TEI value at the user side (UserTEIValue) is m, if 0≤m≤63, x must be equal to m，otherwise the link between the local and network sides can't be established；if m>63, x may not equal m and can be any value within the range of 0≤x≤126. The default value of x is 0.
Description	TotalNetLinker sets the total number of the digital trunks at local end working in network-side mode; NetPcmLink establishes the mapping relation between each digital trunk working in network-side mode and its logical number; NetTEIValue sets the TEI value of each digital trunk working in network-side mode
Note	● If N>0, both NetPcmLink and NetTEIValue must be set N times; ● This configuration item is only applicable to the mode of network side.

3.1.1.4.9 Essential Configuration Items for China SS1 (SHD Series)

3.1.1.4.9.1 ProtocolType

Configuration Item	**ProtocolType**
Section	**[SS1Config]**
Written Format	ProtocolType=m

Value Range	m=0: Use SS1 signaling (default);
	m=1: Use the LineSide protocol;
	m=2: Use the ASB protocol;
	m=3: Use the LineSide (OPS-FX) protocol;
	m=4: Use the VPS protocol;
	m=5: Use the ASBEL7 protocol.

Description	Sets the signaling mode for the channel currently using SS1 signaling by default. Here gives a brief explanation for the different values of ProtocolType. Customers can configure it according to the requirements of the remote PBX. When m = 1, 3 or 4, the CD value of the sent CAS is equal to the value of the configuration item TxCas_CD. The channel, if it receives a CAS with the AB value of 0 in an idle state, will turn into the ringing state. When you pick up the phone, if m=1, the AB value of the sent CAS is 2; if m=3, the AB value of the sent CAS is 3; if m=4, the AB value of the sent CAS is 0x0f. When m = 2, the AB value of the sent CAS is equal to the value of the configuration item TxCas_CD. The channel, if it receives a CAS with the ABCD value of 2 in an idle state, will turn into the ringing state. When you pick up the phone, the CD value of the sent CAS is 2. When m = 5, the ASBEL7 protocol is being used, and these two configuration items should be set Ss1SendIdleState=2 and TxCas_CD=0.

3.1.1.5 SHN Series

3.1.1.5.1 Common Configuration Items for SHN Series

3.1.1.5.1.1 LocalSipIp

Note: This configuration item is named LocalIp and can not be found in Section [SIP] for SHN B series VoIP boards in versions below SynCTI 5.1.0.0.

Refer to LocalSipPort

3.1.1.5.1.2 LocalSipPort

Note: This configuration item is named LocalPort and can not be found in Section [SIP] for SHN B series VoIP boards in versions below SynCTI 5.1.0.0

Configuration Item	**LocalSipIp** **LocalSipPort**
Section	**[SIP]**
Written Format	LocalSipIp=m LocalSipPort=n
Value Range	m: Sets the network address carried by SIP signaling. m is a character string in the form of Ipv4 address, containing less than 256 characters. The exceptional case is m=0.0.0.0: Select a random IP address of the local host as the network address carried by SIP signaling. n: Sets the network intercept port carried by SIP. It is 5060 by default and provides the value range of 1024 ~ 65536.
Description	Sets the network address and port carried by SIP signaling for VoIP boards

3.1.1.5.1.3 SipSetConnectionIp

Configuration Item	**SipSetConnectionIp**
Section	**[BoardId=x]**
Written Format	SipSetConnectionIp=z
Value Range	z=a.b.c.d, the contents in the Connectioninformation field in the SDP message body of the INVITE or 200 OK message for SIP. The default value is 0.0.0.0, which means not to set the contents in the Connectioninformation field in the SDP message body of the INVITE or 200 OK message for SIP.
Description	Sets the contents in the Connectioninformation field in the SDP message body of the INVITE or 200 OK message for SIP.
Note	● The function SsmSipSetConnectionInforOfSDP can implement the same feature.

3.1.1.5.2 Essential Configuration Items for SHN A Series

3.1.1.5.2.1 Setting Protocol Type

3.1.1.5.2.1.1 ProtocolType

Configuration Item	**ProtocolType**
Section	**[BoardId=x]**
Written Format	ProtocolType=m
Value Range	m: Sets the protocol used by this VoIP board. See below for details: m=1: Use the SIP protocol. m=2: Set the VoIP board to be a VoIP resource board.
Description	ProtocolType is used to set the protocol for the VoIP board or set the VoIP board to be a VoIP resource board.

3.1.1.5.2.2 Essential Configuration Items for SIP

3.1.1.5.2.2.1 RTPRange

Configuration Item	**RTPRange**
Section	**[SIP]**
Written Format	RTPRange= t_L, t_H
Value Range	t_L: Minimum RTP port range (1024≤t_L, with the default value of 6000); t_H: Maximum RTP port range (t_H≤65535, with the default value of 10000).
Description	Sets the RTP port range for the SHN A series boards.

3.1.1.5.2.2.2 AudioCodecList

Refer to **SendDtmfType**

3.1.1.5.2.2.3 SendDtmfType

Configuration Item	**AudioCodecList** **SendDtmfType**
Section	**[BoardId=x]**
Written Format	AudioCodecList= p_1,p_2,…. p_n SendDtmfType=j

Value Range	$p_1,p_2,..p_n$: The voice codecs list for calls. n≤10 and the value staying ahead has a higher priority. See below for what they mean exactly: p_x=6: G.711 A-Law; p_x=7: G.711 μ-Law; p_x=4: G.723_1; p_x=49: GSM; p_x=9: G722; p_x=96: AMR; p_x=98: ILBC; p_x=100: SILK(16); p_x=102: OPUS(8); p_x=131: G.729A. B-type board supports: A-Law, μ-Law, G.729A, G.723_1 and GSM; C-type board supports all. j: Sets the DTMF transmission mode. See below for details: j=0: Use RFC2833 mode to send DTMF; j=1: Use Signaling mode to send DTMF; j=2: Use In-band mode to send DTMF.
Description	Set the voice codecs and the DTMF transmission mode for the SHN series boards.
Note	The configuration item SendDtmfType is valid for all the VoIP boards. However, for the HMP series, the In-band mode is not supported.

3.1.1.5.2.2.4 RecvDtmfType

Configuration Item	**RecvDtmfType**
Section	**[BoardId=x]**
Written Format	RecvDtmfType=j

Value Range	j: Sets the DTMF reception mode. See below for details:
	j=0: Support three modes In-band, Signaling and RFC2833 to receive DTMF;
	j=1: Support two modes Signaling and RFC2833 to receive DTMF;
	j=2: Only support the mode In-band to receive DTMF;
	j=3: Only support the mode Signaling to receive DTMF.

Description	Set the DTMF reception mode for the SHN A series.

3.1.1.5.3 Essential Configuration Items for SHN B Series

3.1.1.5.3.1 Essential Configuration Items for SIP

3.1.1.5.3.1.1 BoardIP

Note: This configuration item is named LocalIp and works as the common IP used for both signaling and RTP in versions below SynCTI 5.1.0.0.

Refer to SendDtmfType

3.1.1.5.3.1.2 SubMask

Refer to SendDtmfType

3.1.1.5.3.1.3 Gateway

Refer to SendDtmfType

3.1.1.5.3.1.4 DNS

Refer to SendDtmfType

3.1.1.5.3.1.5 AudioCodecList

Refer to SendDtmfType

3.1.1.5.3.1.6 RTPRange

Refer to SendDtmfType

3.1.1.5.3.1.7 SendSipMsg

Refer to SendDtmfType

3.1.1.5.3.1.8 SendDtmfType

Configuration Item	**BoardIP** **SubMask** **Gateway** **DNS** **AudioCodecList** **RTPRange** **SendSipMsg** **SendDtmfType**
Section	**[BoardId=x]**
Written Format	BoardIP=m Submask=v Gateway=p DNS=L AudioCodecList= q_1,q_2,…. q_n RTPRange= t_L, t_H SendSipMsg=n SendDtmfType=j

Value Range	m: Sets the IP address of the network card equipped on the SHN B-type board. v: Sets the subnet mask for the SHN B-type Series. v is a character string in the form of Ipv4 address, containing less than 256 characters. By default, v=255.255.255.0. p: Sets the gateway for the SHN B-type Series. L: Sets the DNS address. L is a character string in the form of Ipv4 address, containing less than 256 characters. By default, L=192.168.1.100. $q_1,q_2,\ldots q_n$: The voice codecs list for calls, with the default value of 6,7,131,49. n≤10 and the value staying ahead has a higher priority. See below for what they mean exactly. q_x=6: G.711 A-Law; q_x=7: G.711 μ-Law; q_x=131: G.729A; q_x=49: GSM. t_L: Minimum RTP port range (1024≤t_L, with the default value of 6000); t_H: Maximum RTP port range (t_H≤65535, with the default value of 10000). Please note that the RTP ports under 6000 or above 10000 may have been occupied by the driver, therefore we suggest you use the default value of t_L and t_H. n: Sets the switch for playing voices upon receiving rings to the remote end on an SIP channel. This configuration item is valid only when n=183. Once the driver receives the invite call, it will automatically respond 183+SDP instead of sending the 180 message. This configuration item is invalid as long as n≠183. It is only applicable to B-type, C-type and HMP boards, not to A-type boards. j: Sets the DTMF transmission mode. See below for details: j=0: Use RFC2833 mode to send DTMF; j=1: Use Signaling mode to send DTMF; j=2: Use In-band mode to send DTMF.
Description	Set the network address carried by SIP signaling, the DTMF transmission mode and other SIP operational parameters for the SHN B series.
Note	The configuration item SendDtmfType is valid for all the VoIP boards. However, for the HMP series, the In-band mode is not supported.

3.1.1.5.3.1.9 RecvDtmfType

Configuration Item	**RecvDtmfType**
Section	**[BoardId=x]**
Written Format	RecvDtmfType=j

Value Range	j: Sets the DTMF reception mode. See below for details:
	j=0: Support three modes In-band, Signaling and RFC2833 to receive DTMF;
	j=1: Support two modes Signaling and RFC2833 to receive DTMF;
	j=2: Only support the mode In-band to receive DTMF;
	j=3: Only support the mode Signaling to receive DTMF.

Description	Set the DTMF reception mode for the SHN B series.

3.1.1.5.3.1.10 TelephoneEventsPt

Configuration Item	**TelephoneEventsPt**
Section	**[BoardId=x]**
Written Format	TelephoneEventsPt=m
Value Range	m: m is a number representing the DTMF RFC2833 payload format for local reception, with the value range of 0~127 and the default value of 101.
Description	Sets the DTMF RFC2833 payload format for local reception.

3.1.1.5.3.1.11 Send180After183

Configuration Item	**Send180After183**

Section	**[BoardId=x]**
Written Format	Send180After183=y
Value Range	y: Whether to send the 180 message after 183. 0: No (default); 1: Yes.
Description	Sets whether to send the 180 message after 183.
Note	It is required to set SendSipMsg=183.

3.1.1.5.3.1.12 Send183After180

Configuration Item	**Send183After180**
Section	**[BoardId=x]**
Written Format	Send183After180=y
Value Range	y: Whether to send the 183 message after 180. 0: No (default); 1: Yes.
Description	Sends the 183 message after 180 once receiving the prack message.
Note	If SendSipMsg=183 is set, this configuration item goes invalid.

3.1.1.5.4 Essential Configuration Items for SHN C Series

3.1.1.5.4.1 Special Configuration Items for SHN C Series

3.1.1.5.4.1.1 OctMac0

Configuration Item	**OctMac0**
Section	**[BoardId=x]**
Written Format	OctMac0 =00-0c-90-2d-05-02
Description	Sets the MAC address of Ethernet 1 on the SHN C series board.

3.1.1.5.4.1.2 DhcpServer

Configuration Item	**DhcpServer**
Section	**[DHCP]**
Written Format	DhcpServer=201.123.115.65
Description	Sets the IP address of the on-PC Ethernet port for control messages, which is used to load the firmware chip and sent/receive the control messages to/from the chip.

3.1.1.5.4.1.3 ProcessorCtrlMac

Configuration Item	**ProcessorCtrlMac**
Section	**[DHCP]**
Written Format	ProcessorCtrlMac=00-1F-D0-5E-AA-36
Description	Sets the MAC address of the on-PC Ethernet port for control messages, which is used to load the firmware chip and sent/receive the control messages to/from the chip.

3.1.1.5.4.1.4 FilterMacRange

Configuration Item	**FilterMacRange**
Section	**[DHCP]**
Written Format	FilterMacRange=00:0c:90:2d:04:e6- 00:0c:90:2d:04:45- 00:0c:90:2d:04:66
Description	Sets the MAC address whose IP can be allocated by the DHCP server. Only those network devices with such MAC address can be recognized by the DHCP server and given the IP address upon request.

3.1.1.5.4.1.5 BootFileName

Configuration Item	**BootFileName**
Section	**[DHCP]**
Written Format	BootFileName=oct2200.img (default)
Description	Sets the name of the firmware to be loaded. And the driver will load the firmware by tftp server. If there is a DHCP server in the local network, this configuration item can be deleted or set to null, and the driver will load the firmware by tftp client. In such case, the IP address allocated by the DHCP server must be in the same network segment of the firmware, otherwise the loading of firmware will fail.

3.1.1.5.4.1.6 LogLevel

Configuration Item	**LogLevel**
Section	**[DHCP]**
Written Format	LogLevel=All (default)
Value Range	Sets it to All to enable the log feature; or sets it to None to disable the log feature.
Description	Sets the switch for DHCP and TFTP logs.

3.1.1.5.4.1.7 DHCPRange

Configuration Item	**DHCPRange**
Section	**[DHCP]**
Written Format	DHCPRange=201.123.115.240-201.123.115.244
Description	Sets the range of DHCP IP address.

3.1.1.5.4.2 Essential Configuration Items for SIP

3.1.1.5.4.2.1 BoardIP

Note: This configuration item is named LocalIp and works as the common IP used for both signaling and RTP in versions below SynCTI 5.1.0.0.

Refer to SendDtmfType

3.1.1.5.4.2.2 SubMask

Refer to SendDtmfType

3.1.1.5.4.2.3 Gateway

Refer to SendDtmfType

3.1.1.5.4.2.4 DNS

Refer to SendDtmfType

3.1.1.5.4.2.5 AudioCodecList

Refer to SendDtmfType

3.1.1.5.4.2.6 RTPRange

Refer to SendDtmfType

3.1.1.5.4.2.7 SendSipMsg

Refer to SendDtmfType

3.1.1.5.4.2.8 SendDtmfType

Configuration Item	**BoardIP** **SubMask** **Gateway** **DNS** **AudioCodecList** **RTPRange** **SendSipMsg** **SendDtmfType**
Section	**[BoardId=x]**
Written Format	BoardIP=m Submask=v Gateway=p DNS=L AudioCodecList= q_1,q_2,…. q_n RTPRange= t_L, t_H SendSipMsg=n SendDtmfType=j

Value Range	m: Sets the IP address of the network card equipped on the SHN C-type board. v: Sets the subnet mask for the SHN C-type Series. v is a character string in the form of Ipv4 address, containing less than 256 characters. By default, v=255.255.255.0. p: Sets the gateway for the SHN C-type Series. L: Sets the DNS address. L is a character string in the form of Ipv4 address, containing less than 256 characters. By default, L=192.168.1.100. $q_1,q_2,\ldots q_n$: The voice codecs list for calls, with the default value of 6, 7, 131, 4, 9, 96, 98. n≤10 and the value staying ahead has a higher priority. See below for what they mean exactly. q_x=6: G.711 A-Law; q_x=7: G.711 μ-Law; q_x=131: G.729A; q_x=4: G723_1 (G723_1 can simultaneously support up to 280 channels' communication in the case that the voice is normal); q_x=9: G722; q_x=96: AMR (AMR can simultaneously support up to 310 channels' communication in the case that the voice is normal); q_x=98: ILBC. t_L: Minimum RTP port range (1024≤t_L, with the default value of 6000); t_H: Maximum RTP port range (t_H≤65535, with the default value of 10000). Please note that the RTP ports under 6000 or above 10000 may have been occupied by the driver, therefore we suggest you use the default value of t_L and t_H. n: Sets the switch for playing voices to the remote end upon receiving rings on an SIP channel. This configuration item is valid only when n=183. Once the driver receives the invite call, it will automatically respond 183+SDP instead of sending the 180 message. This configuration item is invalid as long as n≠183. It is only applicable to B-type, C-type and HMP boards, not to A-type boards. j: Sets the DTMF transmission mode. See below for details: j=0: Use RFC2833 mode to send DTMF; j=1: Use Signaling mode to send DTMF; j=2: Use In-band mode to send DTMF.
Description	Set the network address carried by SIP signaling, the DTMF transmission mode and other SIP operational parameters for the SHN C series.
Note	The configuration item SendDtmfType is valid for all the VoIP boards. However, for the HMP series, the In-band mode is not supported.

3.1.1.5.4.2.9 RecvDtmfType

Configuration Item	**RecvDtmfType**
Section	**[BoardId=x]**
Written Format	RecvDtmfType=j

Value Range	j: Sets the DTMF reception mode. See below for details:
	j=0: Support three modes In-band, Signaling and RFC2833 to receive DTMF;
	j=1: Support two modes Signaling and RFC2833 to receive DTMF;
	j=2: Support only one mode In-band to receive DTMF.
	j=3: Support only one mode Signaling to receive DTMF.

Description	Set the DTMF reception mode for the SHN C series.

3.1.1.5.4.2.10 TelephoneEventsPt

Configuration Item	**TelephoneEventsPt**
Section	**[BoardId=x]**
Written Format	TelephoneEventsPt=m

Value Range	m: m is a number representing the DTMF RFC2833 payload format for local reception, with the value range of 0~127 and the default value of 101.
Description	Sets the DTMF RFC2833 payload format for local reception.

3.1.1.5.4.2.11 Send180After183

Configuration Item	**Send180After183**
Section	**[BoardId=x]**
Written Format	Send180After183=y
Value Range	y: Whether to send the 180 message after 183. 0: No (default); 1: Yes.
Description	Sets whether to send the 180 message after 183.
Note	It is required to set SendSipMsg=183.

3.1.1.5.4.2.12 Send183After180

Configuration Item	**Send183After180**
Section	**[BoardId=x]**
Written Format	Send183After180=y
Value Range	y: Whether to send the 183 message after 180. 0: No (default); 1: Yes.
Description	Sends the 183 message after 180 once receiving the prack message.
Note	If SendSipMsg=183 is set, this configuration item goes invalid.

3.1.1.6 Essential Configuration Items for DST Series (REC Series only)

3.1.1.6.1 PBXType

Configuration Item	**PBXType**
Section	**[BoardId=x]**
Written Format	PBXType=s
Value Range	s: Denotes the string of the PBX model. For more information about the available models, refer to 'DST Series Supported PBX Models'
Description	Sets the model of the PBX connected with DST Series boards
Example	PBXType=ALCATEL4300

3.1.1.6.2 PhoneType

Configuration Item	**PhoneType**
Section	**[BoardId=x]**
Written Format	PhoneType=n_1,n_2,n_3,n_4,n_5,n_6,n_7,n_8,n_9,n_{10},n_{11},n_{12},n_{13},n_{14},n_{15},n_{16}
Value Range	n_1~n_{16}: The number denotes the model of the digital phone. For the available value, refer to 'DST Series Supported PBX Models' in Chapter 1
Description	Sets the value of the model of the digital phone connected with the DST board

3.1.1.6.3 DstRecRawData

Configuration Item	**DstRecRawData**
Section	**[BoardId=x]**
Written Format	DstRecRawData=n
Value Range	n=0 (default): Common working mode n=1: Raw data acquisition mode

Description	Allows to set the raw data acquisition mode to acquire the raw wave data on the line for analysis and troubleshooting.
Note	● This configuration item is only applicable to DST series B-type boards.

3.1.1.6.4 SetAnalogCtrlEnable

Configuration Item	**SetAnalogCtrlEnable**
Section	**[BoardId=x]**
Written Format	SetAnalogCtrlEnable=n
Value Range	n=0: The configuration item AnalogCtrl goes invalid; n=1: The configuration item AnalogCtrl becomes valid.
Description	Sets whether to enable the configuration item AnalogCtrl.
Note	● This configuration item is only applicable to DST series B-type boards. ● This configuration item is not necessary to be added in most situations. If you need, please set it under the instruction from our technicians.

3.1.1.6.5 AnalogCtrl

Configuration Item	**AnalogCtrl**
Section	**[BoardId=x]**
Written Format	AnalogCtrl =0xabcd,0xabcd,0xabcd

<table>
<tr><td>Value Range</td><td>Hexadecimal data. These three data respectively correspond to three modules. Refer to the following table for reference values of analog switches. You may adjust in real situation if necessary.
<table>
<tr><th>PBX Model</th><th>Reference Value of Analog Switch</th><th>PBX Model</th><th>Reference Value of Analog Switch</th></tr>
<tr><td>Aastra Nexspan</td><td>0x6</td><td>Alcatel4200/4400</td><td>0x2</td></tr>
<tr><td>AscotelIntelligate</td><td>0x4</td><td>Aspect</td><td>0x6</td></tr>
<tr><td>AvayaDefinity-2w</td><td>0x2</td><td>AvayaDefinity-4w</td><td>0x4</td></tr>
<tr><td>AvayaIndex</td><td>0x4</td><td>AvayaMerlingMagix</td><td>0x2</td></tr>
<tr><td>AvayaMLX</td><td>0x4</td><td>Belgacom</td><td>0x2</td></tr>
<tr><td>BoschIntegral2w</td><td>0x2</td><td>BoschIntegral4w</td><td>0x4</td></tr>
<tr><td>BoschIntegral552w</td><td>0x2</td><td>BRI ISDN</td><td>0x4</td></tr>
<tr><td>BRI ISDNI1</td><td>0x4</td><td>Comdial</td><td>0x4</td></tr>
<tr><td>Eon_EQ</td><td>0x4</td><td>Ericsson ELU25/28</td><td>0x4</td></tr>
<tr><td>Ericsson ELU5</td><td>0x4</td><td>Fujitsu F9600</td><td>0x0</td></tr>
<tr><td>Harris2B</td><td>0x4</td><td>Harris_1B_optic</td><td>0x6</td></tr>
<tr><td>Harris_1B_workstation</td><td>0x6</td><td>IAP</td><td>0x4</td></tr>
<tr><td>Intecom2-wire</td><td>0x2</td><td>Intercom4-wire</td><td>0x2</td></tr>
<tr><td>Intel-Tel</td><td>0x4</td><td>iwatsu256k</td><td>0x2</td></tr>
<tr><td>Iwatsu512k</td><td>0x2</td><td>LG_StareXCS</td><td>0x4</td></tr>
<tr><td>MirtelSx2000</td><td>0x4</td><td>NEC</td><td>0x4</td></tr>
<tr><td>nitsuko</td><td>0x2</td><td>Nortel Matra</td><td>0x6</td></tr>
<tr><td>Nortel Meridian1</td><td>0x4</td><td>Nortel Norstar</td><td>0x4</td></tr>
<tr><td>panasonic_DBS</td><td>0x0</td><td>Panasonic KX</td><td>0x2</td></tr>
<tr><td>Panasonic 7600se</td><td>0x0</td><td>PhilipsSOPHOis30002w</td><td>0x2</td></tr>
<tr><td>PhilipsSOPHOis30004w</td><td>0x4</td><td>RockwellSpectrum</td><td>0x4</td></tr>
<tr><td>SamsungDCS-828</td><td>0x4</td><td>SamsungINFOREX</td><td>0x4</td></tr>
<tr><td>Siemens Hicom</td><td>0x2</td><td>Siemens ISDT2W</td><td>0x0</td></tr>
<tr><td>Siemens Realifis DT</td><td>0x4</td><td>Siemens RolmLink</td><td>0x2</td></tr>
<tr><td>Tadran Coral</td><td>0x2</td><td>Teltronics(Harris2020)</td><td>0x6</td></tr>
<tr><td>ToshibaStrataDK/CTX</td><td>0x4</td><td></td><td></td></tr>
</table></td></tr>
<tr><td>Description</td><td>Sets the value for analog switches.</td></tr>
<tr><td>Note</td><td>● This configuration item is only applicable to DST series B-type boards.
● This configuration item is only valid when SetAnalogCtrlEnable=1.
● This configuration item is not necessary to be added in most situations. Please contact our technical support should you have any query about this configuration item.</td></tr>
</table>

3.1.1.6.6 SetVoxChSelectEnable

Configuration Item	**SetVoxChSelectEnable**
Section	**[BoardId=x]**
Written Format	SetVoxChSelectEnable =n

Value Range	n=0: The configuration item VoxChSelect goes invalid; n=1: The configuration item VoxChSelect becomes valid.
Description	Sets whether to enable the configuration item VoxChSelect.
Note	● This configuration item is only applicable to DST series B-type boards. ● This configuration item is not necessary to be added in most situations. If you need, please set it under the instruction from our technicians.

3.1.1.6.7 VoxChSelect

Configuration Item	**VoxChSelect**
Section	**[BoardId=x]**
Written Format	VoxChSelect =0xabcd,0xabcd,0xabcd
Value Range	Hexadecimal data, bits 0-15 respectively correspond to Channel 0-7. Each channel occupies 2 bits. To be exact, 00 – B0; 01 – B1; 10 – B2.
Description	When a DST series B-type board is used for recording, it will give the two voice channels on each channel two different bit values (B0, B1 or B2). By designating the bit values, this configuration item sets the voice channels to be recorded. By default, the B0 voice channels will be recorded.
Example	VoxChSelect=0x4924,0x0,0x5555 indicates that the voices channels to be recorded on Channel 0-7 are respectively B0, B1, B2, B0, B1, B2, B0, B1, those on Channel 8-15 are all B0, and those on Channel16-23 are all B1.
Note	● This configuration item is only applicable to DST series B-type boards. ● This configuration item is only valid when SetVoxChSelectEnable =1. ● This configuration item is not necessary to be added in most situations. If you need, please set it under the instruction from our technicians.

3.1.1.6.8 SetFilterSwitchEnable

Configuration Item	**SetFilterSwitchEnable**
Section	**[BoardId=x]**
Written Format	SetFilterSwitchEnable =n
Value Range	n=0: The configuration item FiiterSwitch goes invalid; n=1: The configuration item FiiterSwitch becomes valid.
Description	Sets whether to enable the configuration item FiiterSwitch.
Note	● This configuration item is only applicable to DST series B-type boards. ● This configuration item is not necessary to be added in most situations. If you need, please set it under the instruction from our technicians.

3.1.1.6.9 FilterSwitch

Configuration Item	**FilterSwitch**
Section	**[BoardId=x]**
Written Format	FilterSwitch =0xabcd,0xabcd,0xabcd
Value Range	Hexadecimal data, varying on actual condition. These three data respectively correspond to three modules.
Description	Sets the filter switch.

Note	● This configuration item is only applicable to DST series B-type boards. ● This configuration item is only valid when SetFilterSwitchEnable =1. ● This configuration item is not necessary to be added in most situations. If you need, please set it under the instruction from our technicians.

3.1.1.6.10 SetVoltageReferenceEnable

Configuration Item	**SetVoltageReferenceEnable**
Section	**[BoardId=x]**
Written Format	SetVoltageReferenceEnable =n
Value Range	n=0: The configuration item VoltageReference goes invalid; n=1: The configuration item VoltageReference becomes valid.
Description	Sets whether to enable the configuration item VoltageReference.
Note	● This configuration item is only applicable to DST series B-type boards. ● This configuration item is not necessary to be added in most situations. If you need, please set it under the instruction from our technicians.

3.1.1.6.11 VoltageReference

Configuration Item	**VoltageReference**
Section	**[BoardId=x]**
Written Format	VoltageReference =0xabcd,0xabcd,0xabcd
Value Range	Hexadecimal data, varying on actual condition. These three data respectively correspond to three modules.
Description	Sets initial maximum level and reference voltage amplitude.
Note	● This configuration item is only applicable to DST series B-type boards. ● This configuration item is only valid when SetVoltageReferenceEnable =1. ● This configuration item is not necessary to be added in most situations. If you need, please set it under the instruction from our technicians.

3.1.1.7 Essential Configuration Items for IPR Series (REC Series only)

3.1.1.7.1 Essential Configuration Items for SynIPRecorder in Master

3.1.1.7.1.1 RecMasterIP

Configuration Item	**RecMasterIP**
Section	**[BoardId=x]**
Written Format	RecMasterIP =m
Value Range	m: m is a character string in the form of Ipv4 address. The default value is 127.0.0.1
Description	Host IP address of SynIPRecorder Master.

3.1.1.7.1.2 RecMasterPort

Configuration Item	**RecMasterPort**
Section	**[BoardId=x]**
Written Format	RecMasterPort =m
Value Range	m: m is a number with the default value of 9888.
Description	Host monitoring port of SynIPRecorder Master.

3.1.1.7.1.3 RTPTimeOut

Configuration Item	**RTPTimeOut**
Section	**[BoardId=x]**
Written Format	RTPTimeOut =m
Value Range	m: m is a number. When a recording channel does not detect the RTP data in a long period of time, it is assumed that the corresponding Session to this channel is disconnected and the recording will be stopped automatically. When m=0, RTP timeout detection will not be performed. The default value is 15 (calculated by seconds).
Description	Sets the time period for RTP timeout detection.

3.1.1.7.2 Essential Configuration Items for SynIPAnalyzer

3.1.1.7.2.1 MonitorNIC

Configuration Item	**MonitorNIC**
Section	**[BoardId=x]**
Written Format	MonitorNIC =m
Value Range	m: m is a character string, representing the identification of the monitored network card
Description	The monitored network card in SynIPAnalyzer.

3.1.1.7.2.2 ForwardIP

Configuration Item	**ForwardIP**
Section	**[BoardId=x]**
Written Format	ForwardIP =m
Value Range	m: m is a character string in the form of IPv4 address. The default value is 127.0.0.1
Description	The IP address that SynIPAnalyzer uses to forward the RTP data.

3.1.1.7.2.3 ForwardPort

Configuration Item	**ForwardPort**
Section	**[BoardId=x]**
Written Format	ForwardPort =m
Value Range	m: m is a number with the default value of 20000.
Description	The port that SynIPAnalyzer uses to forward the RTP data.

3.1.1.7.2.4 MonitorType

Configuration Item	**MonitorType**
Section	**[BoardId=x]**
Written Format	MonitorType =m
Value Range	m=0: Monitor All (default); m=1: Monitor By Station
Description	The monitoring type of SynIPAnalyzer.

Note	● If the monitoring type is set to Monitor By Station (m=1), you need to invoke the function SsmIPRAddStationToMap and SsmIPRRmvStationFromMap(Ex) explicitly to control the Station. If the Station to be monitored is not added by the function SsmIPRAddStationToMap, the driver can not monitor any phone. ● To set 'Monitor By Station' as the original monitoring type, we suggest you do it via this configuration item instead of the function SsmIPRSetMonitorType; or the driver may detect the session and Station before you invoke this function and may probably waste extra time and resources to deal with the monitored session.

3.1.1.7.2.5 RTPTimeOut

Configuration Item	**RTPTimeOut**
Section	**[BoardId=x]**
Written Format	RTPTimeOut =m
Value Range	m: m is a number. When a Session does not detect the RTP data in a long period of time, it is assumed this session is disconnected automatically. When m=0, RTP timeout detection will not be performed. The default value is 15 (calculated by seconds).
Description	Sets the time period for RTP timeout detection.

3.1.1.7.2.6 RtpFwdCtrl

Configuration Item	**RtpFwdCtrl**
Section	**[BoardId=x]**
Written Format	RtpFwdCtrl =m
Value Range	m=0: Forwards RTP if both or either of the addresses in E_RCV_IPR_MEDIA_SESSION_STARTED is identical to that in RTP (default); m=1: Forwards RTP only when both addresses in E_RCV_IPR_MEDIA_SESSION_STARTED are identical to those in RTP.
Description	Sets the rule of forwarding RTP for SynIPAnalyzer.

3.1.1.7.2.7 ThreadPairs

Configuration Item	**ThreadPairs**
Section	**[BoardId=x]**
Written Format	ThreadPairs =m
Value Range	m: m is a nonzero integer, which is used to set the main work thread amount. The default value is 2. For a large-capacity IP recording environment, it is recommended to set this configuration item with the value equal to the CPU amount. The largest value is 64.
Description	Sets the main work thread amount for SynIPAnalyzer.

3.1.1.7.2.8 IGMPEventEnable

Configuration Item	**IGMPEventEnable**
Section	**[BoardId=x]**
Written Format	IGMPEventEnable =m
Value Range	m=0: Don't detect the E_RCV_IPR_IGMP_MESSAGE event (default); m=1: Detect the E_RCV_IPR_IGMP_MESSAGE event.
Description	Sets whether to detect the E_RCV_IPR_IGMP_MESSAGE event of SynIPAnalyzer. Usually only in the ROIP recording is this switch required to enable.

3.1.1.7.2.9 OpusCodec

Configuration Item	**OpusCodec**
Section	**[BoardId=x]**
Written Format	OpusCodec=m
Value Range	m is an integer.
Description	OPUS does not have a fixed encoding value. This configuration is used to set the encoding value of the OPUS format RTP monitored by SynIPAnalyzer.
Note	OPUS format RTP recording under LINUX environment is not supported at present.

3.1.1.1 Essential Configuration Items for HMP Series (CTI Series only)

3.1.1.1.1 Essential Configuration Items for HMP Server

3.1.1.1.1.1 TotalMediaForward

Configuration Item	**TotalMediaForward**
Section	**[HMPRouter]**
Written Format	TotalMediaForward =n
Value Range	n: n is a number. The value range is 0~8, with the default value of 0.
Description	Sets the amount of HMP Client.

3.1.1.1.1.2 LocalIP[n]

Configuration Item	**LocalIP**
Section	**[HMPRouter]**
Written Format	LocalIP[n] =m
Value Range	n: HMP Client number. The value range is 0~ TotalMediaForward-1. m: m is a character string in the form of IPv4 address. By default, m=127.0.0.1.
Description	Sets the IP address of the host where HMP Server locates. This is used for RTP receiving/transmitting between HMP Server and HMP Client and for the control of HMP Client.

3.1.1.1.1.3 TotalCh[n]

Configuration Item	**TotalCh**
Section	**[HMPRouter]**
Written Format	TotalCh[n]=m
Value Range	n: HMP Client number. The value range is 0~ TotalMediaForward-1. m: m is a number, with the default value of 0.
Description	Sets the amount of HMP Client channels supporting CODEC.

3.1.1.1.1.4 LocalPort[n]

Configuration Item	**LocalPort**
Section	**[HMPRouter]**
Written Format	LocalPort[n] =m
Value Range	n: HMP Client number. The value range is 0~ TotalMediaForward-1. m: Port number, with the default value of 5051.
Description	Sets the port of the host where HMP Server locates, used to control HMP Client.

3.1.1.1.1.5 RemoteIP[n]

Configuration Item	**RemoteIP**

Section	**[HMPRouter]**
Written Format	RemoteIP[n] =m
Value Range	n: HMP Client number. The value range is 0~ TotalMediaForward-1. m: m is a character string in the form of IPv4 address. By default, m=127.0.0.1.
Description	Sets the IP address of the host where HMP Client locates.

3.1.1.1.1.6 RemotePort[n]

Configuration Item	**RemotePrt**
Section	**[HMPRouter]**
Written Format	RemotePort[n] =m
Value Range	n: HMP Client number. The value range is 0~ TotalMediaForward-1. m: Port number, with the default value of 5050.
Description	Sets the port of the host where HMP Client locates.

3.1.1.1.1.7 RtpIP[n]

Configuration Item	**RtpIP**
Section	**[HMPRouter]**
Written Format	RtpIP[n] =m
Value Range	n: HMP Client number. The value range is 0~ TotalMediaForward-1. m: m is a character string in the form of IPv4 address. By default, m=192.168.0.1.
Description	Sets the host IP address which is used by HMP Client to receive and transmit RTP, to communicate with the external part. This is equivalent to the RTP address of the B-type IP board. Depending on the amount and configuration of network ports of the host where HMP Client locates, the value of it can be the same as or different from that of RemoteIP. However, it cannot be set to 127.0.0.1; otherwise, the RTP from the remote end will fail to reach the local end.

3.1.1.1.1.8 MaxMediaThread

Configuration Item	**MaxMediaThread**
Section	**[HMPRouter]**
Written Format	MaxMediaThread =m
Value Range	m: Sets the thread amount of HMP Server for RTP CODEC and voice processing. Range of value: 0<m≤64, with the default value of 4.
Description	More threads are needed if HMP Server processes more RTP and voice channels. You can assign the treads appropriately according to the amount of the CPU cores and channels.

3.1.1.1.2 Essential Configuration Items for SIP

3.1.1.1.2.1 BoardIP

Note: This configuration item is named LocalIp and works as the common IP used for both signaling and RTP in versions below SynCTI 5.1.0.0.

Refer to SendDtmfType

3.1.1.1.2.2 SubMask

Refer to SendDtmfType

3.1.1.1.2.3 Gateway

Refer to SendDtmfType

3.1.1.1.2.4 DNS

Refer to SendDtmfType

3.1.1.1.2.5 AudioCodecList

Refer to SendDtmfType

3.1.1.1.2.6 RTPRange

Refer to SendDtmfType

3.1.1.1.2.7 SendSipMsg

Refer to SendDtmfType

3.1.1.1.2.8 SendDtmfType

Configuration Item	**BoardIP** **SubMask** **Gateway** **DNS** **AudioCodecList** **RTPRange** **SendSipMsg** **SendDtmfType**
Section	**[BoardId=x]**
Written Format	BoardIP=m Submask=v Gateway=p DNS=L AudioCodecList= $q_1,q_2,\ldots q_n$ RTPRange= t_L, t_H SendSipMsg=n SendDtmfType=j

Value Range	m: Sets the IP address of the network card equipped on the HMP board. v: Sets the subnet mask for the HMP Series. v is a character string in the form of Ipv4 address, containing less than 256 characters. By default, v=255.255.255.0. p: Sets the gateway for the HMP Series. L: Sets the DNS address. L is a character string in the form of Ipv4 address, containing less than 256 characters. By default, L=192.168.1.100. $q_1,q_2,\ldots q_n$: The voice codecs list for calls, with the default value of 6, 7, 131, 49, 96, 98. n≤10 and the value staying ahead has a higher priority. See below for what they mean exactly. q_x=6: G.711 A-Law; q_x=7: G.711 μ-Law; q_x=131: G.729A; q_x=49: GSM; q_x=96: AMR; q_x=98: ILBC. t_L: Minimum RTP port range (1024≤t_L, with the default value of 6000); t_H: Maximum RTP port range (t_H≤65535, with the default value of 10000). Please note that the RTP ports under 6000 or above 10000 may have been occupied by the driver, therefore we suggest you use the default value of t_L and t_H. n: Sets the switch for playing voices upon receiving rings to the remote end on an SIP channel. This configuration item is valid only when n=183. Once the driver receives the invite call, it will automatically respond 183+SDP instead of sending the 180 message. This configuration item is invalid as long as n≠183. It is only applicable to B-type, C-type and HMP boards, not to A-type boards. j: Sets the DTMF transmission mode. See below for details: j=0: Use RFC2833 mode to send DTMF; j=1: Use Signaling mode to send DTMF; j=2: Use In-band mode to send DTMF.
Description	Set the network address carried by SIP signaling, the DTMF transmission mode and other SIP operational parameters for the HMP series.
Note	The configuration item SendDtmfType is valid for all the VoIP boards. However, for the HMP series, the In-band mode is not supported.

3.1.1.1.2.9 RecvDtmfType

Configuration Item	**RecvDtmfType**
Section	**[BoardId=x]**
Written Format	RecvDtmfType=j

Value Range	j: Sets the DTMF reception mode. See below for details:
	j=0: Support three modes In-band, Signaling and RFC2833 to receive DTMF;
	j=1: Support two modes Signaling and RFC2833 to receive DTMF;
	j=2: Only support the mode In-band to receive DTMF;
	j=3: Only support the mode Signaling to receive DTMF.

Description	Set the DTMF reception mode for the HMP series.

3.1.1.1.2.10 TelephoneEventsPt

Configuration Item	**TelephoneEventsPt**
Section	**[BoardId=x]**
Written Format	TelephoneEventsPt=m
Value Range	m: m is a number representing the DTMF RFC2833 payload format for local reception, with the value range of 0~127 and the default value of 101.
Description	Sets the DTMF RFC2833 payload format for local reception.

3.1.1.1.2.11 Send180After183

Configuration Item	**Send180After183**
Section	**[BoardId=x]**
Written Format	Send180After183=y
Value Range	y: Whether to send the 180 message after 183. 0: No (default); 1: Yes.
Description	Sets whether to send the 180 message after 183.
Note	It is required to set SendSipMsg=183.

3.1.1.1.2.12 Send183After180

Configuration Item	**Send183After180**
Section	**[BoardId=x]**
Written Format	Send183After180=y
Value Range	y: Whether to send the 183 message after 180. 0: No (default); 1: Yes.
Description	Sends the 183 message after 180 once receiving the prack message.
Note	If SendSipMsg=183 is set, this configuration item goes invalid.

3.1.2 Common Configuration Items

3.1.2.1 Setting Protocol for SHD/DTP Series Board

3.1.2.1.1 CardType

Configuration Item	**CardType**
Section	**[BoardId=x]**
Written Format	CardType=n
Value Range	n=0: E1 interface n=1: T1 interface n=2: J1 interface The default value is 0.
Description	This configuration item and the PcmSSx configuration item can enable the system to run based on T1/J1 ISDN protocol. However, you need to modify the following configuration items according to the voice CODECs used on the line. 1. DefaultVoiceFormat under Section [BoardId=x] which is used to configure the voice CODEC for DSP reception/transmission; 2. UserVoiceFormat or NetVoiceFormat under Section [ISDN] if ISDN lines are used.

3.1.2.2 Setting Events Thrown out by Driver

3.1.2.2.1 DefaultEventOutput

Configuration Item	**DefaultEventOutput**
Section	**[SystemConfig]**
Written Format	DefaultEventOutput=m
Value Range	m=0: Not output any events m=1: Only output commonly-used events. For more information about the commonly-used events, refer to the related information in Chapter 1; m=2: Output all of the events
Description	Sets the event filter of the driver. For more information, refer to 'Event Filter' in Chapter 1

3.1.2.2.2 OvrlEnrgEventOut

Configuration Item	**OvrlEnrgEventOut**
Section	**[SystemConfig]**
Written Format	OvrlEnrgEventOut=n
Value Range	n=0: not output E_CHG_OvrlEnrgLevel n=1: output E_CHG_OvrlEnrgLevel, any other value except 0 will be regarded as 1 The default value is 0.
Description	Sets the event filter for E_CHG_OvrlEnrgLevel in the driver. To output E_CHG_OvrlEnrgLevel, the precondition is the value of DefaultEventOutput is not 0.

3.1.2.3 Setting Depth of Internal Event Queue

3.1.2.3.1 MaxEventPerChannel

Configuration Item	**MaxEventPerChannel**
Section	**[SystemConfig]**
Written Format	MaxEventPerChannel=n
Value Range	n>0, calcaulated by the total number of events, with the default value of 100
Description	Sets the depth of the event-queue of one channel. When the application invokes the function of SsmSetEvent using the parameter wEvent=0xffff to enable the event polling or event callback mode, the driver will allocate memory for the event queue using the set value of this configuration item

3.1.2.3.2 MaxUserEventSize

Configuration Item	**MaxUserEventSize**
Section	**[SystemConfig]**
Written Format	MaxUserEventSize=n
Value Range	n≥0, with the default value of 0 which means the application is forbidden to add self-defined event to the driver
Description	During the time interval (8ms) of one hardware interruption, the maximum number of the self-defined events that the application may add to the interval event output queue in the driver.

3.1.2.4 Setting General Parameters for State Machine

3.1.2.4.1 MaxWaitAutoDialAnswerTime

Configuration Item	**MaxWaitAutoDialAnswerTime**
Section	**[SystemConfig] / [SS1Config] / [ISUP] / [TUP] / [ISDN] / [SIP]**
Written Format	MaxWaitAutoDialAnswerTime=t
Value Range	t: Maximum wait time (seconds), the default value varies according to the channel type: ✧ Analog trunk channel: 25 seconds; ✧ SS1 channel: 90 seconds; ✧ ISUP channel: 180 seconds; ✧ TUP channel: 60 seconds; ✧ ISDN channel: 60 seconds; ✧ SIP channel: 60 seconds.

Description	The maximum wait time waiting for the called party pickup after the channel state turns to 'WaitAnswer', during an outgoing call. For the execution process of AutoDial, refer to the following parts of Chapter 1: ✧ Analog trunk channel: 'Analog Trunk Channel State Machine'; ✧ SS1 channel: Timer T6 in China SS1 State Machine '; ✧ ISUP channel: Timer T4 in 'ISUP Channel State Machine'; ✧ TUP channel: Timer T4 in 'TUP Channel State Machine'; ✧ ISDN channel: Timer WaitAnswerTimer in 'ISDN Channel State Machine'; ✧ SIP channel: Timer evTimeout(*1) in 'SIP Channel State Machine'.
Note	● For the configuration of analog trunk channels, use the configuration section [SystemConfig]; for the configuration of SS1 channels, use the configuration section [SS1Config]; for the configuration of ISUP channels, use the configuration section [ISUP]; for the configuration of TUP channels, use the configuration section [TUP]; for the configuration of ISDN channels, use the configuration section [ISDN]; for the configuration of SIP channels, use the configuration section [SIP].

3.1.2.4.2 RingToPending

Configuration Item	**RingToPending**
Section	**[TUP] / [ISUP]**
Written Format	RingToPending=m
Value Range	m=0: Response to the 'disconnected' message from the other end, start to tear down the connection; m=1: Not to response to the tear-down message from the other end, the channel is transitioned to the 'pending' state. The reason of pending is also set to be PEND_RemoteHangupOnRinging, so that when to tear down the connection is determined by the application itself; The default value is 0
Description	During incoming call, when the local end is in the 'ringing' state, if the remote PBX quits call, it sends tear-down message to the local end. When the driver receives the tear-down message, this configuration item determines which next state of the channel will be entered. For more information about this configuration item, refer to the channel state transition in Chapter 1
Note	● This configuration item is only applicable to TUP and ISUP channels ● For the configuration of TUP channels, use the configuration section [TUP], for the configuration of ISUP channel, use the configuration section [ISUP]

3.1.2.4.3 AutoClearCallerIdBufOnHangup

Configuration Item	**AutoClearCallerIdBufOnHangup**
Section	**[SystemConfig]**
Written Format	AutoClearCallerIdBufOnHangup=b
Value Range	b=0: No b=1: Yes(default)
Description	When the call is finished, this configuration item determines whether the driver will automatically clear the buffer area storing the calling party ID and the extended buffer area. If it's ISUP channel or TUP channel, the buffer area storing the called party number and 1st called party number will also be cleared
Note	This configuration item requires the driver version 2.1 or above

3.1.2.4.4 AutomaticAisGeneration

Configuration Item	**AutomaticAisGeneration**
Section	**[BoardId=x]**
Written Format	AutomaticAisGeneration[n]=b
Value Range	n: Logical PCM number, numbered from 0; b=0: No; (default) b=1: Yes
Description	When the input signal is interrupted, it determines whether to enable the automatic generation of the AIS message
Note	● This configuration item is only applicable to 8E1/16E1 links; ● When the current configuration item for PCM 4m is configured, the AutomaticAisGeneration for PCM 4*m+1, PCM 4*m+2 and PCM 4*m+3 will be automatically set by the system simultaneously (m is the natural number which makes the logical PCM index meaningful); ● This configuration item requires the driver version 4.7.3.1 or above

3.1.2.4.5 AutomaticRaiGereration

Configuration Item	**AutomaticRaiGereration**
Section	**[BoardId=x]**
Written Format	AutomaticRaiGereration =b
Value Range	b=0: No; (default) b=1: Yes
Description	When the input signal is lost, it determines whether to enable the automatic generation of the RAI message
Note	● This configuration item is only applicable to 8E1/16E1 links ● When the current configuration item for PCM 4m is configured, the AutomaticRAIGeneration for PCM 4*m+1, PCM 4*m+2 and PCM 4*m+3 will be automatically set by the system simultaneously (m is the natural number which makes the logical PCM index meaningful); ● This configuration item requires the driver version 4.7.3.1 or above

3.1.2.4.6 IgnoreBlockInGra

Configuration Item	**IgnoreBlockInGra**
Section	**[TUP]**
Written Format	IgnoreBlockInGra =b
Value Range	b=0: No; (default) b=1: Yes
Description	Whether to ignore the status indication field in GRA
Note	This configuration item is only valid to the TUP connection

3.1.2.4.7 AllowTimeoutInSpyISDN

Configuration Item	**AllowTimeoutInSpyISDN**
Section	**[BoardId=x]**
Written Format	AllowTimeoutInSpyISDN=b

Value Range	b=1: Reset the channel state back to S_SPY_STANDBY if time's out after the channel keeps in the S_SPY_RCVPHONUM state for a long time (default); b=0: Not to reset the channel state back to S_SPY_STANDBY if time's out after the channel keeps in the S_SPY_RCVPHONUM state for a long time.
Description	Sets whether to reset the channel state back to S_SPY_STANDBY if time's out after the channel keeps in the S_SPY_RCVPHONUM state for a long time.
Note	This configuration item is only applicable to the Synway DTP series boards. Forbidding the use of this configuration item may cause an incomplete call so as to suspend the system in the S_SPY_RCVPHONUM state for long time.

3.1.2.5 Setting Voice CODECs on Digital Lines

3.1.2.5.1 DefaultVoiceFormat

Configuration Item	**DefaultVoiceFormat**
Section	**[BoardId=x]**
Written Format	DefaultVoiceFormat=m
Value Range	m=6:A-Law; m=7:μ-Law
Description	Sets the voice data encoding format of the B-channel on digital lines (includes digital lines connecting with DST Series boards and digital trunks connecting with SHD/DTP Series boards). In north America and Japan, μ-Law is adopted; In Europe and China, A-Law is adopted. The encoding format of the B-channel voice data can be determined by referring to related manuals of the PBX and telephone, and can also be determined by experiment
Note	● This configuration item is only applicable to DST, SHD, DTP Series and C-type fax boards; ● For C-type fax boards, this configuration item is used to set the format of bit stream (A-Law or μ-Law); ● If the set value of this configuration item is not the same as the actual voice data encoding format of the B-channel, the recorded voice data will have noises

3.1.2.6 Setting Tone Detector

3.1.2.6.1 Setting Start/stop of Tone Detector on Digital Voice Board

3.1.2.6.1.1 DefaultToneCheckState

Configuration Item	**DefaultToneCheckState**
Section	**[BoardId=x]**
Written Format	DefaultToneCheckState =n
Value Range	n=0: Tone detector disabled (Default); n=1: Tone detector starts and stops automatically; n=2: Tone detector's start and stop are controlled by the application.
Description	Set the switch to enable or disable the tone detector for digital boards or IP boards. When n=1, the tone detector's start and stop will be affected by the channel state transition, just like analog trunk boards. When n = 2, the tone detector's start and stop are controlled by the application program through the API functions SsmStartToneAnalyze and SsmCloseToneAnalyze.

3.1.2.6.2 Setting Parameters of Noise Filter

3.1.2.6.2.1 MinimumVoiceDetermineEnergy

Configuration Item	**MinimumVoiceDetermineEnergy**
Section	**[SystemConfig]**
Written Format	MinimumVoiceDetermineEnergy=e
Value Range	e: e≥700, with the default value of 100000. For the conversion between the value of this parameter and the DB value, refer to 'FFT' in 'Tone Detector' of Chapter 1
Description	Sets the threshold value for noise detection on the line. For more information, refer to SsmSetMinVocDtrEnergy.

3.1.2.6.3 Setting Parameters of DTMF Pulse-width Filter

3.1.2.6.3.1 ToneHighFilterPoint

Refer to **ToneLowFilterPoint**

3.1.2.6.3.2 ToneLowFilterPoint

Configuration Item	**ToneHighFilterPoint** **ToneLowFilterPoint**
Section	**[SystemConfig]**
Written Format	ToneHighFilterPoint=t_H ToneLowFilterPoint=t_L
Value Range	t_H: The minimum duration (ms) of the tone at on state, it must be integral times of 16ms, t_H >0, with the default value of 64ms t_L: The minimum duration (ms) of the tone at off state, it must be integral times of 16ms, t_L >0, with the default value of 160 ms
Description	ToneHighFilterPoint sets the minimum duration of the tone. If the incoming signal is the same as that of the tone, but its duration is less than the set value of this parameter, the driver will regard it as the interfering signal; ToneLowFilterPoint sets the minimum duration that the tone disappears. If it's determined that the incoming signal is the same as that of the tone, and also the duration that this signal disappears is larger than the set value of this parameter, the driver confirms that this signal has disappeared.

3.1.2.6.4 Setting 1st Call Progress Tone Detector

3.1.2.6.4.1 Setting Parameters of Frequency Detector

3.1.2.6.4.1.1 TonePara

Configuration Item	**TonePara**
Section	**[SystemConfig]**
Written Format	TonePara =f1,b1,f2,b2,r

Value Range	f1: The first center frequency (Hz), range of value: 300~3400, with the default value of 450 b1: The bandwidth (Hz) of the first center frequency, range of value: 40~120, with the default value of 80 f2: The second center frequency (Hz), range of value: 300~3400, with the default value of 0 b2: The bandwidth (Hz) of the second center frequency, range of value: 40~120, with the default value of 0 r: The threshold value (%), range of value: 15~80, with the default value of 50 For more information, refer to the description of the function SsmSetTonePara
Description	Sets the Parameters of the frequency detector in the call progress tone detector. For more information, refer to Call Progress Tone Detector in Chapter 1

3.1.2.6.4.2 Setting Parameters of Dial Tone Detector

3.1.2.6.4.2.1 IsDialToneDetermineTime

Configuration Item	**IsDialToneDetermineTime**
Section	**[SystemConfig]**
Written Format	IsDialToneDetermineTime=t
Value Range	t: Minimum duration (ms), t≥1300, with the default value of 1500
Description	Sets the parameters of the dial tone detector in the 1st call progress tone detector. For more information, refer to SsmSetIsDialToneDtrTime

3.1.2.6.4.3 Setting Parameters of Echo Tone Detector

3.1.2.6.4.3.1 RingEchoTonePara

Configuration Item	**RingEchoTonePara**
Section	**[SystemConfig]**
Written Format	RingEchoTonePara=t_H,t_L
Value Range	t_H: The duration (ms) of the tone at on state, range of value: 300≤t_H≤2500, with the default value of 1000 t_L: The duration (ms) of the tone at off state, range of value: 800≤t_L≤6000, with the default value of 4000
Description	Sets the durations of the tone at on and off states for the echo tone detector. For more information, refer to the function of SsmSetRingEchoTonePara

3.1.2.6.4.4 Setting Parameters of Busy Tone Detector

3.1.2.6.4.4.1 BusyTonePeriod

Refer to **IsBusyToneDetermineCount**

3.1.2.6.4.4.2 IsBusyToneDetermineCount

Configuration Item	**BusyTonePeriod** **IsBusyToneDetermineCount**
Section	**[SystemConfig]**
Written Format	Format 1:BusyTonePeriod=p_1 Format 2:BusyTonePeriod=p_1,p_2 Format 3:BusyTonePeriod=p_1,p_2,p_3 Format 4:BusyTonePeriod=p_1,p_2,p_3,p_4 IsBusyToneDetermineCount=n

Value Range	p_1,p_2,p_3,p_4: Busy tone cycle (ms), range of value: 200≤p_i≤2000, with the default value of 700ms; n: The minimum number of busy tone cycles, 1≤n≤10, with the default value of 2
Description	Sets the parameters of the busy tone detector in the 1st Call Progress Tone Detector. For more information, refer to Busy Tone Detector in Chapter 1; BusyTonePeriod sets the busy tone cycles, at most 4 busy tone cycles can be set, separated by comma. Format 1, Format 2, Format 3 and Format 4 are applicable to setting 1~4 cycles respectively. For more information, refer to the function SsmSetBusyTonePeriodEx; IsBusyToneDetermineCount sets the minimum number of the busy tone cycles

Example	BusyTonePeriod=700 // Detect one type of busy tone, the cycle is 700ms
	BusyTonePeriod=700,1400 // Detect two types of busy tones, the cycles are 700ms and 1400ms respectively

3.1.2.6.4.5 Setting Parameters for Kewl Start

3.1.2.6.4.5.1 EnableKewlStart

Configuration Item	**EnableKewlStart**
Section	**[SystemConfig]**
Written Format	EnableKewlStart=m
Value Range	m=0: disable the Kewl Start feature (default) m=1: enable the Kewl Start feature
Description	Sets whether to enable the Kewl Start feature to detect remote hangup.

3.1.2.6.4.5.2 KSVoltageThreshold

Configuration Item	**KSVoltageThreshold**
Section	**[SystemConfig]**
Written Format	KSVoltageThreshold=n
Value Range	The threshold voltage to detect remote hangup (V), with the default value of 0 and the value range of n≥0
Description	Sets the threshold voltage for Kewl Start to detect remote hangup. For more information, refer to Change in Analog Phone Line Voltage section in Chapter 1.

3.1.2.6.4.5.3 KSKeepTime

Configuration Item	**KSKeepTime**
Section	**[SystemConfig]**
Written Format	KSKeepTime=m
Value Range	Duration (ms), with the default value of 256 and the value range of 16 ~ 65535.
Description	Sets the filter time for Kewl Start to detect remote hangup. For more information, refer to Change in Analog Phone Line Voltage section in Chapter 1.

3.1.2.6.4.6 Setting Parameters of User-defined Tone Detector

3.1.2.6.4.6.1 AppointedToneAnalyzerSwitch

Configuration Item	**AppointedToneAnalyzerSwitch**
Section	**[SystemConfig]**
Written Format	AppointedToneAnalyzerSwitch=m
Value Range	m=0: Stop the user-defined tone detector; m=1: Start the user-defined tone detector, detect the continuous tone, the configuration item IsAppointedToneDetermineTime is valid here; m=2: Start the user-defined tone detector, detect the periodic tones, the configuration item AppointedTonePara and IsAppointedToneDetermineCount are valid here

Description	Sets the user-defined tone detector in the 1st Call Progress Tone Detector. For more details, refer to User-defined Tone Detector in Chapter 1

3.1.2.6.4.6.2 IsAppointedToneDetermineTime

Configuration Item	**IsAppointedToneDetermineTime**
Section	**[SystemConfig]**
Written Format	IsAppointedToneDetermineTime=t
Value Range	t≥16: Minimum duration (ms), with the default value of 80
Description	Sets the minimum duration of the user-defined continuous tone, it's applicable to the user-defined tone detector in the 1st Call Progress Tone Detector. For more information, refer to User-defined Tone Detector in chapter 1
Note	● Only when the configuration item AppointedToneAnalyzerSwitch is set to be 1, this configuration item is valid; ● If the set value of this configuration item is less than the set value of IsDialToneDetermineTime, the dial tone will not be detected correctly; If the set value of this configuration item is larger than the set value of IsDialToneDetermineTime, the user-defined continuous tone will not be detected correctly

3.1.2.6.4.6.3 AppointedTonePara

Refer to **IsAppointedToneDetermineCount**

3.1.2.6.4.6.4 IsAppointedToneDetermineCount

Configuration Item	**AppointedTonePara** **IsAppointedToneDetermineCount**
Section	**[SystemConfig]**
Written Format	AppointedTonePara=t_H,t_L IsAppointedToneDetermineCount=n
Value Range	t_H: The duration (ms) of the tone at on state, 16≤t_H≤6000, with the default value of 2000 t_L: The duration (ms) of the tone at off state, 16≤t_L≤6000, with the default value of 2000 n: User-defined minimum number of tone cycles, 1≤n≤10, with the default value of 2
Description	Sets parameters of the periodic tones of User-defined Tone Detector in the 1st Call Progress Tone Detector; AppointedTonePara sets the durations of the tone at on and off states; IsAppointedToneDetermineCount sets the minimum number of the tone cycles to be counted.
Note	Only when the configuration item AppointedToneAnalyzerSwitch is set to be 2, this configuration item is valid

3.1.2.6.5 Setting 2nd Call Progress Tone Detector

3.1.2.6.5.1 Setting Operating Status

3.1.2.6.5.1.1 Enable2ndToneAnalyzer

Configuration Item	**Enable2ndToneAnalyzer**
Section	**[SystemConfig]**
Written Format	Enable2ndToneAnalyzer=b
Value Range	b=1: Start 2nd Call Progress Tone Detector; b=0: Stop 2nd Call Progress Tone Detector
Description	Sets the operating status of the 2nd Call Progress Tone Detector. For more information, refer to SsmStart2ndToneAnalyzer

3.1.2.6.5.1.2 Check2ndToneOnAutoDial

Configuration Item	**Check2ndToneOnAutoDial**
Section	**[SystemConfig]**
Written Format	Check2ndToneOnAutoDial=b
Value Range	b=1: Yes b=0: No (default)
Description	Sets whether the detected result of the 2nd Call Progress Tone Detector affects the state transition of the analog trunk channel. For more information, refer to 'Call Progress Tone Detector' in Chapter 1

3.1.2.6.5.2 Setting Parameters of Frequency Detector

3.1.2.6.5.2.1 2ndTonePara

Configuration Item	**2ndTonePara**
Section	**[SystemConfig]**
Written Format	TonePara =f_1,b_1,f_2,b_2,r
Value Range	f1: The first center frequency (Hz), range of value: 300~3400, with the default value of 450 b1: The bandwidth (Hz) of the first center frequency, range of value: 40~120, with the default value of 80 f2: The second center frequency (Hz), range of value: 300~3400, with the default value of 0 b2: The bandwidth (Hz) of the second center frequency, range of value: 40~120, with the default value of 0 r: The threshold value (%), range of value: 15~80, with the default value of 50 For more information, refer to the description of the function SsmSet2ndTonePara
Description	Sets parameters of the frequency detector in the 2nd Call Progress Tone Detector, for more information, refer to Call Progress Tone Detector in Chapter 1

3.1.2.6.5.3 Setting Parameters of Dial Tone Detector

3.1.2.6.5.3.1 2ndIsDialToneDetermineTime

Configuration Item	**2ndIsDialToneDetermineTime**
Section	**[SystemConfig]**
Written Format	2ndIsDialToneDetermineTime=t
Value Range	t: The minimum duration (ms) of the dial tone, n≥1300, with the default value of 1500
Description	Sets the dial tone detection parameters of the 2nd Call Progress Tone Detector. For more information, refer to SsmSet2ndIsDialToneDtrTime

3.1.2.6.5.4 Setting Parameters of Echo Tone Detector

3.1.2.6.5.4.1 2ndRingEchoTonePara

Configuration Item	**2ndRingEchoTonePara**
Section	**[SystemConfig]**
Written Format	2ndRingEchoTonePara=t_H,t_L
Value Range	t_H: The duration (ms) of the tone at on state, range of value: 300≤t_H≤2500, with the default value of 1000 t_L: The duration (ms) of the tone at off state, range of value: 800≤t_L≤6000, with the default value of 4000
Description	Sets the durations of the tone at on and off states for the ringback tone detector in the 2nd Call Progress Tone Detector. For more information, refer to SsmSet2ndRingEchoTonePara

3.1.2.6.5.5 Setting Parameters of Busy Tone Detector

3.1.2.6.5.5.1 2ndBusyTonePeriod

Refer to **2ndIsBusyToneDetermineCount**

3.1.2.6.5.5.2 2ndIsBusyToneDetermineCount

Configuration Item	**2ndBusyTonePeriod** **2ndIsBusyToneDetermineCount**
Section	**[SystemConfig]**
Written Format	Format 1:2ndBusyTonePeriod=p_1 Format 2:2ndBusyTonePeriod=p_1,p_2 Format 3:2ndBusyTonePeriod=p_1,p_2,p_3 Format 4:2ndBusyTonePeriod=p_1,p_2,p_3,p_4 2ndIsBusyToneDetermineCount=n
Value Range	p_1,p_2,p_3,p_4: Busy tone cycle (ms), range of value: 200≤p_i≤2000, default value: 700ms。 n: The minimum number of the busy tone cycles,1≤n≤10, with the default value of 2
Description	Sets parameters of the Busy Tone Detector in the 2nd Call Progress Tone Detector 2ndBusyTonePeriod sets the busy tone cycle, at most 4 busy tone cycles can be set, separated by ‘,’. Format 1, format 2, format 3 and format 4 are applicable to setting 1~4 cycles respectively . For more information, refer to SsmSetBusyTonePeriodEx; 2ndIsBusyToneDetermineCount sets the minimum number of the busy tone cycles

Example	2ndBusyTonePeriod=700 // Detect one type of busy tone, the cycle is 700 ms
	2ndBusyTonePeriod=700,1400 //Detect two types of busy tones, the cycles are 700 ms and 1400 ms respectively

3.1.2.6.5.5.3 MaxBsTnOffTime

Configuration Item	**MaxBsTnOffTime**
Section	**[SystemConfig]**
Written Format	MaxBsTnOffTime=t
Value Range	t (ms) ≥400, with the default value of 2000 ms
Description	Sets the duration of the busy tone detection. For more information, refer to Busy Tone Detector in Chapter 1
Note	This configuration item also affects the 2nd Call Progress Tone Detector

3.1.2.6.5.6 Setting Parameters of User-defined Tone Detector

3.1.2.6.5.6.1 2ndAppointedToneAnalyzerSwitch

Configuration Item	**2ndAppointedToneAnalyzerSwitch**
Section	**[SystemConfig]**
Written Format	2ndAppointedToneAnalyzerSwitch=m

Value Range	m=0: Stop the User-defined Tone Detector;
	m=1, Start the User-defined Tone Detector, detect continuous tone;
	m=2: Start the User-defined Tone Detector, detect periodic tone

Description	Sets the operating status of the User-defined Tone Detector in the 2nd Call Progress Tone Detector. For more information, refer to User-defined Tone Detector in Chapter 1

3.1.2.6.5.6.2 2ndIsAppointedToneDetermineTime

Configuration Item	**2ndIsAppointedToneDetermineTime**
Section	**[SystemConfig]**
Written Format	2ndIsAppointedToneDetermineTime=t

Value Range	t (ms) ≥16: the minimum duration, with the default value of 80
Description	Sets the minimum duration of the user-defined continuous tone, it's applicable to the User-defined Tone Detector in the 2nd Progress Tone Detector. For more information, refer to User-defined Tone Detector in Chapter 1
Note	● Only when the configuration item 2ndAppointedToneAnalyzerSwitch is set 1, this configuration item is valid; ● If the set value of this configuration item is less than the set value of 2ndIsDialToneDetermineTime, the dial tone will not be detected correctly; If the set value of this configuration item is larger than the set value of 2ndIsDialToneDetermineTime, the user-defined continuous tone will not be detected correctly

3.1.2.6.5.6.3 2ndAppointedTonePara

Refer to **2ndIsAppointedToneDetermineCount**

3.1.2.6.5.6.4 2ndIsAppointedToneDetermineCount

Configuration Item	**2ndAppointedTonePara** **2ndIsAppointedToneDetermineCount**
Section	**[SystemConfig]**
Written Format	2ndAppointedTonePara=t_H,t_L 2ndIsAppointedToneDetermineCount=n
Value Range	t_H: The duration (ms) of the tone at on state, 16≤t_H≤6000, with the default value of 2000 t_L: The duration (ms) of the tone at off state, 16≤t_L≤6000, with the default value of 2000 n: User-defined minimum number of tone cycles, 1≤n≤10, with the default value of 2
Description	Sets parameters of the periodic tones of User-defined Tone Detector in the 2nd Call Progress Tone Detector; 2ndAppointedTonePara sets the duration of the tone at off and on states, 2ndIsAppointedToneDetermineCount sets the minimum number of the tone cycles to be counted
Note	Only when the configuration item 2ndAppointedToneAnalyzerSwitch is set to be 2, this configuration item is valid

3.1.2.6.6 Setting Fax Progress Tone Detector

3.1.2.6.6.1 VoiceFreqF1Para

Refer to VoiceFreqF2Para

3.1.2.6.6.2 VoiceFreqF2Para

Configuration Item	**VoiceFreqF1Para** **VoiceFreqF2Para**
Section	**[SystemConfig]**
Written Format	VoiceFreqF1Para=f_1,b_1,r_1,t_1 VoiceFreqF2Para=f_2,b_2,r_2,t_2
Value Range	f_1,f_2: Center frequency (Hz), range of value: 300~3400, the default value of f_1 is 1100, the default value of f_2 is 2100; b_1,b_2: Bandwidth (Hz), range of value: 40~120, with the default value of 80; r_1,r_2: The minimum threshold value (%) for the percentage of in-band energy to overall energy, range of value: 15~80, with the default value of 50 t_1,t_2: The minimum duration (ms), range of value: ≥64, with the default value of 300

Description	Sets parameters of the fax progress tone detector. VoiceFreqF1Para sets the 1st fax progress tone detector; VoiceFreqF2Para sets the 2nd fax progress tone detector. For more information, refer to SsmSetVoiceFxPara.

3.1.2.6.7 Setting Simplified Busy Tone Detector

3.1.2.6.7.1 Setting Operating Status

3.1.2.6.7.1.1 ToneAnalyzeAtRcvFsk

Configuration Item	**ToneAnalyzeAtRcvFsk**
Section	**[BoardId=x]**
Written Format	ToneAnalyzeAtRcvFsk=b
Value Range	b=0: No (default); b=1: Yes
Description	When the FSK receiver is working, whether to start the Simplified Busy Tone Detector. For more information, refer to 'Simplified Busy Tone Detector' in Chapter 1
Note	This configuration item normally is only applicable to analog trunk channels of SHT Series, being valid when ToneDetectorMode=1.

3.1.2.6.7.2 Setting Parameters of Noise Filter

3.1.2.6.7.2.1 TAARFDetermineEnergy

Configuration Item	**TAARFDetermineEnergy**
Section	**[BoardId=x]**
Written Format	TAARFDetermineEnergy=e
Value Range	e≥700, with the default value of 100000; For the conversion between the value of this parameter and the DB value, refer to 'FFT' in 'Tone Detector' of Chapter 1.
Description	Sets the threshold value for noise detection on the line. For more information, refer to Simplified Busy Tone Detector in Chapter 1.
Note	This configuration item is only applicable to simplified busy tone detector.

3.1.2.6.8 Setting Way to Detect Ringback Tones

3.1.2.6.8.1 Setting Way to Detect Ringback Tones

3.1.2.6.8.1.1 IsEchoQuickDetect

Configuration Item	**IsEchoQuickDetect**
Section	**[BoardId=x]**
Written Format	IsEchoQuickDetect=b
Value Range	b=0: A complete ringback tone must be detected, and the ringback tone starts to be measured when it goes at off state (default); b=1: Based on the allowance error set by EchoOnTolerance or EchoOffTolerance, the ringback tone is detected. The ringback tone may start to be measured when it goes either at on or off state.
Description	Setting ringback tone detection method

3.1.2.6.8.2 Sets Allowance Error in Detecting Echo Tone at On State

3.1.2.6.8.2.1 EchoOnTolerance

Configuration Item	**EchoOnTolerance**
Section	**[BoardId=x]**
Written Format	EchoOnTolerance=e
Value Range	0≤e≤50, with the default value of 40
Description	Sets the allowance error in detecting the ringback tone at on state. If the value is set to 40, 40% error is allowed, i.e. for the ringback tone of 1sec on, as long as it is detected to go at on state for 600ms, it is regarded as the ringback tone at on state
Note	Works cooperatively with IsEchoQuickDetect and EchoOffTolerance

3.1.2.6.8.3 Setting Allowance Error in Detecting Echo Tone at Off State

3.1.2.6.8.3.1 EchoOffTolerance

Configuration Item	**EchoOffTolerance**
Section	**[BoardId=x]**
Written Format	EchoOffTolerance =e
Value Range	0≤e≤50, with the default value of 50
Description	Sets the allowance error in detecting the ringback tone at off state. If the value is set to 50, 50% error is allowed, i.e. for the ringback tone of 4sec off, as long as it is detected to go at off state for 2sec, it is regarded as the ringback tone at off state
Note	Works cooperatively with IsEchoQuickDetect and EchoOnTolerance

3.1.2.6.9 Setting CCIR Tone Detector

3.1.2.6.9.1 CCIREnableDetector

Configuration Item	**CCIREnableDetector**
Section	**[SystemConfig]**
Written Format	CCIREnableDetector=b
Value Range	b=0: Disable the CCIR tone detector (default); b=1: Enable the CCIR tone detector.
Description	Sets whether to enable the CCIR tone detector.

Note	This configuration item is only applicable to the following boards: SHT-8A/PCI SHT-8B/PCI SHT-8B/PCI/FAX SHT-8C/PCI/FAX SHT-8C/PCI/EC SHT-16A-CT/PCI SHT-16B-CT/PCI SHT-16B-CT/PCI/FAX SHT-16B-CT/cPCI SHT-16B-CT/cPCI/MP3 SHT-16B-CT/cPCI/FAX SHT-16C-CT/PCI/FAX SHT-16C-CT/PCI/EC SHT-2A/USB SHT-4A/USB SHT-2B/USB SHT-4B/USB

3.1.2.6.10 Setting Selcall Tone Detector

3.1.2.6.10.1 SelcallToneDetectMode

Configuration Item	**SelcallToneDetectMode**
Section	**[SystemConfig]**
Written Format	SelcallToneDetectMode=n

Value Range	n=0: Disable the Selcall Tone detector (default);
	n=1: Receive the Selcall Tone in Calling mode;
	n=2: Receive the Selcall Tone in Date mode.

Description	Sets whether to enable the Selcall Tone detector.
Note	This configuration item is only applicable to SHT and ATP Series.

3.1.2.6.10.2 SelcallToneType

Configuration Item	**SelcallToneType**
Section	**[SystemConfig]**
Written Format	SelcallToneType=n

Value Range	n=0: Read the frequency parameter of Selcall Tone from the configuration item SelcallTonePara (default);
	n=1: CCIR;
	n=2: EIA;
	n=3: EEA;
	n=4: ZVEI-I;
	n=5: ZVEI-II;
	n=6: ZVEI-III;
	n=7: PZVEI;
	n=8: NATEL;
	n=9: DZVEI.

Description	The type of Selcall Tone.
Note	This configuration item is only applicable to SHT and ATP Series.

3.1.2.6.10.3 SelcallTonePara

Configuration Item	**SelcallTonePara**
Section	**[SystemConfig]**

Written Format	SelcallTonePara=p0,p1,p2,…...,p15
Value Range	No limit.
Description	The frequency parameter of Selcall Tone.
Note	It's only valid when SelcallToneTyp=0.

3.1.2.6.11Setting Parameters for Back-to-back Busy Tone Detector

3.1.2.6.11.1 ExToneLevel

Configuration Item	**ExToneLevel**
Section	**[SystemConfig]**
Written Format	ExToneLevel=n
Value Range	Range of value: 50≤n≤1000, calculated by 16ms, with the default value of 187. The driver will use the default value 187 if n is less than 50 or greater than 1000.
Description	Sets the detection cycle of the back-to-back tone detector.
Note	Only the 1st call progress detector supports back-to-back busy tone detection.

3.1.2.7 Setting Answering Machine Detector

3.1.2.7.1 EnableAMD

Configuration Item	**EnableAMD**
Section	**[SystemConfig]**
Written Format	EnableAMD =b
Value Range	b=0: No; b=1: Yes (default).
Description	Sets whether to turn on the answering machine detector while the tone detector is working. For more details, refer to 'Answering Machine Detector' in Chapter 1.

3.1.2.7.2 AMDNoSoundAfterDialTime

Configuration Item	**AMDNoSoundAfterDialTime**
Section	**[SystemConfig]**
Written Format	AMDNoSoundAfterDialTime=n
Value Range	Default value is 15000 (ms).
Description	Uses to judge if the line silence after dial tone lasts overtime or not. If the line silence time is longer than the vaule of AMDNoSoundAfterDialTime, the event E_CHG_AMD will be thrown out (on this condition, the value of E_CHG_AMD is 5)

3.1.2.7.3 AMDNoSoundTime

Configuration Item	**AMDNoSoundTime**
Section	**[SystemConfig]**
Written Format	AMDNoSoundTime=n
Value Range	Default value is 10000 (ms).
Description	Uses to judge if the silence after tone or color ring lasts overtime or not. If the silence time is over the vaule of AMDNoSoundTime, the event E_CHG_AMD will be thrown out (on this condition, the value of E_CHG_AMD is 4)

3.1.2.7.4 AMDTimeOut

Configuration Item	**AMDTimeOut**
Section	**[SystemConfig]**
Written Format	AMDTimeOut =n
Value Range	Default value is 70000 (ms).
Description	Uses to judge if the whole AMD detecting process is overtime or not. If the time is over the vaule of AMDTimeOut, the event E_CHG_AMD will be thrown out (on this condition, the value of E_CHG_AMD is 3)

3.1.2.7.5 AMDToneCount

Configuration Item	**AMDToneCount**
Section	**[SystemConfig]**
Written Format	AMDToneCount=n
Value Range	16ms per unit. Default value is 10, that is 10x16ms=160ms.
Description	Judges if the time threshold for tone is detected. If this time is longer than the value of AMDToneCount, the event E_CHG_AMD will be thrown out (on this condition, the value of E_CHG_AMD is 1).

3.1.2.7.6 AMDTOn

Configuration Item	**AMDTOn**
Section	**[SystemConfig]**
Written Format	AMDTOn =n
Value Range	Default value is 80(ms).
Description	Uses to set the shortest duration for the voice to turn into the ON state.

3.1.2.7.7 AMDTOff

Configuration Item	**AMDTOff**
Section	**[SystemConfig]**
Written Format	AMDTOff =n
Value Range	Default value is 400(ms).
Description	Uses to set the shortest duration for the voice to turn into the OFF state.

3.1.2.7.8 AMDTimeA

Configuration Item	**AMDTimeA**
Section	**[SystemConfig]**
Written Format	AMDTimeA =n
Value Range	Default value is 600(ms).
Description	Uses to set the shortest silence duration before the phone is picked up by a man.

3.1.2.7.9 AMDTimeB

Configuration Item	**AMDTimeB**
Section	**[SystemConfig]**
Written Format	AMDTimeB =n
Value Range	Default value is 80(ms).
Description	Uses to set the shortest duration for the man who picked up the phone to speak.

3.1.2.7.10 AMDTimeC

Configuration Item	**AMDTimeC**
Section	**[SystemConfig]**
Written Format	AMDTimeC =n
Value Range	Default value is 1200(ms).
Description	Uses to set the longest duration for the man who picked up the phone to speak.

3.1.2.7.11 AMDTimeD

Configuration Item	**AMDTimeD**
Section	**[SystemConfig]**
Written Format	AMDTimeD =n
Value Range	Default value is 1000(ms).
Description	Uses to set the shortest silence duration after the phone is picked up by a man.

3.1.2.7.12 AMDSilentEnergy

Configuration Item	**AMDSilentEnergy**
Section	**[BoardId=x]**
Written Format	AMDSilentEnergy =n
Value Range	Energy value. Default value is 17000.
Description	Uses to set an energy value which decides it is voice or silence.

3.1.2.7.13 EnableAMDBeep

Configuration Item	**EnableAMDBeep**
Section	**[SystemConfig]**
Written Format	EnableAMDBeep=b
Value Range	b=0: No (default). Use the old AMD detection mode; b=1: Yes.
Description	Sets whether to enable the feature of distinguishing the ringback music from the cue tone when the tone detector is working.

3.1.2.7.14 AMDTOffEx

Configuration Item	**AMDTOffEx**

Section	**[SystemConfig]**
Written Format	AMDTOffEx =n
Value Range	The default value is equal to the set value of AMDTOff (ms).
Description	Sets the longest duration of the greetings at the OFF state after a call is picked up by a man.

3.1.2.8 Setting Enhanced Tone Detector

3.1.2.8.1 ToneDetectorMode

Configuration Item	**ToneDetectorMode**
Section	**[SystemConfig]**
Written Format	ToneDetectorMode =b
Value Range	b=0: Use the common tone detector (default); b=1: Use the enhanced tone detector.
Description	Selects a kind of tone detector for use.
Note	This configuration item is only applicable to the following boards: SHT-8A/PCI SHT-8B/PCI SHT-8B/PCI/FAX SHT-8C/PCI/FAX SHT-8C/PCI/EC SHT-16A-CT/PCI SHT-16B-CT/PCI SHT-16B-CT/PCI/FAX SHT-16B-CT/cPCI SHT-16B-CT/cPCI/MP3 SHT-16B-CT/cPCI/FAX SHT-16C-CT/PCI/FAX SHT-16C-CT/PCI/EC SHT-2A/USB SHT-4A/USB SHT-2B/USB SHT-4B/USB

3.1.2.8.2 VoiceOffDetermineTime

Configuration Item	**VoiceOffDetermineTime**
Section	**[SystemConfig]**
Written Format	VoiceOffDetermineTime =n
Value Range	n: Calculated by millisecond (ms), usually being 1.2 times the period of a ringback tone, with the default value of 5000ms.
Description	Sets the time of silence detection.

3.1.2.8.3 MaxToneDetectorItem

Configuration Item	**MaxToneDetectorItem**
Section	**[SystemConfig]**
Written Format	MaxToneDetectorItem =N
Value Range	1≦ N≦ 20, with the default value of 5.

Description	Sets there are how many kinds of tones to be detected.

3.1.2.8.4 ToneDetectorItem

Configuration Item	**ToneDetectorItem**
Section	**[SystemConfig]**
Written Format	**ToneDetectorItem[n]** = M,F_1,F_2,F_3,T_1,T_2,T_3,C_{on},C_{off},F_{err},R_{bw},T_{err},P_{end},C_{cnt},T_{clr},P_{pop},E_{event},S_{state},S_{stop},W_e
Value Range	The number of this configuration item varies on the value of MaxToneDetectorItem N. When the value range of n is 0~N-1, the enhanced tone detector supports 3 kinds of tones and the value of each item is shown as follows.

Parameter	Continuous Tone	Periodic Tone	SIT
M	Value Range: =0 Continuous tones, such as dial tones and fax tones F_1, F_2.	Value Range: =1 Periodic tones, such as busy tones, howler tones and ringback tones.	Value Range: =2 SIT_TONE (consists of 3 continuous single tones)
F_1(Hz)	Value Range: (>=300 && <=3400) The 1st mid-frequency	Value Range: (>=300 && <=3400) The 1st mid-frequency	Value Range: (>=300 && <=3400) The frequency of the 1st band
F_2(Hz)	Value Range: (==0 || (>=300 && <=3400)) The 2nd mid-frequency, which is equal to 0 in detection of single tones.	Value Range: (==0 || (>=300 && <=3400)) The 2nd mid-frequency, which is equal to 0 in detection of single tones.	Value Range: (>=300 && <=3400) The frequency of the 2nd band
F_3(Hz)	Invalid	Invalid	Value Range: (>=300 && <=3400) The frequency of the 3rd band
T_1(ms)	Value Range: >= 16 Duration at on state	Value Range: >= 16 Duration at on state	Value Range: >= 16 Duration of the 1st band
T_2(ms)	Invalid	Value Range: >= 16 Duration at off state	Value Range: >= 16 Duration of the 2nd band
T_3(ms)	Invalid	Invalid	Value Range: >= 16 Duration of the 3rd band
C_{on}	Value Range: (>0&&<=100) Filtering point number at on state (time length at each point is 16ms)	Value Range: (>0&&<=100) Filtering point number at on state (time length at each point is 16ms)	Value Range: (>0&&<=100) Filtering point number at on state (time length at each point is 16ms)
C_{off}	Value Range: (>0&&<=100) Filtering point number at off state (time length at each point is 16ms)	Value Range: (>0&&<=100) Filtering point number at off state (time length at each point is 16ms)	Value Range: (>0&&<=100) Filtering point number at off state (time length at each point is 16ms)

	F_{err}(Hz)	Value Range: (>0 && <=3400) Frequency error threshold	Value Range: (>0 && <=3400) Frequency error threshold	Value Range: (>0 && <=3400) Frequency error threshold
	R_{bw}(%)	Value Range: (>=15 && <=80) Threshold value for the minimum percentage of in-band energy to overall energy	Value Range: (>=15 && <=80) Threshold value for the minimum percentage of in-band energy to overall energy	Value Range: (>=15 && <=80) Threshold value for the minimum percentage of in-band energy to overall energy
	T_{err}(%)	Value Range: (>0 && <100) Accepted maximum error threshold for duration at on state (in percentage)	Value Range: (>0 && <100) Accepted maximum error threshold for duration at on or off state (in percentage)	Value Range: (>0 && <100) Accepted maximum error threshold for duration at on state (in percentage)
	P_{end}	Invalid	Value Range: 0 or 1 Way to judge a complete period. 0 indicates the use of threshold value to judge the off state. Judgement condition: Duration at off state >= T_2. 1 indicates the use of error to judge the off state. Judgement condition: T_2*(1-T_{err}(%))<= Duration at off state <=T_2*(1+T_{err}(%))	Invalid
	C_{cnt}	Invalid	Value Range: (>0 && <=10) Throw out the event E_CHG_ToneDetector upon detection of C_{cnt} complete periods.	Invalid
	T_{clr}(ms)	Invalid	Value Range:>= 16 Clear the period count if no complete period is detected during the time T_{clr}(ms)	Invalid
	E_{event}	The type of the event which is thrown out to the application program after tone detection. See Note 1 for details.	The type of the event which is thrown out to the application program after tone detection. See Note 1 for details.	The type of the event which is thrown out to the application program after tone detection. See Note 1 for details.
	S_{state}	Value Range: 0 or 1 Whether to throw out the event to the channel state machine. 0: Not throw out; 1: Throw out.	Value Range: 0 or 1 Whether to throw out the event to the channel state machine. 0: Not throw out; 1: Throw out.	Value Range: 0 or 1 Whether to throw out the event to the channel state machine. 0: Not throw out; 1: Throw out.

S_{stop}	Value Range: 0 or 1 Whether to stop tone detection once a tone is detected. 0: Not stop; 1: Stop.	Value Range: 0 or 1 Whether to stop tone detection once a tone is detected. 0: Not stop; 1: Stop.	Value Range: 0 or 1 Whether to stop tone detection once a tone is detected. 0: Not stop; 1: Stop.
W_e	Value Range: 0 or 1 0 indicates the tone detector is working in FFT analysis; 1 indicates the tone detector is working in FSK reception.	Value Range: 0 or 1 0 indicates the tone detector is working in FFT analysis; 1 indicates the tone detector is working in FSK reception.	Value Range: 0 or 1 0 indicates the tone detector is working in FFT analysis; 1 indicates the tone detector is working in FSK reception.

Note 1: The driver will throw out the event E_CHG_ToneDetector upon tone detection. The 2 lower bits in the parameter dwParam of this event represent the tone type, which are evaluated by E_{vent} in the configuration item ToneDetectorParamItem[n] in the use of the enhanced tone detector. The value range of E_{vent} is 0~0xFFFF, while 0~9 have been defined by the driver. See the table below for details.

Configured Value	Corresponding Flag in Driver	Description	Use analog channel call state machine?	Can be used by application program?
0	CHKTONE_NO_RESULT	Detecting…	N	N
1	CHKTONE_DIALTONE	Dial tone detected	Y	Y
2	CHKTONE_BUSYTONE	Busy tone detected	Y	Y
3	CHKTONE_ECHOTONE	Ringback tone detected	Y	Y
4	CHKTONE_ECHO_NOVOICE	Silent when ringback tone detected	Y	N
5	CHKTONE_NOVOICE	Silence detected	Y	N
6	CHKTONE_VOICE	Speaking voice detected	Y	N
7	CHKTONE_VOICEF1	Tones at the frequency of F_1 detected	Y	Y
8	CHKTONE_VOICEF2	Tones at the frequency of F_2 detected	Y	Y
9	CHKTONE_APPOINTEDTONE	User-specified tone type detected	N	Y
0xA~0x001F	Remained by driver		N	N
0x0020~0xFFFF	Unused		N	Y

In the table above, if CHKTONE_DIALTONE, CHKTONE_BUSYTONE, CHKTONE_ECHOTONE, CHKTONE_VOICEF1, CHKTONE_VOICEF2 are not used by the application program, the analog channel call state machine will not work.

Below is a list of the default setting of ToneDetectorParamItem [n]. Make sure it is identical for both common and enhanced tone detectors.

	Parameter	DIALTONE	BUSYTONE	ECHOTONE	VOICEF1	VOICEF2
	M	0	1	1	0	0
	F_1(Hz)	450	450	450	1100	2100
	F_2(Hz)	0	0	0	0	0
	F_3(Hz)	0	0	0	0	0
	T_1(ms)	1500	350	1000	250	250
	T_2(ms)	0	350	4000	0	0
	T_3(ms)	0	0	0	0	0
	C_{on}	4	4	4	4	4
	C_{off}	10	10	10	10	10
	F_{err}(Hz)	30	30	30	30	30
	R_{bw}(%)	50	50	50	50	50
	T_{err}(%)	20	20	20	20	20
	P_{end}	0	1	0	0	0
	C_{nt}	0	2	1	0	0
	T_{clr}(ms)	0	3000	6000	0	0
	E_{event}	1	2	3	7	8
	S_{state}	1	1	1	1	1
	S_{stop}	0	0	0	0	0
	W_e	0	0	0	0	0
Description	Sets each parameter for tone detection so as to detect all specified tones.					

3.1.2.9 Setting Tone Generator(CTI Series)

3.1.2.9.1 DefaultSendToneFrequence

Refer to DefaultSendToneVolume.

3.1.2.9.2 DefaultSendToneVolume

Configuration Item	**DefaultSendToneFrequence** **DefaultSendToneVolume**
Section	**[BoardId=x]**
Written Format	DefaultSendToneFrequence=f_1,f_2 DefaultSendToneVolume=v_1,v_2
Value Range	f1: The 1st frequency (Hz) of the tone, with the default value of 450; f2: The 2nd frequency (Hz) of the tone, f2 equals to 0 means single tone, greater than 0 means dual tone. Default value is 0; v_1: The 1st volume of the tone, range of value: -7≤v_1≤+6. v_1=-7 means this tone is disabled; if v_1 is greater than 0, it indicates the increase in volume; if v_1 is less than 0, it means the decrease in volume. The set value multiplied by 3 equals to the DB value, with the default value of 0; v_2: The 2nd volume of the tone, the range of value and meaning are the same as v_1, with the default value of -7
Description	Sets the parameters of the tone generator. DefaultSendToneFrequence sets the frequency of the tone, DefaultSendToneVolume sets the volume of the tone

Example	DefaultSendToneFrequence=450, 600 // Using dual tone
	DefaultSendToneVolume=0,0

3.1.2.10 Setting Teleconferencing Parameters (CTI Series)

3.1.2.10.1ClearInVoiceOnRxDtmf

Configuration Item	**ClearInVoiceOnRxDtmf**
Section	**[BoardId=x]**
Written Format	Format 1:ClearInVoiceOnRxDtmf=n,n,n,n Format 2:ClearInVoiceOnRxDtmf=n,n,n,n,n,n,n,n Format 3:ClearInVoiceOnRxDtmf=n,n,n,n,n,n,n,n,n,n,n,n,n,n,n,n Format 4:ClearInVoiceOnRxDtmf=n
Value Range	n=0: Disable the DTMF clamping feature (default); n=1: Enable the DTMF clamping feature
Description	Whether to enable the DTMF clamping feature, that is, whether to cut DTMF signals while putting voices onto bus. For more information, refer to 'Distributed Teleconferencing System' in Chapter 1
Note	● Format 1 is applicable to SHT Series voice boxes and ATP Series recording boxes adopting USB ; ● Format 2 is applicable to 8-channel SHT Series boards; ● Format 3 is applicable to 16-channel SHT Series boards; ● Format 4 is applicable to SHD and SHN Series boards; ● The function of SsmSetFlag (with the parameter of F_ClearInVoiceOnRcvDtmf) may implement the same feature

3.1.2.10.2ClearInVoiceOnRx450Hz

Configuration Item	**ClearInVoiceOnRx450Hz**
Section	**[BoardId=x]**
Written Format	Format 1: ClearInVoiceOnRx450Hz =n,n,n,n,n,n,n,n Format 2: ClearInVoiceOnRx450Hz =n,n,n,n,n,n,n,n,n,n,n,n,n,n,n,n
Value Range	n=0: Not onto bus (default); n=1: Onto bus.
Description	Sets whether to put tones onto bus.
Note	● Format 1 is applicable to 8-channel SHT Series; ● Format 2 is applicable to 16-channel SHT Series.

3.1.2.10.3ConfMaxGroup

Configuration Item	**ConfMaxGroup**
Section	**[SystemConfig]**
Written Format	ConfMaxGroup=n
Value Range	n:n≥0 (with the default value of 64)
Description	Sets the maximum number of the conference rooms that can be created

3.1.2.10.4ConfDefaultMaxGroupMember

Configuration Item	**ConfDefaultMaxGroupMember**
Section	**[SystemConfig]**
Written Format	ConfDefaultMaxGroupMember=n

Value Range	n:n≥0 (with the default value of 64)
Description	Sets the maximum number of the conference channels that can be put into the conference room

3.1.2.10.5ConfDefaultMaxGroupSpeaker

Configuration Item	**ConfDefaultMaxGroupSpeaker**
Section	**[SystemConfig]**
Written Format	ConfDefaultMaxGroupSpeaker=n
Value Range	n: The number of the conference channels, with the default value of 24
Description	Sets the maximum number of the conference channels with speaking rights. The speaking modes of organizer, chairman and dynamic speaker are all provided with speaking rights. Whether the speaking mode of background music is provided with speaking right is determined by the setting of the configuration item BackgroundVoicePriority

3.1.2.10.6ConfDefaultMaxGroupSpeaking

Configuration Item	**ConfDefaultMaxGroupSpeaking**
Section	**[SystemConfig]**
Written Format	ConfDefaultMaxGroupSpeaking=n
Value Range	n≤6, with the default value of 6
Description	Sets the maximum number of conference channels speaking simultanously in a conference room.

3.1.2.10.7ConfJoinedbyEnergy

Configuration Item	**ConfJoinedbyEnergy**
Section	**[SystemConfig]**
Written Format	ConfJoinedbyEnergy=n
Value Range	n=0: Not preemptible; n=2: Support preemptible speaking (default)
Description	Determines whether to support the preemptible speaking in the dynamic speaker mode.

3.1.2.10.8ConfMaxListener

Configuration Item	**ConfMaxListener**
Section	**[SystemConfig]**
Written Format	ConfMaxListener =n
Value Range	n: The conference channel number, with the default value of 480
Description	The maximum number of the listeners that are allowed to join the conference room

3.1.2.10.9ConfDefaultMaxSilenceTime

Configuration Item	**ConfDefaultMaxSilenceTime**
Section	**[SystemConfig]**
Written Format	ConfDefaultMaxSilenceTime=t

Value Range	t: Duration (seconds) , with the default value of 5 seconds
Description	Sets the minimum duration for the channel to keep silent. For more information, refer to 'Distributed Teleconferencing System' in Chapter 1
Note	Now this configuration item has been invalid.

3.1.2.10.10 BackgroundVoicePriority

Configuration Item	**BackgroundVoicePriority**
Section	**[SystemConfig]**
Written Format	BackgroundVoicePriority=n
Value Range	n=1: The priority is higher than the organizer mode. It's provided with special conference mixer; n=2: The priority is lower than the organizer mode, but higher than the chairman mode default value); n=3: The priority is lower than the chairman mode, but higher than the dynamic speaker mode; n=4: The priority is lower than the dynamic mode
Description	Sets the scheduling priority for the speaking mode of the channel which is used to play the background music. For more information, refer to 'Distributed Teleconferencing System' in Chapter 1

3.1.2.10.11 PlayVoiceIsListen

Configuration Item	**PlayVoiceIsListen**
Section	**[SystemConfig]**
Written Format	PlayVoiceIsListen =n
Value Range	n=0: this channel can only speak (can not listen); n=1: this channel can speak and listen
Description	Sets whether to enable the channel which is used to play background music to listen to the teleconference.

3.1.2.10.12 QuitDynKeepingInConf

Configuration Item	**QuitDynKeepingInConf**
Section	**[SystemConfig]**
Written Format	QuitDynKeepingInConf =n
Value Range	n=0(default): Provided the number of people speaking at the same time in the scheduling is n and the max number of speakers is m, if n<m, then m-n people who do not speak this time but have the highest volume in last scheduling will still occupy the local bus and have the speech right. Thus, other conference members may hear slight sound which cannot cause Barge In, such as the background noises. n=1: Cancel the speech right of a conference member in dynamic speaking status once the member stops speaking, to prevent the background noises being heard by other members.
Description	Sets whether to cancel the speech right of a conference member in dynamic speaking status once the member stops speaking, to prevent the background noises being heard by other members. Background noises are slight sound which cannot cause Barge In.
Note	● This configuration requires SynCTI Ver. 5.3.1.3 or above. ● This configuration is applicable for all channels which support conferencing.

3.1.2.11 Setting Voice Playing Operation

3.1.2.11.1 Setting Termination Condition of Voice Playing

3.1.2.11.1.1 DefaultDtmfStopPlay

Refer to DtmfStopPlayCharSet

3.1.2.11.1.2 DtmfStopPlayCharSet

Configuration Item	**DefaultDtmfStopPlay** **DtmfStopPlayCharSet**
Section	**[BoardId=x]**
Written Format	Format 1:DefaultDtmfStopPlay=b,b,b,b DtmfStopPlayCharSet={s,s,s,s} Format 2:DefaultDtmfStopPlay=b,b,b,b,b,b,b,b DtmfStopPlayCharSet={s,s,s,s,s,s,s,s} Format 3:DefaultDtmfStopPlay=b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b DtmfStopPlayCharSet={s,s,s,s,s,s,s,s,s,s,s,s,s,s,s,s} Format 4:DefaultDtmfStopPlay=b DtmfStopPlayCharSet=s
Value Range	b=0: No (default), DtmfStopPlayCharSet is invalid at this time; b=1: Yes, DtmfStopPlayCharSet is valid at this time; s={0123456789*#abcd }, the character set containing any number of DTMF characters, the default value is a full set
Description	DefaultDtmfStopPlay sets whether the voice playing is terminated because the DTMF Detector detects the DTMF character. If it is set to 'yes', DtmfStopPlayCharSet sets the DTMF character set
Note	● Format 1 is applicable to SHT Series USB voice boxes; ● Format 2 and format 3 are applicable to 8-channel and 16-channel SHT Series respectively; ● Format 4 is applicable to SHD and SHN Series.
Example	DefaultDtmfStopPlay=1 DtmfStopPlayCharSet={#} // SHD Series: The voice playing is stopped once '#' is detected, other characters are not effective

3.1.2.11.1.3 HangupStopPlay

Configuration Item	**HangupStopPlay**
Section	**[BoardId=x]**
Written Format	Format 1:HangupStopPlay=b,b,b,b Format 2:HangupStopPlay=b,b,b,b,b,b,b,b Format 3:HangupStopPlay=b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b Format 4:HangupStopPlay=b
Value Range	b=0: No (default); b=1: Yes
Description	Sets whether the voice playing is terminated because the state machine of the driver detects remote hangup
Note	● Format 1 is applicable to SHT Series USB voice boxes; ● Format 2 and format 3 are applicable to 8-channel and 16-channel SHT Series respectively; ● Format 4 is applicable to SHD and SHN Series.

3.1.2.11.1.4 DefaultBargeInStopPlay

Configuration Item	**DefaultBargeInStopPlay**
Section	**[BoardId=x]**
Written Format	Format 1:DefaultBargeInStopPlay=b,b,b,b Format 2:DefaultBargeInStopPlay=b,b,b,b,b,b,b,b Format 3:DefaultBargeInStopPlay=b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b Format 4:DefaultBargeInStopPlay=b
Value Range	b=0: No (default); b=1: Yes
Description	Sets whether the voice playing is terminated because the Barge in Detector detects voice activities
Note	● Format 1 is applicable to SHT Series USB voice boxes; ● Format 2 and format 3 are applicable to 8-channel and 16-channel SHT Series respectively; ● Format 4 is applicable to SHD and SHN Series

3.1.2.11.2 Setting Internal Playback Buffer

3.1.2.11.2.1 PlayBufSize

Configuration Item	**PlayBufSize**
Section	**[SystemConfig]**
Written Format	PlayBufSize=k
Value Range	k: The length (bytes) of the playback buffer area, it must be integral times of 512, with the default value of 32768 bytes
Description	Sets the size of the internal playback buffer area allocated by the driver for each channel

3.1.2.11.3 Setting Parameters of Voice Playing in File List Mode

3.1.2.11.3.1 MaxPlayFileList

Configuration Item	**MaxPlayFileList**
Section	**[SystemConfig]**
Written Format	MaxPlayFileList=n
Value Range	0≤n≤65535, with the default value of 256
Description	Sets the upper limit number of the voice files that the application is allowed to submit to the driver through the function call of SsmAddToFileList

3.1.2.11.4 Setting Parameters of Voice Playing in Preload File Mode

3.1.2.11.4.1 MaxPlayIndexList

Configuration Item	**MaxPlayIndexList**
Section	**[SystemConfig]**
Written Format	MaxPlayIndexList=n
Value Range	0≤n≤512, with the default value of 256.
Description	Sets the upper limit number of preload voice segments allowed to be played upon one call of SsmPlayIndexList or SsmPlayIndexString.

3.1.2.11.5 Setting Parameters of Voice Playing in Buffer List Mode

3.1.2.11.5.1 MaxPlayMemList

Configuration Item	**MaxPlayMemList**
Section	**[SystemConfig]**
Written Format	MaxPlayMemList=n
Value Range	0≤n≤65535, with the default value of 256
Description	Sets the upper limit number of the buffer areas that the application is allowed to submit to the driver through the function call of SsmAddToPlayMemList

3.1.2.11.6 Setting Parameters of Voice Playing in Single File Mode

3.1.2.11.6.1 FastPlayTime

Configuration Item	**FastPlayTime**
Section	**[SystemConfig]**
Written Format	FastPlayTime=t
Value Range	t denotes the step size (ms) of each operation of fast-forward or fast-backward, the with the default value of 1000
Description	Sets the step sizes of the functions of SsmFastFwdPlay and SsmFastBwdPlay

3.1.2.11.7 Setting Volume of Voice Playing

3.1.2.11.7.1 DefaultPlayVolume

Configuration Item	**DefaultPlayVolume**
Section	**[BoardId=x]**
Written Format	Format 1:DefaultPlayVolume=v,v,v,v Format 2:DefaultPlayVolume=v,v,v,v,v,v,v,v Format 3:DefaultPlayVolume=v,v,v,v,v,v,v,v,v,v,v,v,v,v,v,v Format 4:DefaultPlayVolume=v
Value Range	-7≤v≤+6, with the default value of 0
Description	Sets the gain of the volume adjuster A1 of SHD/SHT Series
Note	● Format 1 is applicable to SHT Series USB voice boxes and ATP Series USB recording boxes; ● Format 2 is applicable to 8-channel SHT and ATP Series; ● Format 3 is applicable to 16-channel SHT, ATP and DST Series; ● Format 4 is applicable to SHD, SHN and DTP Series

3.1.2.11.8 Setting Encoding Format of Voice Playing

3.1.2.11.8.1 DefaultPlayFormat

Configuration Item	**DefaultPlayFormat**
Section	**[BoardId=x]**

Written Format	Format 1: DefaultPlayFormat =v,v,v,v Format 2: DefaultPlayFormat =v,v,v,v,v,v,v,v Format 3: DefaultPlayFormat =v,v,v,v,v,v,v,v,v,v,v,v,v,v,v,v Format 4: DefaultPlayFormat =v
Value Range	Please refer to the 9 formats of nFormat in SsmRecToFile,. The default value is v=6. For the format supported by each board, please refer to 1.3.2 Board Supported CODECs
Description	Sets the channel's defaulted encoding format of voice playing
Note	● Format 1 is applicable to SHT Series USB voice boxes and ATP Series USB recording boxes; ● Format 2 is applicable to 8-channel SHT and ATP Series; ● Format 3 is applicable to 16-channel SHT, ATP and DST Series; ● Format 4 is applicable to SHD, SHN and DTP Series

3.1.2.11.9Setting Other Parameters of Voice Playing

3.1.2.11.9.1 DefaultPausePlayOnRxDtmf

Configuration Item	**DefaultPausePlayOnRxDtmf**
Section	**[BoardId=x]**
Written Format	Format 1:DefaultPausePlayOnRxDtmf=b,b,b,b Format 2:DefaultPausePlayObRxDtmf=b,b,b,b,b,b,b,b Format 3:DefaultPausePlayObRxDtmf=b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b Format 4:DefaultPausePlayObRxDtmf=b Format 5:DefaultPausePlayObRxDtmf=[0,29]:b
Value Range	b=0: No; b=1: Yes (default)
Description	When the DTMF Detector detects DTMF signals in the incoming call, if the voice is playing on the channel, in order to ensure the correct operation of the DTMF Detector, the driver needs to suspend the voice playing. The driver will automatically resume the voice playing until the DTMF signal disappears. This configuration item sets whether to enable this feature or not. Normally, it's set to be 0 for recording channels and 1 for other channels
Note	● Format 1 is applicable to SHT and ATP Series USB voice boxes/recording boxes; ● Format 2 is applicable to 8-channel SHT and ATP Series; ● Format 3 is applicable to 16-channel SHT, ATP and ATP Series; ● Format 4 is applicable to SHD Series, SHN Series; ● Format 5 is applicable to the channel bank of 30 channels ● This configuration item is inapplicable to 16-channel C-type SHT series.

3.1.2.11.9.2 PlayFilterFlag

Configuration Item	**PlayFilterFlag**
Section	**[BoardId=x]**
Written Format	Format 1: PlayFilterFlag =b,b,b,b,b,b,b,b Format 2: PlayFilterFlag =b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b
Value Range	b=0: No (default); b=1: Yes.
Description	If voices are played in great volume, the DTMF detector probably fails to acquire the correct DTMF digits sent from the remote end. In such situation, the filter should be enabled to avoid the effect of the great volume on DTMF reception. This configuration item just determines whether to enable the filter or not.
Note	● This configuration item is only applicable to SHT-8C/PCI/EC, SHT-16C/PCI/EC and SHT-16C/PCI/FAX boards; ● Format 1 is applicable to 8-channel SHT Series; ● Format 2 is applicable to 16-channel SHT Series.

3.1.2.12 Setting Recording Operation

3.1.2.12.1 Setting Termination Condition of Voice Recording

3.1.2.12.1.1 DtmfStopRecCharSet

Configuration Item	**DtmfStopRecCharSet**
Section	**[BoardId=x]**
Written Format	Format 1:DtmfStopRecCharSet={s,s,s,s} Format 2:DtmfStopRecCharSet={s,s,s,s,s,s,s,s} Format 3:DtmfStopRecCharSet={s,s,s,s,s,s,s,s,s,s,s,s,s,s,s,s} Format 4:DtmfStopRecCharSet=s
Value Range	's' is the DTMF character set with the format of {***}, in which * may be any one character of '0123456789*#abcd', the number of characters is set based on the actual needs. If 's' is an empty set, this feature is disabled; If 's' is not empty, this feature is enabled. Default value is an empty set
Description	Sets whether the recording is terminated because the DTMF Detector detects the DTMF character
Note	● Format 1 is applicable to the SHT Series USB voice box and the ATP Series USB recording box; ● Format 2 is applicable to 8-channel SHT and ATP Series; ● Format 3 is applicable to 16-channel SHT, ATP and DST Series; ● Format 4 is applicable to SHD, DTP and SHN Series.
Example	DtmfStopRecCharSet={0123456789*#abcd } //SHD Series: The recording stops once any of the characters in the set is detected

3.1.2.12.1.2 HangupStopRec

Configuration Item	**HangupStopRec**
Section	**[BoardId=x]**
Written Format	Format 1:HangupStopRec=b,b,b,b Format 2:HangupStopRec=b,b,b,b,b,b,b,b Format 3:HangupStopRec=b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b Format 4:HangupStopRec=b
Value Range	b=0: No (default); b=1: Yes
Description	Sets whether the voice recording is terminated because the state machine of the driver detects remote hangup
Note	● Format 1 is applicable to the SHT Series USB voice box; ● Format 2 is applicable to 8-channel SHT Series; ● Format 3 is applicable to 16-channel SHT Series; ● Format 4 is applicable to SHD and SHN Series.

3.1.2.12.2 Setting Recording Signal Sources

3.1.2.12.2.1 DefaultRecordVolume

Refer to **DefaultRecordMixerVolume**

3.1.2.12.2.2 DefaultRecordMixerVolume

Configuration Item	**DefaultRecordVolume** **DefaultRecordMixerVolume**
Section	**[BoardId=x]**
Written Format	Format 1:DefaultRecordVolume=v_3,v_3,v_3,v_3 DefaultRecordMixerVolume=v_2,v_2,v_2,v_2 Format 2:DefaultRecordVolume=v_3,v_3,v_3,v_3,v_3,v_3,v_3,v_3 DefaultRecordMixerVolume=v_2,v_2,v_2,v_2,v_2,v_2,v_2,v_2 Format 3:DefaultRecordVolume= v_3,v_3,v_3,v_3,v_3,v_3,v_3,v_3,v_3,v_3,v_3,v_3,v_3,v_3,v_3,v_3 DefaultRecordMixerVolume=v_2,v_2,v_2,v_2,v_2,v_2,v_2,v_2,v_2,v_2,v_2,v_2,v_2,v_2,v_2,v_2 Format 4:DefaultRecordVolume=v_3 DefaultRecordMixerVolume=v_2
Value Range	v_3,v_2: Volume of the volume adjuster, $-7 \le v_3 \le +6$,$-7 \le v_2 \le +6$. -7 implies the volume adjuster is stopped. Other values represent the volume gains ($v_3/v_2 \times 3$ being the decibel value (db)), those greater than 0 indicating the increase in volume and those less than 0 reflecting the decrease in volume. The default value of v_3 is 0 while the default value of v_2 is -7.
Description	Sets the gain of signals entering the recording mixer M3. DefaultRecordVolume sets the gain of the volume adjuster A3, while DefaultRecordMixerVolume sets the gain of the volume adjuster A2. For more information, refer to the operation principle of corresponding boards in Chapter 1
Note	● Format 1 is applicable to the SHT Series USB voice box and the ATP Series USB recording box; ● Format 2 is applicable to 8-channel SHT and ATP Series; ● Format 3 is applicable to 16-channel SHT and ATP Series (DefaultRecordVolume is also applicable to DST Series); ● Format 4 is applicable to SHD, SHN and DTP Series
Example	DefaultRecordVolume=1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1 // Increase the volume by 3DB

3.1.2.12.3Setting Tail-Truncation Feature for File Recording

3.1.2.12.3.1 TruncateTailOnRecordToFile

Configuration Item	**TruncateTailOnRecordToFile**
Section	**[SystemConfig]**
Written Format	TruncateTailOnRecordToFile=t
Value Range	t=0: Disable this feature (default); t>0: The length (ms) of the voice data truncated from the file tail
Description	Sets the time length for tail truncation during the file recording. For more information about the file tail truncation during file recording, refer to the description of the function SsmSetTruncateTail.
Note	This configuration item requires SynCTI Ver. 2.1 or above

3.1.2.12.4Setting Encoding Format of Voice Recording

3.1.2.12.4.1 DefaultRecordFormat

Configuration Item	**DefaultRecordFormat**
Section	**[BoardId=x]**

Written Format	Format 1: DefaultRecordFormat =v,v,v,v Format 2: DefaultRecordFormat =v,v,v,v,v,v,v,v Format 3: DefaultRecordFormat =v,v,v,v,v,v,v,v,v,v,v,v,v,v,v,v Format 4: DefaultRecordFormat =v
Value Range	Please refer to the 9 formats of nFormat in SsmRecToFile. The default value is v=6, A-law; For the format supported by each board, please refer to 1.3.2 Board Supported CODECs.
Description	Sets the channel's defaulted encoding format of voice recording.
Note	● Format 1 is applicable to SHT Series USB voice boxes and ATP Series USB recording boxes; ● Format 2 is applicable to 8-channel SHT and ATP Series; ● Format 3 is applicable to 16-channel SHT, ATP and DST Series; ● Format 4 is applicable to SHD, SHN and DTP Series

3.1.2.12.5Setting Internal Recording Buffer

3.1.2.12.5.1 RecordBufSize

Configuration Item	**RecordBufSize**
Section	**[SystemConfig]**
Written Format	RecordBufSize=k
Value Range	k: The length of the buffer area, calculated by byte. It must be the multiple of 512, with the minimum value of 4096 and the default value of 32768. When GsmCodecEnable is enabled, the minimum value of this configuration item turns to be 16384.
Description	Sets the size of the internal recording buffer area allocated by the driver for each channel

3.1.2.12.6Setting WAV Format for Recording

3.1.2.12.6.1 RecAsWavFormat

Configuration Item	**RecAsWavFormat**
Section	**[SystemConfig]**
Written Format	RecAsWavFormat =K
Value Range	K=0: Use the format of the file itself for recording, i.e. only the file with an extension name of wav or mp3 is recorded in wav format. K=1: All files are recorded in wav format.
Description	Sets the use of wav format to record files.

3.1.2.12.6.2 Mp3IsOnlyRead

Configuration Item	**Mp3IsOnlyRead**
Section	**[SystemConfig]**
Written Format	Mp3IsOnlyRead =K
Value Range	K=0: The recorded MP3 file is in the common state, readable and writable; K=1: The recorded MP3 file is in the read-only state, read only, not writable.
Description	Sets the property of the recorded MP3 files.

3.1.2.12.7Setting AGC Parameters

3.1.2.12.7.1 AGCMAXGAIN

Refer to AGCMINGAINDOWNLINK

3.1.2.12.7.2 AGCMINGAIN

Refer to AGCMINGAINDOWNLINK

3.1.2.12.7.3 AGCMAXGAINUPLINK

Refer to AGCMINGAINDOWNLINK

3.1.2.12.7.4 AGCMINGAINUPLINK

Refer to AGCMINGAINDOWNLINK

3.1.2.12.7.5 AGCMAXGAINDOWNLINK

Refer to AGCMINGAINDOWNLINK

3.1.2.12.7.6 AGCMINGAINDOWNLINK

Configuration Item	**AGCMAXGAIN** **AGCMINGAIN** **AGCMAXGAINUPLINK** **AGCMINGAINUPLINK** **AGCMAXGAINDOWNLINK** **AGCMINGAINDOWNLINK**
Section	**[BoardId=x]**
Written Format	AGCMAXGAIN=G_{max} AGCMINGAIN=G_{min} AGCMAXGAINUPLINK=G_{max} AGCMINGAINUPLINK=G_{min} AGCMAXGAINDOWNLINK=G_{max} AGCMINGAINDOWNLINK=G_{min}
Value Range	G_{max}: $1 \le G_{max} \le 255$, with the default value of 255. When G_{max}=255, the maximum AGC gain is 24dB. This parameter is used to control the maximum gain increase of small signals. The greater the value is, the easier the level of small signals is increased; however, the background noise gets larger at the same time. G_{min}: $1 \le G_{min} \le 255$, with the default value of 6. When G_{min}=6, the minimum AGC gain is -8.5dB. This parameter is used to control the maximum gain decrease of large signals. This parameter should be equal to or greater than 16 if the AGC controller is required to provide gain increase only.
Description	Parameters to control the range of DSP gain increase and decrease
Note	● The default setting of AGC is 16, which indicates the gain of 0dB (neither increase nor decrease). When the output signal level is lower than the expected level, the AGC value gets greater; otherwise it drops. The actual level gain of AGC is 20log (gain value/16) dB. This configuration item is an advanced one. Our default AGC settings are suitable for most application environments. If you are required to reset the parameters for AGC in particular cases, please do it under the instruction of our technicians. ● As the special AGC configuration items for DST-24B/PCI, DST-24B/PCI+, DST-24B/PCIe, DST-24B/PCIe+ boards, the configuration items AGCMAXGAINUPLINK, AGCMINGAINUPLINK, AGCMAXGAINDOWNLINK and AGCMINGAINDOWNLINK are used to set the uplink and downlink AGC parameters.

3.1.2.12.7.7 AGCMAXLEVEL

Refer to AGCMINLEVELDOWNLINK

3.1.2.12.7.8 AGCMINLEVEL

Refer to AGCMINLEVELDOWNLINK

3.1.2.12.7.9 AGCMAXLEVELUPLINK

Refer to AGCMINLEVELDOWNLINK

3.1.2.12.7.10 AGCMINLEVELUPLINK

Refer to AGCMINLEVELDOWNLINK

3.1.2.12.7.11 AGCMAXLEVELDOWNLINK

Refer to AGCMINLEVELDOWNLINK

3.1.2.12.7.12 AGCMINLEVELDOWNLINK

Configuration Item	**AGCMAXLEVEL** **AGCMINLEVEL** **AGCMAXLEVELUPLINK** **AGCMINLEVELUPLINK** **AGCMAXLEVELDOWNLINK** **AGCMINLEVELDOWNLINK**
Section	**[BoardId=x]**
Written Format	AGCMAXLEVEL=L_{max} AGCMINLEVEL=L_{min} AGCMAXLEVELUPLINKL=L_{max} AGCMINLEVELUPLINK=L_{min} AGCMAXLEVELDOWNLINKL=L_{max} AGCMINLEVELDOWNLINK=L_{min}
Value Range	L_{max}: AGCMINLEVEL≤L_{max}≤25600000, with the default value of 3200000. The maximum output voltage level. If the actual output voltage level is greater than L_{max}, the AGC module starts reducing the gain and doesn't stop until the minimum gain set by the configuration item AGCMINGAIN is reached. L_{min}: 0≤L_{min}≤AGCMAXLEVEL, with the default value of 2560000. The minimum output voltage level. If the actual output voltage level is less than L_{min} and keeps for a certain period of time, the AGC module starts increasing the gain and doesn't stop until the maximum gain set by the configuration item AGCMAXGAIN is reached.
Description	Parameters to control the expected value of the AGC output voltage level
Note	● AGC adjusts the gain to control the output signals within a specified range. If the output voltage level is expected to be as stable as possible, you may set the two parameters to be nearly equal but not completely equal; otherwise, the frequent adjustment of gain will lead to the fluctuation of the AGC output voltage level even if the input voltage level is quite stable. The full amplitude of the voltage level is 25600000. If the full-amplitude level is assumed to be 0dB, the level value should be set to 10log (set value/25600000) dB, with the default values 3200000 and 2560000 respectivley corresponding to -9dB and -10dB. This configuration item is an advanced one. Our default settings are suitable for most application environments. If you are required to reset this item, please do it under the instruction of our technicians. ● As the special AGC configuration items for DST-24B/PCI, DST-24B/PCI+, DST-24B/PCIe, DST-24B/PCIe+ boards, the configuration items AGCMAXLEVELUPLINK, AGCMINLEVELUPLINK, AGCMAXLEVELDOWNLINK and AGCMINLEVELDOWNLINK are used to set the uplink and downlink AGC parameters.

3.1.2.12.7.13 AGCDOWNRATIO

Refer to AGCKEEPTIME

3.1.2.12.7.14 AGCUPRATIO

Refer to AGCKEEPTIME

3.1.2.12.7.15 AGCDOWNRATIOUPLINK

Refer to AGCKEEPTIME

3.1.2.12.7.16 AGCUPRATIOUPLINK

Refer to AGCKEEPTIME

3.1.2.12.7.17 AGCDOWNRATIODOWNLINK

Refer to AGCKEEPTIME

3.1.2.12.7.18 AGCUPRATIODOWNLINK

Refer to AGCKEEPTIME

3.1.2.12.7.19 AGCKEEPTIME

Configuration Item	**AGCDOWNRATIO** **AGCUPRATIO** **AGCDOWNRATIOUPLINK** **AGCUPRATIOUPLINK** **AGCDOWNRATIODOWNLINK** **AGCUPRATIODOWNLINK** **AGCKEEPTIME**
Section	**[BoardId=x]**
Written Format	AGCDOWNRATIO=R_{down} AGCUPRATIO=R_{up} AGCDOWNRATIOUPLINK=R_{down} AGCUPRATIOUPLINK=R_{up} AGCDOWNRATIODOWNLINK=R_{down} AGCUPRATIODOWNLINK=R_{up} AGCKEEPTIME=t
Value Range	R_{down}: $0 \le R_{down} \le 100$, with the default value of 35. The step size of automatic gain decrease (5 denotes 5%), used to control the speed of gain decrease. Usually it should be set to a high speed so as to avoid signal chopped distortion. R_{up}: $0 \le R_{up} \le 100$, with the default value of 1. The step size of automatic gain increase (5 denotes 5%), used to control the speed of gain increase. Usually it should be set to a low speed so as to eliminate the background noise for output signals. However, if the speed is set too low, small signals cannot be amplified in time. t: $0 \le t \le 32768$, with the default value of 0. Duration, each unit is 8ms. There may keep a periodof time before the gain turns from decrease to increse. Duration=(set value+1)*8ms. The setting of this parameter can help eliminate the backgroud noise for voice interval, but will delaly the gain increase of small signals. It is not necessary for gain decrease.
Description	Parameters to control the speed of AGC gain decrease/increase

Note	● The speed of gain decrease/increase set by DOWN_TIMEand UP_TIME is 20log(1/(1-set value/100))dB/16ms. The defualt decrease speed is 3.7dB/16ms. That is, the gain can decrease from the maximum value to 0dB in approximately 100ms. The defualt increase speed is 0. 087dB/16ms. That is, the gain can increase from 0dB to the maximum value in about 4 seconds. This configuration item is an advanced one. Our default settings are suitable for most application environments. If you are required to reset this item, please do it under the instruction of our technicians. ● As the special AGC configuration items for DST-24B/PCI, DST-24B/PCI+, DST-24B/PCIe, DST-24B/PCIe+ boards, the configuration items AGCDOWNRATIOUPLINK, AGCUPRATIOUPLINK, AGCDOWNRATIODOWNLINK and AGCUPRATIODOWNLINK are used to set the uplink and downlink AGC parameters

3.1.2.12.7.20 OpenRecEnAndPlayEnOnIdle

Configuration Item	**OpenRecEnAndPlayEnOnIdle**
Section	**[BoardId=x]**
Written Format	Format 1:OpenRecEnAndPlayEnOnIdle=b,b,b,b Format 2:OpenRecEnAndPlayEnOnIdle=b,b,b,b,b,b,b,b Format 3:OpenRecEnAndPlayEnOnIdle=b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b
Value Range	b=0: No (default); b=1: Yes
Description	Sets whether to enable the AGC feature during the recording operation when the analog trunk channel is in the idle state
Note	● This configuration item is only applicable to SHT Series; ● This configuration item requires SynCTI Ver. 4.7.2.0 or above

3.1.2.12.7.21 AutoRecAgcSwitch

Configuration Item	**AutoRecAgcSwitch**
Section	**[BoardId=x]**
Written Format	AutoRecAgcSwitch=a
Value Range	a=0: disabled (default); a>0: for DST series B-type digital station tap boards, Bit0, Bit1, Bit2, Bit3 and Bit4 of a are valid: Bit0=0, the AGC downlink is disabled; Bit0=1, the AGC downlink is enabled; Bit1=0, the AGC uplink is disabled; Bit1=1, the AGC uplink is enabled; Bit2=0 and Bit3=0, DC-isolated is disabled; Bit2=1 and Bit3=1, DC-isolated is enabled; Bit4=1, the stereo is enabled; Bit4=0, the stereo is disabled. For other boards, Bit0 is valid. Bit0=0, the AGC is disabled; Bit0=1, the AGC is enabled.
Description	Sets whether to automatically enable the AGC feature when the channel is being recorded.
Note	● This configuration item requires SynCTI Ver5.3.1.1 or above. ● For SHT Series, if the AGC feature should always be enabled, besides using this configuration item you should set the configuration item OpenRecEnAndPlayEnOnIdle to 1.

3.1.2.12.8Setting Other Parameters

3.1.2.12.8.1 RecEliDTDurTalking

Configuration Item	**RecEliDTDurTalking**
Section	**[BoardId=x]**
Written Format	RecEliDTDurTalking =b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b
Value Range	b=0: No (default); b=1: Yes.

Description	Determines whether to clamp DTMF digits from the call during the recording process.
Note	This configuration item is only applicable to the SHT-16C/PCI/EC board.

3.1.2.12.9 Setting Bit Rate for MP3 Recording

3.1.2.12.9.1 Mp3EncodingBitRate

Configuration Item	**Mp3EncodingBitRate**
Section	**[SystemConfig]**
Written Format	Mp3EncodingBitRate=b
Value Range	b=1: MP3 recording at the bit rate of 8Kbps(default); b=2: MP3 recording at the bit rate of 16Kbps; b=3: MP3 recording at the bit rate of 32Kbps.
Description	Sets the bit rate for MP3 recording.

3.1.2.12.10 Setting Bit Rate for PCM16 Recording

3.1.2.12.10.1 PCM16_16K

Configuration Item	**PCM16_16K**
Section	**[BoardId=x]**
Written Format	PCM16_16K=b
Value Range	b=0: PCM16 recording at the bit rate of 8Kbps (default); b=1: PCM16 recording at the bit rate of 16Kbps.
Description	Sets the bit rate for PCM16 recording.
Note	This configuration item is only applicable to SynHMP.

3.1.2.13 Setting Parameters Required for Both Recording and Playing Operations

3.1.2.13.1 Setting Minimum Size of Pingpong Buffer Area

3.1.2.13.1.1 RecordAndPlayUseAsIP

Configuration Item	**RecordAndPlayUseAsIP**
Section	**[SystemConfig]**
Written Format	RecordAndPlayUseAsIP=b
Value Range	b=0: No (default); b=1: Yes
Description	Sets whether the recording/playing operation supports the mini-buffer. For more information, refer to SsmPlayMemBlock and SsmRecordMemBlock

3.1.2.13.2 Setting Playing Type in Pingpong Buffer Mode

3.1.2.13.2.1 BlockPlayType

Configuration Item	**BlockPlayType**
Section	**[SystemConfig]**

Written Format	BlockPlayType=b
Value Range	b=0: Starvation Mode (default); b=1: Non-starvation Mode
Description	Sets whether to select Starvation Mode for voice playing in the Pingpong Buffer Mode. Starvation Mode: The driver will invoke the callback function repeatedly according to the timer until the internal buffer of the driver is filled full with the data submitted by the application. Non-starvation Mode: The driver will not invoke the callback function until the data submitted by the application are all played out.

3.1.2.13.3Setting Software-based GSM Format

3.1.2.13.3.1 GsmCodecEnable

Configuration Item	**GsmCodecEnable**
Section	**[SystemConfig]**
Written Format	GsmCodecEnable=b
Value Range	b=0: No (default); b=1: Yes
Description	It decides whether the driver will automatically load macmcvt.dll which is the ACM component provided by the SynCTI driver for audio encoding and decoding. This component enables software-based GSM/Mp3 encoding/decoding and G.729A decoding. If the application needs to implement the GSM, Mp3 or G.729A formatted recording/playing on Synway boards which don't have corresponding hardware encoders/decoders, the SynCTI driver can acquire the capability of software encoding and decoding by configuring this item to load the external ACM program.
Note	● If b is set to be 1, it must be ensured that the corresponding audio encoder/decoder (ACM) has been installed in the Windows system; ● This configuration item is only applicable to the Windows operating system. In the Linux operating system, only hardware-based GSM, MP3 and G.729A encodings are supported. For detailed information about the setting of the hardware-based encoding, refer to the configuration item DspCoder; ● When BusPlayListen=0, GsmCodecEnable=0 will be invalid and the driver will load the record/playback software engine to support the monitoring in compressed format.

3.1.2.14 Setting DTMF Detector

3.1.2.14.1Starting/Stopping DTMF Detector

3.1.2.14.1.1 AlwaysEnableRxDtmf

Configuration Item	**AlwaysEnableRxDtmf**
Section	**[BoardId=x]**
Written Format	AlwaysEnableRxDtmf=m
Value Range	m=0: The operating status of the DTMF Detector is always automatically controlled by the driver according to the channel state transition status. When the channel enters the 'Talking' state, it is started automatically; When the channel exits the 'Talking' state, it is stopped automatically. But the application can control the DTMF Detector by the function SsmEnableRxDtmf. m=1: DTMF Detector is always started
Description	Sets the operating status of the DTMF Detector
Note	● This configuration item is applicable to SHD and DTP Series; ● In case m=0 and in the 'Ringing' state, DTMF can be received in ISDN but not in ISUP or TUP.

3.1.2.14.1.2 RcvDtmfOnIdle

Configuration Item	**RcvDtmfOnIdle**
Section	**[BoardId=x]**
Written Format	Format 1:RcvDtmfOnIdle=b,b,b,b Format 2:RcvDtmfOnIdle=b,b,b,b,b,b,b,b Format 3:RcvDtmfOnIdle=b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b Format 4:RcvDtmfOnIdle=[0,29]:b
Value Range	b=0: Stop the DTMF Detector (default) b=1: Start the DTMF Detector
Description	When the station channel is in the 'Idle' state, sets the operating status of DTMF Detector
Note	● This configuration item only supports the station channel; ● Format 1 is only applicable to the SHT Series USB voice boxes; ● Format 2 is applicable to 8-channel SHT Series; ● Format 3 is applicable to 16-channel SHT Series; ● Format 4 is applicable to the 30-channel channel bank.

3.1.2.14.2 Setting Buffer Size for DTMF Detector

3.1.2.14.2.1 RxDtmfBufSize

Configuration Item	**RxDtmfBufSize**
Section	**[SystemConfig]**
Written Format	RxDtmfBufSize=k
Value Range	k: The size (bytes) of the buffer area, with the default value of 200
Description	Sets the size of the buffer area for the DTMF Detector

3.1.2.14.3 Setting Parameters of DTMF Filter

3.1.2.14.3.1 DualAndAllFreqEnScale

Configuration Item	**DualAndAllFreqEnScale**
Section	**[BoardId=x]**
Written Format	Format 1:DualAndAllFreqEnScale=n,n,n,n Format 2:DualAndAllFreqEnScale=n,n,n,n,n,n,n,n Format 3:DualAndAllFreqEnScale=n,n,n,n,n,n,n,n,n,n,n,n,n,n,n,n Format 4:DualAndAllFreqEnScale=m
Value Range	0≤n (%) ≤58, with the default value of 32; 0≤m(%) ≤58, with the default value of 32
Description	Sets the threshold value in percentage for the rate of DTMF in-band energy to overall energy. For more information about this configuration item, refer to DTMF Filter

Note	● Format 1 is applicable to the SHT Series USB voice boxes and the ATP Series USB recording boxes; ● Format 2 is applicable to 8-channel SHT Series and ATP Series; ● Format 3 is applicable to 16-channel SHT, ATP and DST Series; ● Format 4 is applicable to SHD, SHN and DTP Series. However, the following board models require SynCTI Ver. 4.5.1.7 or above: ✧ SHD-240A-CT/cPCI ✧ SHD-240S-CT/cPCI ✧ SHD-480A-CT/cPCI ✧ SHD-480S-CT/cPCI ● The function SsmSetFlag (with the parameter of F_DualAndAllFreqEnScale) may implement the same features; ● Set this configuration item to 15 especially for 8B, 16B, 8C, 16C series boards tp receive abcd.

3.1.2.14.3.2 HighAndLowFreqEnScale

Configuration Item	**HighAndLowFreqEnScale**
Section	**[BoardId=x]**
Written Format	Format 1:HighAndLowFreqEnScale=n,n,n,n Format 2:HighAndLowFreqEnScale=n,n,n,n,n,n,n,n Format 3:HighAndLowFreqEnScale=n,n,n,n,n,n,n,n,n,n,n,n,n,n,n,n Format 4:HighAndLowFreqEnScale=m
Value Range	0≤n (%) ≤100, with the default value of 10 0≤m (%) ≤100, with the default value of 20
Description	Sets the proportion of the high-frequency DTMF energy to the low-frequency in the DTMF signals. Only if the proportion is greater than the set value of this configuration item will the driver confirm that those detected are DTMF signals.
Note	● Format 1 is applicable to the SHT Series USB voice boxes and the ATP Series USB recording boxes; ● Format 2 is applicable to 8-channel SHT and ATP Series; ● Format 3 is applicable to 16-channel SHT, ATP and DST Series; ● Format 4 is applicable to SHD, SHN and DTP Series; ● The same purpose can be achieved by the function SsmSetFlag (with the parameter F_HighAndLowFreqEnScale).

3.1.2.14.3.3 DtmfEnergy

Configuration Item	**DtmfEnergy**
Section	**[BoardId=x]**
Written Format	Format 1:DtmfEnergy=n,n,n,n Format 2:DtmfEnergy=n,n,n,n,n,n,n,n Format 3:DtmfEnergy=n,n,n,n,n,n,n,n,n,n,n,n,n,n,n,n Format 4:DtmfEnergy=m
Value Range	n≥0, with the default value of 0. For more information, refer to the configuration item MinimumVoiceDetermineEnergy
Description	Sets the absolute value of DTMF in-band energy. Only if the DTMF in-band energy is greater than the set value of this configuration item will the driver confirm that those detected are DTMF signals.
Note	● Format 1 is applicable to the SHT Series USB voice boxes and the ATP Series USB recording boxes; ● Format 2 is applicable to 8-channel SHT and ATP Series; ● Format 3 is applicable to 16-channel SHT, ATP and DST Series; ● Format 4 is applicable to SHD, SHN and DTP Series

3.1.2.14.3.4 DTMFFreqOffset

Configuration Item	**DTMFFreqOffset**
Section	**[BoardId=x]**
Written Format	Format 1: DTMFFreqOffset=n,n,n,n Format 2: DTMFFreqOffset=n,n,n,n,n,n,n,n Format 3: DTMFFreqOffset=n,n,n,n,n,n,n,n,n,n,n,n,n,n,n,n Format 4: DTMFFreqOffset=m
Value Range	SHT Series: 0≤n≤100, calculated by %, with the default value of 16 ATP Series: 0≤m≤30, calculated by %, with the default value of 15 SHD Series: 0≤m≤100, calculated by %, with the default value of 18
Description	Sets the maximum offset error of the DTMF signal (offset error = (detected frequency – standard frequency)*1024 / detected frequency).
Note	● Format 1 is applicable to the SHT Series USB voice boxes; ● Format 2 is applicable to 8-channel SHT Series; ● Format 3 is applicable to 16-channel SHT Series; ● Format 4 is applicable to ATP and SHD Series

3.1.2.14.3.5 DtmfModel

Configuration Item	**DtmfModel**
Section	**[BoardId=x]**
Written Format	Format 1: DtmfModel=n,n,n,n Format 2: DtmfModel=n,n,n,n,n,n,n,n Format 3: DtmfModel=n,n,n,n,n,n,n,n,n,n,n,n,n,n,n,n
Value Range	n=0: No (default); n=1: Yes.
Description	Sets whether to only use the energy within the range of 697HZ~1633HZ as the reference overall energy for DTMF detection.
Note	● Format 1 is applicable to the SHT Series USB voice boxes; ● Format 2 is applicable to 8-channel SHT Series; ● Format 3 is applicable to 16-channel SHT Series.

3.1.2.14.4Setting Parameters of DTMF Pulse-width Filter

3.1.2.14.4.1 ReceiveDtmfSensitive

Configuration Item	**ReceiveDtmfSensitive**
Section	**[BoardId=x]**
Written Format	ReceiveDtmfSensitive=n
Value Range	n is comprised of 8 bits (namely Bit7…Bit0), below is the meaning of each bit: Bit3~Bit0: The minimum duration of the DTMF signal at on state. Range of value is 1~6, the unit is 16ms, with the default value of 3; Bit7~Bit4: The minimum duration of the DTMF signal at off state. Range of value is 0~6, the unit is 16ms, with the default value of 0; Note: If the value of Bit7~Bit4 is 0, the minimum duration of DTMF signal at off state is determined by the minimum duration at on state. If the minimum duration of the DTMF signal at on state is less than 3, the value of the minimum duration of the DTMF signal at off state is 1; otherwise, the value of the minimum duration of the DTMF signal at off state is 2. n is represented by hexadecimal number, e.g. ReceiveDtmfSensitive=0x43. This formula means that the minimum durations of the DTMF signals at on and off states are 48ms and 64ms respectively. ReceiveDtmfSensitive=0x03: the minimum durations of the DTMF signal at on and off states are 48ms and 32ms respectively.

Description	The minimum durations of the DTMF signal at on and off states. For more information, refer to DTMF Pulse-width Filter in Chapter 1
Note	It is applicable to SHD, SHT, ATP and DTP Series. However, the following board models among the SHD Series require SynCTI Ver. 4.5.1.7 or above: ✧ SHD-240A-CT/cPCI ✧ SHD-240S-CT/cPCI ✧ SHD-480A-CT/cPCI ✧ SHD-480S-CT/cPCI

3.1.2.14.4.2 OmitABCD

Configuration Item	**OmitABCD**
Section	**[BoardId=x]**
Written Format	Format 1:OmitABCD=b,b,b,b Format 2:OmitABCD=b,b,b,b,b,b,b,b Format 3:OmitABCD=b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b Format 4:OmitABCD=b
Value Range	b=0: discard; b=1: Not to discard Default value is 0
Description	Sets whether to discard the characters of ‘a’, ‘b’, ‘c’, and ‘d’ received by the DTMF Detector. For more information about this configuration item, refer to DTMF Pulse-width Filter in Chapter 1
Note	● Format 1 is applicable to the SHT Series USB voice boxes and the ATP Series USB recording boxes; ● Format 2 is applicable to 8-channel SHT and ATP Series; ● Format 3 is applicable to 16-channel SHT, ATP and DST Series; ● Format 4 is applicable to SHD, DTP and SHN Series ● When you enable the reception of abcd codes, the value of DualAndAllFreqEnScale is by default adjusted to 15. If the reception fails, you may need to lower the value and try again. ● For SHN Series boards, this configuration item is only valid for DTMF detection using the In-band mode, but not applicable for that using the RFC2833 or Signaling mode.

3.1.2.14.4.3 DtrmOnDtmfHighLevel

Configuration Item	**DtrmOnDtmfHighLevel**
Section	**[BoardId=x]**
Written Format	Format 1:DtrmOnDtmfHighLevel=n,n,n,n Format 2:DtrmOnDtmfHighLevel=n,n,n,n,n,n,n,n Format 3:DtrmOnDtmfHighLevel=n,n,n,n,n,n,n,n,n,n,n,n,n,n,n,n Format 4:DtrmOnDtmfHighLevel=n
Value Range	n=0: Only when the duration of the DTMF signals at off state exceeds the minimum duration designated by the configuration item ReceiveDtmfSensitive will the driver output the DTMF characters; n=1: When the duration of the DTMF signals at on state exceeds the minimum duration designated by the configuration item ReceiveDtmfSensitive will the driver output the DTMF characters immediately; Default value is 0
Description	Sets the timing of outputting the DTMF characters. For more information about this configuration item, refer to DTMF Pulse-width Filter in Chapter 1
Note	● Format 1 is applicable to the SHT Series USB voice boxes and the ATP Series USB recording boxes; ● Format 2 is applicable to 8-channel SHT and ATP Series; ● Format 3 is applicable to 16-channel SHT, ATP and DST Series; ● Format 4 is applicable to SHD, SHN and DTP Series

3.1.2.15 Setting DTMF Generator (CTI Series)

3.1.2.15.1 Setting Size of Transmission Buffer Area

3.1.2.15.1.1 TxDtmfBufSize

Configuration Item	**TxDtmfBufSize**
Section	**[SystemConfig]**
Written Format	TxDtmfBufSize=n
Value Range	n≥1, Default value is 50 characters
Description	Sets the size of the transmission buffer area for the DTMF generator

3.1.2.15.2 Setting Parameters for DTMF Generator

3.1.2.15.2.1 TxDtmfTimePara

Configuration Item	**TxDtmfTimePara**
Section	**[BoardId=x]**
Written Format	Format 1:TxDtmfTimePara={t_H,t_L},{t_H,t_L},{t_H,t_L},{t_H,t_L} Format 2:TxDtmfTimePara={t_H,t_L},{t_H,t_L},{t_H,t_L},{t_H,t_L},{t_H,t_L},{t_H,t_L},{t_H,t_L},{t_H,t_L} Format 3:TxDtmfTimePara={t_H,t_L},{t_H,t_L},{t_H,t_L},{t_H,t_L},{t_H,t_L},{t_H,t_L},{t_H,t_L},{t_H,t_L}, {t_H,t_L},{t_H,t_L},{t_H,t_L},{t_H,t_L},{t_H,t_L},{t_H,t_L},{t_H,t_L},{t_H,t_L} Format 4:TxDtmfTimePara={t_H,t_L}
Value Range	t_H≥40: The duration (ms) of the DTMF signal at on state, it must be integral times of 8ms, with the default value of 80 t_L ≥40: The duration (ms) of the DTMF signal at off state, it must be integral times of 8ms, with the default value of 80
Description	Sets the durations (ms) of DTMF signals generated by the DTMF Generator respectively at off and on states. For more information, refer to DTMF Generator in Chapter 1
Note	● Format 1 is applicable to the SHT Series USB voice boxes; ● Format 2 and Format 3 are applicable to 8-channel and 16-channel SHT Series respectively; ● Format 4 is applicable to SHD, SHN Series

3.1.2.15.2.2 DefaultTxDelayTime

Configuration Item	**DefaultTxDelayTime**
Section	**[SystemConfig]**
Written Format	DefaultTxDelayTime=n
Value Range	n: The duration (ms) of silence, with the default value of 2000
Description	Sets the delay time of the DTMF characters when being transmitted. For more information, refer to SsmTxDtmf

3.1.2.15.2.3 TxDtmfAmp

Configuration Item	**TxDtmfAmp**
Section	**[BoardId=x]**
Written Format	TxDtmfAmp=0xxxxx
Value Range	The default value is 0x3f3f (digital serial board) or 0x8976 (analog serial board)

Description	Sets the amplitude of the transmitted DTMF signal, among which the 8 higher bits set the amplitude of low frequency and the 8 lower bits set the amplitude of high frequency. The feature of this function is the same as that of SsmSetFlag (with the parameter F_TXDTMFAMP)

3.1.2.16 Setting Echo Canceller (CTI Series)

3.1.2.16.1 Setting Operating Status

3.1.2.16.1.1 EnableEchoCancellor

Configuration Item	**EnableEchoCancellor**
Section	**[BoardId=x]**
Written Format	Format 1：EnableEchoCancellor=b,b,b,b Format 2：EbableEchoCancellor=b,b,b,b,b,b,b,b Format 3：EnableEchoCancellor=b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b Format 4：EnableEchoCancellor=b
Value Range	b=0：Disable the echo canceller b=1：Enable the echo canceller (default)
Description	Sets the operating status of the echo canceller. For more information, refer to Echo Canceller In Chapter 1
Note	● Format 1 is applicable to the SHT Series USB voice boxes and the ATP Series USB recording boxes; ● Format 2 is applicable to the 8-channel SHT Series; ● Format 3 is applicable to the 16-channel SHT Series; ● Format 4 is applicable to SHD, SHN Series

3.1.2.16.2 Setting Parameters

3.1.2.16.2.1 EchoCancelDelaySize

Configuration Item	**EchoCancelDelaySize**
Section	**[BoardId=x]**
Written Format	EchoCancelDelaySize=k
Value Range	k：the coefficient of the filter delay. Both the value range and the default value are determined by the configuration item LoadFskBin: LoadFskBin=0：0≤k≤7, default value：6; LoadFskBin>0：0≤k≤31, default value：12
Description	Sets the coefficient of the filter delay of the echo canceller
Note	This configuration item is only applicable to SHD and SHT Series USB voice boxes

3.1.2.17 Setting Barge-in Detector (CTI Series)

3.1.2.17.1 Setting Way to Start Barge-in Detector

3.1.2.17.1.1 AlwaysDetectBargeIn

Configuration Item	**AlwaysDetectBargeIn**
Section	**[BoardId=x]**
Written Format	AlwaysDetectBargeIn=b

Value Range	b=0: Under the automatic control of the driver. When the channel goes into the S_CALL_TALKING state and join the conference as a dynamic speaker, the driver will automatically start the Barge-in detector; when the channel goes out of the S_CALL_TALKING state or any dynamic speaker channel leaves the conference room, the driver will automatically stop the Barge-in detector; b=1: Always being started. The default value is 0.
Description	Sets the way to start the Barge-in Detector.
Note	This configuration item is applicable to SHD, SHN, DTP and SHV Series. For DTP and SHV series boards, AlwaysDetectBargeIn=0 means the Barge in detector is thoroughly disabled and the driver will no longer control the detector automatically. To find more information, refer to 'Barge in Detector' in Chapter 1.

3.1.2.17.2Setting Parameters for Barge-in Detector

3.1.2.17.2.1 VoiceEnergyMinValue

Configuration Item	**VoiceEnergyMinValue**
Section	**[BoardId=x]**
Written Format	VoiceEnergyMinValue=k
Value Range	k: The threshold value for noise detection, with the default value of 100,000.
Description	Sets the threshold value to judge noises for the Barge-in detector.

3.1.2.17.2.2 BargeInSensitive

Configuration Item	**BargeInSensitive**						
Section	**[BoardId=x]**						
Written Format	Format 1: BargeInSensitive=k,k,k,k Format 2: BargeInSensitive=k,k,k,k,k,k,k,k Format 3: BargeInSensitive=k,k,k,k,k,k,k,k,k,k,k,k,k,k,k,k Format 4: BargeInSensitive=k						
Value Range	0≤k≤31, with the default value of 6. The greater the value is, the more sensitive the detector is.						
Description	Sets the sensitivity of the Barge-in detector.						
Note	● Format 1 is applicable to the SHT Series USB voice boxes and the ATP Series USB recording boxes; ● Format 2 is applicable to 8-channel SHT and ATP Series; ● Format 3 is applicable to 16-channel SHT and DST Series; ● Format 4 is applicable to SHV, SHD, SHN and DTP Series; ● See the function SsmSetBargeInSens for details.						
DB Value	Value of BargeInSensitive	0	1	2	3	4	5
	DB Value	0	-1	-2	-2.9	-5.9	-8.9
	Value of BargeInSensitive	6	7	8	9	10	11
	DB Value	-11.9	-14.9	-18	-21	-24	-27
	Value of BargeInSensitive	12	13	14	15	16	17~31
	DB Value	-29.9	-32.9	-35.2	-37.1	-39.5	-48
	Note: Only the DB values for ATP series boards are listed in the table.						

3.1.2.17.2.3 BargeInDtrmTime

Configuration Item	**BargeInDtrmTime**

Section	**[BoardId=x]**
Written Format	Format 1: BargeInDtrmTime=k,k,k,k Format 2: BargeInDtrmTime=k,k,k,k,k,k,k,k Format 3: BargeInDtrmTime=k,k,k,k,k,k,k,k,k,k,k,k,k,k,k,k Format 4: BargeInDtrmTime=k
Value Range	k≥16 and must be the multiple of 16, calculated by millisecond (ms), with the default value of 32.
Description	Sets the minimum signal duration for the Barge-in detector.
Note	● Format 1 is applicable to the SHT Series USB voice boxes and the ATP Series USB recording boxes; ● Format 2 is applicable to 8-channel SHT Series; ● Format 3 is applicable to 16-channel SHT and DST Series; ● Format 4 is applicable to SHD, SHN and DTP Series; ● See the function SsmSetIsBargeInDtrmTime for details.

3.1.2.17.2.4 IsNoSoundDtrmTime

Configuration Item	**IsNoSoundDtrmTime**
Section	**[SystemConfig]**
Written Format	IsNoSoundDtrmTime=k
Value Range	The minimum duration for the line to keep silence, calculated by millisecond (ms). k≥16 and must be the multiple of 16, with the default value of 5000ms.
Description	Sets the minimum duration for the line to keep silence. See the function SsmSetNoSoundDtrmTime for details.
Example	IsNoSoundDtrmTime= 5000

3.1.2.17.2.5 DefaultDtmfIsSound

Configuration Item	**DefaultDtmfIsSound**
Section	**[BoardId=x]**
Written Format	Format 1: DefaultDtmfIsSound=b,b,b,b Format 2: DefaultDtmfIsSound=b,b,b,b,b,b,b,b Format 3: DefaultDtmfIsSound=b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b Format 4: DefaultDtmfIsSound=b
Value Range	b=0: No; b=1: Yes (default).
Description	Sets whether to regard the DTMF signal in the incoming call as the voice signal.
Note	● Format 1 is applicable to the SHT Series USB voice boxes and the ATP Series USB recording boxes; ● Format 2 is applicable to 8-channel SHT and ATP Series; ● Format 3 is applicable to 16-channel SHT and DST Series; ● Format 4 is applicable to SHD, SHN and DTP Series; ● This configuration item requires SynCTI Ver. 4.5.2.9 or above.

3.1.2.18 Setting TDM Connection

3.1.2.18.1InVoiceToBus

Configuration Item	**InVoiceToBus**
Section	**[BoardId=x]**
Written Format	Format 1: InVoiceToBus=b,b,b,b Format 2: InVoiceToBus=b,b,b,b,b,b,b,b Format 3: InVoiceToBus=b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b Format 4: InVoiceToBus=b

Value Range	b=0: No; b=1: Yes (default).
Description	Determines whether to connect incoming signals to the TDM bus upon the system initialization. The function SsmSetFlag (with the parameter F_InVoiceToBus) can be used for the same purpose.
Note	● Format 1 is applicable to the SHT Series USB voice boxes and the ATP Series USB recording boxes; ● Format 2 is applicable to 8-channel SHT and 8-channel ATP Series; ● Format 3 is applicable to 16-channel SHT and 16-channel DST Series; ● Format 4 is applicable to SHD, DTP, SHN and 24-channel DST Series; ● This configuration item requires SynCTI Ver. 4.0 or above.

3.1.2.18.2LinkFromStopPlayAndTone

Configuration Item	**LinkFromStopPlayAndTone**
Section	**[BoardId=x]**
Written Format	LinkFromStopPlayAndTone=b
Value Range	b=1: Immediately terminates the task of voice playing or the task performed by the tone generator (default); b=0: Waits for the task of voice playing to go into an end and then executes the off-bus command.
Description	See the Operation Principle of SHD Series section in Chapter 1 for details.
Note	● This configuration item is only applicable to the 30/60/120-channel SHD Series; ● It requires SynCTI Ver. 4.5.2.9 or above. ● If the configuration item UsageMode is set to 1, this configuration item is invalid and uses the default value.

3.1.2.18.3DefaultSpeakVolume

Configuration Item	**DefaultSpeakVolume**
Section	**[BoardId=x]**
Written Format	Format 1: DefaultSpeakVolume=k,k,k,k Format 2: DefaultSpeakVolume=k,k,k,k,k,k,k,k Format 3: DefaultSpeakVolume=k,k,k,k,k,k,k,k,k,k,k,k,k,k,k,k Format 4: DefaultSpeakVolume=k
Value Range	-7≤k≤6, k×3 equals the decibel value (db), with the default value of 0. If k is greater than 0, the volume is increased; if it is less than 0, the volume is decreased.
Description	Sets the volume of this channel's voice signals before they pass the TDM bus and go down to the off-bus mixer M2 of another channel. See sections about the board operation principle in Chapter 1 for details.
Note	● Format 1 is applicable to the SHT Series USB voice boxes and the ATP Series USB recording boxes; ● Format 2 is applicable to 8-channel SHT and ATP Series; ● Format 3 is applicable to 16-channel SHT, ATP and DST Series. ● Format 4 is applicable to SHD and SHN series.

3.1.2.18.4EnableCommonTimeSlot

Configuration Item	**EnableCommonTimeSlot**
Section	**[SystemConfig]**
Written Format	EnableCommonTimeSlot=b

Value Range	b=0: Not open (default); b=1: Open all the time slots on the TDM bus.
Description	Opens all the common time slots on the TDM bus
Note	● This configuration item is only applicable to the situation where Synway boards are interoperated with third party boards; ● It requires SynCTI Ver. 5.2.0.0 or above versions.

3.1.2.19 Setting Caller ID Detector for Analog Phone Line (SHT＋ATP Series)

3.1.2.19.1 Setting Operating Mode of Caller ID Detector

3.1.2.19.1.1 CallerIdStyle

Configuration Item	**CallerIdStyle**
Section	**[BoardId=x]**
Written Format	Format 1: CallerIdStyle=m,m Format 2: CallerIdStyle=m,m,m,m Format 3: CallerIdStyle=m,m,m,m,m,m,m,m Format 4: CallerIdStyle=m,m,m,m,m,m,m,m,m,m,m,m,m,m,m,m
Value Range	m=0: DTMF Mode. The configuration items DtmfCallerIDStyleLength and DtmfCallerIDInterTimeout are effective. m=1: FSK Mode. The configuration items CloseCallerIdOnReceived and FSKCallerIdDtrmTime are effective. The default value is 1.
Description	Sets the mode for the remote PBX to send the calling party number on the analog phone line. Refer to the Caller ID on Analog Phone Line section in Chapter 1 for details.
Note	● This configuration item is only applicable to the analog trunk channel on the SHT Series boards; ● Format 1 and Format 2 are respectively applicable to 2-/4-channel USB voice boxes and USB recording boxes; Format 3 is applicable to 8-channel boards; Format 4 is applicable to 16-channel boards; ● The same purpose can be achieved via the function call of SsmSetFlag (with the parameter F_CALLERIDTYPE); ● It is invalid to the recording channel on ATP-24A series boards.

3.1.2.19.2 Setting Parameters for FSK Mode

3.1.2.19.2.1 CloseCallerIdOnReceived

Configuration Item	**CloseCallerIdOnReceived**
Section	**[SystemConfig]**
Written Format	CloseCallerIdOnReceived=b
Value Range	b=0: Enabled; b=1: Disabled (default).
Description	This configuration item is set to determine whether the Caller ID detector will be automatically disabled by the driver after it receives the complete calling party number in the FSK mode. Refer to the Caller ID on Analog Phone Line section in Chapter 1 for more information.
Note	This configuration item is only applicable to SHT and ATP Series.

3.1.2.19.2.2 FSKCallerIdDtrmTime

Configuration Item	**FSKCallerIdDtrmTime**

Section	**[BoardId=x]**
Written Format	Format 1: FSKCallerIdDtrmTime=k,k Format 2: FSKCallerIdDtrmTime=k,k,k,k Format 3: FSKCallerIdDtrmTime=k,k,k,k,k,k,k,k Format 4: FSKCallerIdDtrmTime=k,k,k,k,k,k,k,k,k,k,k,k,k,k,k,k
Value Range	k: The time to judge the completion, calculated by millisecond (ms),100≤k≤500, with the default value of 500ms.
Description	When the Caller ID detector is started in the FSK mode, since there is no clear symbol to tell the completion in transmitting the FSK calling party number, the driver has to depend on the relativity between time and data to judge the completion of the process. That is, if the received Caller ID character keeps unchanged in length for the period of time set by this parameter, the driver will consider the signal transmission of the FSK Caller ID to be ended.
Note	● This configuration item is only applicable to the analog trunk channel on the SHT Series boards and the recording channel on the ATP Series boards; ● Format 1 and Format 2 are respectively applicable to 2-/4-channel USB voice boxes and USB recording boxes; Format 3 is applicable to 8-channel boards; Format 4 is applicable to 16-channel boards;

3.1.2.19.2.3 FilterInvalidCID

Configuration Item	**FilterInvalidCID**
Section	**[BoardId=x]**
Written Format	Format 1: FilterInvalidCID=b,b,b,b,b,b,b,b Format 2: FilterInvalidCID=b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b
Value Range	b=0: No; b=1: Yes (default).
Description	Sets whether to enable the 'Filter Out Wrong FSK Caller ID' feature. The FSK calling party number received by the driver may be incorrect: ✧ The number is a false one formed by echoes in the sound, not sent by the remote PBX; ✧ The number is exactly sent by the remote PBX, but becomes incomplete because of signals being damaged by disturbance or other elements during the transmission.
Note	● Format 1 and Format 2 are respectively applicable to 8-channel and 16-channel SHT/ATP Series; ● This configuration item requires SynCTI Ver. 4.7.1.8 or above.

3.1.2.19.2.4 FSKMinGate

Configuration Item	**FSKMinGate**
Section	**[BoardId=x]**
Written Format	Format 1: FSKMinGate =b,b,b,b,b,b,b,b Format 2: FSKMinGate =b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b Format 3: FSKMinGate =b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b Format 4: FSKMinGate =b
Value Range	b=n (n>0): The energy threshold value; b=150: The default value (SHT/ATP Series); b=60: The default value (SHD Series).
Description	Sets the energy threshold value for FSK reception.
Note	● Format 1, Format 2 and Format 3 are respectively applicable to 8-channel, 16-channel and 24-channel SHT/ATP Series. ● Format 4 is applicable to SHD Series. ● This configuration item requires SynCTI Ver. 5.0.2.0 or above; however, Format 4 requires SynCTI Ver. 5.3.1.3 or above.

3.1.2.19.2.5 FskNoPhase

Configuration Item	**FskNoPhase**
Section	**[BoardId=x]**

Written Format	FskNoPhase=m
Value Range	m=0: The syn code of the 0/1 up-and-down waveform followed by the flag code composed of all 1s (default); m=1: The syn code of 0804 followed by the flag code composed of all 1s.
Description	Sets the format of the syn code in the FSK calling party number.
Note	This configuration item is only applicable to the analog trunk channel on the SHT Series boards and the recording channel on the ATP Series boards.

3.1.2.19.3Setting Parameters for DTMF Mode

3.1.2.19.3.1 DtmfCallerIDStyleLength

Refer to **DtmfCallerIDInterTimeOut**

3.1.2.19.3.2 DtmfCallerIDInterTimeOut

Configuration Item	**DtmfCallerIDStyleLength** **DtmfCallerIDInterTimeOut**
Section	**[SystemConfig]**
Written Format	DtmfCallerIDStyleLength=n DtmfCallerIDInterTimeOut=t
Value Range	n: The minimum number of DTMF digits, 1≤n≤29, with the default value of 1. t: The time interval, calculated by millisecond (ms), t≥300, with the default value of 500.
Description	When the Caller ID detector works in the DTMF mode, if the number of DTMF digits received by the driver is less than or equal to n, all received digits will be abandoned; if the time interval between the receptions of two DTMF digits is greater than t, all digits previously received will be abandoned.
Note	● DtmfCallerIDInterTimeOut requires SynCTI Ver. 4.5.2.1 or above; ● This configuration item is only applicable to the SHT and ATP Series boards.

3.1.2.20 Setting Ringing Current Signal Detector (SHT+ATP Series)

3.1.2.20.1RingDetectFilterPara

Configuration Item	**RingDetectFilterPara**
Section	**[SystemConfig]**
Written Format	RingDetectFilterPara=t_H, t_L
Value Range	t_H: The minimum duration of signals at on state. t_H≥1 and is calculated by the multiple of 8ms, with the default value of 6 (i.e. 48ms). Only if the signal stays at on state for t_H or longer than t_H will the driver confirm the detection of a ringing current signal. The greater t_H is, the less sensitive the ringing current detector is. t_L: The minimum duration of signals at off state. t_L≥1 and is calculated by the multiple of 8ms, with the default value of 40 (i.e. 320ms). Only if the signal stays at off state for t_L or longer than t_L will the driver confirm the disappearance of ringing current signals.
Description	Sets the filter paramters for the ringing current detector, that is, the parameters used to detect rings on the analog trunk channel and the magnet channel. The driver takes a sample of ringing signals on the analog trunk channel and the magnet channel every 8 ms. A ringing signal is regarded gone when the count of signals at off state goes equal to or greater than the value for judging signals at off state (the 2nd parameter); a ringing signal is confirmed detected when the count of signals at on state goes equal to or greater than the value for judging signals at on state (the 1st parameter). Increasing the value for judging signals at on state will lead to the decrease in the ringing detection's sensitivity; vice versa.

Note	This configuration item is only applicable to the analog trunk channel and the magnet channel on the SHT Series boards as well as the analog recording channel on the ATP Series boards.

3.1.2.20.2CallBackRingDetectFilterPara

Configuration Item	**CallBackRingDetectFilterPara**
Section	**[SystemConfig]**
Written Format	CallBackRingDetectFilterPara=t_H, t_L
Value Range	t_H: The maximum duration of signals at on state. t_H≥1 and is calculated by the multiple of 8ms, with the default value of 25 (i.e. 200ms). t_L: The maximum duration of signals at off state. t_L≥1 and is calculated by the multiple of 8ms, with the default value of 40 (i.e. 320ms).
Description	Sets the filter parameters for the ringback current detector, that is the parameters used to detect rings on the analog trunk channel and the magnet channel. The driver takes a sample of ringing signals on the analog trunk channel and the magnet channel every 8ms. A signal, when the count of it at on and off states is consistent with the setting of RingDetectFilterPara, with the duration at on state less than t_H and the duration at off state less than t_L, is judged as a ringback signal. This feature can be used to distinguish two different ringing signals. At the same time, the driver will throw out the events E_CHG_RingCount and E_CHG_CallBackRingCount.
Note	● This configuration item is applicable to the analog trunk channel and the magnet channel on SHT series boards as well as the analog recording channel on ATP series boards; ● RingDetectFilterPara is used to configure the parameters of 'Ring'; while CallBackRingDetectFilterPara is used to configure the parameters of 'Callback Ring'. 'Callback Ring' is a particular ring which satisfies the condition of 'the duration of RingDetectFilterPara at on state< the duration of voice at on state< the duration of CallBackRingDetectFilterPara at on state, and the duration of RingDetectFilterPara at off state< the duration of voice at off state< the duration of CallBackRingDetectFilterPara at off state'..

3.1.2.20.3RingEndDtrTime

Configuration Item	**RingEndDtrTime**
Section	**[SystemConfig]**
Written Format	RingEndDtrTime=k
Value Range	k>0: The maximum waiting time (ms) for the ringing current signal to be terminated. If the ringing current does not terminate within the maximum waiting time, it will be recounted from 0. The default value of k is 6000(ms).
Description	This configuration item is used to set the maximum durations of the ringing current at the off state which is 'tL+RingEndDtrTime'. 'tL' is the minimum duration at the off state, which can be obtained via the configuration item RingDetectFilterPara. The ringing current can be detected only when the condition 'tL ≦ duration at the off state ≦ tL+RingEndDtrTime' is satisfied. When the period of a ringing current signal lasts too long, the value of k should be increased to prevent the lost of any ringing current signal.
Note	This configuration item is only applicable to the analog trunk channel and the magnet channel on the SHT Series boards as well as the analog recording channel on the ATP Series boards.

3.1.2.20.4AlwaysToRingingOnRingCntX

Configuration Item	**AlwaysToRingingOnRingCntX**
Section	**[SystemConfig]**
Written Format	AlwaysToRingingOnRingCntX=k
Value Range	k=0: The channel state transfers to 'Ringing' when the driver detects two (the Caller ID detector is set to work in the DTMF or FSK mode) complete ringing current signals; k≥1: No matter the Caller ID detector is working in which mode, the channel state will transfer to 'Ringing' once the driver detects k complete ringing current signals. The default value is 0.
Description	Sets the parameters for the ringing current detector. Refer to the Ringing Current on Analog Phone Line section in Chapter 1 for more information.
Note	● This configuration item may be covered by the configuration item ChToRingingOnRingCnt. Refer to the Ringing Current on Analog Phone Line section in Chapter 1 for more information; ● The parameters set by this configuration item can also be configured by the function SsmSetFlag (with the parameter F_ChToRingingOnRingCnt); ● This configuration item is only applicable to the analog trunk channel on the SHT Series boards.

3.1.2.20.5ChToRingingOnRingCnt

Configuration Item	**ChToRingingOnRingCnt**
Section	**[BoardId=x]**
Written Format	Format 1: ChToRingingOnRingCnt=n,n,n,n Format 2: ChToRingingOnRingCnt=n,n,n,n,n,n,n,n Format 3: ChToRingingOnRingCnt=n,n,n,n,n,n,n,n,n,n,n,n,n,n,n,n
Value Range	n=0: The channel state transfers to 'Ringing' when the driver detects two (the Caller ID detector is set to work in the DTMF or FSK mode) complete ringing current signals; n≥1: No matter the Caller ID detector is working in which mode, the channel state will transfer to 'Ringing' once the driver detects n complete ringing current signals. The default value is 0.
Description	Sets the parameters for the ringing current detector. Refer to the Ringing Current on Analog Phone Line section in Chapter 1 for more information.
Note	● This configuration item is only applicable to the analog trunk channel on the SHT Series boards; ● If this configuration item is specified for certain channel and n>0, the parameters of the configuration item AlwaysToRingingOnRingCntX for this channel will be covered. Refer to the Ringing Current on Analog Phone Line section in Chapter 1 for more information. ● The parameters set by this configuration item can also be configured by the function SsmSetFlag (with the parameter F_ChToRingingOnRingCnt); ● Format 1 is applicable to USB voice boxes with analog trunk; ● Format 2 is applicable to 8-channel analog trunk boards; ● Format 3 is applicable to 16-channel analog trunk boards.

3.1.2.21 Setting FSK Transceiver (CTI Series)

3.1.2.21.1 Setting Parameters for FSK Transceiver

3.1.2.21.1.1 FreqBit0

Refer to MdlAmp

3.1.2.21.1.2 FreqBit1

Refer to MdlAmp

3.1.2.21.1.3 Baudrate

Refer to MdlAmp

3.1.2.21.1.4 MdlAmp

Configuration Item	**FreqBit0** **FreqBit1** **Baudrate** **MdlAmp**
Section	**[FskConfig]**
Written Format	FreqBit0=f_0 FreqBit1=f_1 Baudrate=k MdlAmp=a
Value Range	f_0: The carrier frequency of Bit 0 (Hz), 300≤f_0≤3400, with the default value of 2200Hz. f_1: The carrier frequency of Bit 1 (Hz), 300≤f_1≤3400, with the default value of 1200Hz. k: The baud rate (bps), 300≤k≤2400, with the default value of 1200bps. a: The signal amplitude, 0≤n≤255, with the default value of 128.
Description	FreqBit0 and FreqBit1 are respectively used to set the carrier frequency of Bit 0 and Bit 1; Baudrate sets the baud rate while MdlAmp configures the modulating-signal amplitude. Regarding the FSK Transceiver, see the FSK Transceiver section in Chapter 1 for more information.

3.1.2.22 Setting FSK Transceiver (CTI Series)

3.1.2.22.1 FskMarkSignal

Configuration Item	**FskMarkSignal**
Section	**[BoardId=x]**
Written Format	Format 1: FskMarkSignal=m_0,m_1,m_2,m_3,m_4,m_5,m_6,m_7 Format 2: FskMarkSignal=m_0,m_1,m_2,m_3,m_4,m_5,m_6,m_7,m_8,m_9,m_{10},m_{11},m_{12},m_{13},m_{14},m_{15} Format 3: FskMarkSignal=m
Value Range	m, m_i: The number of digits in the identification code (The subscript i represents the physical channel number). =0: The identification code is composed of twelve 1s; =1: The identification code is composed of eight 1s.
Description	Sets the format of the identification code in the FSK data stream.

Note	● Format 1, Format 2 are respectively applicable to 8-channel and 16-channel SHT Series; ● Format 3 is applicable to the SHD Series boards.

3.1.2.22.2FskFrameMode

Configuration Item	**FskFrameMode**
Section	**[BoardId=x]**
Written Format	FskFrameMode=m
Value Range	m=0: Use the format 'flag code+data frame'; m=1: Use the format 'Sync code+flag code+data frame' (default). For more information, refer to Section FSK Transceiver in Chapter 1.
Description	Sets the frame format mode that accepted by the FSK receiver.
Note	● This configuration item is applicable to all SHT Series and SHD Series boards.

3.1.2.22.3FskEchoCancelDelay

Configuration Item	**FskEchoCancelDelay**
Section	**[BoardId=x]**
Written Format	Format 1: FskEchoCancelDelay=b_0,b_1,b_2,b_3,b_4,b_5,b_6,b_7,b_8,b_9,b_{10},b_{11},b_{12},b_{13},b_{14},b_{15} Format 2: FskEchoCancelDelay=b_0,b_1,b_2,b_3,b_4,b_5,b_6,b_7
Value Range	b_i=0: Disable; b_i=1: Enable (default). Note: The subscript i represents the physical channel number
Description	Sets whether to disable the echo canceller when the FSK transceiver is working.
Note	● Format1, Format2 are respectively applicable to 8-channel and 16-channel SHT Series; ● When the FSK transceiver is enabled on the channel, if this configuration item is set to 1, the echo canceller will be disabled compulsorily; otherwise, the status of the echo canceller is determined by the configuration item EnableEchoCancellor; ● This configuration item requires SynCTI Ver4.5.6.2 or above; ● The same purpose can be achieved by the function SsmSetFlag (with the parameter F_EchoCancelInFsk).

3.1.2.23 Setting Parameters for Fax Channel

3.1.2.23.1MaxSpeed

Configuration Item	**MaxSpeed**
Section	**[Fax]**
Written Format	MaxSpeed=k
Value Range	k: The faxing rate (bps). It can be set to 4800, 9600, 14400 or 33600. The default value is 9600.
Description	Sets the maximum faxing rate used in receiving or transmitting faxes.

3.1.2.23.2RcvDisTime

Configuration Item	**RcvDisTime**
Section	**[Fax]**
Written Format	RcvDisTime=n

Value Range	0≤n≤10, with the default value of 4.
Description	This configuration item is used to set the times to repeat the Dis detection in case the Dis signal is undetected or invalid during the faxing process. It is supported by SynCTI Ver. 4.7.2.0.
Note	This configuration item requires SynCTI Ver. 4.7.2.0 or above.

3.1.2.23.3ResendForRTN

Configuration Item	**ResendForRTN**
Section	**[Fax]**
Written Format	ResendForRTN=k
Value Range	k: 0 not resend the page data upon the reception of an RTN message (default); k:1 resend the page data upon the reception of an RTN message.
Description	Sets whether to resend the page data upon the reception of an RTN message.

3.1.2.23.4ResetRcvForRTN

Configuration Item	**ResetRcvForRTN**
Section	**[Fax]**
Written Format	ResetRcvForRTN =k
Value Range	k: 0 send an RTN message upon the reception of pages with incomplete and/or incorrect fax data (default); k: 1 hang up the phone directly instead of sending an RTN message upon the reception of pages with incomplete and/or incorrect fax data.
Description	Determines whether to send an RTN message or hang up the phone immediately upon the reception of pages with incomplete and/or incorrect fax data.

3.1.2.23.5KeepPageForRTN

Configuration Item	**KeepPageForRTN**
Section	**[Fax]**
Written Format	KeepPageForRTN =k
Value Range	k: 0 not keep the received pages with incomplete and/or incorrect fax data; k: 1 keep the received pages with incomplete and/or incorrect fax data (default).
Description	Dedermines whether to keep the received pages with incomplete and/or incorrect fax data.

3.1.2.23.6PercentageForRTN

Configuration Item	**PercentageForRTN**
Section	**[Fax]**
Written Format	PercentageForRTN =n
Value Range	0≤n≤100, with the default value of 30.
Description	If the percentage received is greater than the value of this configuration item, the page is considered to be with incomplete and/or incorrect fax data; otherwise, it is regarded as the one reaching the standard.

3.1.2.23.7 FaxCodeMode

Configuration Item	**FaxCodeMode**
Section	**[Fax]**
Written Format	FaxCodeMode =k

Value Range	k=0: at or below MH (default);
	k=1: at or below MR;
	k=2: at or below MMR.

Description	Sets the upper limit of the fax CODECs.

3.1.2.23.8 FaxSendDisTime

Configuration Item	**FaxSendDisTime**
Section	**[Fax]**
Written Format	FaxSendDisTime =n
Value Range	3000≤n≤6000, calculated by millisecond (ms), with the default value of 6000.
Description	Sets the time interval for DIS command retransmission.

3.1.2.24 Setting Voltage Detector

3.1.2.24.1 Ignoring Detected Result of Voltage Detector

3.1.2.24.1.1 IgnoreLineVoltage

Configuration Item	**IgnoreLineVoltage**
Section	**[BoardId=x]**
Written Format	Format 1: IgnoreLineVoltage=b,b,b,b Format 2: IgnoreLineVoltage=b,b,b,b,b,b,b,b Format 3: IgnoreLineVoltage=b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b Format 4: IgnoreLineVoltage=b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b
Value Range	b=0: No (default); b=1: Yes.
Description	Sets whether to ignore the on-line voltage detected result. Refer to the Change in Analog Phone Line Voltage section in Chapter 1 for details.
Note	● This configuration item is only applicable to the recording channel on the ATP Series boards (including the analog recording module and the microphone module); ● Format 1 is applicable to the ATP Series USB recording boxes; ● Format 2 is applicable to the 8-channel ATP Series; ● Format 3 is applicable to the 16-channel ATP Series; ● Format 4 is applicable to the 24-channel ATP Series.
Related Function	SsmSetIgnoreLineVoltage

3.1.2.24.2 Setting Voltage Threshold for Ring Detection

3.1.2.24.2.1 JudgeLineVoltage

Configuration Item	**JudgeLineVoltage**
Section	**[SystemConfig]**
Written Format	JudgeLineVoltage=k

Value Range	0<k<256: The difference between the current line voltage in ringing state and that of 8ms before, calculated by 0.5Volt (0.5V), with the default value of 7V (15*0.5V).
Description	Set the voltage threshold value for different rings on the analog line.
Note	● This configuration item is only applicable to the recording channel on the ATP/SHT Series boards (including the analog recording module and the trunk module).

3.1.2.24.3Setting Judgement on Pickup/Hangup

3.1.2.24.3.1 IsHangupDtrmVoltage

Configuration Item	**IsHangupDtrmVoltage**
Section	**[BoardId=x]**
Written Format	Format 1: IsHangupDtrmVoltage=k,k,k,k Format 2: IsHangupDtrmVoltage=k,k,k,k,k,k,k,k Format 3: IsHangupDtrmVoltage=k,k,k,k,k,k,k,k,k,k,k,k,k,k,k,k
Value Range	k: The judging voltage (V), with the default value of 26V.
Description	Set the voltage used to judge the pickup/hangup behavior of the recording channel on the ATP Series boards. Refer to the Change in Analog Phone Line Voltage section in Chapter 1 for details.
Note	● This configuration item is only applicable to the recording channel on the ATP Series boards (including the analog recording module and the microphone module); ● Format 1 is applicable to the ATP Series USB recording boxes; ● Format 2 is applicable to the 8-channel ATP Series; ● Format 3 is applicable to the 16-channel ATP Series.
Related Function	SsmSetDtrmLineVoltage

3.1.2.24.3.2 LineOncnt

Configuration Item	**LineOncnt**
Section	**[SystemConfig]**
Written Format	LineOncnt=t
Value Range	t: The minimum value is one time, 1≤t≤65536, each time equals to 8ms; the default value is 25 times (i.e. 200ms).
Description	Only if the line voltage meets the set voltage for judging the pickup behavior and lasts for the minimum duration specified by this configuration item will the driver confirm the pickup behavior on the line. Refer to the Change in Analog Phone Line Voltage section in Chapter 1 for more information.
Note	● This configuration item is only applicable to the recording channel on the ATP or SHT Series boards; ● It requires SynCTI Ver. 4.7.2.0 or above.

3.1.2.24.4Setting Off-line Detection

3.1.2.24.4.1 OffLineSet

Configuration Item	**OffLineSet**
Section	**[SystemConfig]**
Written Format	OffLineSet=k
Value Range	k: The threshold voltage value (V), which is two times the actual voltage value. The default value is 5, which corresponds to the actual voltage of 2.5V.
Description	Sets the threshold voltage value for judging the disconnection of the analog phone line. Refer to the Change in Analog Phone Line Voltage section in Chapter 1 for details.
Note	This configuration item is only applicable to the analog trunk recording channel.

3.1.2.24.4.2 OffLineDetermineTime

Configuration Item	**OffLineDetermineTime**
Section	**[SystemConfig]**
Written Format	OffLineDetermineTime=t
Value Range	t: The minimum duration, calculated by millisecond (ms), with the default value of 200.
Description	Sets the minimum duration for judging the disconnection of the analog phone line. Refer to the Change in Analog Phone Line Voltage section in Chapter 1 for details.
Note	This configuration item is only applicable to the recording channel on the ATP Series boards and the analog trunk channel on the SHT Series boards.

3.1.2.24.5 Setting Effect of Polarity Reversal Signal

3.1.2.24.5.1 DisablePolarReverse

Configuration Item	**DisablePolarReverse**
Section	**[BoardId=x]**
Written Format	Format 1: DisablePolarReverse=b,b,b,b Format 2: DisablePolarReverse=b,b,b,b,b,b,b,b Format 3: DisablePolarReverse=b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b Format 4: DisablePolarReverse= [0,29]:b
Value Range	b=0: Affect (default); b=1: Not affect
Description	Sets whether the polarity reversal signal affects the channel state transition. Refer to the Change in Analog Phone Line Voltage section in Chapter 1 for details.
Note	● This configuration item is only applicable to the analog trunk channel on the SHT Series boards; ● Format 1 is applicable to the SHT Series USB recording boxes; ● Format 2 is applicable to the 8-channel SHT Series; ● Format 3 is applicable to the 16-channel SHT Series; ● Format 4 is applicable to the 30-channel channel bank.

3.1.2.24.6 Setting Threshold Voltage to Ignore Interfering Signal

3.1.2.24.6.1 PolarIgnore

Configuration Item	**PolarIgnore**
Section	**[BoardId=x]**
Written Format	**PolarIgnore** =b
Value Range	b>=0, with the default value of 0.
Description	Sets a threshold voltage to ignore interfering signals on a line lest they are mistaken by the driver for polarity reversal signals. To be exact, when PolarIgnore=b, all signals with voltage values less than or equal to b will be ignored, that is, not be regarded as polarity reversal signals. Refer to the Change in Analog Phone Line Voltage section in Chapter 1 for details.
Note	This configuration item is applicable to analog trunk channels on the SHT Series boards and analog recording channels on the ATP Series boards.

3.1.2.25 Setting Speaker of USB Recording/voice Box

3.1.2.25.1USBLine0Output

Configuration Item	**USBLine0Output**
Section	**[BoardId=x]**
Written Format	USBLine0Output=m
Value Range	m=0: Send the voice to the on-board speaker for play; m=1: Send the voice to the external speaker for play via the on-board speaker output jack
Description	Enable or disable the on-board speaker in the USB recording/voice box. See the Operation Principle of ATP Series or Operation Principle of SHT Series section in Chapter 1 for more information.
Note	● This configuration item is only applicable to the ATP Series USB recording boxes and the SHT Series USB voice boxes; ● It requires SynCTI Ver. 4.7.1.7 or above.

3.1.2.26 Common Configuration Item for SHT Series (CTI Series)

3.1.2.26.1Special Configuration Item for Analog Trunk Channel

3.1.2.26.1.1 Setting Analog Trunk Channel to Be Recording Channel

3.1.2.26.1.1.1 SetAnalogChToRecCh

Configuration Item	**SetAnalogChToRecCh**
Section	**[Boardid=x]**
Written Format	Format 1: SetAnalogChToRecCh =b,b,b,b Format 2: SetAnalogChToRecCh =b,b,b,b,b,b,b,b Format 3: SetAnalogChToRecCh =b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b
Value Range	b=1: Channel open, working as a recording channel; b=0: Channel closed (default), this configuration item being invalid.
Description	Mandatorily sets the analog trunk channel to be a recording channel
Note	● This configuration item is only effective to SynCTI 5.0.1.0 or above versions; ● This configuration item is only applicable to the analog trunk channel, including SHT Series boards with analog trunk modules (except D-type boards) and USB voice boxes; ● Format 1 is applicable to the SHT Series USB voice boxes; ● Format 2 is applicable to the 8-channel SHT Series; ● Format 3 is applicable to the 16-channel SHT Series.

3.1.2.26.1.2 Setting Parameters for Outgoing Call

3.1.2.26.1.2.1 MaxWaitDialToneTime

Configuration Item	**MaxWaitDialToneTime**
Section	**[SystemConfig]**
Written Format	MaxWaitDialToneTime=t
Value Range	t: The maximum waiting time, calculated by millisecond (ms), 0≤t≤30, with the default value of 3ms.

Description	Sets the maximum time to wait for the dial tone. The driver should detect the dial tone first after the application invokes the function SsmAutoDial to start the AutoDial task. If the dial tone can not be detected within the time specified by this configuration item, the AutoDial task fails. See the Analog Trunk Channel State Machine section in Chapter 1 for more information about the AutoDial task.
Note	This configuration item is only applicable to the analog trunk channel.

3.1.2.26.1.3 Setting Adjustment on Analog Trunk Channel

3.1.2.26.1.3.1 AdjustImmediately

Configuration Item	**AdjustImmediately**
Section	**[BoardId=x]**
Written Format	AdjustImmediately =b
Value Range	b=0: Enable (default); b=1: Disable.
Description	Sets whether to enable the adjustment feature for the analog trunk channels when it picks up the call.

3.1.2.26.2 Setting Flash Signal and Duration

3.1.2.26.2.1 DefaultTxFlashChar

Configuration Item	**DefaultTxFlashChar**
Section	**[SystemConfig]**
Written Format	DefaultTxFlashChar =s
Value Range	s: The character used to represent the flash, with the default value of '!', not allowed to be standard DTMF digits or '?'.
Description	Sets the character to represent the flash. A flash signal will be sent upon this character being put into the function SsmTxDtmf.

3.1.2.26.2.2 DefaultTxFlashTime

Configuration Item	**DefaultTxFlashTime**
Section	**[SystemConfig]**
Written Format	DefaultTxFlashTime=t
Value Range	32≤t≤1000, calculated by millisecond (ms), with the default value of 500.
Description	Sets the duration of the flash signal generated on the analog trunk via the function call of SsmTxDtmf where the character '!' is used as a parameter. Refer to the Generating Flash Signal on Analog Line section in Chapter 1 for more information.

3.1.2.26.2.3 Keep Channel State after Flash

3.1.2.26.2.3.1 AfterFlashNotAffectChState

Configuration Item	**AfterFlashNotAffectChState**
Section	**[SystemConfig]**
Written Format	AfterFlashNotAffectChState=b
Value Range	b=0: No (default); b=1: Yes.
Description	Sets whether to keep the channel state after the flash operation.
Note	This configuration item is only applicable to the analog trunk channel.

3.1.2.26.2.4 Remote Pickup Detector

3.1.2.26.2.4.1 Setting Parameters for Ordinary Remote Pickup Detector

3.1.2.26.2.4.1.1 WaitAfterDialTime

Configuration Item	**WaitAfterDialTime**
Section	**[SystemConfig]**
Written Format	WaitAfterDialTime=t
Value Range	t: The maximum waiting time, calculated by millisecond (ms), 16≤t≤60000, with the default value of 18000.
Description	After the function SsmAutoDial is invoked on the analog trunk channel, the driver will throw out the E_CHG_ToneAnalyze event (with the parameter CHKTONE_NOVOICE) if it neither detects ringback tones on the line following the transmission of the complete called party number to the remote PBX, nor detects voice activities within the time specified by this configuration item. See the Ordinary Remote Pickup Detector section in Chapter 1 for more information.
Note	This configuration item is applicable to the analog trunk channel on the SHT Series boards. Besides, this configuration item can implement the similar feature for SHD Series Boards and IP boards when DefaultToneCheckState is set to 1.

3.1.2.26.2.4.1.2 MaxWaitVocAfterEcho

Configuration Item	**MaxWaitVocAfterEcho**
Section	**[SystemConfig]**
Written Format	MaxWaitVocAfterEcho=t
Value Range	t: The maximum waiting time, calculated by second (s), with the default value of 10.
Description	After the function SsmAutoDial is invoked on the analog trunk channel, the driver will throw out the E_CHG_ToneAnalyze event (with the parameter CHKTONE_ECHO_NOVOICE) if it detects ringback tones on the line following the transmission of the complete called party number to the remote PBX, but does not detect any voice activity within the time specified by this configuration item. See the Ordinary Remote Pickup Detector section in Chapter 1 for more information.
Note	This configuration item is only applicable to the analog trunk channel on the SHT Series boards.

3.1.2.26.2.4.1.3 VoiceOnDetermineTime

Configuration Item	**VoiceOnDetermineTime**
Section	**[SystemConfig]**
Written Format	VoiceOnDetermineTime=t
Value Range	16≤t≤1024 and must be the multiple of 16, calculated by millisecond (ms), with the default value of 96ms.
Description	Sets the minimum duration of voice activities. Only if continuous voice activities occur on the line and last for a period of time longer than the preset value of this configuration item will the driver confirm there are real voice activities available on the line. See the Ordinary Remote Pickup Detector section in Chapter 1 for more information.
Note	This configuration item is applicable to the analog trunk channel on the SHT Series boards. Besides, this configuration item can implement the similar feature for SHD Series Boards and IP boards when DefaultToneCheckState is set to 1.

3.1.2.26.2.4.1.4 EchoNoVoiceToTalking

Configuration Item	**EchoNoVoiceToTalking**
Section	**[SystemConfig]**

Written Format	EchoNoVoiceToTalking=N
Value Range	N=0: Disabled (default); N=1: Enabled.
Description	When the pickup pulse on the line following ringback tones is very small or approaching zero, the event 'no voice after ringback' will be thrown out. Setting this configuration at the time the event 'no voice after ringback' is generated can make the channel go into the talking state automatically. It is applicable to some particular occasions. For more information, see the Ordinary Remote Pickup Detector section in Chapter 1.
Note	This configuration item is only applicable to the analog trunk channel on the SHT Series boards.

3.1.2.26.2.4.2 Setting Parameters for Enhanced Remote Pickup Detector

3.1.2.26.2.4.2.1 RelativeEngyHookDetect

Configuration Item	**RelativeEngyHookDetect**
Section	**[SystemConfig]**
Written Format	RelativeEngyHookDetect=b
Value Range	b=0: Disable b=1: Enable (default)
Description	Sets the operating mode of the enhanced remote pickup detector on the analog trunk channel. Refer to the Enhanced Remote Pickup Detector section in Chapter 1 for more information.
Note	● This configuration item is only applicable to the analog trunk channel on the SHT Series boards and requires SynCTI Ver. 3.5.2.4 or above; ● The same purpose can be achieved via the function call of SsmSetFlag (with the parameter F_RELATIVEENGYHOOKDETECT).

3.1.2.26.2.4.2.2 HookEngyConfigMulti

Refer to **HookValidEngyCnt**

3.1.2.26.2.4.2.3 HookValidEngyCnt

Configuration Item	**HookEngyConfigMulti** **HookValidEngyCnt**
Section	**[SystemConfig]**
Written Format	HookEngyConfigMulti=k HookValidEngyCnt=t
Value Range	k: The sensitivity, 2≤k≤65535, with the default value of 6. The greater the value is, the less sensitive, which means the increased miss detection rate and the decreased false detection rate. t: The minimum duration, 32≤t≤65535, calculated by millisecond (ms), which has to be the multiple of 16, with the default value of 48ms. The greater the value is, the less sensitive, which means the increased miss detection rate and the decreased false detection rate.
Description	Sets the parameters for the enhanced remote pickup detector on the analog trunk channel. HookEngyConfigMulti sets the sensitivity of the pickup detection while HookValidEngyCnt configures the minimum duration to eliminate the interference of the signal dithering on the detected result. Only when the driver detects an instantaneous pickup behavior which lasts for a time equal to or longer than t will it conclude there is a real pickup behavior performed on the line. Refer to the Enhanced Remote Pickup Detector section in Chapter 1 for more information.
Note	● This configuration item is only applicable to the analog trunk channel on the SHT Series boards, being valid only when the configuration item RelativeEngyHookDetect is set to 1; ● It requires SynCTI Ver. 3.5.2.4 or above.

3.1.2.26.3 Special Configuration Item for Station Channel

3.1.2.26.3.1 AutoSendDialTone

Configuration Item	**AutoSendDialTone**
Section	**[BoardId=x]**
Written Format	Format 1: AutoSendDialTone=b,b,b,b Format 2: AutoSendDialTone=b,b,b,b,b,b,b,b Format 3: AutoSendDialTone=b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b Format 4: AutoSendDialTone=[S_1,E_1]:b+…+ [S_m,E_m]:b
Value Range	b=1: Enable; b=0: Disable (default). S_1: The start channel number, with the default value of 0. E_1: The end channel number, the default value being the total number of on-board channels. m: The number of parameter fields, being equal to or greater than 1 but not exceeding the total number of on-board channels.
Description	Sets whether the dial tone is automatically sent by the driver upon detecting the pickup behavior of the phone connected to the station channel.
Note	● This configuration item is only applicable to the station channel, including the SHT Series boards equipped with the station module and the SHD Series boards connected to the channel bank; ● Format 1 is applicable to the SHT Series USB voice boxes; ● Format 2 is applicable to the 8-channel SHT Series; ● Format 3 is applicable to the 16-channel SHT Series; ● Format 4 is applicable to the SHD Series boards connected to the channel bank. See the Connection to Channel Bank section in Chapter 1 for more information.
Example	When the SHD-30A-CT/PCI/SS1 board is connected to the channel bank, you can follow the configuration below to enable this feature for the first 15 on-board channels and disable this feature for the last 15 channels. AutoSendDialTone=[0,14]:1+[15,29]:0 If this feature is enabled for all 30 on-board channels, you can configure as follows. AutoSendDialTone=[0, 29]:1

3.1.2.26.3.2 StopSendDialToneOnDtmf

Configuration Item	**StopSendDialToneOnDtmf**
Section	**[BoardId=x]**
Written Format	Format 1: StopSendDialToneOnDtmf=b,b,b,b Format 2: StopSendDialToneOnDtmf=b,b,b,b,b,b,b,b Format 3: StopSendDialToneOnDtmf=b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b Format 4: StopSendDialToneOnDtmf =[S_1,E_1]:b+…+ [S_m,E_m]:b
Value Range	b=1: Enable (default); b=0: Disable. S_1: The start channel number, with the default value of 0. E_1: The end channel number, the default value being the total number of on-board channels. m: The number of parameter fields, being equal to or greater than 1 but not exceeding the total number of on-board channels.
Description	This configuration item is used to set whether to enable the feature of ‘auto-stop of dial tone transmission by driver’ upon the driver’s detection of DTMF digits in sending tones to the phone.

Note	● This configuration item is only applicable to the station channel, including the SHT Series boards equipped with the station module and the SHD Series boards connected to the channel bank; ● Format 1 is applicable to the SHT Series USB voice box; ● Format 2 is applicable to the SHT Series 8-channel boards; ● Format 3 is applicable to the SHT Series 16-channel boards; ● Format 4 is applicable to the SHD Series boards connected to the channel bank. Refer to the Connection to Channel Bank section in Chapter 1 for more information. See the example given in the configuration item AutoSendDialTone for reference.

3.1.2.26.3.3 MaxLocalFlashTime

Configuration Item	**MaxLocalFlashTime**
Section	**[SystemConfig]**
Written Format	MaxLocalFlashTime=t
Value Range	32≤t≤2000, calculated by millisecond (ms), with the default value of 700.
Description	Sets the maximum duration for judging the flash signal.
Note	This configuration item is only applicable to the station channel.

3.1.2.26.3.4 UserOnHookFilterTime

Configuration Item	**UserOnHookFilterTime**
Section	**[BoardID=x]**
Written Format	Format 1: UserOnHookFilterTime=t,t,t,t Format 2: UserOnHookFilterTime=t,t,t,t,t,t,t,t Format 3: UserOnHookFilterTime=t,t,t,t,t,t,t,t,t,t,t,t,t,t,t,t Format 4: UserOnHookFilterTime =[S_1,E_1]:t+…+ [S_m,E_m]:t
Value Range	t: The minimum duration of the pickup signal, calculated by millisecond (ms), which has to be the multiple of 16. The less the value is, the more sensitive. The default value is 64ms. S_1: The start channel number, with the default value of 0. E_1: The end channel number, the default value being the total number of on-board channels. m: The number of parameter fields, being equal to or greater than 1 but not exceeding the total number of on-board channels.
Description	Sets the minimum duration of the pickup signal.
Note	● This configuration item is only applicable to the station channel, including the SHT Series boards equipped with the station module and the SHD Series boards connected to the channel bank; ● Format 1 is applicable to the SHT Series USB voice boxes; ● Format 2 is applicable to the 8-channel SHT Series; ● Format 3 is applicable to the 16-channel SHT Series; ● Format 4 is applicable to the SHD Series boards connected to the channel bank. Refer to the Connection to Channel Bank section in Chapter 1 for more information. See the example given in the configuration item AutoSendDialTone for reference. ● Usually there is no need to reconfigure this item as its default value is adequate for most application instances.

3.1.2.26.3.5 UserChGenerateRingMode

Configuration Item	**UserChGenerateRingMode**
Section	**[BoardId=x]**
Written Format	UserChGenerateRingMode=m
Value Range	m=0: 1sec on and 4sec off (default); m=1: Always at on state until the application sends the stop command.
Description	Sets the way how the station channel or the magnet channel generates the ringing signal.

3.1.2.26.3.6 UserSendPolar

Configuration Item	**UserSendPolar**
Section	**[BoardId=x]**
Written Format	Format 1: UserSendPolar=b,b,b,b Format 2: UserSendPolar=b,b,b,b,b,b,b,b Format 3: UserSendPolar=b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b
Value Range	b=0: No (default); b=1: Yes.
Description	Sets whether the station channel has the capability of generating the polarity reversal signal. Refer to the 'Generating Polarity Reversal Signal on Phone Line' section in Chapter 1 for more information.
Note	● This configuration item is only applicable to the station channel on the SHT Series boards; ● Format 1 is applicable to the SHT Series USB voice boxes; ● Format 2 is applicable to the 8-channel SHT Series; ● Format 3 is applicable to the 16-channel SHT Series.

3.1.2.26.3.7 LocalHookFilterTime

Configuration Item	**LocalHookFilterTime**
Section	**[SystemConfig]**
Written Format	LocalHookFilterTime=t
Value Range	16≤t≤MaxLocalFlashTime, calculated by millisecond (ms), with the default value of 16.
Description	Sets the minimum duration for judging the flash signal.
Note	This configuration item is only applicable to the station channel.

3.1.2.26.4Special Configuration Item for Fax Channel

3.1.2.26.4.1 MaxFaxChannel

Configuration Item	**MaxFaxChannel**
Section	**[BoardId=x]**
Written Format	MaxFaxChannel=n
Value Range	Both the range of value and the default value depend on the board model. See the Setting Number of Fax Channels section in Chapter 1 for more information.
Description	Sets the total number of fax channels.

3.1.2.26.4.2 DSP3WORKMODE

Configuration Item	**DSP3WORKMODE**
Section	**[BoardId=x]**
Written Format	DSP3WORKMODE=n
Value Range	n=1: For SHT-16B-CT/PCI/FAX, enable this board to have faxing capability (default); n=2: For SHT-16B-CT/PCI/FAX, enable this board to support GSM recording in hardware; n=3: For SHT-16B-CT/PCI/MP3 and SHT-16B-CT/cPCI/MP3, enable them to support MP3 recording in hardware.
Description	For the SHT-16B-CT/PCI/FAX board, this configuration item is used to enable the capability of faxing or GSM recording in hardware; for the SHT-16B-CT/PCI/MP3 and SHT-16B-CT/cPCI/MP3 boards, this configuration item is used to enable the capability of MP3 recording in hardware. See the Brief Introduction of SHT Series section in Chapter 1 for more information.
Note	The board model of SHT-16B-CT/PCI/MP3 displayed by configuration tools is SHT-16B-CT/PCI/FAX. Therefore, don't forget to configure DSP3WORDMODE=3 when you are using the SHT-16B-CT/PCI/MP3 board.

3.1.2.26.5 Special Configuration Item for Composite Module

3.1.2.26.5.1 UnimoduleState

Configuration Item	**UnimoduleState**
Section	**[BoardId=x]**
Written Format	Format 1: UnimoduleState=b,b,b,b Format 2: UnimoduleState=b,b,b,b,b,b,b,b Format 3: UnimoduleState=b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b
Value Range	b=0: Yes; b=1: No(default).
Description	Sets whether to connect the analog trunk channel on the composite module directly to the station channel.
Note	● Format 1 is applicable to the SHT Series USB voice boxes; ● Format 2 is applicable to the 8-channel SHT Series; ● Format 3 is applicable to the 16-channel SHT Series.

3.1.2.26.6 Special Configuration Item for Non-module channel

3.1.2.26.6.1 NoModuleChBusRec

Configuration Item	**NoModuleChBusRec**
Section	**[BoardId=x]**
Written Format	Format 1: NoModuleChBusRec=b,b,b,b Format 2: NoModuleChBusRec=b,b,b,b,b,b,b,b Format 3: NoModuleChBusRec=b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b
Value Range	b=0: Disable (default); b=1: Enable.
Description	Sets whether to input the voice signals (not the incoming signals) which come from the bus to the Barge-in Detector. See the Non-module Channel section in Chapter 1 for more information.
Note	● Format 1 is applicable to the SHT Series USB voice boxes; ● Format 2 is applicable to the 8-channel SHT Series; ● Format 3 is applicable to the 16-channel SHT Series.

3.1.2.26.7 Magnet Channel

3.1.2.26.7.1 IsMagnetModule

Configuration Item	**IsMagnetModule**
Section	**[BoardId=x]**
Written Format	Format 1: IsMagnetModule=b,b,b,b Format 2: IsMagnetModule=b,b,b,b,b,b,b,b Format 3: IsMagnetModule=b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b
Value Range	b=0: Analog trunk module (default); b=1: Magnet Module
Description	Designates whether the module fixed on the SHT Series board is the analog trunk module or the magnet module. If it is the analog trunk module, this configuration item should be set to 0; if it is the magnet module, this configuration item should be set to 1. See the Magnet Channel section in Chapter 1 for more information.

Note	● Format 1 is applicable to the SHT Series USB voice boxes; ● Format 2 is applicable to the 8-channel SHT Series; ● Format 3 is applicable to the 16-channel SHT Series.

3.1.2.27 Common Configuration Item for SHD Series (CTI Series)

3.1.2.27.1 Setting Number-receiving Rule for Incoming Call

3.1.2.27.1.1 DefaultRcvPhoNumLen

Refer to RcvPhoNumCfgLen

3.1.2.27.1.2 DefaultRcvCallerID

Refer to RcvPhoNumCfgLen

3.1.2.27.1.3 RcvPhoNumCfgLen

Configuration Item	**DefaultRcvPhoNumLen** **DefaultRcvCallerID** **RcvPhoNumCfgLen**
Section	**[TUP] / [ISUP] / [ISDN] / [SS1Config]**
Written Format	DefaultRcvPhoNumLen=n DefaultRcvCallerID=m RcvPhoNumCfgLen=b
Value Range	n: The way to receive the number. n=0: Prefix Mode. The configuration items MaxPhoNumRule and Rule are used to set particular rules in this mode. 1≤n≤40: Fixed-length Mode. n is the length of the called party number. Receiving all the n digits of the called party number in the incoming call puts the local end's number reception into an end. At this time, the configuration items DefaultRcvCallerID and RcvPhoNumCfgLen are valid. The default value of n is 0. m: Determines whether it is necessary to receive the calling party number m=0: Not necessary (default); m=1: Necessary. b: The way to save the called party number. b=0: Save all the received digits of the called party number (default); b=1: Only save some digits first received. The number of digits to be saved is specified by DefaultRcvPhoNumLen. The remaining digits received are all discarded.
Description	DefaultRcvPhoNumLen sets the way to receive the number in an incoming call. DefaultRcvCallerID determines whether it is necessary to receive the calling party number. This configuration item does not support ISDN protocol. RcvPhoNumCfgLen sets the way for the driver to save the received called party number to its internal buffer.

Note	● The configuration item DefaultRcvCallerID becomes valid only when the configuration item DefaultRcvPhoNumLen is set to Fixed-length Mode. ● RcvPhoNumCfgLen is only applicable to the TUP channel; ● This configuration item is only applicable to TUP, ISUP, ISDN, SS1 channels; ● The SS1 channel uses the configuration section **[SS1Config]**; ● The ISUP channel uses the configuration section **[ISUP]**; ● The TUP channel uses the configuration section **[TUP]**; ● The ISDN channel uses the configuration section **[ISDN]**. For the ISDN channel, the number receiving rule set by this configuration item will be ignored if the driver works in the Tandem Exchange mode. Refer to the Terminating Office Mode vs. Tandem Exchange Mode section in Chapter 1 for more information.

3.1.2.27.1.4 MaxPhoNumRule

Refer to **Rule**

3.1.2.27.1.5 Rule

Configuration Item	**MaxPhoNumRule** **Rule**
Section	**[TUP] / [ISUP] / [ISDN] / [SS1Config]**
Written Format	MaxPhoNumRule=N Format 1: Rule[i]="a…x",n,f,y,z Format 2: Rule[i]="a…x",n,f,y Format 3: Rule[i]="a…x",n,f
Value Range	N: The times to set the number receiving rule. This configuration item is not required to be set if the set value of DefaultRcvPhoNumLen is greater than 0; otherwise, N must be set greater than 0 and the configuration item Rule should be set for N times, beginning with i=0. The default value of N is 0. I : The rule number, numbered from 0, 0 ≤ i < N 'a…x': The string of the prefix, such as '114', '127', etc. n: The set length for receiving the prefix string. n must be greater than or equal to the actual length of the prefix string. Except for SS1 and ISDN channels, this parameter gets valid only when y is set to 0. f: Determines whether it is necessary to receive the calling party number =1: Necessary; =0: Not necessary. y: Determines whether the length of the phone number is fixed. =1: Not fixed. The parameters n and z are invalid. The driver will ask the application to control the number receiving process upon reception of the office number. After receiving the complete call number, the application can invoke the function SsmSetKB to refuse or accept the incoming call, and then asks the driver to control the process again. =0: Fixed (default). The parameters n and z are valid. z: The way to save the called party number. z=0: Save all the received digits of the called party number (default); z=1: Only save some digits first received. The number of digits to be saved is determined by n. The remaining digits received are all discarded.
Description	MaxPhoNumRule is used to specify the times to set the number receiving rule; Rule is used to set details of the number receiving rule. Format 1 is only applicable to the TUP channel; Format 2 is only applicable to TUP and ISUP channels; Format 3 is applicable to SS1 and ISDN channels.

Note	● This configuration item becomes valid only when DefaultRcvPhoNumLen is set to Prefix Mode; ● This configuration item is only applicable to TUP, ISUP, ISDN, SS1 channels; ● The SS1 channel uses the configuration section **[SS1Config]**; ● The ISUP channel uses the configuration section **[ISUP]**; ● The TUP channel uses the configuration section **[TUP]**; ● The ISDN channel uses the configuration section **[ISDN]**. For the ISDN channel, the number receiving rule set by this configuration item will be ignored if the driver works in the Tandem Exchange mode. Refer to the Terminating Office Mode vs. Tandem Exchange Mode section in Chapter 1 for more information.

3.1.2.27.1.6 MfcLenCtrlPos

Refer to **CallerIdEnTable**

3.1.2.27.1.7 MfcLengthTable

Refer to **CallerIdEnTable**

3.1.2.27.1.8 CallerIdEnTable

Configuration Item	**MfcLenCtrlPos** **MfcLengthTable** **CallerIdEnTable**
Section	**[SS1Config]**
Written Format	MfcLenCtrlPos=n MfcLengthTable=m,m,m,m,m,m,m,m,m,m CallerIdEnTable=b,b,b,b,b,b,b,b,b,b
Value Range	n: The length control bit taken as the index for receiving the called party number, n≥1; m: The length of the called party number to be received; b: Determines whether it is necessary to receive the calling party number =1: Necessary; =0: Not necessary.
Description	MfcLenCtrlPos sets the length control bit for receiving the called party number on the trunk channel; MfcLengthTable sets the index of the length control bit for MFC number receiving on the trunk channel; CallerIdEnTable sets the table controlling whether to enable the trunk channel to receive the calling party number or not. The digit at the position MfcLenCtrlPos in the called party number can be taken as the index for searching the corresponding length in the length control table MfcLengthTable; also it can be taken as the index to check if the corresponding calling party number should be received in CallerIdEnTable.
Note	This configuration item is only applicable to the SS1 channel.

3.1.2.27.2 Setting 'Auto-answer of Incoming Call' Feature

3.1.2.27.2.1 AutoSendKB

Refer to NetSideAutoSendAck

3.1.2.27.2.2 AutoSendACM

Refer to NetSideAutoSendAck

3.1.2.27.2.3 UserSideAutoSendAck

Refer to NetSideAutoSendAck

3.1.2.27.2.4 NetSideAutoSendAck

Configuration Item	**AutoSendKB** **AutoSendACM** **UserSideAutoSendAck** **NetSideAutoSendAck**
Section	**[SS1Config] / [TUP] / [ISUP] / [ISDN]**

Written Format	AutoSendKB=b //use [SS1Config]
	AutoSendACM=b //use [TUP] or [ISUP]
	UserSideAutoSendAck=b //use [ISDN]
	NetSideAutoSendAck=b //use [ISDN]

Value Range	b=0: No; b=1: Yes (default).
Description	Sets whether to enable the 'Auto-answer of Incoming Call' feature or not. AutoSendKB is used to set the SS1 channel. Refer to the China SS1 State Machine section in Chapter 1 for more information. AutoSendACM is used to set ISUP and TUP channels. Refer to the ISUP Channel State Machine or TUP Channel State Machine section in Chapter 1 for more information. UserSideAutoSendAck is used to set the ISDN channel (user-side) while NetSideAutoSendAck is used to set the ISDN channel (network-side). Refer to the ISDN Channel State Machine section in Chapter 1 for more information about the k2 switch.
Note	The same purpose can be achieved via the function call of SsmEnableAutoSendKB.

3.1.2.27.3Setting Calling Party Number for Outgoing Call

3.1.2.27.3.1 TxCallerId

Refer to **CalloutCallerId**

3.1.2.27.3.2 CalloutCallerId

Configuration Item	**TxCallerId** **CalloutCallerId**
Section	**[SS1Config] / [TUP] / [ISUP] / [ISDN]**
Written Format	Format 1: TxCallerId=s Format 2: TxCallerId[m]=s Format 1: CalloutCallerId=s Format 2: CalloutCallerId[m]=s
Value Range	s: The ASCII string storing the calling party number. The number of the characters is up to 20 for the ISDN channel, up to 15 for the TUP channel, and up to 50 for other channels. s has to be made up of the digits '0'~'9'. Other types of characters will be ignored. It being empty indicates the calling party number will not be sent to the remote PBX in the outgoing call. m: The logical number of the digital truck in the application. It starts from 0, the range of value: 0≤m<M (M is the set value of the configuration item TotalPcm). The default value is empty.
Description	Sets the phone number of the local end, i.e. the calling party number, in the outgoing call. TxCallerId is used to set the SS1 channel (using Section [SS1Config]); CalloutCallerId is used to set the ISUP channel (using Section [ISUP]), the TUP channel (using Section [TUP]) and the ISDN channel (using Section [ISDN]). Format 1 is used to set the global channels; Format 2 is used to set the channel with a specific PCM number.

Note	● Some PBXes using TUP and ISUP protocols requires the calling party number string sent by the calling party to contain the ST signal. The configuration item SetSTSignal can be used for this purpose. ● If Format 1 and Format 2 exist at the same time, Format 2 has the priority.

3.1.2.27.3.3 SetSTSignal

Configuration Item	**SetSTSignal**
Section	**[TUP] / [ISUP]**
Written Format	SetSTSignal=b
Value Range	b=0: No (default); b=1: Yes.
Description	Sets whether the calling party number string sent by the local end to the remote PBX contains the ST signal in the outgoing call.
Note	This configuration item is only applicable to TUP and ISUP channels, the ISUP channel using the configuration section [ISUP] while the TUP channel using [TUP].

3.1.2.27.4Setting Called Party Number for Outgoing Call

3.1.2.27.4.1 SetCalledSTSignal

Configuration Item	**SetSTSignal**
Section	**[TUP] / [ISUP]**
Written Format	SetCalledSTSignal=b
Value Range	b=0: No (default); b=1: Yes.
Description	Sets whether the called party number string sent by the local end to the remote PBX contains the ST signal in the outgoing call.
Note	This configuration item is only applicable to TUP and ISUP channels, the ISUP channel using the configuration section [ISUP] while the TUP channel using [TUP].

3.1.2.27.5Connecting to Channel Bank

3.1.2.27.5.1 UsageMode

Configuration Item	**UsageMode**
Section	**[BoardId=x]**
Written Format	UsageMode=b
Value Range	b=0: No (default); b=1: Yes.
Description	Sets whether to connect the SHD Series boards to the channel bank. See the Connection to Channel Bank section in Chapter 1 for details.

Note	This configuration item is only applicable to the following board models in the SHD Series. ✓ SHD-30A-CT/PCI/SS1 ✓ SHD-30A-CT/PCI/ISDN ✓ SHD-30A-CT/PCI/SS7 ✓ SHD-30A-CT/cPCI ✓ SHD-30B-CT/PCI/FAX ✓ SHD-60A-CT/PCI/SS1 ✓ SHD-60A-CT/PCI/ISDN ✓ SHD-60A-CT/PCI/SS7 ✓ SHD-60A-CT/cPCI ✓ SHD-60B-CT/PCI/SS7/FAX ✓ SHD-60B-CT/cPCI/FAX ✓ SHD-120A-CT/PCI/SS1 ✓ SHD-120A-CT/PCI/ISDN ✓ SHD-120A-CT/PCI/SS7 ✓ SHD-120A-CT/cPCI ✓ SHD-120D-CT/PCI/CAS ✓ SHD-240D-CT/PCI/CAS ✓ SHD-30E-CT/PCIe ✓ SHD-30E-CT/PCIe/FAX ✓ SHD-30E-CT/PCIe/EC ✓ SHD-60E-CT/PCIe ✓ SHD-60E-CT/PCIe/FAX ✓ SHD-60E-CT/PCIe/EC ✓ SHD-120E-CT/PCIe ✓ SHD-120E-CT/PCIe/FAX ✓ SHD-120E-CT/PCIe/EC ✓ SHD-240E-CT/PCIe ✓ SHD-240E-CT/PCIe/FAX ✓ SHD-240E-CT/PCIe/EC ✓ SHD-240E-CT/PCIe/VAR ✓ SHD-30E-CT/PCI(SSW) ✓ SHD-30E-CT/PCI/FAX(SSW) ✓ SHD-30E-CT/PCI/EC(SSW) ✓ SHD-60E-CT/PCI(SSW) ✓ SHD-60E-CT/PCI/FAX(SSW) ✓ SHD-60E-CT/PCI/EC(SSW) ✓ SHD-120E-CT/PCI(SSW) ✓ SHD-120E-CT/PCI/FAX(SSW) ✓ SHD-120E-CT/PCI/EC(SSW) ✓ SHD-240E-CT/PCI(SSW) ✓ SHD-240E-CT/PCI/FAX(SSW) ✓ SHD-240E-CT/PCI/EC(SSW)

3.1.2.27.5.2 CBProtocolType

Configuration Item	**CBProtocolType**
Section	**[BoardId=x]**
Written Format	CBProtocolType=m
Value Range	m=0: CCS; m=1: CAS (default).
Description	Sets the signaling mode used in connecting the SHD Series boards to the channel bank.
Note	This configuration item becomes valid only when the configuration item UsageMode is set to 1.

3.1.2.27.5.3 CBChannelType

Configuration Item	**CBChannelType**
Section	**[BoardId=x]**
Written Format	CBChannelType=m
Value Range	m=0: Not use this channel; m=1: Station channel (default); m=2: E/M channel.
Description	Sets the channel type used in connecting the SHD Series boards to the channel bank.
Note	This configuration item becomes valid only when the configuration item UsageMode is set to 1.

3.1.2.27.5.4 CBChangeChannelType

Configuration Item	**CBChangeChannelType**
Section	**[BoardId=x]**
Written Format	CBChangeChannelType=m
Value Range	m=0: Do not use this feature (default); m=1: Compulsively set the channel type to station channel when its initialization is failed.
Description	Generally, the large-capacity channel banks connect with each other through the optical-fiber circuit. When the channel type used in connecting the SHD Series boards to the channel bank is -1 upon initialization, you can use this configuration item to set it to station channel compulsively.
Note	This configuration item becomes valid only when the configuration item UsageMode is set to 1.

3.1.2.27.6 Setting Board Operating Mode

3.1.2.27.6.1 RunAsSpy

Configuration Item	**RunAsSpy**
Section	**[BoardId=x]**
Written Format	RunAsSpy =b
Value Range	b=0: Normal mode (default); b=1: Monitoring mode.
Description	Sets the operating mode for the SHD Series boards.
Note	● This configuration item is invalid to the signaling-monitoring board as it always works in the monitoring mode; ● This configuration item is only applicable to A-type and C-type digital boards.

3.1.2.27.7 Advanced Setting for SS1

3.1.2.27.7.1 Selecting Country

3.1.2.27.7.1.1 mfcr2_Protocol

Configuration Item	**mfcr2_Protocol**
Section	**[SS1Config]**
Written Format	mfcr2_Protocol=s

Value Range	s is a string which is described below. **Value** / **Supported Country/Protocol** al / ALGERIA ar / ARGENTINA bh / BAHRAIN bo / BOLIVIA br / BRAZIL cl / CHILE cn / CHINA co-land / COLOMBIA_LAND co-cell / COLOMBIA_CELL cz / CZECH cd / DEMO_CONGO eg / EGYPT gn / GHANA hd / HONDURAS in / INDIA id / INDONESIA iq / IRAQ itu / ITU kr / KOREA kw / KUWAIT my / MALAYSIA mx / MEXICO ng / NIGERIA pa / PANAMA ph / PHILIPINNES ro / ROMANIA sa / SAUDI_ARABIA sg / SINGAPORE th / THAILAND ve / VENEZUELA The default value of s is cn.
Description	Sets the country to use SS1.
Note	This configuration item becomes valid only if the configuration item ProtocolType is set to 0; This configuration item is also applicable to the SS1 channel for DTP series.

3.1.2.27.7.2 Setting R2 Parameters for SS1 Connection

When the value of mfcr2_Protocol is not cn (CHINA), the following configuraiton items can be used to set the R2 parameters for SS1 connection as the expansion of the SS1 MFC-R2 protocol variant.

3.1.2.27.7.2.1 tonesgroupA

Configuration Item	**tonesgroupA**
Section	**[SS1Config]**
Written Format	tonesgroupA=k
Value Range	k: Composed of 16 bits. Each hexade (4 bits) of the parameter contains one request. Low to high hexade: 1. Send next DID (bit0-3). 2. Send Group I category (bit4-7). 3. Send next ANI (bit8-11). 4. Send Group II tone (and switch to group B tone reception) (bit12-15).
Description	Sets backward Group A tones. The driver uses them to send requests to the calling party during the compelled sequence.

Note	This configuration item is only applicable to the SS1 channel on the SHD Series boards.
Example	If the MFC-R2 protocol defines that A1 sends next DID, A5 sends Group I category, A5 sends next ANI, A3 sends Group II tone (and switch to group B tone reception), it should be set tonesgroupA=0x3551.

3.1.2.27.7.2.2 tonesgroupB

Configuration Item	**tonesgroupB**
Section	**[SS1Config]**
Written Format	tonesgroupB=k
Value Range	k: Composed of 16 bits. Each hexade (4 bits) of the parameter contains one Group B indication. Low to high hexade: 1. Indicate congestion (bit0-3). 2. Indicate unallocated number (bit4-7). 3. Indicate busy (bit8-11). 4. Indicate line out of order (bit12-15).
Description	Sets some backward Group B tones. The driver uses them to send the final indication of the compelled sequence to the calling party.
Note	This configuration item is only applicable to the SS1 channel on the SHD Series boards.

3.1.2.27.7.2.3 Tonesendofinfo

Configuration Item	**Tonesendofinfo**
Section	**[SS1Config]**
Written Format	tonesendofinfo=k
Value Range	k: Composed of 16 bits. Every 4 bits constitute a hexade. Low to high hexade: 1. End of DID (bit0-3). In some countries, a tone that signals the end of the DID digits does not exist. In this case, the first hexade will be 0. 2. Spare (bit4-7). 3. End of ANI - caller ID available (bit8-11). 4. End of ANI - called ID restricted (bit12-15). In most countries, there is no distinction for MFC-R2 between restricted and non-restricted caller ID. In this case, the fourth hexade is 0.
Description	Sets forward tones that indicate the end or the non-availability of certain types of information.
Note	This configuration item is only applicable to the SS1 channel on the SHD Series boards.

3.1.2.27.7.2.4 Tonesanswer

Configuration Item	**Tonesanswer**
Section	**[SS1Config]**
Written Format	tonesanswer=k
Value Range	k: Composed of 16 bits. Each hexade (4 bits) of the parameter contains one distinct type of acceptance indication. Low to high hexade: 1. Call accepted in Group B - charge. 2. Call accepted in Group B – free call. 3. Call accepted in Group A. 4. Spare.
Description	Sets backward tones indicating acceptance of the call.
Note	This configuration item is only applicable to the SS1 channel on the SHD Series boards.

3.1.2.27.7.2.5 Tonesrepeatrequest

Configuration Item	**Tonesrepeatrequest**
Section	**[SS1Config]**
Written Format	tonesrepeatrequest=k

Value Range	k: Composed of 16 bits. Every 4 bits constitute a hexade. Consider the DID that the outbound side played last to be DID n. Low to high hexade: 1. Repeat DID n-1 2. Repeat DID n-2 3. Repeat DID n-3 4. Repeat all DIDs (restart dialing)
Description	Sets backward Group A tones the inbound side plays to request a digit repetition from the outbound side.
Note	This configuration item is only applicable to the SS1 channel on the SHD Series boards.

3.1.2.27.7.2.6 TonesAnswerA

Configuration Item	**TonesAnswerA**
Section	**[SS1Config]**
Written Format	TonesAnswerA=k
Value Range	k: Composed of 16 bits. Every 4 bits constitute a hexade. Low to high hexade: 1. Group A network busy. 2. Undefined number in Group A. 3. Particular tones in Group B. 4. xxx in Group II.
Description	Sets some R2 signaling parameters.
Note	This configuration item is only applicable to the SS1 channel on the SHD Series boards.

3.1.2.27.7.3 Setting Basic Parameters for SS1 Connection

3.1.2.27.7.3.1 TxCas_CD

Configuration Item	**TxCas_CD**
Section	**[SS1Config]**
Written Format	TxCas_CD=k
Value Range	k: The high 6 bits should be set to 0, being reserved; the low 2 bits are C/D signaling code. Bit1: Signaling Code C, with the default value of 1; Bit0: Signaling Code D, with the default value of 1.
Description	Sets the value of CD in the ABCD signaling codes sent by the local end to the remote PBX.
Note	This configuration item is only applicable to the SS1 channel on the SHD Series boards, being valid only if the configuration item ProtocolType is set to 1,2,3, 4 or 5.

3.1.2.27.7.3.2 RxCASFilterTime

Configuration Item	**RxCASFilterTime**
Section	**[SS1Config]**
Written Format	RxCASFilterTime=t
Value Range	t: The minimum duration of ABCD signaling codes sent out by the remote PBX, calculated by millisecond (ms), which has to be the multiple of 8, with the default value of 0.
Description	Sets the minimum duration of ABCD signaling codes sent out by the remote PBX. Only when the on-line ABCD signaling codes vary and the new value keeps for more than the time specified by this configuration item will the driver confirm the change of ABCD codes. Otherwise, the driver will believe there are undesired dithering signals on the line.

Note	● This configuration item is only applicable to the SS1 channel on the SHD Series boards; ● It is used for those E1/T1 lines providing a relative low quality of signals. The severe signal dithering is likely to cause the misdetection of the remote pickup by the driver. This configuration item just helps set a minimum duration to eliminate the possibility of signal dithering. Note: The set value of this configuration item being too great may bring in some side effect. That is, in case the called party sends the 'Called party first hangs up' signaling (i.e. A_b/B_b=1/1) immediately following the 'Called party answers' signaling (i.e. A_b/B_b=0/1), the 'Called party answers' signaling is probably filtered out by the filter as dithering signals.

3.1.2.27.7.3.3 MaxWaitMfcTime

Configuration Item	**MaxWaitMfcTime**
Section	**[SS1Config]**
Written Format	MaxWaitMfcTime=t
Value Range	t: The maximum waiting time, calculated by second, with the default value of 10sec.
Description	Sets the timer T2 for the SS1 state machine. Refer to the China SS1 State Machine section in Chapter 1 for more information.
Note	This configuration item is only applicable to the SS1 channel on the SHD Series boards.

3.1.2.27.7.3.4 RxR2FilterTime

Configuration Item	**RxR2FilterTime**
Section	**[BoardId=x]**
Written Format	RxR2FilterTime=t
Value Range	t: The minimum duration of the R2 signal, calculated by millisecond (ms), which has to be the multiple of 16, with the default value of 16. The range of value is 16~96.
Description	Sets the minimum duration of the MFC R2 signal.
Note	● This configuration item is only applicable to the SS1 channel on the SHD Series boards; ● The same purpose can be achieved by the function call of SsmSetFlag (with the parameter F_RXR2FILTERTIME).

3.1.2.27.7.3.5 LSRingTimeout

Configuration Item	**LSRingTimeout**
Section	**[SS1Config]**
Written Format	LSRingTimeout =t
Value Range	t: Calculated by millisecond (ms). Range of value is 1000~20000, with the default value of 6000.
Description	Sets the overtime for being in the 'Ringing' state. If a channel which is in the 'Ringing' state doesn't receive any connection message within the overtime, it will go into the 'Waiting for release' state.
Note	● This configuration item is only applicable to the SS1 channel on the SHD Series boards; ● This configuration item is valid only under the 'LineSide(ProtocolType=1)' mode.

3.1.2.27.7.4 Setting Operating Mode of SS1 Channel State Machine

3.1.2.27.7.4.1 EnableAutoCall

Configuration Item	**EnableAutoCall**
Section	**[BoardId=x]**
Written Format	EnableAutoCall[n]=b

Value Range	n: The internal digital trunk number on the board, 0 ≤ n ≤ M-1 (M is the total number of on-board digital trunks); b: Determines whether to use the state machine provided by the SynCTI driver. b=1: Yes (default); b=0: No.
Description	Sets the operating mode of the SS1 channel state machine.
Note	This configuration item is only applicable to the SS1 channel on the SHD Series boards.

3.1.2.27.7.4.2 AutoCallInTimeSlot

Configuration Item	**AutoCallInTimeSlot**
Section	**[BoardId=x]**
Written Format	AutoCallInTimeSlot[n]=i,k
Value Range	n: The internal digital trunk number on the board, 0 ≤ n ≤ M-1 (M is the total number of on-board digital trunks); i: The initial time slot set for the incoming call. k: The number of channels (time slots) set for the incoming call.
Description	Sets k time slots from TS i on PCM n to process incoming calls and others to process outgoing calls.
Note	This configuration item is only applicable to the SS1 channel on the SHD Series boards.

3.1.2.27.7.5 Setting Parameters for China SS1 State Machine

3.1.2.27.7.5.1 MfcKB

Configuration Item	**MfcKB**
Section	**[SS1Config]**
Written Format	MfcKB=k
Value Range	1≤k≤6. The default value varies on the configuration item mfcr2 Protocol: 1 for Saudi Arabia, Mexico, China, Thailand and Brazil; 5 for Kuwait, 6 for other countries. When k is set to the default value, it indicates the called party is free and receives the incoming call. For the physical meanings of the value of KB, refer to the function SsmSetKB and the SS1 protocol of relevant countries.
Description	Sets the value of the KB signal sent to the remote PBX by the SS1 channel upon automatic reception of an incoming call. Refer to the China SS1 State Machine section in Chapter 1 for more information.
Note	This configuration item becomes valid only when the configuration item AutoSendKB is set to 1.

3.1.2.27.7.5.2 MaxWaitSetKBTime

Configuration Item	**MaxWaitSetKBTime**
Section	**[SS1Config]**
Written Format	MaxWaitSetKBTime=t
Value Range	t: The maximum waiting time, calculated by second, with the default value of 3sec.
Description	Sets the maximum time to wait for the application to configure the KB signal. Sets the maximum waiting time in the SS1 channel state machine for the application to respond to this incoming call request (i.e. the timer T4). Refer to the China SS1 State Machine section in Chapter 1 for more information.

3.1.2.27.7.5.3 MaxWaitKDTime

Configuration Item	**MaxWaitKDTime**
Section	**[SS1Config]**
Written Format	MaxWaitKDTime=t
Value Range	t: The maximum waiting time, calculated by second, with the default value of 60sec.

Description	Sets in the SS1 channel state machine the maximum time to wait for the remote PBX to send the KD signal (i.e. the timer T3). Refer to the China SS1 State Machine section in Chapter 1 for more information.

3.1.2.27.7.5.4 PcmSyncMask

Configuration Item	**PcmSyncMask**
Section	**[SS1Config]**
Written Format	PcmSyncMask=x
Value Range	x: The mask code for line synchronization value. It is 0x206 by default.
Description	Sets whether to shield the sync status of a PCM link on the E1 trunk. When PcmSyncMask is set to 0x0206, it indicates all sync statuses of a PCM link except those represented by bit1, bit2 and bit9 will be ignored. Refer to SsmGetPcmLinkStatus for more information.

3.1.2.27.7.5.5 PhoNumHoldup

Refer to **A3pTime**

3.1.2.27.7.5.6 A1ToA3pWaitTime

Refer to **A3pTime**

3.1.2.27.7.5.7 A3pTime

<table>
<tr><td>Configuration Item</td><td>PhoNumHoldup
A1ToA3pWaitTime
A3pTime</td></tr>
<tr><td>Section</td><td>[SS1Config]</td></tr>
<tr><td>Written Format</td><td>PhoNumHoldup=m
A1ToA3pWaitTime=t_{wait}
A3pTime=t_{keep}</td></tr>
<tr><td>Value Range</td><td>m: The switch to control the ‘Called Number Hold-up’ feature. Its value and the corresponding meaning are shown in the table below.
<table>
<tr><th>m</th><th>Description</th></tr>
<tr><td>0</td><td>Disable the ‘Called Number Hold-up’ feature.</td></tr>
<tr><td>1</td><td>The ‘Called Number Hold-up’ feature is enabled, allowing the holdup of 1 digit of the called party number. When the driver receives the last digit of the called party number according to the preset number receiving rule, it will send the A1 but not A3 signal to the remote PBX and keep waiting for the subsequent digit.
✧ If the driver receives the subsequent digit within t_{wait}, it will save this digit into the Callee ID buffer and finish the reception of the called party number. At that time, the driver will resume the A3 signal transmission in interworking mode to the remote PBX and the normal progress;
✧ If the driver does not receive the subsequent digit within t_{wait}, it will send the A3 signal in the form of pulse to the remote PBX and resume the normal progress.</td></tr>
<tr><td>2</td><td>The ‘Called Number Hold-up’ feature is enabled, allowing the holdup of several digits of the called party number. When the driver receives the last digit of the called party number according to the preset number receiving rule, it will continue to send the A1 and keep waiting for the subsequent digit. If the driver receives the subsequent digit within t_{wait}, it will save this digit into the Callee ID buffer and repeat the process described above. Only if the driver does not receive the subsequent digit within t_{wait}, it will send the A3 signal in the form of pulse to the remote PBX and resume the normal progress.</td></tr>
</table>
The default value of m is 0.
t_{wait}: The maximum waiting time, calculated by second, with the default value of 1000, being valid when m>0;
t_{keep}: The pulse width, calculated by second, with the default value of 150, being valid when m>0.</td></tr>
</table>

Description	Sets the 'Called Number Hold-up' feature. PhoNumHoldup is used to set the control switch of this feature; A1ToA3pWaitTime is to set the threshold value of the time to wait for the subsequent digit; A3pTime is to set the pulse width of the A3 signal.
Note	The purpose achieved by PhoNumHoldup also can be realized via the function call of SsmSetFlag (with the parameter F_RCVPHONUMHOLDUP).

3.1.2.27.7.5.8 Setting Parameters in State Machine for Outgoing Call

3.1.2.27.7.5.8.1 MaxWaitOccupyAckTime

Configuration Item	**MaxWaitOccupyAckTime**
Section	**[SS1Config]**
Written Format	MaxWaitOccupyAckTime=t
Value Range	t≥1, calculated by second, with the default value of 60.
Description	Sets the value of the timer T5. Refer to the China SS1 State Machine section in Chapter 1 for more information.

3.1.2.27.7.5.8.2 MfcKD

Configuration Item	**MfcKD**
Section	**[SS1Config]**
Written Format	MfcKD=k
Value Range	1≤k≤6, with the default value of 3 (local call). See description on the function SsmSetKD for more information about the KD signal.
Description	Sets the originating service type, i.e. KD, for the outgoing call.

3.1.2.27.7.5.8.3 MfcKA

Configuration Item	**MfcKA**
Section	**[SS1Config]**
Written Format	MfcKA=k
Value Range	1≤k≤10, with the default value of 1 (ordinary/regular). See description on the function SsmSetKA for more information about the KA signal.
Description	Sets the KA signal (calling party's category at the local end) sent in an outgoing call.

3.1.2.27.7.5.8.4 MaxWaitKBTime

Configuration Item	**MaxWaitKBTime**
Section	**[SS1Config]**
Written Format	MaxWaitKBTime=t
Value Range	t: The maximum waiting time, calculated by second, with the default value of 60sec;
Description	Sets the maximum time to wait for the KB signal from the remote PBX.

3.1.2.27.7.5.9 Connecting with Dialogic SS1 Channel

3.1.2.27.7.5.9.1 ToRingingDelayTime

Configuration Item	**ToRingingDelayTime**
Section	**[BoardId=x]**
Written Format	ToRingingDelayTime=t
Value Range	t: The delay time, calculated by second, with the default value of 0.
Description	Sets the delay time. Upon the connection of a Synway board and a Dialogic board through the SS1 channel, if the SS1 channel on the Synway board serves as the incoming end, a period of waiting time is required before the channel goes into the 'Ringing' state following the reception of the complete called party number.

Note	This configuration item is only used in connecting the Synway SHD boards with the Dialogic boards.

3.1.2.27.7.5.9.2 RepeatPhoNumOn1stR2bwdIsA5

Configuration Item	**RepeatPhoNumOn1stR2bwdIsA5**
Section	**[BoardId=x]**
Written Format	RepeatPhoNumOn1stR2bwdIsA5=m
Value Range	m=0: The driver sets the pending reason value to be SS1OUT_BWD_A5 and transfers the channel state to S_CALL_PENDING; m=1: If the local end receives the A5 signal from the remote end only after sending the 1st digit of the called party number, it will send the 1st digit again; otherwise, the driver will set the pending reason value to be SS1OUT_BWD_A5 and transfers the channel state to S_CALL_PENDING. The default value is 1.
Description	This configuration item is used to select the driver's subsequent behavior after the reception of the A5 signal (unallocated-number signal) from the remote end in an outgoing call.
Note	This configuration item is only used in connecting the Synway SHD boards with the Dialogic boards.

3.1.2.27.7.5.10 Setting Remote Blocked at Application's Exit

3.1.2.27.7.5.10.1 IsBlockSS1In

Configuration Item	**IsBlockSS1In**
Section	**[SS1Config]**
Written Format	IsBlockSS1In=b
Value Range	b=1: Send (default); b=0: Not send.
Description	This configuration item is used to set whether the driver will automatically sent the blocking command to the remote PBX in order to inform it not to start a call towards the local end again at the time the application program exits.
Note	This configuration item is only applicable to the SS1 channel on the SHD Series boards.

3.1.2.27.7.5.11 Outputting Debugging Information

3.1.2.27.7.5.11.1 MfcR2ToRxCallerIdBuf

Configuration Item	**MfcR2ToRxCallerIdBuf**
Section	**[SS1Config]**
Written Format	MfcR2ToRxCallerIdBuf=b
Value Range	b=0: No (default); b=1: Yes.
Description	Sets whether to save the R2 signal sent by the remote PBX in the outgoing call to the Extended Caller ID buffer so as to facilitate the observation on the call process. If this feature is enabled, the driver will put the calling party number into the RxCallerIdExBuf buffer and throw out the E_CHG_CIDExBuf event to the application every time when the R2 signal sent by the remote PBX varies, including the appearance and disappearance of the R2 signal. The function SsmGetCallerIdEx is used to take out the strings stored in this buffer.

3.1.2.27.7.6 Setting Parameters for LineSide Protocol

3.1.2.27.7.6.1 LSWaitPickupTime

Configuration Item	**LSWaitPickupTime**
Section	**[SS1Config]**
Written Format	LSWaitPickupTime=t
Value Range	16≤t≤10000, calculated by millisecond (ms), which has to be the multiple of 8, with the default value of 96. If the set value is out of the range of value, the driver will automatically modify it to 96ms.
Description	Sets the timer T1 in the LineSide state machine. Refer to the Line Side State Machine section in Chapter 1 for more information.
Note	This configuration item is not applicable to SS1, being valid only when the configuration item ProtocolType is set to 1, 2, 3 or 4.

3.1.2.27.7.6.2 Ss1SendIdleState

Configuration Item	**Ss1SendIdleState**
Section	**[SS1Config]**
Written Format	Ss1SendIdleState=m

Value Range	When the configuration item ProtocolType is set to 1 or 3:
	m=0: Use CAS 00xx;
	m=1: Use CAS 01xx (especially for Alcatel PBXes).
	When the configuration item protocolType is set to 2:
	m=0: Use CAS 00xx;
	m=1: Use CAS 01xx;
	m=2: Use CAS CD01 (CD is the set value of the configuration item TxCas_CD)

Description	Sets the ABCD signaling codes used in sending the idle indicator to the remote PBX.
Note	This configuration item is applicable to the SS1 channel using the LineSide protocol and the channel using the ASB protocol and the channel using the ASBEL protocol, being valid only when the configuration item ProtocolType is set to 1, 2, 3 or 5.

3.1.2.27.7.6.3 LSTxCas

Configuration Item	**LSTxCas**
Section	**[SS1Config]**
Written Format	LSTxCas=0xabcd
Value Range	=0: This configuration item is invalid; =Other values: each number indicates the ab value of the sent CAS in one state. Below are the meanings of the bits from low to high: d (ab value of the sent CAS in idle state); c (ab value of the sent CAS in pickup state); b (ab value of the sent CAS upon the start of flash signal); a (ab value of the sent CAS at the end of flash signal).
Description	Sets the ab value of the sent CAS in four states: idle, pickup, the start of flash signal and the end of flash signal.
Note	This configuration item is applicable to the SS1 channel using the LineSide protocol, being valid only when the configuration item ProtocolType is set to 1 or 3.

3.1.2.27.8 Advanced Setting for SS7

3.1.2.27.8.1 Setting TS16 Property

3.1.2.27.8.1.1 UseTS16AsCircuit

Refer to **Ss7SignalingTS**

3.1.2.27.8.1.2 Ss7SignalingTS

Configuration Item	**UseTS16AsCircuit** **Ss7SignalingTS**
Section	**[BoardId=x]**
Written Format	UseTS16AsCircuit[n]=b Ss7SignalingTS[n]=k

Value Range	n: The physical digital trunk number, with the value range of 0~M-1. M is the set value of the configuration item PcmNumber. b: Determines whether this digital trunk contains SS7 signaling links. b=0: Yes (default). The configuration item Ss7SignalingTS determines the specific time slot to provide signaling links; b=1: No. The time slot specified by the configuration item Ss7SignalingTS can serve as neither the signaling link nor the voice path. k: The time slot number, with the default value of 16. 1) For the driver versions below SynCTI 4.5.8.0, k has the fixed value of 16. 2) For the driver versions equal to or above SynCTI 4.5.8.0 and below SynCTI 4.8.0.0, the value of k depends on the board model. If the board model is one of the followings, ✧ SHD-240A-CT/cPCI ✧ SHD-240S-CT/cPCI ✧ SHD-480A-CT/cPCI ✧ SHD-480S-CT/cPCI k must be set to 16. If the board model is one of others in SHD Series, k can be set to 1 or 16. 3) For SynCTI 4.8.0.0 or above, if the board model is one of the followings, ✧ SHD-240A-CT/cPCI ✧ SHD-240S-CT/cPCI ✧ SHD-480A-CT/cPCI ✧ SHD-480S-CT/cPCI k must be set ≥0 and ≤32. When k is equal to 0 or 32, all time slots from TS1 to TS31 on this digital trunk are used as voice paths; other values except 0 and 32 indicate the particular signaling time slot number. If the board model is one of the followings, ✧ SHD-30D-CT/PCI ✧ SHD-60D-CT/PCI ✧ SHD-240D-CT/PCI ✧ SHD-240D-CT/PCI/EC ✧ SHD-120D-CT/PCI ✧ SHD-120D-CT/PCI/EC ✧ SHD-60B-CT/PCI/FJ ✧ All E-type digital trunk boards k must be set >0 and <33. To be exact, In E1 mode, k=32 indicates all the 31 time slots on this PCM are used to deliver voice data. For SHD-60B-CT/PCI/FJ, k=32 indicates all the 31 time slots on the monitored PCM are used to deliver voice data, and this feature is only supported for E1 and SS7. (Note: While using SS1 or ISDN, k is set to 16 by default and cannot be modified.) In T1/J1 mode, 1≤k≤25. k=25 indicates all of the 24 time slots on this PCM are used to deliver voice data. (Note: While using SS1, k is set to 25 by default; while using ISDN, k is set to 24 by default. Both cannot be modified.) If the board model is one of others in SHD Series, k can be set to 1 or 16.
Description	UseTS16AsCircuit is used to set whether this digital trunk contains SS7 signaling links.
Note	● It is an advanced configuration item, only applicable to the SS7 signaling; ● If the value of the configuration item PcmNumber is greater than 1, this configuration item should be set for each digital trunk.

3.1.2.27.8.2 Setting Corresponding Characters for Spare Address Codes

3.1.2.27.8.2.1 AddressSignal

Configuration Item	**AddressSignal**
Section	**[TUP] / [ISUP]**
Written Format	AddressSignal[n]=c

Value Range	n: The spare address code in the TUP/ISUP message, 10≤n≤14. c: The character corresponding to n, not allowed to be any one of '0'~'9'. If c includes more than one character, what n corresponds to is the first character of c.
Description	Sets the corresponding character for each spare address code. See the Setting Corresponding Characters for Spare Address Codes section in Chapter 1 for more information.
Note	● This configuration item is only applicable to the TUP/ISUP channel. ● The ISUP channel uses the configuration section **[ISUP]** while the TUP channel uses **[TUP]**. ● Example: AddressSignal[11]=b.

3.1.2.27.8.3 Setting IP Address of SS7 Server

3.1.2.27.8.3.1 Ss7ServerIP

Refer to **SecondServerIP**

3.1.2.27.8.3.2 SecondServerIP

Configuration Item	**Ss7ServerIP** **SecondServerIP**
Section	**[SS7]**
Written Format	Ss7ServerIP=a.b.c.d SecondServerIP=a.b.c.d
Value Range	a.b.c.d: The address of the SS7 server, with the default value of 127.0.0.1 (indicating there is only one SS7 server available, running with the application program on the local PC).
Description	Sets the address of the SS7 server.
Note	If both the SS7 server and the application program are running on the local PC, these two configuration items are not necessarily set; if the system contains two SS7 server, Ss7ServerIP and SecondServerIP are respectively used to set the IP address of the master server and that of the slave server.
Example	Ss7ServerIP=201.123.123.1

3.1.2.27.8.4 Setting IP Address of Local PC

3.1.2.27.8.4.1 LocalIP

Configuration Item	**LocalIP**
Section	**[SS7]**
Written Format	LocalIP=a.b.c.d
Value Range	a.b.c.d: The address of the local PC, with the default value of 127.0.0.1 (indicating both the SS7 server and the application program are running on the local PC).
Description	Sets the IP address of the SS7 server or the gateway.
Example	LocalIP=201.123.123.5

3.1.2.27.8.5 SS7 Server Outputting Data on Voice Time Slot

3.1.2.27.8.5.1 LoadShp_a3AsSIU

Configuration Item	**LoadShp_a3AsSIU**
Section	**[SystemConfig]**
Written Format	LoadShp_a3AsSIU=b

Value Range	b=0: No (default); b=1: Yes.
Description	Sets whether the SS7 server outputs the data on voice time slots. When the SS7 server is used not to run any service system but to provide SS7 service, it only process the data on signaling time slots of the digital trunk but not those on voice time slots. If the digital trunk that connects to the SS7 server has voice time slots on it besides the signaling links, the SS7 server is required to exchange the data from a voice time slot to another and then output the data from there. Such an operation is based on the two-way connection between all voice time slots of the digital trunk with the physical number of 0 and the corresponding voice time slots of the digital trunk with the physical number of 1. Refer to the Supplying SS7 Service for Third-party Board section in Chapter 4 for more information.
Note	It is only applicable to the SHD-60A-CT/PCI/SS7 board.

3.1.2.27.8.6 Setting SS7 Circuit for Digital Trunk

3.1.2.27.8.6.1 Ss7CircuitMap[pcm]

Configuration Item	**Ss7CircuitMap[pcm]**
Section	**[BoardId=x]**
Written Format	Ss7CircuitMap[pcm]=b
Value Range	pcm: The physical number of the digital trunk on the board. For more information, refer to related manuals. Range of value: 0≤pcm≤31; b: composed of 32 bits, with the default value of 0xFFFFFFFF. The values of bit0-bit31 respectively indicate whether TS0-TS31 for a particular PCM are used as SS7 circuits: 0=not used as SS7 ciruit; 1=used as SS7 circuit. In an E1 system, TS0 and the time slot for processing SS7 signaling will be ignored.
Description	Sets the corresponding relationship between SS7 circuits and time slots for a particular PCM. The time slot which is set to 'not used as SS7 circuit' will not be reset by the driver and the inbound call from the other end to this time slot will be blocked. Meanwhile, the corresponding channel will turn to the 'unused' state.
Example	Ss7CircuitMap[0]= 0xFFFFFFFF

3.1.2.27.8.7 Advanced Configuration Item for TUP

3.1.2.27.8.7.1 Setting Range Field in Circuit Group Message

3.1.2.27.8.7.1.1 SendGRMRange

Configuration Item	**SendGRMRange**
Section	**[TUP]**
Written Format	SendGRMRange=m
Value Range	m=0: Use all-0 field. That is, the time slot number in the CIC field of the circuit group message is 1 and the Range field is 0, indicating this circuit group message covers all time slots TS1~31; m=1: Use not all-0 field. The range of time slots covered by the circuit group message depends on the configuration item Ss7SignalingTS. Assuming the set value of Ss7SignalingTS is N: ✧ If N=1, the time slot number in the CIC field of the circuit group message is 2 and the Range field is 29; ✧ If N=16, the circuit group message will be divided into 2 messages: the time slot number in the CIC field of one message is 1 and the Range field is 14; the time slot number in the CIC field of the other message is 17 and the Range field is 14.
Description	This configuration item is used to set the range of time slots covered by the circuit group message when the local driver sends the message to the remote PBX.

Note	This configuration item will affect all circuit group messages, including the circuit group reset message and the circuit group blocking/unblocking message.

3.1.2.27.8.7.1.2HangupRingSendCBK

Configuration Item	**HangupRingSendCBK**
Section	**[TUP]**
Written Format	HangupRingSendCBK=m
Value Range	m=0: Send the CFL message (default); m=1: Send the CBK message.
Description	When the channel stays in the S_CALL_RINGING state, the function call of SsmHangup or SsmHangupEx by the application will prompt the driver to send the call rejection message to the remote PBX. The call rejection message can be CBK or CFL. It is this configuration item that determines which one is used exactly.
Note	It requires SynCTI Ver. 4.7.1.5 or above.

3.1.2.27.8.7.2 Incoming Call: Customizing ACM Message

3.1.2.27.8.7.2.1DefaultACM

Configuration Item	**DefaultACM**
Section	**[TUP]**
Written Format	DefaultACM=0xab
Value Range	ab：8-bit data, represented by hexadecimal digits. The 8 bits are arranged from high to low as **HGFEDCBA**. **Bit** / **Implication** / **Value** / **Description** **HG** / **Spare** **F** / Signaling Channel Indicator / 0 / Any channel 1 / All are SS7 channels **E** / Call Forwarding Indicator / 0 / Call non-forwarding 1 / Call forwarding **D** / Incoming Echo Suppression Indicator / 0 / Incoming half-echo suppressor not included 1 / Incoming half-echo suppressor included **C** / Subscriber Free Indicator / 0 / Not instructed 1 / The subscriber is free. **BA** / Address Complete Signal Type Indicator / 00 / Address complete signal 01 / Address complete signal, charge 10 / Address complete signal, no charge 11 / Address complete signal, payphone The default value is 0x05.
Description	In the incoming call, the local end is required to send the ACM message to the remote PBX after receiving the complete called party number and other information. This ACM message should contain the message indicator field to indicate the user's status which is set via this configuration item. Refer to the TUP Channel State Machine section in Chapter 1 for more information.
Example	DefaultACM=0x05

3.1.2.27.8.7.3 Incoming Call: Customizing GRQ Message

3.1.2.27.8.7.3.1ReqTypeIndicators

Configuration Item	**ReqTypeIndicators**
Section	**[TUP]**
Written Format	ReqTypeIndicators=0xk

<table>
<tr><td>Value Range</td><td>k: 8-bit data, represented by hexadecimal digits. The 8 bits are arranged from high to low as HGFEDCBA.
<table>
<tr><th>Bit</th><th>Implication</th><th>Value</th><th>Description</th></tr>
<tr><td>HG</td><td>Spare</td><td></td><td></td></tr>
<tr><td rowspan="2">F</td><td rowspan="2">Echo Suppressor Request Indicator (also is applicable to the echo canceller)</td><td>0</td><td>Not requested</td></tr>
<tr><td>1</td><td>Requested</td></tr>
<tr><td rowspan="2">E</td><td rowspan="2">Request Holding Indicator</td><td>0</td><td>Not requested</td></tr>
<tr><td>1</td><td>Requested</td></tr>
<tr><td rowspan="2">D</td><td rowspan="2">Malicious Call Identification Indicator</td><td>0</td><td>Not encountered</td></tr>
<tr><td>1</td><td>Encountered</td></tr>
<tr><td rowspan="2">C</td><td rowspan="2">Original Called Party's Address Indicator</td><td>0</td><td>Not requested</td></tr>
<tr><td>1</td><td>Requested</td></tr>
<tr><td rowspan="2">B</td><td rowspan="2">Calling Line Identification Indicator</td><td>0</td><td>Not requested</td></tr>
<tr><td>1</td><td>Requested</td></tr>
<tr><td rowspan="2">A</td><td rowspan="2">Calling Party's Category Indicator</td><td>0</td><td>Not requested</td></tr>
<tr><td>1</td><td>Requested</td></tr>
</table>
The default value is 0x03.</td></tr>
<tr><td>Description</td><td>Sets the request type indicator in the GRQ message.</td></tr>
<tr><td>Example</td><td>ReqTypeIndicators=0x03</td></tr>
</table>

3.1.2.27.8.7.4 Outgoing Call: Customizing IAI/IAM Message

3.1.2.27.8.7.4.1 ConnectReqMsg

Configuration Item	**ConnectReqMsg**
Section	**[TUP]**
Written Format	ConnectReqMsg=m
Value Range	m=0: Automatically selected by the driver. If the driver's internal outgoing Caller ID buffer (configurable via the function SsmSetTxCallerId or the configuration item TxCallerId) or original Callee ID buffer (configurable via the function SsmSetTxOriginalCallerID) is not empty, use the IAI message; otherwise, use the IAM message. m=1: Use the IAM message. m=2: Use the IAI message. The default value is 0.
Description	This configuration item is to set the initial address message type used by the local end to start an outgoing call.

3.1.2.27.8.7.4.2 CalloutIAM_CAT

Configuration Item	**CalloutIAM_CAT**
Section	**[TUP]**
Written Format	CalloutIAM_CAT=0xk

<table>
<tr><td>Value Range</td><td>k: 8-bit data, represented by hexadecimal digits. The 8 bits are arranged from high to low as HGFEDCBA.
<table>
<tr><th colspan="4">Low bits: HGFEDCBA</th></tr>
<tr><th>Bit</th><th>Implication</th><th>Value</th><th>Description</th></tr>
<tr><td>HG</td><td>Spare</td><td>00</td><td></td></tr>
<tr><td rowspan="32">FEDCBA</td><td rowspan="32">Calling Party’s Category</td><td>000000</td><td>Unknown, used in international semi-automatic working</td></tr>
<tr><td>000001</td><td>Operator, language French, used in international semi-automatic working</td></tr>
<tr><td>000010</td><td>Operator, language English, used in international semi-automatic working</td></tr>
<tr><td>000011</td><td>Operator, language German, used in international semi-automatic working</td></tr>
<tr><td>000100</td><td>Operator, language Russian, used in international semi-automatic working</td></tr>
<tr><td>000101</td><td>Operator, language Spanish, used in international semi-automatic working</td></tr>
<tr><td>000110</td><td>A particular language selected by mutual agreement (Chinese), used in international semi-automatic working</td></tr>
<tr><td>000111</td><td>A particular language selected by mutual agreement, used in international semi-automatic working</td></tr>
<tr><td>001000</td><td>A particular language selected by mutual agreement (Japanese), used in international semi-automatic working</td></tr>
<tr><td>001001</td><td>National operator (presentation allowed)</td></tr>
<tr><td>001010</td><td>Ordinary subscriber, used in international-to-long-distance, international-to-local network</td></tr>
<tr><td>001011</td><td>Subscriber with priority, used in international-to-long-distance, international-to-local, local-to-local network</td></tr>
<tr><td>001100</td><td>Data call</td></tr>
<tr><td>001101</td><td>Test call</td></tr>
<tr><td>001110</td><td>001110~001111 Spare</td></tr>
<tr><td>010000</td><td>Ordinary, free, used in local-to-international network</td></tr>
<tr><td>010001</td><td>Ordinary, regular, used in local-to-international network</td></tr>
<tr><td>010010</td><td>Ordinary, subscriber list, instant, used in local-to-international network</td></tr>
<tr><td>010011</td><td>Ordinary, printer, instant, used in local-to-international network</td></tr>
<tr><td>010100</td><td>With priority, free, used in local-to-international network</td></tr>
<tr><td>010101</td><td>With priority, regular, used in local-to-international network</td></tr>
<tr><td>010110</td><td>Spare</td></tr>
<tr><td>010111</td><td>Spare</td></tr>
<tr><td>011000</td><td>Ordinary subscriber, used in local-to-local network</td></tr>
<tr><td>011001</td><td>011001~111111 Spare</td></tr>
</table>
The default value is 0x18。</td></tr>
<tr><td>Description</td><td>Sets the calling party’s category field in the IAI/IAM message.</td></tr>
<tr><td>Example</td><td>CalloutIAM_CAT=0x18</td></tr>
</table>

3.1.2.27.8.7.4.3CalloutIAM_MsgPntr

Configuration Item	**CalloutIAM_MsgPntr**
Section	**[TUP]**
Written Format	CalloutIAM_MsgPntr=0xabcd

<table>
<tr><td>Value Range</td><td>abcd: 16-bit data, represented by hexadecimal digits, the low 12 bits being valid. Bit11~Bit0 are written as LKJIHGFEDCBA, each of which is described below.
<table>
<tr><th>Bit</th><th>Implication</th><th>Value</th><th>Description</th></tr>
<tr><td rowspan="2">H</td><td rowspan="2">Incoming International Call Indicator</td><td>0</td><td>Not incoming international call</td></tr>
<tr><td>1</td><td>Incoming international call</td></tr>
<tr><td rowspan="2">G</td><td rowspan="2">Outgoing Echo Suppressor Indicator</td><td>0</td><td>Outgoing echo suppressor not included</td></tr>
<tr><td>1</td><td>Outgoing echo suppressor included</td></tr>
<tr><td rowspan="4">FE</td><td rowspan="4">Continuity Check Indicator</td><td>00</td><td>Continuity check not required</td></tr>
<tr><td>01</td><td>Continuity check required on this circuit</td></tr>
<tr><td>10</td><td>Continuity check performed on a previous circuit</td></tr>
<tr><td>11</td><td>Spare</td></tr>
<tr><td rowspan="3">DC</td><td rowspan="3">Nature of Circuit Indicator</td><td>00</td><td>No satellite circuit in the connection</td></tr>
<tr><td>01</td><td>satellite circuit available in the connection</td></tr>
<tr><td>Others</td><td>Spare</td></tr>
<tr><td rowspan="4">BA</td><td rowspan="4">Nature of Address Indicator</td><td>00</td><td>Local subscriber number</td></tr>
<tr><td>01</td><td>Spare</td></tr>
<tr><td>10</td><td>Valid national number</td></tr>
<tr><td>11</td><td>International number</td></tr>
<tr><td>L</td><td>Spare</td><td></td><td></td></tr>
<tr><td rowspan="2">K</td><td rowspan="2">Signaling Channel Indicator</td><td>0</td><td>Any channel</td></tr>
<tr><td>1</td><td>All are SS7 channels</td></tr>
<tr><td rowspan="2">J</td><td rowspan="2">All Digital Channel Required Indicator</td><td>0</td><td>Ordinary call</td></tr>
<tr><td>1</td><td>All digital channels required</td></tr>
<tr><td rowspan="2">I</td><td rowspan="2">Call Transfer Indicator</td><td>0</td><td>Call transfer required</td></tr>
<tr><td>1</td><td>Call transfer not required</td></tr>
</table>
The default value is 0x0000。</td></tr>
<tr><td>Description</td><td>Sets the message indicator field in the IAI/IAM message.</td></tr>
<tr><td>Example</td><td>CalloutIAM_MsgPntr=0x0000</td></tr>
</table>

3.1.2.27.8.7.4.4CallingIndicatorBit

Configuration Item	**CallingIndicatorBit**
Section	**[TUP]**
Written Format	CallingIndicatorBit=n
Value Range	n=0x10: Use Bit E； n=0x04: Use Bit C (default)
Description	There is a first indicator (octet) in the IAI message, Bit7~Bit0 being written as HGFEDCBA, among which Bit C is the calling party information indicator, Bit E is the calling line identification indicator. In case the IAI message contains the calling party number, some PBXes requires the use of Bit E, some requires the use of Bit C. Which one is exactly used can be set via this configuration item depending on the remote PBX's requirement.
Example	CallingIndicatorBit=0x10

3.1.2.27.8.7.4.5OriginalCalleeAddrInd

Configuration Item	**OriginalCalleeAddrInd**
Section	**[TUP]**
Written Format	OriginalCalleeAddrInd=0xn

<table>
<tr><td>Value Range</td><td>n: 8-bit data, represented by hexadecimal digits, the low 4 bits being valid. Bit3~Bit0 are written as DCBA, each of which is described below.
<table>
<tr><th>Bit</th><th>Implication</th><th>Value</th><th>Description</th></tr>
<tr><td>DC</td><td>Spare</td><td>00</td><td></td></tr>
<tr><td rowspan="4">BA</td><td rowspan="4">Nature of Address Indicator</td><td>00</td><td>Local subscriber number</td></tr>
<tr><td>01</td><td>Spare national number</td></tr>
<tr><td>10</td><td>Valid national number</td></tr>
<tr><td>11</td><td>International number</td></tr>
</table>
The default value is 0x02.</td></tr>
<tr><td>Description</td><td>Sets the address indicator in the original called party address field of the IAI message.</td></tr>
<tr><td>Example</td><td>OriginalCalleeAddrInd=0x02</td></tr>
</table>

3.1.2.27.8.7.4.6CallerAddrInd

<table>
<tr><td>Configuration Item</td><td>CallerAddrInd</td></tr>
<tr><td>Section</td><td>[TUP]</td></tr>
<tr><td>Written Format</td><td>CallerAddrInd =0xn</td></tr>
<tr><td>Value Range</td><td>n: Address indicator, 8-bit and hexadecimal, the low 4 bits being valid. Bit3~Bit0 are written as DCBA, each of which is described below.
<table>
<tr><th>Bit</th><th>Implication</th><th>Value</th><th>Description</th></tr>
<tr><td rowspan="2">D</td><td rowspan="2">Calling party number incomplete indicator</td><td>0</td><td>Not incomplete</td></tr>
<tr><td>1</td><td>Incomplete</td></tr>
<tr><td rowspan="2">C</td><td rowspan="2">Calling party number presentation indicator</td><td>0</td><td>Presentation unrestricted</td></tr>
<tr><td>1</td><td>Presentation restricted</td></tr>
<tr><td rowspan="4">BA</td><td rowspan="4">Nature of address indicator</td><td>00</td><td>Local subscriber number</td></tr>
<tr><td>01</td><td>Spare national number</td></tr>
<tr><td>10</td><td>Valid national number</td></tr>
<tr><td>11</td><td>International number</td></tr>
</table>
The default value is 0x02.</td></tr>
<tr><td>Description</td><td>Sets the address indicator in the calling line identification field in the IAI message.</td></tr>
<tr><td>Example</td><td>CallerAddrInd =0x02</td></tr>
<tr><td>Note</td><td>● This configuration item requires SynCTI Ver. 5.3.1.2 or above.</td></tr>
</table>

3.1.2.27.8.7.5 Outgoing Call: Setting Way to Reply with GRQ Message

3.1.2.27.8.7.5.1AutoSendGSM

<table>
<tr><td>Configuration Item</td><td>AutoSendGSM</td></tr>
<tr><td>Section</td><td>[TUP]</td></tr>
<tr><td>Written Format</td><td>AutoSendGSM=m</td></tr>
<tr><td>Value Range</td><td>m=0: Taken over by the application. The outgoing call is resumed after the channel state transfers to ‘Pending’ and the application program invoke the function SsmSetTxCallerId to set the calling party number.
m=1: Automatically replied by the driver (default).</td></tr>
<tr><td>Description</td><td>In the outgoing call, once the GRQ message is received from the remote PBX, this configuration item is used to determine whether this message is automatically replied by the driver via sending the GSM message. See the TUP Channel State Machine section in Chapter 1 for more information.</td></tr>
</table>

3.1.2.27.8.7.6 Not Using SynCTI-provided TUP State Machine

3.1.2.27.8.7.6.1AutoHandleTup

<table>
<tr><td>Configuration Item</td><td>AutoHandleTup</td></tr>
<tr><td>Section</td><td>[SS7]</td></tr>
<tr><td>Written Format</td><td>AutoHandleTup=b</td></tr>
</table>

Value Range	b=0: No; b=1: Yes (default).
Description	Sets whether to use the TUP state machine provided by the SynCTI driver or not.

3.1.2.27.8.7.7 Outputting Debugging Information

3.1.2.27.8.7.7.1 DebugViewTupCallProc

Configuration Item	**DebugViewTupCallProc**
Section	**[SS7]**
Written Format	DebugViewTupCallProc=b
Value Range	b=0: No (default); b=1: Yes.
Description	Sets whether to output the debugging information of the TUP state machine to the software tool DebugView. If Yes, DebugView is required to run first.
Note	Enabling the TUP debugging feature will reduce the system running efficiency. Hence this feature is only used for debugging.

3.1.2.27.8.8 Advanced Configuration Item for ISUP

3.1.2.27.8.8.1 Incoming Call: Customizing Message Type Used by Local End to Reply Incoming Call

3.1.2.27.8.8.1.1 DefaultCalledPickupMsg

Configuration Item	**DefaultCalledPickupMsg**
Section	**[ISUP]**
Written Format	DefaultCalledPickupMsg=k
Value Range	k=0x09: Use the ANM message (response); k=0x07: Use the CON message (address complete and phone picked up) The default value is 0x09.
Description	This configuration item is used to set the type of the message sent to the remote PBX when the ISUP channel stays in the 'Ringing' state and the application invokes the function SsmPickup or SsmPickupANX.
Note	It requires SynCTI Ver. 4.7.1.8 or above.
Example	DefaultCalledPickupMsg=0x09

3.1.2.27.8.8.2 Incoming Call: Customizing Backward Call Indicator

3.1.2.27.8.8.2.1 DefaultBackwardCallInd

Configuration Item	**DefaultBackwardCallInd**
Section	**[ISUP]**
Written Format	DefaultBackwardCallInd=k

<table>
<tr><td>Value Range</td><td>k: The parameter value of the backward call indicator field, including 2 bytes. The bits arranged from high (Bit15) to low (Bit0) are written as PONMLKJIHGFEDCBA, each of which is described below.
<table>
<tr><th>Bit</th><th>Implication</th><th>Value and Description</th></tr>
<tr><td>BA</td><td>Charge Indicator</td><td>00: No indication
01: No charge
10: Charge
11: Spare</td></tr>
<tr><td>DC</td><td>Called Party’s Status Indicator</td><td>00: No indication
01: Subscriber free
10: Connect when free
11: Spare</td></tr>
<tr><td>FE</td><td>Called Party’s Category Indicator</td><td>00: No indication
01: Ordinary subscriber
10: payphone
11: Spare</td></tr>
<tr><td>HG</td><td>End-to-end Method Indicator (Note)</td><td>00: No end-to-end method available (only link-by-link method available)
01: Pass-along method available
10: SCCP method available
11: Pass-along and SCCP methods available</td></tr>
<tr><td>I</td><td>Interworking Indicator (Note)</td><td>0: No interworking encountered
1: Interworking encountered</td></tr>
<tr><td>J</td><td>End-to-end Information Indicator (Note)</td><td>0: No end-to-end information available
1: End-to-end information available</td></tr>
<tr><td>K</td><td>ISDN User Part Indicator (Note)</td><td>0: ISDN user part not used all the way
1: ISDN user part used all the way</td></tr>
<tr><td>L</td><td>Holding Indicator</td><td>0: Holding not requested
1: Holding requested</td></tr>
<tr><td>M</td><td>ISDN Access Indicator</td><td>0: Terminating access non-ISDN
1: Terminating access ISDN</td></tr>
<tr><td>N</td><td>Echo Control Device Indicator</td><td>0: Incoming half-echo control device not included
1: Incoming half-echo control device included</td></tr>
<tr><td>PO</td><td>SCCP Method Indicator (Note)</td><td>00: No indication
01: Connectionless method available
10: Connection oriented method available
11: Connectionless and connection oriented methods available</td></tr>
</table>
NOTE: Bits G-K and O-P constitute the protocol control indicator.
The default value is 0x1416.</td></tr>
<tr><td>Description</td><td>Sets the backward call indicator field in the ACM and CON messages.</td></tr>
<tr><td>Note</td><td>It requires SynCTI Ver. 4.7.1.8 or above.</td></tr>
<tr><td>Example</td><td>DefaultBackwardCallInd=0x1416</td></tr>
</table>

3.1.2.27.8.8.3 Incoming Call: Customizing REL Message

3.1.2.27.8.8.3.1 DefaultHangupRELInd

Configuration Item	**DefaultHangupRELInd**
Section	**[ISUP]**
Written Format	DefaultHangupRELInd=0xk

<table>
<tr><td>Value Range</td><td>The values and corresponding meanings of k are listed in the table below.
<table>
<tr><th>Value</th><th>Macro in shpa3api.h</th><th>Description</th></tr>
<tr><td>0x01</td><td>C_ISUP_REL_UNN</td><td>Number unallocated</td></tr>
<tr><td>0x10</td><td>C_ISUP_REL_NORMAL_REL</td><td>Normal release</td></tr>
<tr><td>0x11</td><td>C_ISUP_REL_BUSY</td><td>User busy</td></tr>
<tr><td>0x12</td><td>C_ISUP_REL_NOREPLY</td><td>No answer</td></tr>
<tr><td>0x15</td><td>C_ISUP_REL_DENY</td><td>Refused</td></tr>
<tr><td>0x1c</td><td>C_ISUP_REL_LACKADDR</td><td>Address incomplete</td></tr>
<tr><td>0x1f</td><td>C_ISUP_REL_NORMAL</td><td>Normal</td></tr>
<tr><td>0x2a</td><td>C_ISUP_REL_BLOCK</td><td>Exchange device blocked</td></tr>
</table>
The default value is 0x15 (C_ISUP_REL_DENY).</td></tr>
<tr><td>Description</td><td>This configuration item is used to set the release cause value carried by the REL message when a call comes into the channel but has not yet been picked up, and the driver is prompted by the function call of SsmHangup to send the REL message to the remote PBX.</td></tr>
<tr><td>Note</td><td>● If the set value is out of the range shown above, the driver will automatically adjust the value to 0x15;
● It requires SynCTI Ver. 4.7.1.5 or above.</td></tr>
<tr><td>Example</td><td>DefaultHangupRELInd=0x15</td></tr>
</table>

3.1.2.27.8.8.3.2DefaultCauseInd

<table>
<tr><td>Configuration Item</td><td>DefaultCauseInd</td></tr>
<tr><td>Section</td><td>[ISUP]</td></tr>
<tr><td>Written Format</td><td>DefaultCauseInd=n</td></tr>
<tr><td>Value Range</td><td>n: 16-bit data, represented by hexadecimal digits. Each bit in it is described below.
<table>
<tr><td colspan="4">Low-bit Byte: HGFEDCBA</td></tr>
<tr><th>Bit</th><th>Implication</th><th>Value</th><th>Description</th></tr>
<tr><td rowspan="2">H</td><td rowspan="2">Extension Indicator</td><td>0</td><td>Information continues in the next octet (octet suggested)</td></tr>
<tr><td>1</td><td>Last octet</td></tr>
<tr><td rowspan="2">GF</td><td rowspan="2">Coding Standard</td><td>00</td><td>ITU-T standard code</td></tr>
<tr><td>Others</td><td>Reserved</td></tr>
<tr><td>E</td><td>Spare</td><td>0</td><td>Should be set to 0</td></tr>
<tr><td rowspan="10">DCBA</td><td rowspan="10">Location</td><td>0000</td><td>User (U)</td></tr>
<tr><td>0001</td><td>Local private network (LPN)</td></tr>
<tr><td>0010</td><td>Local public network (LN)</td></tr>
<tr><td>0011</td><td>Transfer network (TN)</td></tr>
<tr><td>0100</td><td>Remote public network (RLN)</td></tr>
<tr><td>0101</td><td>Remote private network (RPN)</td></tr>
<tr><td>0111</td><td>Internet(INTL)</td></tr>
<tr><td>1010</td><td>Network formed by points except two connected ones on the SS7 public network (BI)</td></tr>
<tr><td>Others</td><td>Spare</td></tr>
<tr><td colspan="4">High-bit Byte: HGFEDCBA</td></tr>
<tr><th>Bit</th><th>Implication</th><th>Value</th><th>Description</th></tr>
<tr><td>H</td><td>Extension Indicator</td><td>1</td><td>Last octet</td></tr>
<tr><td rowspan="5">GFEDCBA</td><td rowspan="5">Recommendation</td><td>0000000</td><td>Q.931</td></tr>
<tr><td>0000011</td><td>X.21</td></tr>
<tr><td>0000100</td><td>X.25</td></tr>
<tr><td>0000101</td><td>Public Land Mobile Network (PLMN), Q.1031/Q.1051</td></tr>
<tr><td>Others</td><td>Reserved</td></tr>
</table>
The default value is 0x0000.</td></tr>
<tr><td>Description</td><td>Sets the coding standard, location and recommendation parameters for the cause indicator field. The recommendation parameter is optional: if not selected, the extension octet of the low 8 bits must be set to 1 and the value of the high 8 bits will be ignored by the driver; If selected, the extension octet of the low 8 bits must be set to 0.</td></tr>
<tr><td>Example</td><td>DefaultCauseInd=0x0000</td></tr>
</table>

3.1.2.27.8.8.4 Incoming Call: Setting Other Parameters

3.1.2.27.8.8.4.1 SaveRGNTo1stPhoNumStr

Configuration Item	**SaveRGNTo1stPhoNumStr**
Section	**[ISUP]**
Written Format	SaveRGNTo1stPhoNumStr=b
Value Range	b=0: No; b=1: Yes (default).
Description	Sets whether the functions SsmGet1stPhoNumStr and SsmGet1stPhoNumStrA only return the original called party number information in the IAM message. See description on those two functions for more information.
Note	● This configuration item is only applicable to the ISUP channel; ● It requires SynCTI Ver 4.7.1.7 or above.

3.1.2.27.8.8.5 Outgoing Call: Customizing IAM Message

3.1.2.27.8.8.5.1 DefaultNatureOfConnectionInd

Configuration Item	**DefaultNatureOfConnectionInd**
Section	**[ISUP]**
Written Format	DefaultNatureOfConnectionInd=0xk
Value Range	k: The nature of connection indicator, represented by hexadecimal digits, with the default value of 0x00. Regarding the specific value, see Appendix 2 Optional ISUP Parameters and Descriptions. Usually it is provided by the remote PBX according to the particular configuration.
Description	Sets the nature of connection indicator in the IAM message.
Example	DefaultNatureOfConnectionInd=0x00

3.1.2.27.8.8.5.2 DefaultIAM_ForwardCallInd

Configuration Item	**DefaultIAM_ForwardCallInd**
Section	**[ISUP]**
Written Format	DefaultIAM_ForwardCallInd=0xk
Value Range	k: The forward call indicator, represented by hexadecimal digits, with the default value of 0x0040. Regarding the specific value, see Appendix 2 Optional ISUP Parameters and Descriptions. Usually it is provided by the remote PBX according to the particular configuration.
Description	Sets the forward call indicator in the IAM message.
Example	DefaultIAM_ForwardCallInd=0x0040

3.1.2.27.8.8.5.3 DefaultIAM_CAT

Configuration Item	**DefaultIAM_CAT**
Section	**[ISUP]**
Written Format	DefaultIAM_CAT=0xk
Value Range	k: The calling party's category indicator, with the default value of 0x0a. Regarding the specific value, see Appendix 2 Optional ISUP Parameters and Descriptions. Usually it is provided by the remote PBX according to the particular configuration.
Description	Sets the calling party's category indicator in the IAM message.
Example	DefaultIAM_CAT=0x0a

3.1.2.27.8.8.5.4 DefaultIAM_TransmissionMediumRequirment

Configuration Item	**DefaultIAM_TransmissionMediumRequirment**

Section	**[ISUP]**
Written Format	Format 1: DefaultIAM_TransmissionMediumRequirment=0xk Format 2: DefaultIAM_TransmissionMediumRequirment[N]=0xk
Value Range	k: The transmission medium requirement parameter, represented by hexadecimal digits, with the default value of 0x02. Regarding the specific value, see Appendix 2 Optional ISUP Parameters and Descriptions. Usually it is provided by the remote PBX according to the particular configuration. N: The PCM number for the link it lies on. If the whole system uses the same transmission medium, use Format 1; if links in the system use different transmission medium, use Format 2, remaining Format 1 and setting k to 0xff.
Description	Sets the transmission medium requirement parameter in the IAM message.
Note	Don't forget to remain Format 1 and set k to 0xff whiling using Format 2.
Example	Format 1: DefaultIAM_TransmissionMediumRequirment=0x02 Format 2: DefaultIAM_TransmissionMediumRequirment=0xff; must set DefaultIAM_TransmissionMediumRequirment[0]=0x02; use 0x02 for PCM 0,1 DefaultIAM_TransmissionMediumRequirment[1]=0x02 DefaultIAM_TransmissionMediumRequirment[2]=0x04; use 0x04 for PCM 2,3 DefaultIAM_TransmissionMediumRequirment[3]=0x04

3.1.2.27.8.8.5.5DefaultIAM_CalleeParam

Configuration Item	**DefaultIAM_CalleeParam**
Section	**[ISUP]**
Written Format	DefaultIAM_CalleeParam=0xk
Value Range	k: 16-bit data, represented by hexadecimal digits. Each bit in it is described below.

Low-bit Byte: HGFEDCBA

Bit	Implication	Value	Description
H	Odd/even Indicator	0	Even number of address signals
		1	Odd number of address signals
GFEDCBA	Nature of Address Indicator	0000001	Subscriber number
		0000011	National number
		0000100	International number
		Others	Spare

High-bit Byte: HGFEDCBA

Bit	Implication	Value	Description
H	Internal Network Number Indicator	0	Routing to internal network number allowed
		1	Routing to internal network number not allowed
GFE	Numbering Plan Indicator	001	ISDN (Telephony) numbering plan (Recommendations E.164, E.163)
		011	Data numbering plan(Recommendation X.164)
		100	Telex numbering plan (Recommendation F.164)
		Others	Spare
DCBA	Spare		

The default value is 0x1003.

Description	Sets the called party number parameter field in the IAM message.
Example	DefaultIAM_CalleeParam=0x1003

3.1.2.27.8.8.5.6DefaultIAM_CallerParam

Configuration Item	**DefaultIAM_CallerParam**
Section	**[ISUP]**
Written Format	DefaultIAM_CallerParam=0xk

<table>
<tr><td>Value Range</td><td>k:16-bit data, represented by hexadecimal digits. Each bit in it is described below.
<table>
<tr><td colspan="4">Low-bit Byte: HGFEDCBA</td></tr>
<tr><th>Bit</th><th>Implication</th><th>Value</th><th>Description</th></tr>
<tr><td>H</td><td>Odd/even Indicator</td><td>0</td><td>Even number of address signals</td></tr>
<tr><td></td><td></td><td>1</td><td>Odd number of address signals</td></tr>
<tr><td>GFEDCBA</td><td>Nature of Address Indicator</td><td>0000001</td><td>Subscriber number</td></tr>
<tr><td></td><td></td><td>0000011</td><td>National number</td></tr>
<tr><td></td><td></td><td>0000100</td><td>International number</td></tr>
<tr><td></td><td></td><td>Others</td><td>Spare</td></tr>
<tr><td colspan="4">High-bit Byte: HGFEDCBA</td></tr>
<tr><th>Bit</th><th>Implication</th><th>Value</th><th>Description</th></tr>
<tr><td>H</td><td>Number Incomplete Indicator</td><td>0</td><td>Complete</td></tr>
<tr><td></td><td></td><td>1</td><td>Incomplete</td></tr>
<tr><td>GFE</td><td>Numbering Plan Indicator</td><td>001</td><td>ISDN (Telephony) numbering plan (Recommendations E.164, E.163)</td></tr>
<tr><td></td><td></td><td>011</td><td>Data numbering plan(Recommendation X.164)</td></tr>
<tr><td></td><td></td><td>100</td><td>Telex numbering plan (Recommendation F.164)</td></tr>
<tr><td></td><td></td><td>Others</td><td>Spare</td></tr>
<tr><td>DC</td><td>Address Presentation Restricted Indicator</td><td>00</td><td>Presentation allowed</td></tr>
<tr><td></td><td></td><td>01</td><td>Presentation restricted</td></tr>
<tr><td></td><td></td><td>10</td><td>Address not available</td></tr>
<tr><td></td><td></td><td>Others</td><td>Spare</td></tr>
<tr><td>BA</td><td>Screening Indicator</td><td>00</td><td>User provided, not verified (national use)</td></tr>
<tr><td></td><td></td><td>01</td><td>User reserved, verified and passed</td></tr>
<tr><td></td><td></td><td>10</td><td>User provided, verified and not passed (national use)</td></tr>
<tr><td></td><td></td><td>11</td><td>Network provided</td></tr>
</table>
The default value is 0x1001.</td></tr>
<tr><td>Description</td><td>Sets the calling party number parameter field in the IAM message.</td></tr>
<tr><td>Example</td><td>DefaultIAM_CallerParam=0x1001</td></tr>
</table>

3.1.2.27.8.8.5.7 DefaultIAM_OriginalCalleeParam

Configuration Item	**DefaultIAM_OriginalCalleeParam**
Section	**[ISUP]**
Written Format	DefaultIAM_OriginalCalleeParam=0xk
Value Range	k:16-bit data, represented by hexadecimal digits, with the default value of 0x1001. Regarding the specific value, see Appendix 2 Optional ISUP Parameters and Descriptions.
Description	Sets the first two bytes of the original called party number in the IAM message, including the nature of address indicator, numbering plan indicator and address presentation restricted indicator. Regarding the specific value, see Appendix 2 Optional ISUP Parameters and Descriptions. Usually it is provided by the remote PBX according to the particular configuration.
Example	DefaultIAM_OriginalCalleeParam=0x1001

3.1.2.27.8.8.5.8 DefaultIAM_RedirectingNumber

Configuration Item	**DefaultIAM_RedirectingNumber**
Section	**[ISUP]**
Written Format	DefaultIAM_RedirectingNumber=0xk
Value Range	k:16-bit data, represented by hexadecimal digits, with the default value of 0x1001. Regarding the specific value, see Appendix 2 Optional ISUP Parameters and Descriptions.

Description	Sets the first two bytes of the redirecting number in the IAM message, including the nature of address indicator, numbering plan indicator and address presentation restricted indicator. Regarding the specific value, see Appendix 2 Optional ISUP Parameters and Descriptions. Usually it is provided by the remote PBX according to the particular configuration.
Example	DefaultIAM_RedirectingNumber=0x1001

3.1.2.27.8.8.5.9bSubscriberSI

Refer to SubscriberSI

3.1.2.27.8.8.5.10 SubscriberSI

Configuration Item	**bSubscriberSI** **SubscriberSI**
Section	**[ISUP]**
Written Format	bSubscriberSI=k SubscriberSI=0xk_1, 0xk_2,…0xk_i,…, 0xk_{11}
Value Range	k: =0: Not include (default); =1: Include. The specific value of the user server information parameter is set in the configuration item SubscriberSI. k_i: The user service information parameter, represented by hexadecimal digits, with the default value of 0x80,0x90,0xa3. It is applicable to Huawei PBXes. Up to 11 k_i are allowed to be configured. Regarding the specific value, see Appendix 2 Optional ISUP Parameters and Descriptions. Usually it is provided by the remote PBX according to the particular configuration.
Description	bSubscriberSI is used to set whether the IAM message contains the user service information; SubscriberSI is used to set the content of the user service information, being valid only if bSubscriberSI is set to 1.
Example	bSubscriberSI=1 SubscriberSI=0x80,0x90,0xa3

3.1.2.27.8.8.5.11 bOptionalFCI

Refer to OptionalFCI

3.1.2.27.8.8.5.12 OptionalFCI

Configuration Item	**bOptionalFCI** **OptionalFCI**
Section	**[ISUP]**
Written Format	bOptionalFCI=b OptionalFCI=0xk

<table>
<tr><td>Value Range</td><td>b: =0: Not include (default);
=1: Include.
k: 8-bit data, represented by hexadecimal digits. The bits arranged from high to low are written as HGFEDCBA, each of which is described below.
<table>
<tr><th>Bit</th><th>Implication</th><th>Value</th><th>Description</th></tr>
<tr><td rowspan="2">H</td><td rowspan="2">Connected Line Identity Request Indicator</td><td>0</td><td>Not requested</td></tr>
<tr><td>1</td><td>Requested</td></tr>
<tr><td>GFED</td><td>Spare</td><td></td><td></td></tr>
<tr><td rowspan="2">C</td><td rowspan="2">Simple Segmentation Indicator</td><td>0</td><td>No additional information will be sent</td></tr>
<tr><td>1</td><td>Additional information will be sent in a segmentation message</td></tr>
<tr><td rowspan="4">BA</td><td rowspan="4">Closed User Group Call Indicator</td><td>00</td><td>Non-CUG call</td></tr>
<tr><td>01</td><td>Spare</td></tr>
<tr><td>10</td><td>Closed user group call, outgoing access allowed</td></tr>
<tr><td>11</td><td>Closed user group call, outgoing access not allowed</td></tr>
</table>
The default value is 0. If necessary, it is usually provided by the remote PBX according to the particular configuration</td></tr>
<tr><td>Description</td><td>bOptionalFCI is used to set whether the IAM message contains the optional forward call indicator. If this indicator is contained, its value is specified by the configuration item OptionalFCI.
OptionalFCI is used to set the specific value of the optional forward call indicator.</td></tr>
<tr><td>Example</td><td>bOptionalFCI=1
OptionalFCI=0x01</td></tr>
</table>

3.1.2.27.8.8.5.13 Usr2UsrInfo

Configuration Item	**Usr2UsrInfo**
Section	**[ISUP]**
Written Format	Usr2UsrInfo=s_1,s_2,…,s_i,…,s_N
Value Range	s_i: A character represented by hexadecimal digits. The value is unallocated in default, i.e. this field is not included. Up to 131 s_i are allowed to be configured.
Description	Sets the user-to-user information field in the IAM message. The same purpose can be achieved via the function SsmSetIsupParameter.
Example	Usr2UsrInfo=a0,31,6c

3.1.2.27.8.8.5.14 LocationNumber

Configuration Item	**LocationNumber**
Section	**[ISUP]**
Written Format	LocationNumber =s_1,s_2,…,s_i,…,s_N
Value Range	s_i: A character represented by hexadecimal digits. The value is unallocated in default, i.e. this field is not included. Up to 50 s_i are allowed to be configured.
Description	Sets the LocationNumber information field in the IAM message. The same purpose can be achieved via the function SsmSetIsupParameter.
Example	LocationNumber=0x04,0x13,0x19,0x89,0x45,0x90,0x09,0x20

3.1.2.27.8.8.6 Outgoing Call: Setting Way to Reply with INF Message

3.1.2.27.8.8.6.1AutoSendINF

Configuration Item	**AutoSendINF**
Section	**[ISUP]**
Written Format	AutoSendINF=m

Value Range	m=0: Taken over by the application; m=1: Automatically replied by the driver (default).
Description	In the outgoing call, once the INR message is received from the remote PBX, this configuration item is used to determine whether this message is automatically replied by the driver via sending the INF message. See the ISUP Channel State Machine section in Chapter 1 for details.

3.1.2.27.8.8.7 Outgoing Call: Setting Maximum Waiting Time for ACM Message

3.1.2.27.8.8.7.1 MaxWaitACMTime

Configuration Item	**MaxWaitACMTime**
Section	**[ISUP]**
Written Format	MaxWaitACMTime=n
Value Range	n≥0: Maximum time to wait for the remote end to return the ACM message, calculated by s, with the default value of 25.
Description	Sets the maximum time to wait for the remote end to return the ACM message in an outgoing call.

3.1.2.27.8.8.8 Setting Maximum Waiting Time for Auto State Transferring from Pending to Idle

3.1.2.27.8.8.8.1 MaxWaitPendingToIdleTime

Configuration Item	**MaxWaitPendingToIdleTime**
Section	**[ISUP]**
Written Format	MaxWaitPendingToIdleTime=n
Value Range	n≥0: Maximum time to wait for the driver in the Pending state to automatically transfer to the Idle state, calculated by s, with the default value of 60.
Description	Sets the maximum time to wait for the driver in the Pending state to automatically transfer to the Idle state.

3.1.2.27.8.8.9 Replacing WaitRlc by Pending

3.1.2.27.8.8.9.1 WaitRlcToPending

Configuration Item	**WaitRlcToPending**
Section	**[ISUP]**
Written Format	WaitRlcToPending=n
Value Range	n=0: The driver uses the WaitRlc state (default); n=1: The driver uses the Pending state instead of the WaitRlc state.
Description	Sets whether the driver uses the Pending state to replace the WaitRlc state.

3.1.2.27.8.8.10 Not Using SynCTI-provided ISUP State Machine

3.1.2.27.8.8.10.1 AutoHandleIsup

Configuration Item	**AutoHandleIsup**
Section	**[SS7]**
Written Format	AutoHandleIsup=b
Value Range	b=0: No; b=1: Yes (default).
Description	Sets whether to use the ISUP state machine provided by the SynCTI driver or not.

3.1.2.27.8.8.10.2 CircuitReset

Configuration Item	**CircuitReset**
Section	**[ISUP]**
Written Format	CircuitReset=m
Value Range	m=0: After the SS7 service is enabled, the circuit goes into an idle state without sending a circuit reset message. m=1: After the SS7 service is enabled, the circuit sends a circuit reset message and goes into an idle state (default value). It is a boolean configuration item, not applicable to TUP.
Description	Determines if the circuit goes into an idle state or is reset after the ISUP service is enabled.

3.1.2.27.8.8.11 Setting Interval to Send GRS Message if Channels in Circuit Reset

3.1.2.27.8.8.11.1 SendGrsTime

Configuration Item	**SendGrsTime**
Section	**[ISUP]**
Written Format	SendGrsTime=n
Value Range	n is greater than 0, calculated by s, with the default value of 5s.
Description	Sets the interval to send a GRS message if the channel is in the state of circuit reset.

3.1.2.27.8.9 Advanced Configuration Item for SCCP

3.1.2.27.8.9.1 AutoHandleSccp

Configuration Item	**AutoHandleSccp**
Section	**[SS7]**
Written Format	AutoHandleSccp=b
Value Range	b=0: No; b=1: Yes (default).
Description	Sets if the SCCP message is processed by the SynCTI driver.
Note	It requires SynCTI Ver. 4.7.1.7 or above.

3.1.2.27.8.10 Setting Way to Output SS7 MSU

3.1.2.27.8.10.1 GetMsuOnAutoHandle

Configuration Item	**GetMsuOnAutoHandle**
Section	**[SS7]**
Written Format	GetMsuOnAutoHandle=b
Value Range	b=0: Not to output the original SS7 MSU in auto call connection which uses the driver's state machine to process user-level protocols (default); b=1: Allowed to obtain the original SS7 MSU in auto call connection which uses the driver's state machine to process user-level protocols.
Description	Sets the way to output SS7 MSU. SS7 MSU。When the driver's state machine is not used to process user-level protocols, i.e. when the auto call connection is disabled, the driver outputs the original SS7 MSU; when the driver's state machine is used to process user-level protocols, i.e. when the auto call connection is enabled, the driver's output of SS7 MSU is determined by the configuration item GetMsuOnAutoHandle.
Note	It requires SynCTI Ver.5.0.3.0 or above. All driver versions published before Ver.5.0.3.0 only support the operation based on GetMsuOnAutoHandle=0.

3.1.2.27.8.11 Setting Way to Handle SS7 MTP2 MSU

3.1.2.27.8.11.1 AppHandleMtp2Msu

Configuration Item	**AppHandleMtp2Msu**
Section	**[SS7]**
Written Format	AppHandleMtp2Msu=b
Value Range	b=0: SS7 MTP2 MSU handled by the driver (default); b=1: SS7 MTP2 MSU handled by the application program; b=2: Both SS7 MTP2 MSU and SS7 MTP2 link command and status handled by the application program.
Description	Sets the way to handle SS7 MTP2 MSU.
Note	This configuration item requires SynCTI Ver.5.1.0.0 or above. All driver versions published before Ver.5.1.0.0 only support the operation based on AppHandleMtp2Msu=0. The configuration AppHandleMtp2Msu=2 requires SynCTI Ver.5.3.2.1 or above.

3.1.2.27.8.12 Configuration Items for SS7 High Load System

3.1.2.27.8.12.1 RefreshWatchDogInSys

Configuration Item	**RefreshWatchDogInSys**
Section	**[BoardId=x]**
Written Format	RefreshWatchDogInSys=b
Value Range	b=0: Refresh the watchdog by the upper layer of the driver (default); b=1: Refresh the watchdog by the SYS layer.
Description	Sets the way to refresh the watchdog.
Note	This configuration item requires SynCTI Ver.5.3.1.5 or above. If the local end processor failure always occurs in SS7 system, we recommend you to set this configuration item to 1.

3.1.2.27.8.12.2 ReplySLTAInSys

Configuration Item	**ReplySLTAInSys**
Section	**[BoardId=x]**
Written Format	ReplySLTAInSys=b
Value Range	b=0: Reply the SLTA message by the upper layer of the driver (default); b=1: Reply the SLTA message by the SYS layer.
Description	Sets the way to reply the SLTA message.
Note	This configuration item requires SynCTI Ver.5.3.1.5 or above, and it only supports the SHD Series D-type and E-type boards. If the SS7 signaling link on the remote PBX often disconnects in SS7 system, we recommend you to set this configuration item to 1.

3.1.2.27.8.13 Setting SS7 Protocol Type

3.1.2.27.8.13.1 Ss7Type

Configuration Item	**Ss7Type**
Section	**[SS7]**
Written Format	Ss7Type=b
Value Range	b=0: ITU protocol (default); b=1: ANSI protocol.
Description	Sets the protocol type of SS7.
Note	This configuration item requires SynCTI Ver.5.4.0.0 or above.

3.1.2.27.9 Advanced Setting for ISDN

3.1.2.27.9.1 Setting ISDN Processing Mode

3.1.2.27.9.1.1 AutoHandleIsdn

Configuration Item	**AutoHandleIsdn**
Section	**[ISDN]**
Written Format	AutoHandleIsdn=b
Value Range	b=0: No; b=1: Yes (default).
Description	Sets whether ISDN messages are processed by the driver-provided ISDN state machine. Refer to the Advanced ISDN Programming Interface section in Chapter 1 for more information.
Note	It is an advanced configuration item, only required to be set when the application wants to process ISDN message by itself.

3.1.2.27.9.2 Setting Parameters for User-side/Network-side Digital Trunk

3.1.2.27.9.2.1 UserCrcMode

Refer to **NetCrcMode**

3.1.2.27.9.2.2 NetCrcMode

Configuration Item	**UserCrcMode** **NetCrcMode**
Section	**[ISDN]**
Written Format	UserCrcMode[n]=b NetCrcMode[n]=b
Value Range	n: The user-side (UserCrcMode) or network-side (NetCrcMode) digital trunk number b: The switch to control the CRC check b=0: Disable; b=1: Enable.
Description	Sets whether to enable the feature of CRC check for the digital trunk. UserCrcMode is applicable to the user side while NetCrcMode to the network side.
Note	● It is an advanced configuration item; ● If the set value of the configuration item TotalUserLinker (assumed as N) is greater than 0, UserCrcMode should be configured for N times; if the set value of the configuration item TotalNetLinker (assumed as N) is greater than 0, NetCrcMode should be configured for N times.

3.1.2.27.9.2.3 UserChIdentify

Refer to **NetChIdentify**

3.1.2.27.9.2.4 NetChIdentify

Configuration Item	**UserChIdentify** **NetChIdentify**
Section	**[ISDN]**
Written Format	UserChIdentify[n]=k NetChIdentify[n]=k

Value Range	n: The user-side (UserChIdentify) or network-side (NetChIdentify) digital trunk number k: The way to represent the channel identification message k=0x11: In the form of Time Slot Diagram; k=0x10: In the form of Number (default). **Note: The default value of k varies on the version of the SynCTI driver: if the driver used is below Ver. 4.7.3.1, the default value of k is 0x11; otherwise, it is 0x10.**
Description	Sets the way to represent channel identification messages on the digital trunk. UserChIdentify is applicable to the user side while NetChIdentify to the network side. See the Representation of Channel Identification Message section in Chapter 1 for more information.

3.1.2.27.9.2.5 UserVoiceFormat

Refer to **NetVoiceFormat**

3.1.2.27.9.2.6 NetVoiceFormat

Configuration Item	**UserVoiceFormat** **NetVoiceFormat**
Section	**[ISDN]**
Written Format	UserVoiceFormat[n]=k NetVoiceFormat[n]=k
Value Range	n: The user-side (UserVoiceFormat) or network-side (NetVoiceFormat) digital trunk number k: The voice CODEC, k=6: A-law (default); k=7: μ-law.
Description	Sets the voice CODEC used on the digital trunk. UserVoiceFormat is applicable to the user side while NetVoiceFormat to the network side.

3.1.2.27.9.2.7 UserRestarTime

Refer to **NetRestarTime**

3.1.2.27.9.2.8 NetRestarTime

Configuration Item	**UserRestarTime** **NetRestarTime**
Section	**[ISDN]**
Written Format	UserRestarTime=t NetRestarTime=t
Value Range	100≤t≤120, calculated by second, with the default value of 100.
Description	Sets the time for the local end to respond to the link restart command from the remote end, i.e. from the moment the local link receives the link restart command from the remote end until it is successfully restarted. If the link fails to be restarted within the time specified by this configuration item, all channels on relative digital trunks enter into the 'Unusable' state (i.e. S_CALL_UNAVAILABLE). UserRestarTime is applicable to the user side while NetRestarTime to the network side.
Note	● It is an advanced configuration item, not required to be set in most cases. ● It is applicable to all digital trunks in the system.

3.1.2.27.9.2.9 UserEstablishTime

Refer to **NetEstablishTime**

3.1.2.27.9.2.10 NetEstablishTime

Configuration Item	**UserEstablishTime** **NetEstablishTime**
Section	**[ISDN]**
Written Format	UserEstablishTime=t NetEstablishTime=t
Value Range	5≤t≤10, calculated by second, with the default value of 5.
Description	Sets the interval for the driver to establish links between the local and remote ends. If the driver fails to establish links within the time specified by this configuration item, it will automatically start another attempt. UserEstablishTime is applicable to the user side while NetEstablishTime to the network side.
Note	● It is an advanced configuration item, not required to be set in most cases. ● It is applicable to all user-side and network-side digital trunks in the system.

3.1.2.27.9.3 Setting Parameters for Outgoing Call

3.1.2.27.9.3.1 UserCalledNoSet

Refer to **NetCallingNoSet**

3.1.2.27.9.3.2 NetCalledNoSet

Refer to **NetCallingNoSet**

3.1.2.27.9.3.3 UserCallingNoSet

Refer to **NetCallingNoSet**

3.1.2.27.9.3.4 NetCallingNoSet

Configuration Item	**UserCalledNoSet** **NetCalledNoSet** **UserCallingNoSet** **NetCallingNoSet**
Section	**[ISDN]**
Written Format	UserCalledNoSet[n]=k NetCalledNoSet[n]=k UserCallingNoSet[n]=g NetCallingNoSet[n]=g

Value Range	n: The user-side (UserCalledNoSet, UserCallingNoSet) or network-side (NetCalledNoSet, NetCallingNoSet) digital trunk number
	k: The type of number (TON) and numbering scheme, represented by hexadecimal digits.
	k=0xa1: National Number (default);
	k=0x91: International Number;
	k=0xc1: Subscriber Number;
	k=0xb1: Specified Number;
	k=0x81: Unknown.
	g: The type of number (TON) and numbering scheme, represented by hexadecimal digits.
	g=0x21: National Number (default);
	g=0x11: International Number;
	g=0x41: Subscriber Number;
	g=0x31: Specified Number;
	g=0x01: Unknown.

Description	Sets the type of number (TON) and numbering scheme for the calling and called party numbers in the SETUP message during the outgoing call. UserCalledNoSet, NetCalledNoSet are used to set called party numbers respectively for the user side and the network side; UserCallingNoSet, NetCallingNoSet are used to set calling party numbers respectively for the user side and the network side.
Note	● It is an advanced configuration item; ● If the set value of the configuration item TotalUserLinker (assumed as N) is greater than 0, UserCalledNoSet and UserCallingNoSet should be configured for N times; if the set value of the configuration item TotalNetLinker (assumed as N) is greater than 0, NetCalledNoSet and NetCallingNoSet should be configured for N times.

3.1.2.27.9.3.5 UserNumIsFull

Refer to **NetNumIsFull**

3.1.2.27.9.3.6 NetNumIsFull

Configuration Item	**UserNumIsFull** **NetNumIsFull**
Section	**[ISDN]**
Written Format	UserNumIsFull=b NetNumIsFull=b
Value Range	b=0: No (default); b=1: Yes.
Description	Sets whether the 'Called Number Complete' parameter is included in the SETUP message during the outgoing call. UserNumIsFull is applicable to the user side while NetNumIsFull to the network side.
Note	It is an advanced configuration item.

3.1.2.27.9.3.7 UserChPreference

Refer to **NetChPreference**

3.1.2.27.9.3.8 NetChPreference

Configuration Item	**UserChPreference** **NetChPreference**
Section	**[ISDN]**
Written Format	UserChPreference=b NetChPreference=b
Value Range	b=0: No (default); b=1: Yes.
Description	Sets whether to allow the preferential channel selection. UserChPreference is applicable to the user side while NetChPreference to the network side.
Note	It is an advanced configuration item.

3.1.2.27.9.3.9 UserHighLayerCompatible

Refer to **NetHighLayerCompatible**

3.1.2.27.9.3.10 NetHighLayerCompatible

Configuration Item	**UserHighLayerCompatible** **NetHighLayerCompatible**

Section	**[ISDN]**
Written Format	UserHighLayerCompatible=b NetHighLayerCompatible=b
Value Range	b=0: No; b=1: Yes (default).
Description	Sets whether the 'High Layer Compatibility' field is included in the SETUP message. UserHighLayerCompatible is applicable to the user side while NetHighLayerCompatible to the network side.
Note	It is an advanced configuration item.

3.1.2.27.9.3.11 UserLowLayerCompatible

Refer to **NetLowLayerCompatible**

3.1.2.27.9.3.12 NetLowLayerCompatible

Configuration Item	**UserLowLayerCompatible** **NetLowLayerCompatible**
Section	**[ISDN]**
Written Format	UserLowLayerCompatible=b NetLowLayerCompatible=b
Value Range	b=0: No; b=1: Yes (default).
Description	Sets whether the 'Low Layer Compatibility' field is included in the SETUP message. UserLowLayerCompatible is applicable to the user side while NetLowLayerCompatible to the network side.
Note	It is an advanced configuration item.

3.1.2.27.9.3.13 UserDialTime

Refer to **NetDialTime**

3.1.2.27.9.3.14 NetDialTime

Configuration Item	**UserDialTime** **NetDialTime**
Section	**[ISDN]**
Written Format	UserDialTime=t NetDialTime=t
Value Range	t: The waiting time, calculated by second, with the default value of 60sec
Description	After the application calls the function SsmSearchIdleCallOutCh on an ISDN channel, the driver will start a timer. If the application does not start an outgoing call before the timer overflows, the driver will reset the channel back to the idle state and then transfer it to the S_CALL_STANDBY state. This configuration item is used to set the value of this timer. See the ISDN Channel State Machine section in Chapter 1 for more information. UserDialTime is applicable to the user side while NetDialTime to the network side.
Note	It is an advanced configuration item.

3.1.2.27.9.3.15 TransferCapability

Configuration Item	**TransferCapability**
Section	**[ISDN]**
Written Format	**TransferCapability=n**
Value Range	=0: Voice (default) =1: 3.1k audio
Description	Sets the 'Transfer Capability' filed in the signaling message

Note	It is an advanced configuration item.

3.1.2.27.9.3.16 PresentNumber

Configuration Item	**PresentNumber**
Section	**[ISDN]**
Written Format	PresentNumber =n
Value Range	=0: not allowed to show number =1: allowed to show number (default)
Description	Sets the field that determines if it is allowed to show the calling party number in the signaling message.
Note	It is an advanced configuration item.

3.1.2.27.9.3.17 UserT303

Refer to **NetT303**

3.1.2.27.9.3.18 NetT303

Configuration Item	**UserT303** **NetT303**
Section	**[ISDN]**
Written Format	UserT303=t NetT303=t
Value Range	t: The length of time for the timer T303, calculated by second, with the default value of 4sec
Description	Sets the timer T303. UserT303 is applicable to the user side while NetT303 to the network side. See the ISUP Channel State Machine section in Chapter 1 for more information.

3.1.2.27.9.3.19 UserT304

Refer to **NetT304**

3.1.2.27.9.3.20 NetT304

Configuration Item	**UserT304** **NetT304**
Section	**[ISDN]**
Written Format	UserT304=t NetT304=t
Value Range	t: The length of time for the timer T304, calculated by second, with the default value of 30sec
Description	Sets the timer T304. UserT304 is applicable to the user side while NetT304 to the network side. See the ISUP Channel State Machine section in Chapter 1 for more information.

3.1.2.27.9.3.21 UserWaitAfterCallProceeding

Configuration Item	**UserWaitAfterCallProceeding**
Section	**[ISDN]**
Written Format	UserWaitAfterCallProceeding=n
Value Range	n: The waiting time, with the default value of 15s.
Description	Sets the maximum time that the local end waits for the remote end to send back the acknowledgement message in an outgoing call. If the local end doesn't receive the acknowledgement message from the remote end before timeout, it will actively disconnect the call.
Note	It is an advanced configuration item only used for user-side outgoing calls.

3.1.2.27.9.3.22 NetWaitAfterCallProceeding

Configuration Item	**NetWaitAfterCallProceeding**
Section	**[ISDN]**
Written Format	NetWaitAfterCallProceeding =n
Value Range	n: The waiting time, with the default value of 20s.
Description	Sets the maximum time that the local end waits for the remote end to send back the acknowledgement message in an outgoing call. If the local end doesn't receive the acknowledgement message from the remote end before timeout, it will actively disconnect the call.
Note	It is an advanced configuration item only used for network-side outgoing calls.

3.1.2.27.9.3.23 UserTxCallingPartyNum

Configuration Item	**UserTxCallingPartyNum**
Section	**[ISDN]**
Written Format	UserTxCallingPartyNum = b
Value Range	b=0: Not to send the 6c unit (i.e. the calling party number) in the ISDN User Setup message for an outgoing call; b=1: Send the 6c unit (i.e. the calling party number) in the ISDN User Setup message for an outgoing call (default).
Description	Sets whether to send the 6c unit (i.e. the calling party number) in the ISDN User Setup message for an outgoing call.

3.1.2.27.9.3.24 NetTxCallingPartyNum

Configuration Item	**NetTxCallingPartyNum**
Section	**[ISDN]**
Written Format	NetTxCallingPartyNum = b
Value Range	b=0: Not to send the 6c unit (i.e. the calling party number) in the ISDN Net Setup message for an outgoing call; b=1: Send the 6c unit (i.e. the calling party number) in the ISDN Net Setup message for an outgoing call (default).
Description	Sets whether to send the 6c unit (i.e. the calling party number) in the ISDN Net Setup message for an outgoing call.

3.1.2.27.9.4 Setting Parameters for Incoming Call

3.1.2.27.9.4.1 UserSideDefaultAckTimer

Refer to **NetSideDefaultAck**

3.1.2.27.9.4.2 UserSideDefaultAck

Refer to **NetSideDefaultAck**

3.1.2.27.9.4.3 NetSideDefaultAckTimer

Refer to **NetSideDefaultAck**

3.1.2.27.9.4.4 NetSideDefaultAck

<table>
<tr><td>Configuration Item</td><td>UserSideDefaultAckTimer
UserSideDefaultAck
NetSideDefaultAckTimer
NetSideDefaultAck</td></tr>
<tr><td>Section</td><td>[ISDN]</td></tr>
<tr><td>Written Format</td><td>UserSideDefaultAckTimer=t
UserSideDefaultAck=k
NetSideDefaultAckTimer=t
NetSideDefaultAck=k</td></tr>
<tr><td>Value Range</td><td>t: The set value of the timer, calculated by second, t<120, with the default value of 20.
k: The driver's behavior after the timer overflows. Its values and corresponding meanings are shown in the table below.
<table>
<tr><th>K</th><th>Meaning</th><th>Driver's Behavior</th></tr>
<tr><td>1</td><td>Called party idle</td><td>Send the ALERTING message</td></tr>
<tr><td>2</td><td>Called party busy</td><td>Send the DISCONNECT message.
Reason=Called party busy</td></tr>
<tr><td>4</td><td>Call refused</td><td>Send the DISCONNECT message.
Reason=Call refused</td></tr>
<tr><td>5</td><td>No answer</td><td>Send the DISCONNECT message.
Reason=Called party no answer</td></tr>
<tr><td>Others</td><td>Spare</td><td></td></tr>
</table>
The default value of k is 1.</td></tr>
<tr><td>Description</td><td>After the driver receives the complete number in an incoming call, this configuration item is used to set whether the reply message is automatically sent by the driver. Refer to the ISUP Channel State Machine section in Chapter 1 for more information.
UserSideDefaultAckTimer, UserSideDefaultAck are applicable to the user side while NetSideDefaultAckTimer, NetSideDefaultAck to the network side.</td></tr>
<tr><td>Note</td><td>It is an advanced configuration item.</td></tr>
</table>

3.1.2.27.9.4.5 UserIsReceivePhoNum

Refer to **NetIsReceivePhoNum**

3.1.2.27.9.4.6 NetIsReceivePhoNum

<table>
<tr><td>Configuration Item</td><td>UserIsReceivePhoNum
NetIsReceivePhoNum</td></tr>
<tr><td>Section</td><td>[ISDN]</td></tr>
<tr><td>Written Format</td><td>UserIsReceivePhoNum=b
NetIsReceivePhoNum=b</td></tr>
<tr><td>Value Range</td><td>b=0: Tandem Exchange Mode;
b=1: Terminating Office Mode (default).</td></tr>
<tr><td>Description</td><td>✧ Sets the driver operating mode during an incoming call. For more information, refer to the section Terminating Office Mode vs. Tandem Exchange Mode in Chapter 1.
UserIsReceivePhoNum is applicable to the user side while NetIsReceivePhoNum to the network side.</td></tr>
<tr><td>Note</td><td>● Usually this configuration item is set to 0 provided the remote PBX is Alcatel or AVAYA;
● The configuration item UserTieStationMode supported by earlier versions of our driver has the same function but is only applicable to the user side. Hence it can be replaced by UserIsReceivePhoNum.</td></tr>
</table>

3.1.2.27.9.4.7 UserIsSendChIdentify

Refer to NetIsSendChIdentify

3.1.2.27.9.4.8 NetIsSendChIdentify

Configuration Item	**UserIsSendChIdentify** **NetIsSendChIdentify**
Section	**[ISDN]**
Written Format	UserIsSendChIdentify=b NetIsSendChIdentify=b
Value Range	b=0: No; b=1: Yes (default).
Description	Sets whether the channel identification message is included in the corresponding reply message (such as CALL PROCEEDING, ALERT, etc.) after the local end receives the SETUP message from the remote PBX during an incoming call. UserIsSendChIdentify is applicable to the user side while NetIsSendChIdentify to the network side.

3.1.2.27.9.4.9 UserT302

Refer to NetT302

3.1.2.27.9.4.10 NetT302

Configuration Item	**UserT302** **NetT302**
Section	**[ISDN]**
Written Format	UserT302=t NetT302=t
Value Range	t: The length of time for the timer T302, calculated by second, with the default value of 15sec
Description	Sets the timer T302. UserT302 is applicable to the user side while NetT302 to the network side. See the ISUP Channel State Machine section in Chapter 1 for more information.

3.1.2.27.9.4.11 UserT313

Configuration Item	**UserT313**
Section	**[ISDN]**
Written Format	UserT313=t
Value Range	t: The length of time for the timer T313, calculated by second, with the default value of 4sec
Description	Sets the timer T313. See the ISUP Channel State Machine section in Chapter 1 for more information.
Note	This configuration item is only applicable to the user side.

3.1.2.27.9.4.12 ProgressExt

Configuration Item	**ProgressExt**
Section	**[ISDN]**
Written Format	ProgressExt=n

Value Range	n: Progress indicator, with the default value of 0 that implies no progress indicator is included. The value of n is represented by 8 bits. Each bit means as follows. Bit8=1 Bit7, Bit 6=00----ITU-T coding standard Bit5=0----Spare Bit4~ Bit1= 0000----User (U) 0001----Local private network (LPN) 0010----Local public network (LN) 0011----Transfer network (TN) 0100----Remote public network (RLN) 0101----Remote private network (RPN) 1010----Network formed by points except two connected on the SS7 public network (BI)
Description	In an incoming call, if the ALERT or SETUP message is sent by the local end when the value of ProgressExt is not 0, it will contain the progress indicator.
Note	It is an advanced configuration item, only applicable to incoming calls.

3.1.2.27.9.5 Setting Timers

3.1.2.27.9.5.1 UserT305

Refer to NetT306

3.1.2.27.9.5.2 NetT305

Refer to NetT306

3.1.2.27.9.5.3 NetT306

Configuration Item	**UserT305** **NetT305** **NetT306**
Section	**[ISDN]**
Written Format	UserT305=t NetT305=t NetT306=t
Value Range	t: The length of time for the timer, calculated by second, with the default value of 30 sec.
Description	Sets the timers T305 and T306. UserT305 is applicable to the user side while NetT305 and NetT306 to the network side. ✧ To the user side, the driver will start T305 after the local end sends the DISCONNECT message to the remote PBX, and automatically stop it after the local end receives the RELEASE or DISCONNECT message from the remote PBX. ✧ To the network side, after the local end sends the DISCONNECT message to the remote PBX, the driver will start T306 if the DISCONNECT message contains the progress indicator field and the value of this field is equal to 8, or start T305 otherwise. After the local end receives the RELEASE or DISCONNECT message from the remote PBX, the driver will automatically stop T305 or T306. Once T305 or T306 overflows, the driver will send the RELEASE message.

3.1.2.27.9.5.4 UserT308

Refer to NetT308

3.1.2.27.9.5.5 NetT308

Configuration Item	**UserT308** **NetT308**
Section	**[ISDN]**
Written Format	UserT308=t NetT308=t
Value Range	t: The length of time for the timer, calculated by second, with the default value of 4 sec.
Description	Sets the timer T308. UserT308 is applicable to the user side while NetT308 to the network side. The driver will start T308 after the local end sends the RELEASE message to the remote PBX, and automatically stop it after the local end receives the RELEASE or RELEASE COMPLETE message from the remote PBX. If T308 overflows, the driver will resend the RELEASE message.

3.1.2.27.9.6 Setting Other Parameters

3.1.2.27.9.6.1 WaitHangupTime

Configuration Item	**WaitHangupTime**
Section	**[ISDN]**
Written Format	WaitHangupTime=b
Value Range	20<b<10000, calculated by millisecond (ms), with the default value of 1600.
Description	Sets the ISDN pending time.
Note	It is an advanced configuration item.

3.1.2.27.9.6.2 UserStatusReason

Refer to NetStatusReason

3.1.2.27.9.6.3 NetStatusReason

Configuration Item	**UserStatusReason** **NetStatusReason**
Section	**[ISDN]**
Written Format	UserStatusReason=m NetStatusReason=m
Value Range	m=1: 3-byte Format (default); m=0: 2-byte Format
Description	When the local end receives the STATUS ENQUIRY message (for status query) from the remote PBX, the driver will automatically reply by sending the STATUS message. This configuration item specifies the reason field in the STATUS message to use the 2-byte or 3-byte format. Using which format should be determined according to the parameter setting for the remote PBX. UserStatusReason is applicable to the user side while NetStatusReason to the network side.
Note	● It is an advanced configuration item; ● It requires SynCTI Ver. 4.7.1.8 or above.

3.1.2.27.9.6.4 EnAutoAlert02

Refer to EnAutoAlert03

3.1.2.27.9.6.5 EnAutoAlert03

Configuration Item	**EnAutoAlert02** **EnAutoAlert03**
Section	**[ISDN]**
Written Format	EnAutoAlert02=m EnAutoAlert03=m
Value Range	m=0: This feature disabled (default); m=1: This feature enabled.
Description	If EnAutoAlert02=1, it indicates the 02 (CALL PROCEEDING) message is received. In such situation, if the progress indicator turns to be 8 or 1, the system will go into the state of auto alert. If EnAutoAlert03=1, it indicates the 03 (PROGRESS) message is received. In such situation, if the progress indicator turns to be 8 or 1, the system will go into the state of auto alert.
Note	● It is an advanced configuration item; ● It requires SynCTI Ver. 5.2.0.0 or above.

3.1.2.27.9.6.6 GetRedirectionReason

Configuration Item	**GetRedirectionReason**
Section	**[ISDN]**
Written Format	GetRedirectionReason=m
Value Range	m=0: Obtains the redirection reason of the information element Facility (0x1c) (default); m=1: Obtains the redirection reason of the information element Redirecting Number (0x74).
Description	If GetRedirectionReason=1, it indicates that the function SsmGetRedirectionInfReason obtains the redirection reason of the information element 0x74. If GetRedirectionReason=0, it indicates that the function SsmGetRedirectionInfReason obtains the redirection reason of the information element 0x1c.
Note	● It is an advanced configuration item; ● It requires SynCTI Ver. 5.3.2.3 or above.

3.1.2.27.9.7 NetCallingNoSet

Configuration Item	**UserRedirectingNoSet** **NetRedirectingNoSet**
Section	**[ISDN]**
Written Format	UserRedirectingNoSet[n]=k NetRedirectingNoSet[n]=k

Value Range	n: The user-side (UserCalledNoSet, UserCallingNoSet) or network-side (NetCalledNoSet, NetCallingNoSet) digital trunk number
	k: The type of number (TON) and numbering scheme, represented by hexadecimal digits.
	g=0x21: National Number (default);
	g=0x11: International Number;
	g=0x41: Subscriber Number;
	g=0x31: Specified Number;
	g=0x01: Unknown.

Description	Sets the type of number (TON) and numbering scheme for the calling and called party numbers in the SETUP message during the outgoing call to modify the number attribute in the 0x74 field respectively for the user side and the network side.
Note	It is an advanced configuration item.

3.1.2.27.10 Setting Way to Use DSP Chip

3.1.2.27.10.1 LoadFskBin

Configuration Item	**LoadFskBin**
Section	**[BoardId=x]**
Written Format	LoadFskBin=m
Value Range	m=0: Enhanced Echo Canceller Mode. In this mode, the voice channel is unable to record voices and receive or send FSK data. The DSP resources standing idle are used to enhance the effect of the echo canceller; m=1: Normal Running Mode (default); m=2: TS16 Recording Mode. The driver will directly pass the original signaling data from TS16 to the application by recording. When m=1, each on-board DSP chip processes data from 15 voice time slots. Therefore, two DSP chips are needed to process data from the 30 time slots on each digital trunk, one for TS_1~TS_{15} and the other for TS_{17}~TS_{31}. When m=2, the DSP chip previously used to process data from TS_{17}~TS_{31} will turn to process those from TS_{16}~TS_{30}, and the voice data from TS_{31} will be abandoned. **Note:** In such case, the physical channel number 15 corresponds not to TS_{17}, but to TS_{16}. And the same change also happens to the other channels.
Description	Selects the operating mode for DSP chips on the SHD Series boards.
Note	It is an advanced configuration item, only applicable to the 30/60/120-channel SHD Series A-type boards.

3.1.2.27.11 Setting Signal Recording Mode

3.1.2.27.11.1 ShdDEToneRec

Configuration Item	**ShdDEToneRec**
Section	**[BoardId=x]**
Written Format	ShdDEToneRec=b
Value Range	b=0: No (default); b=1: Yes.
Description	Sets whether to have the tone and the DTMF signal sent by the SHD Series D-type or E-type boards enter the recording buffer area.
Note	It is only applicable to the SHD Series D-type and E-type boards.

3.1.2.27.12 Setting the Filtration Threshold of the Back Noise

3.1.2.27.12.1 NoiseFilteringMinGate

Configuration Item	**NoiseFilteringMinGate**
Section	**[BoardId=x]**
Written Format	NoiseFilteringMinGate=b
Value Range	b=0: Disabled (default); b=n (0<n≤1000): Threshold value (To set it, the recommended value is b=300).
Description	Sets the filtration threshold of the back noise.
Note	● It is only applicable to the SHD Series D-type and E-type boards; ● If the value of this item is in the range of 1~1000, fax failure may occur; ● If the value of this item is not in the range of 1~1000, this feature is considered to be disabled.

3.1.2.27.13 Setting PCM Working Status

3.1.2.27.13.1 PcmPowerDown

Configuration Item	**PcmPowerDown**
Section	**[BoardId=x]**
Written Format	PcmPowerDown[i]=d
Value Range	i denotes the logical number of the digital trunk, range of value: 0~N-1 (N is the set value of the configuration item TotalPcm, it can be obtained by the function call of SsmGetMaxPcm) d=0: normal working mode (default); d=1: idle mode (It is similar to the state that the PCM is unconnected).
Description	Sets the status (working or Idle) of the PCM.
Note	● It is only applicable to the SHD Series D-type and E-type boards.

3.1.2.28 Common Configuration Item for SHN/HMP Series (CTI Series)

3.1.2.28.1 Advanced General Settings for SHN/HMP Series

3.1.2.28.1.1 EnableSIPStun

Refer to HeartInterval

3.1.2.28.1.2 SIPStunServerIP

Refer to HeartInterval

3.1.2.28.1.3 LogLevel

Refer to HeartInterval

3.1.2.28.1.4 LogFile

Refer to HeartInterval

3.1.2.28.1.5 EventThreadNum

Refer to HeartInterval

3.1.2.28.1.6 SipCallCheckInterval

Refer to HeartInterval

3.1.2.28.1.7 SendOptionsOutCallIP

Refer to HeartInterval

3.1.2.28.1.8 SendOptionsOutCallInterval

Refer to HeartInterval

3.1.2.28.1.9 RemoteCrashCheckInterval

Refer to HeartInterval

3.1.2.28.1.10 IgnoreUserName

Refer to HeartInterval

3.1.2.28.1.11 AutoDetectRemoteRTPAddress

Refer to HeartInterval

3.1.2.28.1.12 HeartInterval

Note: This configuration item is named UdpHeartTime in versions below SynCTI 5.1.0.0.

Configuration Item	**EnableSIPStun** **SIPStunServerIP** **StunMaskAddr** **LogLevel** **EventThreadNum** **SipCallCheckInterval** **SendOptionsOutCallIP** **SendOptionsOutCallInterval** **RemoteCrashCheckInterval** **IgnoreUserName** **AutoDetectRemoteRTPAddress** **HeartInterval**
Section	**[SIP]**
Written Format	EnableSIPStun=p SIPStunServerIP=y StunMaskAddr=d LogLevel=m EventThreadNum=j SipCallCheckInterval=n SendOptionsOutCallIP=k SendOptionsOutCallInterval=q RemoteCrashCheckInterval=i IgnoreUserName=I AutoDetectRemoteRTPAddress=z HeartInterval=x

Value Range	p: p=0, disable the Stun feature in SIP; p=1, enable the Stun feature in SIP.
	y: The IP address of the Stun server in SIP.
	d: The network address of the IP board.
	m: Sets the output log level. See below for details.
	m=1: SEGMENT, serious error (default);
	m=2 : FATAL, error in protocol message;
	m=3: ERROR, logic error;
	m=4: WARNING, logic warning;
	m=5: INFO1, logic information
	m=6: INFO2, SIP message information;
	m=7: INFO3, RTP message information;
	m=8: INFO4, original SIP message.
	j: Sets the size of the thread pool used to get SIP protocol events. The value range is 0<j<6 and the default value is 3.
	n: Sets the interval between checks of the remote end's abnormal hangup, in second. n=0: Not use this feature.
	k: Automatically sends the destination IP address of the signaling message options in case of non registration.
	q: Sends the time interval of the signaling message options, in second. q=0: Disable this feature.
	i: Sets the time interval (calculated by s) to detect the abnormal hangup of the remote end by RTP. i=0 means this feature is disabled. Note: This feature is valid only when n=0. The minimum time for the C-type VoIP board is 10s.
	I: Sets whether to enable the feature of ignoring user name. I=0: disable (default); I=1: enable. When a SIP channel which serves as the incoming end sends the register request to the server, if this feature is enabled, the driver will search for idle channels only according to the domain name within the signaling message, ignoring the judgment on user name.
	z: Sets whether to enable the remote RTP address self-adaptive feature. z=0: Disable; z=1, Enable (default). This feature is used to update the RTP reception address or port in the signaling message sent by the remote end when it does not comply with the actual state. Therefore, it can increase the self-adaptive capability of RTP transmission. The 480C board supports the following feature: using the source IP address of the OCT first received RTP packet as the destination IP address to send the RTP packet.
	x: Sets the heartbeat time for the UDP port. The value range is 0≤x≤1000 (ms) and the default value is x=0 (disable the heartbeat).

Description	Sets other parameters.
Note	This item is valid only when the configuration item ProtocolType is set to 1.

3.1.2.28.1.13 UseRport

Configuration Item	**UseRport**
Section	**[SIP]**
Written Format	UseRport=z
Value Range	z=1: The Invite message includes the rport field (default); z=0: The Invite message does not include the rport field.
Description	Sets whether to have the Invite message include the rport field.

3.1.2.28.1.14 SipTransportProtocol

Configuration Item	**SipTransportProtocol**
Section	**[SIP]**
Written Format	SipTransportProtocol =z

Value Range	z=0, SIP is used over UDP (default);
	z=1, SIP is used over TCP;
	z=2, SIP is used over TLS.

Description	Sets whether SIP is used over TCP or UDP

3.1.2.28.1.15 RetransmitLost200OK

Configuration Item	**RetransmitLost200OK**
Section	**[SIP]**
Written Format	RetransmitLost200OK=z
Value Range	z=1: It is not necessary for the board to wait for the ACK message after sending the 200OK message; z=0: It is necessary for the board to wait for the ACK message after sending the 200OK message (default).
Description	Sets if it is necessary for the board to wait for the ACK message after sending the 200OK message to establish a call.

3.1.2.28.1.16 RegisterTimeOutSpace

Configuration Item	**RegisterTimeOutSpace**
Section	**[SIP]**
Written Format	RegisterTimeOutSpace=q
Value Range	q: time interval, q>0, calculated by s and the default value is 10.
Description	Time interval between different register messages.
Note	Here the time interval exactly means the time from the register timeout to the auto generation of a new register message.

3.1.2.28.1.17 EnableSetUsername

Configuration Item	**EnableSetUsername**
Section	**[SIP]**
Written Format	EnableSetUsername=n
Value Range	n=1: Username can be modified after registration; n=0: Username cannot be modified after registration (default).
Description	Sets whether the username can be modified after registration for SHN Series boards.
Note	This configuration item is only applicable to some special SIP servers, and does not work on common servers.

3.1.2.28.1.18 UserName

Refer to SipTotalMultiRegNum

3.1.2.28.1.19 RegPassword

Refer to SipTotalMultiRegNum

3.1.2.28.1.20 Domain

Refer to SipTotalMultiRegNum

3.1.2.28.1.21 RegRealm

Refer to SipTotalMultiRegNum

3.1.2.28.1.22 RegExpires

Refer to SipTotalMultiRegNum

3.1.2.28.1.23 RegStartCh

Refer to SipTotalMultiRegNum

3.1.2.28.1.24 RegEndCh

Refer to SipTotalMultiRegNum

3.1.2.28.1.25 SipTotalMultiRegNum

Configuration Item	**UserName** **RegPassword** **Domain** **RegRealm** **RegExpires** **RegStartCh** **RegEndCh** **RegOutBoundProxyAddr** **RegAuthUserName** **RegDisplayName** **SipTotalMultiRegNum**
Section	**[SIP]**
Written Format	UserName[m]=s RegPassword[m]=j Domain[m]=x RegRealm[m]=r RegExpires[m]=q RegStartch[m]=o RegEndCh[m]=e RegOutBoundProxyAddr[m]=b RegAuthUserName[m]=a RegDisplayName[m]=d SipTotalMultiRegNum=n
Value Range	m: Number for the multi-channel operation. It must start from 0, range of value: 0≤m≤n(n is the set value of the configuration item SipTotalMultiRegNum) s: Username, a string containing up to 50 characters. When m≠0, s indicates the registered username; when m=0, s indicates the content of the username field in SIP URI. j: Registered user password. It is valid only when m≠0, indicating the registered user password. x: Registered server. It is valid only when m≠0, indicating the registered server address in the format of host[:port]. r: Alias of the SIP server. q: Registration expiration for the multi-channel operation. It indicates how often the board refreshes the registration information to the registered server, with the default value of 3600s and the minimum value of 10s. o: Starting channel number of the multi-channel operation. e: Ending channel number of the multi-channel operation. The multi-channel registration is equal to the single channel registration when o=e.
	b: Bound external address for the multi-channel registration, which is used under the IMS network environment. After this configuration, the registration and invite messages will both be sent to the bound external address. a: Authentication username for the multi-channel registration, which is used to specify the authentication username field in the authentication information. d: Display username for the multi-channel registration, which is used to specify the display username in the registration message. n: Times for performing the multi-channel operation.
Description	Using the configuration items to perform the multi-channel registration for n times is equivalent to invoking the function SsmSipRegister for n times.

Note	Other configuration items in this section are valid only when the set value of the configuration item SipTotalMultiRegNum is greater than 0.

Example	Suppose that there is an SHN-60B-CT/PCI+ board installed in the system, the channels that need to be registered are channels 0~19 (registered to 192.168.1.100:5200 under the username of 1001), channels 20~39 (registered to 192.168.1.101:5200 under the username of 1002) and channels 40~59 (registered to ims.test.com server under the username of 1001 in IMS. The bound external address is 192.168.1.102:5200, and the authentication username is +1001@ims.test.com)
	[SIP] // in the SIP field
	SipTotalMultiRegNum = 3 //indicates the times for performing the multi-channel operation. //The following three groups of configuration items are parameters respectively for three multi-channel registrations. For detailed information on each configuration item, you can refer to the configuration item of board registration.
	//Configuration items for the first multi-channel registration
	Domain[0]=192.168.1.100:5200
	RegExpires[0]=3600
	UserName[0]=1001
	RegPassword[0]=1001
	RegRealm[0]=
	RegStartCh[0]=0 //The starting channel number of the first multi-channel operation. This channel number is decided over the whole program.
	RegEndCh[0]=19 //The ending channel number of the first multi-channel operation. This channel number is decided over the whole program.
	//Configuration items for the second multi-channel registration
	Domain[1]=192.168.1.101:5200
	RegExpires[1]=3600
	UserName[1]=1002
	RegPassword[1]=1002
	RegRealm[1]=
	RegStartCh[1]=20
	RegEndCh[1]=39
	//Configuration items for the third multi-channel registration
	Domain[2]= ims.test.com
	RegExpires[2]=3600
	UserName[2]=1001
	RegPassword[2]=1001

	RegRealm[2]= RegStartCh[2]=40 RegEndCh[2]=59 RegOutBoundProxyAddr[0]= 192.168.1.102:5200 RegAuthUserName[0]= +1001@ims.test.com RegDisplayName[0]=1001 (Optional)

3.1.2.28.1.26 UseGroupByIp

Configuration Item	**UseGroupByIp**
Section	**[SIP]**

Written Format	UseGroupByIp=z
Value Range	z=1: Enable SIP Trunk feature; z=0: Disable SIP Trunk feature (default).
Description	When the SIP trunk feature is enabled and the channel is bound to the IP address by invoking the multi-channel or single channel registration function, the board which serves as the called party can search for the channel using the IP address of the calling party server.

3.1.2.28.1.27 UseSipKB

Configuration Item	**UseSipKB**
Section	**[SIP]**
Written Format	UseSipKB=z
Value Range	z=2: The board works as a registration server. To an incoming call from a registered account, the application will reply the 180 message; to an incoming call from an unregistered account, the driver will reply the 180 message; z=1: The 180 message is automatically replied by the application; z=0: The 180 message is automatically replied by the driver (default).
Description	Sets whether the 180 message is replied by the driver or the application.
Note	● If this configuration item is set to 1, the channel will enter the Pending state after receiving the INVITE message; ● If this configuration item is set to 2 and the board works as a registration server, the channel will enter the Pending state only when the incoming call is from a registered account; otherwise, the channel will enter the Ringing state; ● When the board is set to work as a registration server, that is, the configuration item SipRegMode is set to 1 or 2; if this configuration item is set by default to 0, which may affect the normal working of the registration server, the driver will automatically reset it to 2.

3.1.2.28.1.28 SipProxyAddr

Configuration Item	**SipProxyAddr**
Section	**[SIP]**
Written Format	SipProxyAddr=z
Value Range	z=a.b.c.d: SIP proxy server address, with the default value of 0.0.0.0 which means not to use the SIP proxy server.
Description	Sets the SIP proxy server address.

3.1.2.28.1.29 SipProxyPort

Configuration Item	**SipProxyPort**
Section	**[SIP]**
Written Format	SipProxyPort=n
Value Range	0≤n≤65535: SIP proxy server port, with the default value of 5060.
Description	Sets the SIP proxy server port.

3.1.2.28.1.30 ims

Configuration Item	**ims**
Section	**[SIP]**
Written Format	ims=b
Value Range	b=0: The remote SIP server does not use the ims network (default); b=1: The remote SIP server uses the ims network.
Description	Confirm whether the remote SIP server uses the ims network. If it does, this configuration item should be set to 1, otherwise the local IP board will fail to register to the ims network.

3.1.2.28.1.31 SipNewRegisterMode

Configuration Item	**SipNewRegisterMode**
Section	**[SIP]**
Written Format	SipNewRegisterMode=b
Value Range	b=0: Binds the channel with register information (default); b=1: Unbinds the channel from register information.
Description	Sets whether to bind the channel with register information.

3.1.2.28.1.32 SipBuildBusyReplaceForbidden

Configuration Item	**SipBuildBusyReplaceForbidden**
Section	**[SIP]**
Written Format	SipBuildBusyReplaceForbidden=b
Value Range	b=0: Adopts 403 or 415 message (default); b=1: Uses 486 message to replace 403 or 415 message.
Description	Sets whether to adopt the 486 message to replace the 403 or 415 message.

3.1.2.28.1.33 SipFindRingChFromPreRingCh

Configuration Item	**SipFindRingChFromPreRingCh**
Section	**[SIP]**
Written Format	SipFindRingChFromPreRingCh=b
Value Range	b=0: Searches for an idle channel for ringing in the ascending order of the channel number, starting from Ch0 (default); b=1: Provided the previous ringing channel is Ch(N). Searches for an idle channel for ringing in the ascending order of the channel number, starting from Ch(N+1).
Description	Sets whether to search for an idle channel for ringing from Ch(N+1).

3.1.2.28.1.34 SipDomain

Configuration Item	**SipDomain**
Section	**[SIP]**
Written Format	SipDomain=b
Value Range	b=0: Do not parse the SIP server domain name while checking the Registration state of the channel; b=1: Parse the SIP server domain name while checking the Registration state of the channel (default).
Description	Sets whether to parse the SIP server domain name while checking the Registration state of the channel.

3.1.2.28.1.35 SipSearchChInRegisterChannel

Configuration Item	**SipSearchChInRegisterChannel**
Section	**[SIP]**
Written Format	SipSearchChInRegisterChannel=b
Value Range	b=0: Searches for an idle channel from the nonregistered channels while receiving a peer to peer incoming call (default); b=1: Searches for an idle channel from all channels (both registered and nonregistered) while receiving a peer to peer incoming call.
Description	Sets whether to search for an idle channel only from the registered channels while receiving a peer to peer incoming call.
Configuration Item	SipSessionExpiresMin
Section	[SIP]
Written Format	SipSessionExpiresMin =b

Value Range	b=0, Not to designate the minimum refresh time for the session (default); b>=90, Designate the minimum refresh time for the session.
Description	Sets whether to designate the minimum refresh time for the session. The refresh time of the session in the received invite message should be longer than the value set in this item, or the driver will reply the 422 message to reject the invite message.
Note	This item is valid only when it is used with the configuration item SipSessionExpires.

3.1.2.28.1.36 SipRegNumberAreaLen

Configuration Item	**SipRegNumberAreaLen**
Section	**[SIP]**
Written Format	SipRegNumberAreaLen=z
Value Range	0<z ≤20, It indicates the number of digits in the prefix of a registered number ; z=0, It means the registered number has no prefix (default).
Description	Sets the prefix of a registered number.
Note	This configuration is used for such occasion when the prefix of a registered number doesn't conform to that of the actual incoming number.

3.1.2.28.1.37 SipAddRTPChkSum

Configuration Item	**SipAddRTPChkSum**
Section	**[SIP]**
Written Format	SipAddRTPChkSum =b
Value Range	b=0: The RTP data package does not include the Checksum field (default); b=1: The RTP data package includes the Checksum field.
Description	Sets whether to have the RTP data package include the Checksum field.

3.1.2.28.1.38 StunMaskAddr

Refer to HeartInterval

3.1.2.28.1.39 SipMsgHeaderName

Refer to SipMsgHeaderValue

3.1.2.28.1.40 SipMsgHeaderValue

Configuration Item	**SipMsgHeaderName** **SipMsgHeaderValue**
Section	**[SIP]**
Written Format	SipMsgHeaderName=a SipMsgHeaderValue=b
Value Range	a: The name of the field to be added; b: The content of the field to be added.
Description	Sets the name and content of the field to be added. The function SsmSipMsgSetHeader can implement the same feature.

3.1.2.28.1.41 SipAutoAdaptRtpSrc

Configuration Item	**SipAutoAdaptRtpSrc**
Section	**[SIP]**
Written Format	SipAutoAdaptRtpSrc=b
Value Range	b=0, Not to handle (default); b=1, Handle.

Description	Sets whether to handle the RTP package of which the Source field is always changing for the SHN B-type board.

3.1.2.28.1.42 SipSpecialReg

Configuration Item	**SipSpecialReg**
Section	**[SIP]**
Written Format	SipSpecialReg=z
Value Range	z=1, The from and to fields in the registration message don't carry the sever port; z=0, The from and to fields in the registration message carry the sever port (default).
Description	Sets whether to have the from and to fields in the registration message carry the sever port.

3.1.2.28.1.43 SipTotalWhiteListNum

Configuration Item	**SipTotalWhiteListNum** **SipTotalBlackListNum** **SipWhiteList** **SipBlackList**
Section	**[SIP]**
Written Format	SipTotalWhiteListNum=n SipTotalBlackListNum=m SipWhiteList[i]=x SipBlackList[i]=x
Value Range	n: The amount of the IPs in the white list; m: The amount of the IPs in the black list; x: IP address.
Description	Sets the white/black list.
Example	Set the white list: SipTotalWhiteListNum=3 SipWhiteList[0]=201.123.115.100 SipWhiteList[1]=201.123.115.200 SipWhiteList[2]=201.123.115.300 Set the black list: SipTotalBlackListNum=3 SipBlackList[0]=201.123.115.10 SipBlackList[1]=201.123.115.20 SipBlackList[2]=201.123.115.30

3.1.2.28.1.44 SipTotalBlackListNum

Refer to SipTotalWhiteListNum

3.1.2.28.1.45 SipWhiteList

Refer to SipTotalWhiteListNum

3.1.2.28.1.46 SipBlackList

Refer to SipTotalWhiteListNum

3.1.2.28.1.47 SipSrtp

Configuration Item	**SipSrtp**
Section	**[SIP]**
Written Format	SipSrtp=b

Value Range	b=0, Disable Srtp encryption (default); b=1, Enable Srtp encryption.
Description	Sets Srtp encryption.

3.1.2.28.1.48 SipSessionExpiresMin

Configuration Item	**SipSessionExpiresMin**
Section	**[SIP]**
Written Format	SipSessionExpiresMin=b
Value Range	b=0, Not to designate the minimum refresh time for the session (default); b>=90, Designate the minimum refresh time for the session.
Description	Sets whether to designate the minimum refresh time for the session. The refresh time of the session in the received invite message should be longer than the value set in this item, or the driver will reply the 422 message to reject the invite message.
Note	This item is valid only when it is used with the configuration item SipSessionExpires.

3.1.2.28.1.49 SipSessionExpires

Configuration Item	**SipSessionExpires**
Section	**[SIP]**
Written Format	SipSessionExpires =b
Value Range	b=0, Not to designate the refresh time for the session (default); b>= the minimum refresh time for the session (SipSessionExpiresMin), Designate the refresh time for the session.
Description	Sets whether to have the corresponding field set in this item included in the invite and 200 ok messages.
Note	This item is valid only when it is used with the configuration item SipSessionExpiresMin.

3.1.2.28.1.50 SipNetMultiIP

Configuration Item	**SipNetMultiIP**
Section	**[SIP]**
Written Format	SipNetMultiIP[i] =[n,m]-x
Value Range	i: The subscript, from 0 to 7; n: Starting channel number; m: Ending channel number; x: IP Address.
Description	Sets the IP addresses used for outgoing calls.
Note	● The subscript must start at 0 and increase by 1; ● The dual NIC feature must be enabled before using this configuration item. That is the configuration item SipMutiIpOn must be set to 1.

3.1.2.28.1.51 SipTransferRTPFromHost

Configuration Item	**SipTransferRTPFromHost**
Section	**[SIP]**
Written Format	SipTransferRTPFromHost =b
Value Range	b=0: The RTP packets interact directly between the OCT network port and the external terminal (default); b=1: The RTP packets interact via the host between the OCT network port and the external terminal.
Description	Sets whether to have the RTP packets interact via the host.

3.1.2.28.1.52 SDPAnswerSpecified

Configuration Item	**SDPAnswerSpecified**

Section	**[SIP]**
Written Format	SDPAnswerSpecified=z
Value Range	z=1, The SDP that responded the SIP message takes only one supported codec; z=0, The SDP that responded the SIP message takes all supported codecs (default).
Description	Sets whether the SDP (200, 183, etc.) that responded the SIP message takes only one supported codec.

3.1.2.28.1.53 SipRequestUseContact

Configuration Item	**SipRequestUseContact**
Section	**[SIP]**
Written Format	SipRequestUseContact=z
Value Range	z=0, reply message according to the Contact address of the SIP message (default); z=1, reply message according to the source address of the SIP message.
Description	Enable or disable the network address learning feature.

3.1.2.28.1.54 SipMultiIpOn

Configuration Item	**SipMultiIpOn**
Section	**[SIP]**
Written Format	SipMultiIpOn =b
Value Range	b=0, Not to enable the dual NIC feature (default); b=1, Enable.
Description	Sets whether to enable the dual NIC feature.
Note	Only when this configuration item is enabled will the configuration item (SipNetMultiIP) and the function (SsmSipSetMutilNetIP) work, which are both related to the dual NIC feature.

3.1.2.28.1.55 SipRestartAudioFor183

Configuration Item	**SipRestartAudioFor183**
Section	**[SIP]**
Written Format	SipRestartAudioFor183 =z
Value Range	z=1, Restart the audio triggered by the 183 message when receiving the 200OK message (default); z=0, Not to restart the audio triggered by the 183 message when receiving the 200OK message.
Description	Sets whether to restart the audio triggered by the 183 message when receiving the 200OK message.

3.1.2.28.1.56 SipSingleProtocol

Configuration Item	**SipSingleProtocol**
Section	**[SIP]**
Written Format	SipSingleProtocol =z
Value Range	z=1, Only support TCP or UDP to transport the SIP message; z=0, Support both TCP and UDP to transport the SIP message (default).
Description	Sets whether to support only one or both of TCP and UDP to transport the SIP message.
Note	This configuration item specifies the protocol type based on SipTransportProtocol. If SipTransportProtocol is set to UDP, it only supports UDP to transport the SIP message; if SipTransportProtocol is set to TCP, it only supports TCP to transport the SIP message.

3.1.2.28.1.57 SipSendRequestToSrcIP

Configuration Item	**SipSendRequestToSrcIP**
Section	**[SIP]**

Written Format	SipSendRequestToSrcIP=z
Value Range	z=1, Send the request message to the source address of a session (default); z=0, Send the request message to the contact field in the session message.
Description	Sets whether to send the request message to the source address of a session.

3.1.2.28.1.58 SipExpiresPercent

Configuration Item	**SipExpiresPercent**
Section	**[SIP]**
Written Format	SipExpiresPercent =m
Value Range	m indicates a percentage of SipRegRefreshTime. 0<m≤100, with the default value of 100.
Description	Sets the percentage of SipRegRefreshTime.

3.1.2.28.1.59 UserAgent

Configuration Item	**UserAgent**
Section	**[SIP]**
Written Format	UserAgent =a
Value Range	a: User agent name, the max length of it is 20 characters. The default value is “Synway driver version”, like “Synway/5.3.4.0”.
Description	Sets the user agent name for the SHN series board.
Note	It requires SynCTI Ver. 5.3.1.5 or above.

Example	Some special SIP servers require special soft-terminals to register. In such case, this configuration item can be used to register the Synway SHN series board via the corresponding field in the registration message.
	Supposed it is necessary to register an SHN board in the system to a Huawei soft-switch, follow the settings below:
	Add the configuration item UserAgent manually //under [SIP] section
	UserAgent=HUAWEI OpenEye v3.1 //the terminal name identified by Huawei soft-switch

3.1.2.28.1.60 LocalTLSPort

Configuration Item	**LocalTLSPort**
Section	**[SIP]**
Written Format	LocalTLSPort =n
Value Range	n: Sets the TLS monitoring port. The default value is 5061 and the value range is 1024~65536.
Description	Sets the TLS monitoring port.

3.1.2.28.1.61 SipRegRefreshTime

Configuration Item	**SipRegRefreshTime**
Section	**[SIP]**
Written Format	SipRegRefreshTime=m
Value Range	m indicates the time interval to refresh a registration after it fails. 0<m≤3600, with the default value of 0, calculated by second.
Description	Sets the time interval to refresh a registration after it fails.
Note	● This confguration is only enabled when a registration fails; ● It is valid to the registration failure message that 4xx responds.

3.1.2.28.1.62 SipRegMode

Configuration Item	**SipRegMode**
Section	**[SIP]**
Written Format	SipRegMode =b

Value Range	b=0, Disable(default); b=1, Independent mode. The driver will process the registration request automatically without the participation of the application; b=2, Interactive mode, The application can respond to the registration request and obtain the real-time status of the terminal via the registration server's API, Thus it can take a dynamical manage on the terminal so as to realize the related features of the VoIP proxy server.
Description	Sets registration server mode of the board.

3.1.2.28.1.63 SipDscp

Configuration Item	**SipDscp**
Section	**[SIP]**
Written Format	SipDscp =m
Value Range	m: Sets the Dscp value in the IP header of the SIP signaling packet. 0≤m≤63, with the default value of 0. The bigger m is, the higher priority the network is.
Description	Set the Dscp value in the IP header of the SIP signaling packet to raise the priority of the network.
Note	For the Windows 2000, Windows XP and Windows 2003 systems, the configuration item DisableUserTOSSetting=0 should be added to the position of "HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\TcpIp\Parameters". This configuration item is invalid for system versions above Windows 2003.

3.1.2.28.1.64 SipListOnlyForOptions

Configuration Item	**SipListOnlyForOptions**
Section	**[SIP]**
Written Format	SipListOnlyForOptions =z
Value Range	z=0, Filter the full black list (default); z=1, Only filter the Options message in the black list.
Description	Sets whether to filter the full black list or only filter the options message in the black list.

3.1.2.28.1.65 SipTCPConnectTime

Configuration Item	**SipTCPConnectTime**
Section	**[SIP]**
Written Format	SipTCPConnectTime =b
Value Range	b>0, the TCP connection time, calculated by ms, with the default value of 1000.
Description	Sets the TCP connection time for Sip calls.

3.1.2.28.1.66 SipCallInTickCountLimit

Configuration Item	**SipCallInTickCountLimit**
Section	**[SIP]**
Written Format	SipCallInTickCountLimit =b
Value Range	b>=0, the interval time for incoming calls, calculated by ms, with the default value of 0.
Description	Sets the interval time for incoming calls on a SIP channel.

3.1.2.28.1.67 SipExpiresPercent

Configuration Item	**SipExpiresPercent**
Section	**[SIP]**
Written Format	SipExpiresPercent =m
Value Range	The percent for refreshing the validity period of the registration..Value range: 0<m<=100.
Description	Sets the percent for refreshing the validity period of the registration.

3.1.2.28.1.68 SipLearnNetFromVia

Configuration Item	**SipLearnNetFromVia**
Section	**[SIP]**
Written Format	SipLearnNetFromVia =z
Value Range	z=0, Disable (default); z=1, Enable.
Description	Enable or disable the network address learning feature.
Note	● Only support UDP calls; ● The rport feature must be enabled; ● The first call on the channel is for learning, so it will not change the contact field in the invite message; ● The learning effect is equal to invoking the function SsmSipSetMultiNetIP.

3.1.2.28.1.69 SipSubscribeEvent

Configuration Item	**SipSubscribeEvent**
Section	**[SIP]**
Written Format	SipSubscribeEvent=z
Value Range	z = 0, the driver will reply with 403 after receiving the Subscribe message (default); z = 1, after receiving the Subscribe message, the driver responds with 200 OK; the driver event is thrown, carrying the content of the Subscribe message.
Description	Set the method to process the Subscribe message.

3.1.2.28.1.70 SipGetCalleeNUMFormRequestURI

Configuration Item	**SipGetCalleeNUMFormRequestURI**
Section	**[SIP]**
Written Format	SipGetCalleeNUMFormRequestURI=b
Value Range	b=0, get the called number from the TO field (default); b=1, get the called number from the request line.
Description	Specify the source field of the called number.
Note	Related functions: SsmGetPhoNumStrA, SsmGetPhoNumStr.

3.1.2.28.1.71 ContactSection

Configuration Item	**SipLearnNetFromVia**
Section	**[SIP]**
Written Format	ContactSection=d
Value Range	The IP address and port number for IP communication.
Description	Designates a channel to send the content host and port of the Contact field respectively in the INVITE, 180 and 200 messages.
Note	Same as SsmSipSetContactSection in feature.

3.1.2.28.1.72 HMPRTPBindETH

Configuration Item	**HMPRTPBindETH**
Section	**[SIP]**
Written Format	HMPRTPBindETH=b
Value Range	b=0: Not bound (default); b=1: Bound
Description	Sets whether RTP sending is bound to the specified IP address.
Note	This configuration is only applicable to LINUX system environment.

3.1.2.28.1.73 SipPortsCorrespondToChannels

Configuration Item	SipPortsCorrespondToChannels
Section	[SIP]
Written Format	SipPortsCorrespondToChannels=b
Value Range	b=0: All channels uses the same port during SIP channel registration (default); b=1: Each channel uses an independent port during SIP channel registration.
Description	Set whether to use an independent port for SIP channel registration.
Note	This configuration item is not currently supported in the LINUX environment.

3.1.2.28.2SHN A Series

3.1.2.28.2.1 Advanced Setting for SHN A Series

3.1.2.28.2.1.1 Setting RTP Traversal through NAT and SIP Server

3.1.2.28.2.1.1.1EnableRTPStun

Refer to RegRealm

3.1.2.28.2.1.1.2StunServerIP

Refer to RegRealm

3.1.2.28.2.1.1.3Register

Refer to RegRealm

3.1.2.28.2.1.1.4UserName

Refer to RegRealm

3.1.2.28.2.1.1.5RegPassword

Refer to RegRealm

3.1.2.28.2.1.1.6Domain

Refer to RegRealm

3.1.2.28.2.1.1.7RegExpires

Refer to RegRealm

3.1.2.28.2.1.1.8MapIP

Refer to RegRealm

3.1.2.28.2.1.1.9RegRealm

Configuration Item	**EnableRTPStun** **StunServerIP** **Register** **UserName** **RegPassword** **Domain** **RegExpires** **MapIP** **RegRealm**
Section	**[BoardId=x]**
Written Format	EnableRTPStun=y StunServerIP=p Register=m UserName=s RegPassword=j Domain=x RegExpires=q MapIP=k RegRealm=r
Value Range	y: Sets whether to enable the Stun detection in RTP. p: The IP address of the Stun server in RTP. m: Indicates whether to enable registration. m=0: Disable registration; m≠0: Enable registration. s: User name, containing up to 50 characters. When m≠0, s indicates the registered user name; when m=0, s indicates the content of the 'username' field in SIP URI. j: Registered user password It is valid only when m≠0. x: Registered server It is valid and indicates the registered server address in the format of host[:port] only when m≠0. q: Period of registration validity, indicating how often the board refreshes the registration information to the registered server. k: Mapped IP address of the board. The function SsmSipSetConnectionInforOfSDP can implement the same feature. r: Alias of the SIP server.
Description	Sets the SIP server.
Note	This item is valid only when the configuration item ProtocolType is set to 1.

3.1.2.28.3SHN B/ SHN C Series

3.1.2.28.3.1 Advanced Setting for SHN B/ SHN C Series

3.1.2.28.3.1.1 Setting RTP Traversal through NAT and SIP Server

3.1.2.28.3.1.1.1EnableRTPStun

Refer to RegRealm

3.1.2.28.3.1.1.2StunServerIP

Refer to RegRealm

3.1.2.28.3.1.1.3Register

Refer to RegRealm

3.1.2.28.3.1.1.4UserName

Refer to RegRealm

3.1.2.28.3.1.1.5RegPassword

Refer to RegRealm

3.1.2.28.3.1.1.6Domain

Refer to RegRealm

3.1.2.28.3.1.1.7RegExpires

Refer to RegRealm

3.1.2.28.3.1.1.8MapIP

Refer to RegRealm

3.1.2.28.3.1.1.9RegRealm

Configuration Item	**EnableRTPStun** **StunServerIP** **Register** **UserName** **RegPassword** **Domain** **RegExpires** **MapIP** **RegRealm**
Section	**[BoardId=x]**
Written Format	EnableRTPStun=y StunServerIP=p Register=m UserName=s RegPassword=j Domain=x RegExpires=q MapIP=k RegRealm=r

Value Range	y: Sets whether to enable the Stun detection in RTP. p: The IP address of the Stun server in RTP. m: Indicates whether to enable registration. m=0: Disable registration; m≠0: Enable registration. s: User name, containing up to 50 characters. When m≠0, s indicates the registered user name; when m=0, s indicates the content of the 'username' field in SIP URI. j: Registered user password It is valid only when m≠0. x: Registered server It is valid and indicates the registered server address in the format of host[:port] only when m≠0. q: Period of registration validity, indicating how often the board refreshes the registration information to the registered server. k: Mapped IP address of the board. r: Alias of the SIP server.
Description	Sets the SIP server.
Note	This item is valid only when the configuration item ProtocolType is set to 1.

3.1.2.28.3.1.1.10 DscpFlag

Refer to DscpValue

3.1.2.28.3.1.1.11 DscpValue

Configuration Item	**DscpFlag** **DscpValue**
Section	**[SIP]**
Written Format	DscpFlag=p DscpValue=m
Value Range	p: Sets whether the RTP data package sent by SHN Series B-type or C-type board includes the Dscp value. p=1: Yes; p=0: No (default). m: The Dscp value of the data package, 0≤m≤63, with the default value of 0. Larger m value indicates higher priority.
Description	Sets the Dscp value to enhance the data package's priority in network transmission.

3.1.2.28.3.1.2 Setting RTP Payload

3.1.2.28.3.1.2.1 SizeG711A

Refer to SizeG729

3.1.2.28.3.1.2.2 SizeG711U

Refer to SizeG729

3.1.2.28.3.1.2.3 SizeG729

Configuration Item	**SizeG711A** **SizeG711U** **SizeG729**
Section	**[BoardId=x]**

Written Format	SizeG711A=m SizeG711U=n SizeG729=x
Value Range	m=20 or m=30 n=20 or n=30 x=20 or x=30
Description	Sets the RTP payload in millisecond. Meanwhile adjusts the voice processing capability of the board. That is, sets how soon to take an RTP packet and how big it is.
Note	● The value must be 20 or 30. If it is filled in with other number or none, the system will regard it as 20ms by default. ● These configuration items are applicable only to SHN-B and SHN-C boards.

3.1.2.28.3.1.2.4JitterTime

Configuration Item	**JitterTime**
Section	**[BoardId=x]**
Written Format	JitterTime=x, y
Value Range	40≤x≤800, 160≤y≤1200, calculated by ms, with the default value of x=160, y=400.
Description	Sets the size of RTP JitterBuffer. The parameters x and y indicate the minimum and maximum buffer times respectively. If the reception rate of the RTP packet becomes greater than y or smaller than x, the voice quality may drop.
Note	● This configuration item does not need to be modified unless for special requirements. Otherwise, the voice quality may drop.

3.1.2.29 Common Configuration Items for DST Series (REC Series)

3.1.2.29.1SupplyBoardClockLine

Configuration Item	**SupplyBoardClockLine**
Section	**[BoardId=x]**
Written Format	SupplyBoardClockLine=n
Value Range	n: Board number, numbered from 0
Description	Sets the phone line which offers the reference clock. If this configuration item is not set or set with an illegal value, the board will automatically choose the reference clock.
Note	It is an advanced configuration item which is only applicable to the DST Series boards.

3.1.2.29.2DEventUpdates

Configuration Item	**DEventUpdates**
Section	**[SystemConfig]**
Written Format	DEventUpdates=n
Value Range	n=0: Filtrate events (default); n=1: Not to filtrate events.
Description	Works as a switch to control the event filtration, helping the board to filtrate repeated events once it is enabled.
Note	● This configuration item is only applicable to the DST Series boards.

3.1.2.30 Common Configuration Items for ATP Series (REC Series)

3.1.2.30.1 Setting Input Signal Gain

3.1.2.30.1.1 MicGain

Configuration Item	**MicGain**
Section	**[BoardId=x]**
Written Format	Format 1: MicGain=g,g,g,g Format 2: MicGain=g,g,g,g,g,g,g,g Format 3: MicGain=g,g,g,g,g,g,g,g,g,g,g,g,g,g,g,g
Value Range	g=0: Normal volume g=1: Increased volume
Description	Sets the gain of input signals on the analog recording channel. If the voice signal comes from the analog phone line, the gain should be set to 0DB (g=0); if it comes from the moving coil microphone, the gain should be set to 20DB (g=1).
Note	● Format 1 is applicable to the ATP Series USB recording box; ● Format 2 is applicable to the 8-channel ATP Series boards; ● Format 3 is applicable to the 16-channel ATP Series boards; ● This configuration item is not applicable to ATP-24A series boards.

3.1.2.30.2 Setting Idle Channel's Capability of Energy Detection

3.1.2.30.2.1 EnableIdleChTA

Configuration Item	**EnableIdleChTA**
Section	**[BoardId=x]**
Written Format	Format 1: EnableIdleChTA =b,b,b,b Format 2: EnableIdleChTA =b,b,b,b,b,b,b,b Format 3: EnableIdleChTA =b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b
Value Range	b=0: Disable this capability (default); b=1: Enable this capability
Description	Sets whether the analog recording channel can detect the energy and receive DTMF digits when it is in an idle state.
Note	● Format 1 is applicable to the ATP Series USB recording box; ● Format 2 is applicable to the 8-channel ATP Series boards; ● Format 3 is applicable to the 16-channel ATP Series boards.

3.1.2.30.3 Acquiring More Recording Formats

3.1.2.30.3.1 DspCoder

Configuration Item	**DspCoder**
Section	**[BoardId=x]**
Written Format	DspCoder=n
Value Range	N=131: Load G.729A coder; N=49: Load GSM coder; N=85: Load MP3 8kbps coder; N=86: Load MP3 16kbps coder.
Description	Sets the coder loaded by an on-board DSP which gives more formats for recording.

Note	● It is an advanced configuration item which is only applicable to the following boards (in bracket list the corresponding codecs supported): SHD-30B-CT/PCI/FJ （G.729A，GSM） SHD-60B-CT/PCI/FJ （G.729A，GSM） DTP-30C/PCI+ （G.729A，GSM，MP3 8kbps，MP3 16kbps） DTP-60C/PCI+ （G.729A，GSM，MP3 8kbps，MP3 16kbps） DTP-120C/PCI+ （G.729A，GSM，MP3 8kbps，MP3 16kbps） DTP-30C/PCIe+ （G.729A，GSM，MP3 8kbps，MP3 16kbps） DTP-60C/PCIe+ （G.729A，GSM，MP3 8kbps，MP3 16kbps） DTP-120C/PCIe+ （G.729A，GSM，MP3 8kbps，MP3 16kbps） ATP-24A-PCI+ （G.729A，GSM，MP3 8kbps，MP3 16kbps） ATP-24A-PCIe+ （G.729A，GSM，MP3 8kbps，MP3 16kbps） DST-24B/PCI+ （G.729A，GSM，MP3 8kbps，MP3 16kbps） DST-24B/PCI+(SSW) （G.729A，GSM，MP3 8kbps，MP3 16kbps） DST-24B/PCIe+ （G.729A，GSM，MP3 8kbps，MP3 16kbps） ● It requires SynCTI Ver. 5.0.2.0 or above.

3.1.2.30.4Setting Prerecord Feature

3.1.2.30.4.1 PrerecordEnable

Configuration Item	**PrerecordEnable**
Section	**[SystemConfig]**
Written Format	PrerecordEnable=b
Value Range	b=0: Disable this feature (default); b=1: Enable this feature
Description	Enables or disables the prerecord feature.
Note	● This function is applicable to ATP Series boards, the analog trunk channels of SHT and SHF Series boards, the TUP, ISUP and ISDN channels of SHD Series boards; ● It requires SynCTI Ver. 5.3.3.0 or above.

3.1.2.30.4.2 PrerecordMode

Configuration Item	**PrerecordMode**
Section	**[SystemConfig]**
Written Format	PrerecordMode=m
Value Range	m=0: Start to write the recording data into the internal buffer when the Barge-in detector detects voice activities (default); m=1: Start to write the recording data into the internal buffer when the analog recording channel detects the pickup behavior on the line. The default value is 0.
Description	Sets the specific way to do prerecording.
Note	● This function is applicable to ATP Series boards, the analog trunk channels of SHT and SHF Series boards, the TUP, ISUP and ISDN channels of SHD Series boards; ● It requires SynCTI Ver. 5.3.3.0 or above.

3.1.2.30.4.3 PrerecordInsertTime

Configuration Item	**PrerecordInsertTime**
Section	**[SystemConfig]**
Written Format	PrerecordInsertTime=t
Value Range	0<t<n, calculated by millisecond (ms), with the default value of 500. The maximum n is decided by RecordBufSize. n= RecordBufSize/16.

Description	When the application starts the task of file recording after the prerecord feature is enabled, this parameter is used to specify the length of time to write the prerecording data into the file.
Note	● This function is applicable to ATP Series boards, the analog trunk channels of SHT and SHF Series boards, the TUP, ISUP and ISDN channels of SHD Series boards; ● It requires SynCTI Ver. 5.3.3.0 or above.

3.1.2.30.4.4 PrerecordCodec

Configuration Item	**PrerecordCodec**
Section	**[SystemConfig]**
Written Format	PrerecordCodec=m

Value Range	m=1: PCM8;
	m=6: A-law (default);
	m=7: μ-law;
	m=17: IMA ADPCM (only for boards having a hardware encoder);
	m=49: GSM 6.10;
	m=85: MP3.

Description	Sets the prerecording CODEC.
Note	● This function is applicable to ATP Series boards, the analog trunk channels of SHT and SHF Series boards, the TUP, ISUP and ISDN channels of SHD Series boards; ● It requires SynCTI Ver. 5.3.3.0 or above.

3.1.2.30.5 Detecting Dial Pulse

3.1.2.30.5.1 EnablePulseKeyDetect

Configuration Item	**EnablePulseKeyDetect**
Section	**[SystemConfig]**
Written Format	EnablePulseKeyDetect=b
Value Range	b=0: No (default); b=1: Yes.
Description	Sets whether it is necessary to detect the dial pulse.
Note	This configuration item is only applicable to the analog trunk recording channel.

3.1.2.30.6 Setting Maximum Duration of Flash Signal

3.1.2.30.6.1 MaxRecChFlashFilterTime

Configuration Item	**MaxRecChFlashFilterTime**
Section	**[SystemConfig]**
Written Format	MaxRecChFlashFilterTime=t
Value Range	t: time duration, calculated by millisecond (ms), with the default value of 200ms
Description	Sets the filter time for a recording channel to judge flash signals. Refer to the Change in Analog Phone Line Voltage section in Chapter 1 for more information.

3.1.2.30.7 Setting Guard Time for Clearing Ringing Signal Counter

3.1.2.30.7.1 RecChClearRingDelayTime

Configuration Item	**RecChClearRingDelayTime**

Section	**[SystemConfig]**
Written Format	RecChClearRingDelayTime=t
Value Range	t≥0, calculated by second, with the default value of 0
Description	Sets the guard time for clearing the ringing signal counter after the recording channel detects ringing signals and picks up the call.

3.1.2.30.8 Setting Soft-switch Feature

3.1.2.30.8.1 BusPlayListen

Configuration Item	**BusPlayListen**
Section	**[SystemConfig]**
Written Format	BusPlayListen=b
Value Range	b=0: Disable data exchange via CT-Bus (i.e. enable the soft-switch feature); b=1: Enable data exchange via CT-Bus (default).
Description	Sets whether the driver will enable data exchange via CT-Bus. This item is invalid to boards without CT-bus, such as the ATP series, and in such case, the soft-switch feature will instead be enabled automatically. Refer to the Operation Principle of ATP Series section in Chapter 1 for more information.
Note	● When BusPlayListen=0, GsmCodecEnable=0 will be invalid and the driver will load the record/playback software engine to support the monitoring in compressed format.

3.1.2.30.9 Setting MF Detector

3.1.2.30.9.1 AlwaysEnableRxMF

Configuration Item	**AlwaysEnableRxMF**
Section	**[BoardId=x]**
Written Format	AlwaysEnableRxMF=b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b
Value Range	b=0: No (default); b=1: Yes.
Description	Sets the operating mode of the MF detector
Note	This configuration item is only applicable to 24-channel ATP Series.

3.1.2.30.9.2 MFDualAndAllFreqEnScale

Configuration Item	**MFDualAndAllFreqEnScale**
Section	**[BoardId=x]**
Written Format	HighAndLowFreqEnScale=m
Value Range	0≦m(%)≦100, with the default value of 32
Description	Sets the threshold value in percentage for the rate of MF in-band energy to overall energy
Note	This configuration item is only applicable to the ATP series.

3.1.2.30.9.3 MFFreqOffset

Configuration Item	**MFFreqOffset**
Section	**[BoardId=x]**
Written Format	MFFreqOffset=m
Value Range	0≦m(%)≦30, with the default value of 15
Description	Sets the maximum offset error of the MF signal (offset error = (detected frequency – standard frequency)*1024 / detected frequency).
Note	This configuration item is only applicable to the ATP series.

3.1.2.31 Common Configuration Items for DTP Series (REC Series)

3.1.2.31.1 Saving Monitoring Message to File

Refer to 3.1.3.5 Setting SS1 Monitoring Log, 3.1.3.6 Setting SS7 Monitoring Log, 3.1.3.7 Setting ISDN Monitoring Log, 3.1.3.8 Setting Control Information for Log Creating.

3.1.2.31.2 Setting Signaling Message Output by SS7 State Machine

3.1.2.31.2.1 bOpenSpySS7Msu

Configuration Item	**bOpenSpySS7Msu**
Section	**[SS7Spy]**
Written Format	bOpenSpySS7Msu=b
Value Range	b=0: No (default); b=1: Yes.
Description	Sets whether the ISUP/TUP call state machine provided by the SynCTI driver will output original ISUP/TUP messages to the application while processing them. See the section Operation Principle of DTP Series in Chapter 1 for more information.
Note	● This configuration item is only applicable to the DTP Series boards based on SS7 protocol; ● It requires SynCTI Ver. 4.7.2.0 or above.

3.1.2.31.3 Supporting CorNet Protocol for ISDN Monitoring

3.1.2.31.3.1 ISDNProtocolType

Configuration Item	**ISDNProtocolType**
Section	**[SpyPcm]**
Written Format	ISDNProtocolType=b
Value Range	b=0: No (default); b=1: Yes.
Description	Sets whether to support the CorNet protocol for ISDN monitoring (this protocol is similar to ISDN Q.931).
Note	● This configuration item is only applicable to those DTP Series boards using ISDN signaling; ● It requires SynCTI Ver.5.1.0.0 or above.

3.1.2.31.4 SS1 Monitoring of E1 Networks without R2 Signaling

3.1.2.31.4.1 SpySs1NoR2

Configuration Item	**SpySs1NoR2**
Section	**[SS1Config]**
Written Format	SpySs1NoR2=b
Value Range	b=0: No (default); b=1: Yes.

Description	Sets whether to support the SS1 monitoring of E1 networks without R2 signaling (DTMF transmission mode).
Note	● This configuration item is only applicable to the DTP Series boards using E1 SS1 signaling; ● It requires SynCTI Ver. 5.3.1.3 or above.

3.1.2.31.5 Setting Waiting Time for Channel into Unusable State after Link Async

3.1.2.31.5.1 SpyWaitUnavailableTime

Configuration Item	**SpyWaitUnavailableTime**
Section	**[SystemConfig]**
Written Format	SpyWaitUnavailableTime=t
Value Range	Calculated by millisecond, with the default value of 1280.
Description	If the link keeps asynchronous longer than the set value in this configuration, all digital trunk channels involved will go into the unusable state (i.e. S_CALL_UNAVAILABLE).
Note	● This configuration item is only applicable to the DTP Series boards; ● It requires SynCTI Ver. 5.3.1.3 or above.

3.1.2.31.6 Setting Message Decoding Method

3.1.2.31.6.1 UseHdlc

Configuration Item	**UseHdlc**
Section	**[BoardId=x]**
Written Format	UseHdlc=b
Value Range	b=0: No (default); b=1: Yes.
Description	Sets whether to use the HDLC method to decode messages for the DTP Series C-type boards. The default method is DSP decoding.
Note	● This configuration item is only applicable to the DTP Series C-type boards; ● It requires SynCTI Ver.5.3.1.3 or above.

3.1.2.31.7 Supporting RBS Protocol for SS1 Monitoring

3.1.2.31.7.1 RBSType

Configuration Item	**RBSType**
Section	**[SS1Config]**
Written Format	RBSType=b

Value Range	b=0: Not use RBS protocol (default);
	b=1: Use E&M WINK (R1);
	b=2: Use E&M IMMEDIATE;
	b=3: Use E&M WINK (DTMF).

Description	Sets which kind of RBS protocol is supported in SS1 monitoring of T1 networks.
Note	● This configuration item is only applicable to the DTP Series boards supporting SS1 monitoring of T1 trunk; ● It requires SynCTI Ver. 5.3.1.3 or above.

3.1.2.31.8 Supporting Event Output of Original L2 Message for ISDN Monitoring

3.1.2.31.8.1 bOpenSpyIsdnL2

Configuration Item	**bOpenSpyIsdnL2**
Section	**[ISDN]**
Written Format	bOpenSpyIsdnL2=k
Value Range	k=0: Do not output the event E_RCV_IsdnL2SpyMsu (Default); k=1: Output the event E_RCV_IsdnL2SpyMsu.
Description	Sets whether to have the DTP board output the event E_RCV_IsdnL2SpyMsu. The application program can invoke the function SsmGetIsdnL2SpyMsu to obtain the content of the message upon receiving this event.

3.1.2.32 Common Configuration Items for IPR Series (REC Series)

3.1.2.32.1 Advanced Universal Configuration Items for SynIPRecorder in Master

3.1.2.32.1.1 RcvTimeSpan

Configuration Item	**RcvTimeSpan**
Section	**[BoardId=x]**
Written Format	RcvTimeSpan =m
Value Range	m: m is a number with the default value of 20.
Description	Working period of receiving threads. It indicates how often the RTP data is received in the Slaver end.

3.1.2.32.1.2 CodecInBufferSize

Configuration Item	**CodecInBufferSize**
Section	**[BoardId=x]**
Written Format	CodecInBufferSize =m
Value Range	m: m is a number with the default value of 4096.
Description	Size of the buffer temporarily storing the data that needs to be decoded (in byte)

3.1.2.32.1.3 SlaverLogLevel

Configuration Item	**SlaverLogLevel**
Section	**[BoardId=x]**
Written Format	SlaverLogLevel =m

Value Range	m = 0: system-level failure;
	m = 1: common failure;
	m = 2: warning (default);
	m = 3: common information;
	m = 4: detailed information including call stack;
	m = 5: most detailed information including content of transmitted/received data;
	m = 6: debugging information.

Description	Log level in the Slaver end.

3.1.2.32.1.4 SlaverLogType

Configuration Item	**SlaverLogType**
Section	**[BoardId=x]**
Written Format	SlaverLogType =m
Value Range	m = 0: console output; m = 1: DbgView output in the Slaver end; m = 2: log output in the Slaver end; m = 3: forward back to the Master end (default).
Description	Log output type in the Slaver end.

3.1.2.32.1.5 RecorderMixerType

Configuration Item	**RecorderMixerType**
Section	**[BoardId=x]**
Written Format	RecorderMixerType =m

Value Range	m = 0: record the Primary only;
	m = 1: record the Secondary only;
	m = 2: record both parties (default).

Description	Sets the mixer mode during the recording. It is applicable to all of the SynIPRecorder channels.

3.1.2.32.1.6 RecorderJitterBuffer

Configuration Item	**RecorderJitterBuffer**
Section	**[BoardId=x]**
Written Format	RecorderJitterBuffer =m,n
Value Range	m: m is a number with the default value of 1280. It indicates the minimum Jitter delay time. n: n is a number with the default value of 3200. It indicates the maximum Jitter delay time.
Description	Values corresponding to the configuration items in JitterBuffer which are separated by the punctuation '.'. x indicates the minimum Jitter delay time while y indicates the maximum Jitter delay time in byte (i.e. the number of sampling points)

3.1.2.32.1.7 RecorderVolume

Configuration Item	**RecorderVolume**
Section	**[BoardId=x]**
Written Format	RecorderVolume =m,n
Value Range	m: m is a number with the default value of 0. it indicates the primary volume with the value range of [-7，+6]. n: n is a number with the default value of 0. it indicates the secondary volume with the value range of [-7，+6].
Description	Sets the recording volume.

3.1.2.32.2Configuration Items for SynIPAnalyzer Series IP PBX Monitoring Port

3.1.2.32.2.1 SIP

Configuration item	**MonitorPortNo** **MonitorPort** **MonitorPortType** **ProxyIPAddress** **NonStationAddressNo** **NonStationAddress** **Special** **MixCSProtocol**
Section	**[IPRSIP]**
Written Format	MonitorPortNo =m MonitorPort[x] = p MonitorPortType[x] = t ProxyIPAddress = i NonStationAddressNo = n NonStationAddress[y] = j Special = k MixCSProtocol = q
Value Range	m: number of monitoring ports. The default value is 0 which means SIP monitoring is disabled, and the upper limit is 20. x: indicates which configuration item it is. Range of value: 0~19. p: the monitoring port, with the default value of 5060. t: the type of the monitoring port (0: UDP, 1: TCP), with the default value of 0. i: a character string in the form of IPv4 address. It indicates the proxy address of SIP, with the default value of 0.0.0.0. n: number of non-Station addresses, with the default value of 0 and the upper limit of 20. These non-Station addresses won't be identified as Station. y: indicates which configuration item it is. Range of value: 0~19. j: non-Station IP address in the form of IPv4 address. k: For some special PBXes where both SIP protocol and proprietary protocol are used, auxiliary signaling processing is needed. When k=1, the SIP/AASP protocol is supported; when k=2, the IPA analysis module will acquire the extension number from both the REQUEST and CONTACT fields in extension monitoring. q: If the call establish process and the call release process use different transport protocols (e.g. the former uses TCP and the latter uses UDP), this configuration item should be set to 1.
Description	SIP monitoring.

3.1.2.32.2.2 Cisco SCCP

Configuration Item	**MonitorPort** **MonitorPortType**
Section	**[IPRSCCP]**
Written Format	MonitorPort = p MonitorPortType = t
Value Range	p: the monitoring port. The default value is 0 which indicates SCCP monitoring is disabled. If SCCP monitoring is enabled, the commonly used port is 2000. t: the type of the monitoring port (0: UDP, 1: TCP), with the default value of 1.
Description	SCCP monitoring.

3.1.2.32.2.3 Avaya H323

Configuration Item	**H225CSPortNO** **H225CSPort** **H225CSPortType** **H225RASPortNO** **H225RASPort** **H225RASPortType** **NonStationAddressNo** **NonStationAddress**
Section	**[IPRAvayaH323]**
Written Format	H225CSPortNo =m H225CSPort[x] = p H225CSPortType[x] = t H225RASPortNo =n H225RASPort[x] = q H225RASPortType[x] = u NonStationAddressNo = o NonStationAddress[y] = j
Value Range	m: number of monitoring ports. The default value is 0 which means Avaya H323 monitoring is disabled, and the upper limit is 20. x: indicates which configuration item it is. Range of value: 0~19. p: the CS port, with the default value of 1720 q: the RAS port. The configuration of this port is optional. If it is not used by the PBX, this item should be set to 0. t: the type of the monitoring port (0: UDP, 1: TCP), with the default value of 1. u: the type of the monitoring port (0: UDP, 1: TCP), with the default value of 0. o: number of non-Station addresses, with the default value of 0 and the upper limit of 20. These non-Station addresses won't be identified as Station. y: indicates which configuration item it is. Range of value: 0~19. j: non-Station IP address in the form of IPv4 address.
Description	Avaya H323 monitoring.

3.1.2.32.2.4 Shortel MGCP

Configuration Item	**CallAgentPort** **CallAgentPortType** **GatewayPort** **GatewayPortType**
Section	**[IPRShortelMGCP]**
Written Format	CallAgentPort = p CallAgentPortType = t GatewayPort=q GatewayPortType=u
Value Range	p: port of the CallAgent. The default value is 0, which means Shortel MGCP monitoring is disabled. If Shortel MGCP monitoring is enabled, the commonly used port is 2727. q: port of the Gateway. The default value is 0, which means Shortel MGCP monitoring is disabled. If Shortel MGCP monitoring is enabled, the commonly used port is 2427. t, u: the type of the monitoring port (0: UDP, 1: TCP), with the default value of 0.
Description	Shortel MGCP monitoring.

3.1.2.32.2.5 H323

Configuration Item	**H225CSPortNO** **H225CSPort** **H225CSPortType** **H225RASPortNO** **H225RASPort** **H225RASPortType** **NonStationAddressNo** **NonStationAddress**
Section	**[IPRH323]**
Written Format	H225CSPortNo =m H225CSPort[x] = p H225CSPortType[x] = t H225RASPortNo =n H225RASPort[x] = q H225RASPortType[x] = u NonStationAddressNo = o NonStationAddress[y] = j
Value Range	m: number of monitoring ports. The default value is 0 which means H323 monitoring is disabled, and the upper limit is 20. x: indicates which configuration item it is. Range of value: 0~19. p: the CS port, with the default value of 1720 q: the RAS port. The configuration of this port is optional. If it is not used by the PBX, this item should be set to 0. t: the type of the monitoring port (0: UDP, 1: TCP), with the default value of 1. u: the type of the monitoring port (0: UDP, 1: TCP), with the default value of 0. o: number of non-Station addresses, with the default value of 0 and the upper limit of 20. These non-Station addresses won't be identified as Station. y: indicates which configuration item it is. Range of value: 0~19. j: non-Station IP address in the form of IPv4 address.
Description	H323 monitoring.

3.1.2.32.2.6 Panasonic MGCP

Configuration Item	**CallAgentPort** **CallAgentPortType** **GatewayPort** **GatewayPortType**
Section	**[IPRPanasonicMGCP]**
Written Format	CallAgentPort=p CallAgentPortType=t GatewayPort=q GatewayPortType=u
Value Range	p: Port of the CallAgent. The default value is 0, which means Panasonic MGCP monitoring is disabled. When Panasonic MGCP monitoring is enabled, the commonly used port is 2727. q: Port of the Gateway. The default value is 0, which means Panasonic MGCP monitoring is disabled. When Panasonic MGCP monitoring is enabled, the commonly used port is 2427. t, u: the type of the monitoring port (0: UDP, 1: TCP), with the default value of 0.
Description	Panasonic MGCP monitoring.

3.1.2.32.2.7 Toshiba MEGACO

Configuration Item	**MonitorPort** **MonitorPortType**
Section	**[IPRToshibaMEGACO]**

Written Format	MonitorPort=p MonitorPortType=t
Value Range	p: Monitoring port. The default value is 0, which means Toshiba MEGACO monitoring is disabled. When Toshiba MEGACO monitoring is enabled, the commonly used port is 2944. t: The type of the monitoring port (0: UDP, 1: TCP), with the default value of 1.
Description	Toshiba MEGACO monitoring.

3.1.2.32.2.8 Siemens H323

Configuration Item	**H225CSPort** **H225CSPortType** **ProprietaryPort** **ProprietaryPortType**
Section	**[IPRSiemensH323]**
Written Format	H225CSPort = p H225CSPortType = t ProprietaryPort = q ProprietaryPortType = u
Value Range	p: CS port. The default value is 1720. q: Port for Siemens PBX proprietary protocol messages. The default value is 4060. t, u: The type of the monitoring port (0: UDP, 1: TCP), with the default value of 1.
Description	Siemens H323 monitoring.

3.1.2.32.2.9 Alcatel

Configuration Item	**MonitorPort** **MonitorPortType**
Section	**[IPRALCATEL]**
Written Format	MonitorPort=p MonitorPortType=t
Value Range	p: Monitoring port. The default value is 0, which means Alcatel monitoring is disabled. When Alcatel monitoring is enabled, the commonly used port is 32640, 7775 or 32128. t: The type of the monitoring port (0: UDP, 1: TCP), with the default value of 0.
Description	Alcatel monitoring.

3.1.2.32.2.10 Mitel

Configuration Item	**MonitorPort** **MonitorPortType**
Section	**[IPRMITEL]**
Written Format	MonitorPort=p MonitorPortType=t
Value Range	p: Monitoring port. The default value is 0, which means Mitel monitoring is disabled. When Mitel monitoring is enabled, the commonly used port is 6800. t: The type of the monitoring port (0: UDP, 1: TCP), with the default value of 1.
Description	Mitel monitoring.

3.1.2.32.2.11 LG Nortel

Configuration Item	**LTPSPort** **LTPSPortType** **PhonePort** **PhonePortType** **NonStationAddressNo** **NonStationAddress**
Section	**[IPRLGNortel]**

Written Format	LTPSPort = m LTPSPortType = t PhonePort = n PhonePortType = q NonStationAddressNo = o NonStationAdress[y] = j
Value Range	m: Port of the LTPS. The default value is 0, which means LG Nortel monitoring is disabled. When LG Nortel monitoring is enabled, the commonly used port is 5588. n: Port of the Phone. The default value is 0, which means LG Nortel monitoring is disabled. When LG Nortel monitoring is enabled, the commonly used port is 5588. t, q: The type of the monitoring port (0: UDP, 1: TCP), with the default value of 0. o: Number of non-Station addresses, with the default value of 0 and the upper limit of 20. These non-Station addresses won't be identified as Station. y: Indicates which configuration item it is. Range of value: 0~19. j: Non-Station IP address in the form of IPv4 address.
Description	LG Nortel monitoring.

3.1.2.32.2.12 Samsung

Configuration Item	**MonitorPort** **MonitorPortType**
Section	**[IPRSAMSUNG]**
Written Format	MonitorPort = p MonitorPortType = t
Value Range	p: Monitoring port. The default value is 0, which means Samsung monitoring is disabled. When Samsung monitoring is enabled, the commonly used port is 6000; t: The type of the monitoring port (0: UDP, 1: TCP), with the default value of 0.
Description	Samsung monitoring.

3.1.2.32.2.13 Tadicom MGCP

Configuration Item	**CallAgentPort** **CallAgentPortType** **GatewayPort** **GatewayPortType**
Section	**[IPRTadicomMGCP]**
Written Format	CallAgentPort = p CallAgentPortType = t GatewayPort = q GatewayPortType = u
Value Range	p: Port of the CallAgent. The default value is 0, which means Tadicom MGCP monitoring is disabled. When Tadicom MGCP monitoring is enabled, the commonly used port is 2727; q: Port of the Gateway. The default value is 0, which means Tadicom MGCP monitoring is disabled. When Tadicom MGCP monitoring is enabled, the commonly used port is 2427; t, u: The type of the monitoring port (0: UDP, 1: TCP), with the default value of 0.
Description	Tadicom MGCP monitoring.

3.1.2.32.2.14 Zenitel

Configuration Item	**MonitorPort** **MonitorPortType**
Section	**[IPRZENITEL]**
Written Format	MonitorPort = p MonitorPortType = t
Value Range	p: Monitoring port. The default value is 0, which means Zenitel monitoring is disabled. When Zenitel monitoring is enabled, the commonly used port is 50001. t: The type of the monitoring port (0: UDP, 1: TCP), with the default value of 1.

Description	Zenitel monitoring.

3.1.2.32.2.15 Nortel UNIStim

Configuration Item	**MonitorPort** **MonitorPortType**
Section	**[IPRNortelUNIStim]**
Written Format	MonitorPort = p MonitorPortType = t
Value Range	p: Monitoring port. The default value is 0, which means Nortel UNIStim monitoring is disabled. When Nortel UNIStim monitoring is enabled, the commonly used port is 5000. t: The type of the monitoring port (0: UDP, 1: TCP), with the default value of 0.
Description	Nortel UNIStim monitoring.

3.1.2.32.3 Advanced General Setting for SynIPAnalyzer Series

3.1.2.32.3.1 SynIPAnalyzer RTP-Only Recording

3.1.2.32.3.1.1 RTPCtrlRec

Configuration Item	**RTPCtrlRec**
Section	**[BoardId=x]**
Written Format	RTPCtrlRec =m
Value Range	m=0: Disable the RTP-Only mode (default); m=1: Enable the RTP-Only mode, and the current protocol configuration for the PBX port will become invalid.
Description	Sets the RTP-Only mode switch for SynIPAnalyzer.
Note	If this configuration item is set to 1, the current protocol configuration for the PBX port will become invalid automatically. That is, the RTP-Only mode has a higher priority.

3.1.2.32.3.2 SynIPAnalyzer Log Analysis

3.1.2.32.3.2.1 IPALogType

Configuration Item	**IPALogType**
Section	**[BoardId=x]**
Written Format	IPALogType=m

Value Range	m=0: console output (default);
	m=1: DbgView output;
	m=2: log output.

Description	Sets the SynIPAnalyzer log output type.

3.1.2.32.3.2.2 IPALogLevel

Configuration Item	**IPALogLevel**
Section	**[BoardId=x]**
Written Format	IPALogLevel=m

Value Range	m=0: system-level failure;
	m=1: common failure;
	m=2: warning (default);
	m=3: common information;
	m=4: detailed information including call stack;
	m=5: most detailed information including content of transmitted/received data;
	m=6: debugging information.

Description	Sets the SynIPAnalyzer log level.

3.1.2.32.3.3 SynIPAnalyzer Data Capture

3.1.2.32.3.3.1 DumpIPData

Configuration Item	**DumpIPData**
Section	**[BoardId=x]**
Written Format	DumpIPData=m
Value Range	m=0: Disable SynIPAnalyzer's IP data capture (default); m=1: Enable SynIPAnalyzer's IP data capture.
Description	The switch for SynIPAnalyzer's IP Data capture.

3.1.2.32.3.3.2 DumpPackNumPerFile

Configuration Item	**DumpPackNumPerFile**
Section	**[BoardId=x]**
Written Format	DumpPackNumPerFile=n
Value Range	n: n is a number with the default value of 2000000, which denotes the saved data size of a single capture packet.
Description	The data size for a single capture packet in case SynIPAnalyzer's IP data capture is enabled.

3.1.2.32.3.3.3 DumpFileReserveNum

Configuration Item	**DumpFileReserveNum**
Section	**[BoardId=x]**
Written Format	DumpFileReserveNum=n
Value Range	n: n is a number with the default value of 0. If it is 0, the driver will save all the captured IP data files; if it is a positive value, the driver will only save a specified number of lately captured files and delete all the other automatically.
Description	The number of the saving files in case SynIPAnalyzer's IP data capture is enabled.

3.1.2.32.3.3.4 DumpFilePath

Configuration Item	**DumpFilePath**
Section	**[BoardId=x]**
Written Format	DumpFilePath=s
Value Range	s is the path to save the captured IP data files, the length of which must be less than 256 bytes. By default, it is the same as the API log's.
Description	The path to save the captured SynIPAnalyzer's IP data files.

3.1.2.33 Common Configuration Items for SHV Series

3.1.2.33.1 OldVar

Configuration Item	**OldVar**
Section	**[BoardId=x]**
Written Format	OldVar =n
Value Range	n = 0: new algorithm (default); n = 1: old algorithm.
Description	Sets the algorithm supported by the SHN-120B-CT/PCIe/VAR or SHD-240E-CT/PCIe/VAR board.
Note	● For the SHN-120B-CT/PCIe/VAR board, the new algorithm supports 60 voice alteration channels, while the old algorithm supports 120 voice alteration channels; ● For the SHN-240E-CT/PCIe/VAR board, the new algorithm supports 120 voice alteration channels, while the old algorithm supports 240 voice alteration channels; ● It requires SynCTI Ver. 5.4.0.0 or above.

3.1.3 Advanced Configuration Items

3.1.3.1 Multi-program Support

3.1.3.1.1 MultiCardMultiProcess

Configuration Item	**MultiCardMultiProcess**
Section	**[SystemConfig]**
Written Format	MultiCardMultiProcess=b
Value Range	b=0: No (default); b=1: Yes
Description	Sets whether the SynCTI driver supports more than one application program running at the same time.

3.1.3.2 Outputting API Debugging Information

3.1.3.2.1 ApiLogEnable

Configuration Item	**ApiLogEnable**
Section	**[DebugView]**
Written Format	ApiLogEnable =b
Value Range	b = 1: Output API function calls information to DebugView; b = 2: Output API function calls information to the LOG file; b = 3: Output events information to the LOG file; b = 4: Output both API function calls and events information to the LOG file; b = 5: Output both API function calls and events information to the DebugView file; b<1 or b>5 (default): Not output information.
Description	Sets whether to output API function calls information or not. If the driver is allowed to output API function calls information, it will output such information as the name, the parameter of each API function invoked by the application as well as the result of the function call, in the form of strings. See the Driver-provided Debugging Feature section in Chapter 1 for more information.

Note	● Since this feature is enabled at the expense of the host CPU cost, usually it is only used in debugging application programs. Disable this feature once the application system is put into formal running.

3.1.3.2.2 ApiLogSetEventRange

Configuration Item	**ApiLogSetEventRange**
Section	**[DebugView]**
Written Format	ApiLogSetEventRange =n..m
Value Range	n: The start number of events to be written into LOG. n=-1(default) indicates all events with the number <=m should be output; m: The end number of events to be written into LOG. m=-1(default) indicates all events with the number >=n should be output.
Description	Sets the event range for logging. Each output event with the number between n and m (including n and m) will be written into the LOG file. If both n and m are equal to -1, it means all output events should be written into the LOG file.
Note	This configuration item becomes effective only when ApiLogEnable is set to output to LOG.

3.1.3.2.3 ApiLogSetChRange

Configuration Item	**ApiLogSetChRange**
Section	**[DebugView]**
Written Format	ApiLogSetChRange =n..m
Value Range	n: The start channel number for logging API function calls and output events. n=-1(default) indicates the channel range is <=m. m: The end channel number for logging API function calls and output events. m=-1(default) indicates the channel range is >=n.
Description	Sets the range of channels where API function calls and output events should be written into the log. All API function calls and output events on the channel whose number is between n and m (including n and m) will be written into the LOG file. If both n and m are equal to -1, it means API function calls and output events on all channels should be written into the LOG file.
Note	This configuration item becomes effective only when ApiLogEnable is set to output to LOG.

3.1.3.2.4 ApiLogCreateMode

Configuration Item	**ApiLogCreateMode**
Section	**[DebugView]**
Written Format	ApiLogCreateMode =n
Value Range	n=0: All channel information written into a log file (default) n=1: Each channel has an independent log file.
Description	Sets the way to create logs for API functions and events.

Note	1. This configuration item becomes effective only when ApiLogEnable is set to output to LOG. 2. The log file name is constituted by the following rule: 1) If ApiLogCreateMode=0, the name of the output log file will be "Log type (ShCtiApi)+ year+ month+ date+ time", e.g. ShCtiApi_2012041915.log. 2) If ApiLogCreateMode=1, the name of the output log file will be "Log type (ShCtiApi)+ channel number + year+ month+ date+ time", e.g ShCtiApi_Ch0015_2012041916.log.

3.1.3.2.5 Mask_SsmGetNoSoundTime

Refer to **Mask_SsmGetRecOffset**

3.1.3.2.6 Mask_SsmGetChStateKeepTime

Refer to **Mask_SsmGetRecOffset**

3.1.3.2.7 Mask_SsmGetPlayedTime

Refer to **Mask_SsmGetRecOffset**

3.1.3.2.8 Mask_SsmGetPlayedPercentage

Refer to **Mask_SsmGetRecOffset**

3.1.3.2.9 Mask_SsmGetPlayOffset

Refer to **Mask_SsmGetRecOffset**

3.1.3.2.10 Mask_SsmGetRecTime

Refer to **Mask_SsmGetRecOffset**

3.1.3.2.11Mask_SsmGetRecOffset

Configuration Item	**Mask_SsmGetNoSoundTime** **Mask_SsmGetChStateKeepTime** **Mask_SsmGetPlayedTime** **Mask_SsmGetPlayedPercentage** **Mask_SsmGetPlayOffset** **Mask_SsmGetRecTime** **Mask_SsmGetRecOffset** **Mask_SsmGetChState**
Section	**[DebugView]**
Written Format	Mask_SsmGetNoSoundTime=b Mask_SsmGetChStateKeepTime=b Mask_SsmGetPlayedTime=b Mask_SsmGetPlayedPercebtage=b Mask_SsmGetPlayOffset=b Mask_SsmGetRecTime=b Mask_SsmGetRecOffset=b Mask_SsmGetChState=b
Value Range	b=0: Not output (default); b=1: Output
Description	Sets whether the driver will output debugging information of some query API functions or not. In the name of this configuration item, the string following 'Mask_' is just the function name. See the Driver-provided Debugging Feature section in Chapter 1 for more information.
Note	This configuration item becomes effective only when the 'API log Output' feature is enabled via the configuration item APILogEnable.

3.1.3.2.12Mask_SsmGetChState

Refer to **Mask_SsmGetRecOffset**

3.1.3.3 Setting ISDN Connection Log

3.1.3.3.1 IsdnLogEnable

Configuration Item	**IsdnLogEnable**
Section	**[DebugView]**
Written Format	IsdnLogEnable =n
Value Range	=0: Not output (default); =1: Output to the LOG file; =2: Output to DebugView; =3: Output to the PCAP file.
Description	Sets whether to record a frame of messages on the data link layer of ISDN protocol.

3.1.3.3.2 DecodeIsdnMsg

Configuration Item	**DecodeIsdnMsg**
Section	**[DebugView]**

Written Format	DecodeIsdnMsg=b
Value Range	b=0: No (default); b=1: Yes.
Description	Sets whether to decode ISDN messages before outputting ISDN debugging messages. The original ISDN messages use binary system, not easy to read. The decoded ISDN messages are in text format, easy to read, applicable to both network and user sides.

3.1.3.3.3 IsdnDebugLog

Configuration Item	**IsdnDebugLog**
Section	**[ISDN]**
Written Format	IsdnDebugLog=b
Value Range	b=0: No (default); b=1: Yes.
Description	Sets whether to output ISDN debugging information on condition that IsdnLogEnable is enabled so that developers can monitor the running results. If IsdnLogEnable is disabled, no information will be output.

3.1.3.4 Setting SS1 Connection Log

3.1.3.4.1 Ss1LogEnable

Configuration Item	**Ss1LogEnable**
Section	**[DebugView]**
Written Format	Ss1LogEnable=n
Value Range	=0: Not output (default); =1: Output to the LOG file; =2: Output to DebugView.
Description	Sets whether to output received and sent ABCD codes and R2 to the log file.

3.1.3.4.2 Ss1LogCreateMode

Configuration Item	**Ss1LogCreateMode**
Section	**[DebugView]**
Written Format	Ss1LogCreateMode=b
Value Range	b=0: All channel information written into a log file (default); b=1: Each channel has an independent log file.
Description	Sets the way to create logs for SS1 connection.
Note	This configuration item becomes effective only when Ss1LogEnable is set to output to LOG.

3.1.3.5 Setting SS1 Monitoring Log

3.1.3.5.1 SpySs1LogEnable

Configuration Item	**SpySs1LogEnable**
Section	**[DebugView]**
Written Format	SpySs1LogEnable=n

Value Range	=0: Not output (default); =1: Output to the LOG file; =2: Output to DebugView.
Description	Sets whether to output information about SS1 CAS and R2 that the driver receives to the log file.

3.1.3.5.2 SpySs1CreateMode

Configuration Item	**SpySs1CreateMode**
Section	**[DebugView]**
Written Format	SpySs1CreateMode=b
Value Range	b=0: All channel information written into a log file (default); b=1: Each channel has an independent log file.
Description	Sets the way to create logs for SS1 monitoring.
Note	This configuration item becomes effective only when SypSs1LogEnable is set to output to LOG.

3.1.3.6 Setting SS7 Monitoring Log

3.1.3.6.1 SpySs7LogEnable

Configuration Item	**SpySs7LogEnable**
Section	**[DebugView]**
Written Format	SpySs7LogEnable=n
Value Range	=0: Not output (default); =1: Output to the LOG file; =2: Output to DebugView; =3: Output to the PCAP file.
Description	Sets whether to output the SS7-related information that the driver receives to the LOG file.

3.1.3.7 Setting ISDN Monitoring Log

3.1.3.7.1 SpyIsdnLogEnable

Configuration Item	**SpyIsdnLogEnable**
Section	**[DebugView]**
Written Format	SpyIsdnLogEnable=n
Value Range	=0: Not output (default); =1: Output to the LOG file; =2: Output to DebugView; =3: Output to the PCAP file.
Description	Sets whether to output the ISDN signaling messages the driver receives (including DASS2 and DPNSS messages) to the LOG file. (Note: Unless otherwise noted, the ISDN signaling in this document indicates Euro-ISDN which does not include DASS2 and DPNSS.)

3.1.3.8 Setting System Information Monitoring Log

3.1.3.8.1 SystemInfoLogEnable

Configuration Item	**SystemInfoLogEnable**
Section	**[DebugView]**
Written Format	SystemInfoLogEnable =n
Value Range	=0: Not output; =1: Output to the LOG file (default); =2: Output to DebugView.
Description	Sets whether to record the information about the link sync, the local-end impedance, E1/T1, CRC-4 and other running information.
Note	This is a system operation log which keeps writing and gets out of the control of the configuration item LogMaxPeriod.

3.1.3.9 Setting Digital Call Monitoring Signaling Log

3.1.3.9.1 DSTLogEnable

Configuration Item	**DSTLogEnable**
Section	**[DebugView]**
Written Format	DSTLogEnable=n
Value Range	=0: Not output (Default); =1: Output to the LOG file; =2: Output to DebugView.
Description	Sets whether to output the digital call concerned signaling messages received by the driver to the log file.
Note	This configuration item is valid only for the driver version 5314 or above.

3.1.3.9.2 DSTLogCreateMode

Configuration Item	**DSTLogCreateMode**
Section	**[DebugView]**
Written Format	DSTLogCreateMode=b
Value Range	b=0: All channel information written into a log file (default); b=1: Each channel has an independent log file.
Description	Sets the way to create logs for digital call monitoring of signaling messages.
Note	This configuration item is valid only when DSTLogEnable is set to 1 (Output to the LOG file).

3.1.3.10 Setting Fax Interaction Command Log

3.1.3.10.1FaxLogEnable

Configuration Item	**FaxLogEnable**
Section	**[DebugView]**
Written Format	FaxLogEnable=n

Value Range	=0: Not output log(Default); =1: Output to the LOG file; =2: Output to DebugView.
Description	Sets whether to send the fax interaction commands received by the upper layer of the driver to the log file.

3.1.3.10.2FaxLogCreateMode

Configuration Item	**FaxLogCreateMode**
Section	**[DebugView]**
Written Format	FaxLogCreateMode=b
Value Range	b=0: All channel information written into a log file (default); b=1: Each channel has an independent log file.
Description	Sets the way to create logs for fax interaction commands.
Note	This configuration item is valid only when FaxLogEnable is set to 1 (Output to the LOG file).

3.1.3.11 Setting Oct Debugging and Error Information Log

3.1.3.11.1OctLogEnable

Configuration Item	**OctLogEnable**
Section	**[DebugView]**
Written Format	OctLogEnable=n
Value Range	=0: Not output log (Default); =1: Output to the LOG file; =2: Output to DebugView.
Description	Sets whether to output the debugging and error information of Oct to the log file.

3.1.3.12 Setting Control Information for Log Creating

3.1.3.12.1LogCreatePeriod

Configuration Item	**LogCreatePeriod**
Section	**[DebugView]**
Written Format	LogCreatePeriod=n
Value Range	n>0, calculated by hour, with the default value of 24 hours.
Description	Sets the period to create a log.

Note	This configuration item is valid only to the following logs. **Log Type** / **Definition** ShCtiApiLog — Records information about API function calls and events thrown out by the driver. ShCtiSs1Log — Records received and sent ABCD codes and R2 ShCtiIsdnLog — Records a frame of messages on the data link layer of ISDN protocol ShCtiSpySs7Log — Records a frame of messages that the driver receives ShCtiSpySs1Log — Records information about SS1 CAS and R2 that the driver receives ShCtiSpyIsdnLog — Records a frame of messages that the driver receives ShCtiSystemInfoLog — Records system information, such as the link sync, the local-end impedance, E1/T1, CRC-4 and other running information ShCtiDstLog — Records the digital call monitoring signaling message

3.1.3.12.2LogMaxKeep

Configuration Item	**LogMaxKeep**
Section	**[DebugView]**
Written Format	LogMaxKeep=n
Value Range	n>0 or n=-1. The default setting is n=7.
Description	When the number of generated log files is greater than the set value of this parameter, the first generated files will be deleted automatically. If n=-1, it means there is no limit on the number to save log files and no log files will be deleted.
Note	1. To achieve the purpose of this configuration item, the application cannot be restarted after the log output feature is enabled. 2. This configuration item is valid only to the following logs. **Log Type** / **Definition** ShCtiApiLog — Records information about API function calls and events thrown out by the driver. ShCtiSs1Log — Records received and sent ABCD codes and R2 ShCtiIsdnLog — Records a frame of messages on the data link layer of ISDN protocol ShCtiSpySs7Log — Records a frame of messages that the driver receives ShCtiSpySs1Log — Records information about SS1 CAS and R2 that the driver receives ShCtiSpyIsdnLog — Records a frame of messages that the driver receives ShCtiSystemInfoLog — Records system information, such as the link sync, the local-end impedance, E1/T1, CRC-4 and other running information ShCtiDstLog — Records the digital call monitoring signaling message

3.1.3.12.3LogMaxPeriod

Configuration Item	**LogMaxPeriod**
Section	**[DebugView]**
Written Format	LogMaxPeriod=n
Value Range	n>0 or n=-1. The default setting is n=30.
Description	When the number of log generating periods reaches the set value of this parameter, the log output will be stopped automatically. If n= -1, it means there is no limit on the number to generate log files.
Note	1. To achieve the purpose of this configuration item, the application cannot be restarted after the log output feature is enabled. 2. This configuration item is valid only to the following logs.

Log Type	Definition
ShCtiApiLog	Records information about API function calls and events thrown out by the driver.
ShCtiSs1Log	Records received and sent ABCD codes and R2
ShCtiIsdnLog	Records a frame of messages on the data link layer of ISDN protocol
ShCtiSpySs7Log	Records a frame of messages that the driver receives
ShCtiSpySs1Log	Records information about SS1 CAS and R2 that the driver receives
ShCtiSpyIsdnLog	Records a frame of messages that the driver receives
ShCtiDstLog	Records the digital call monitoring signaling message

3.1.3.12.4LogFilePath

Configuration Item	**LogFilePath**
Section	**[DebugView]**
Written Format	LogFilePath=S
Value Range	S is the path to save the log file. The default setting is the working directory of the current application.
Description	The path to save logs. Note that it must be a path already existing, such as ‘D:\’, or the function call will fail. If it is set to null, the log will be saved to the current working path; if it is set to a name, a folder with such name will be created under the current working path to save the log.

Note	This configuration item is valid only to the following logs. **Log Type** / **Definition** ShCtiApiLog: Records information about API function calls and events thrown out by the driver. ShCtiSs1Log: Records received and sent ABCD codes and R2 ShCtiIsdnLog: Records a frame of messages on the data link layer of ISDN protocol ShCtiSpySs7Log: Records a frame of messages that the driver receives ShCtiSpySs1Log: Records information about SS1 CAS and R2 that the driver receives ShCtiSpyIsdnLog: Records a frame of messages that the driver receives ShCtiSystemInfoLog: Records system information, such as the link sync, the local-end impedance, E1/T1, CRC-4 and other running information ShCtiDstLog: Records the digital call monitoring signaling message ShCtiSipLog: Records the signaling and processing information of the IP board

3.1.3.12.5LogOverWrite

Configuration Item	**LogOverWrite**
Section	**[DebugView]**
Written Format	LogOverWrite=n
Value Range	n=0: The old content of the log will not be overridden but be appended with new content upon application restart (default); n=1: The old content of the log will be overridden upon application restart.
Description	Sets whether to override the old content of the log upon application restart.
Note	This configuration item is valid only to the following logs. **Log Type** / **Definition** ShCtiApiLog: Records information about API function calls and events thrown out by the driver. ShCtiSs1Log: Records received and sent ABCD codes and R2 ShCtiIsdnLog: Records a frame of messages on the data link layer of ISDN protocol ShCtiSpySs7Log: Records a frame of messages that the driver receives ShCtiSpySs1Log: Records information about SS1 CAS and R2 that the driver receives ShCtiSpyIsdnLog: Records a frame of messages that the driver receives ShCtiSystemInfoLog: Records system information, such as the link sync, the local-end impedance, E1/T1, CRC-4 and other running information ShCtiDstLog: Records the digital call monitoring signaling message ShCtiSipLog: Records the SIP log information

3.1.3.12.6CreateDumpWhenCrash

Configuration Item	**CreateDumpWhenCrash**
Section	**[SystemConfig]**
Written Format	CreateDumpWhenCrash=n
Value Range	n=0: Not to create a dump file when the driver crashes; n=1: Create a dump file when the driver crashes (default).
Description	Sets whether to enable the feature of creating a dump file when the driver crashes. If the driver crashed in Windows, a dump file named 'Crash-yyyymmdd(date)-hhmmss(time).dmp' would be generated under the current directory of the application. At the VC platform, the dump file would be output when either the application or the driver runs illegal; at other platforms in Windows, the dump file would be output only when the driver runs illegal. This feature is unsupported in Windows 2000 and below. If the driver crashed in Linux, a file named 'core.PID' (PID: Process Identifier) would be generated under the current directory of the application.

3.1.3.13 Setting DSP Access Mode

3.1.3.13.1DspAddr4

Configuration Item	**DspAddr4**
Section	**[BoardId=x]**
Written Format	DspAddr4=n
Value Range	n=0: DSP addressing is based on 2-byte access (default); n=1: DSP addressing is based on 4-byte access.
Description	Sets whether to enable the feature of addressing DSP based on 4-byte access. For some machines models such as IBM X3650 M3 and Advantech 610L, addressing DSP based on 2-byte access is not supported and can cause blue screen. To solve such problem, this configuration item should be set to 1.

3.1.3.14 Setting DSP Echo Cancellation Effect

3.1.3.14.1DspEc

Configuration Item	**DspEc**
Section	**[BoardId=x]**
Written Format	DspEc=n
Value Range	n=0: Disables the EC (Echo Cancellation) feature for DSP (Default); n≠0: Enables the EC feature for DSP. The value of n indicates the echo cancellation effect.
Description	Sets whether to enable the EC feature for DSP and set the echo cancellation effect. This configuration item should be properly set according to actual situation. Below are the detailed instructions: 1. Set DspEc=0 and record the incoming voice on the Digital board. 2. Select a short segment of steady echo from the recorded file and use cooledit to analyze the extremum of the echo. Provided the extremum is val, DspEc=[(val/4)*(val/4)]*32.

Note	This configuration item is only valid to the SHD EC board.

3.1.3.15 Setting CPU Multi-core Sharing Function

3.1.3.15.1EnableSetDpc

Configuration Item	**EnableSetDpc**
Section	**[SystemConfig]**
Written Format	EnableSetDpc=b
Value Range	b=0: No (default); b=1: Yes.
Description	Sets whether to enable the CPU multi-core sharing function for the SynCTI driver (only applicable to WINDOWS system.)

3.1.3.16 Setting the Filtration of DC Deviation

3.1.3.16.1DcOnOffValue

Configuration Item	**DcOnOffValue**
Section	**[Boardid=x]**
Written Format	DcOnOffValue=v
Value Range	The value range for the SHT/ATP/DST series boards is 0≤v≤100; the default value is 0 for SHT-16B-CT/PCI/FAX and 40 for other SHT/ATP/DST series boards; The value range for SHD-30C-CT/PCI, SHD-60C-CT/PCI, SHD-30C-CT/PCI/Fax and SHD-60C-CT/PCI/Fax boards is 0 or 1, with the default value of 0.
Description	For the SHT/ATP/DST series boards, it is to set the threshold of DC deviation; For SHD-30C-CT/PCI, SHD-60C-CT/PCI, SHD-30C-CT/PCI/Fax and SHD-60C-CT/PCI/Fax boards, it is to set whether to enable the DC removal feature. In order to keep the energy unattenuated, the energy will be multiplied by 2 after the DC removal feature is enabled.

3.1.3.16.2BusDcOnOffValue

Configuration Item	**BusDcOnOffValue**
Section	**[Boardid=x]**
Written Format	BusDcOnOffValue=b
Value Range	The value range is 0 or 1, with the default value of 0.
Description	Sets whether to enable the DC removal feature while putting voices onto bus for the console 1 of voice DSP (0: disable; 1: enable).
Note	This configuration item is only valid to the SHT-16D-CT/PCIe board.

3.1.3.17 Setting Driver Event Processing Mode

3.1.3.17.1MultiBoardClock

Configuration Item	**MultiBoardClock**
Section	**[SystemConfig]**
Written Format	MultiBoardClock=b

Value Range	b=0: No (Default); b=1: Yes.
Description	Sets whether to start the multiple threads to handle the driver events.

3.2 Preload Voice Configuration File ShIndex.ini (CTI Series)

3.2.1 [System] Section

3.2.1.1 MaxIndexSeg

Configuration Item	**MaxIndexSeg**
Section	**[System]**
Written Format	MaxIndexSeg=n
Value Range	0≤n≤65535, with the default value of 0. (Note: The value of 'n×the total number of channels×306' must be less than the available physical memory of the computer, otherwise the initialization would fail.)
Description	Sets the total number of preload voice segments on a single channel.
Note	MaxIndexSeg determines the number of [SegNo=x] sections.

3.2.2 [SegNo=x] Section

It is used to set the file information of each voice segment used for memory playback by index. x is the segment number. More than one [SegNo=x] section should be configured if there are many voice segments available.

3.2.2.1 FileName

Configuration Item	**FileName**
Section	**[SegNo=x]**
Written Format	FileName=s
Value Range	s is the name of the file which contains voice segments, being either the standard wav file or the non-header file. If it is the standard wav file, the format specified by the configuration item CodecFormat will be ignored; otherwise, it is regarded as the non-header file, using the CODEC designated by the configuration item CodecFormat. If the full path is not specified, the driver will search for this file under the current directory of the application.
Description	Sets the voice file name.
Example	FileName=d0.wav

3.2.2.2 Alias

Configuration Item	**Alias**
Section	**[SegNo=x]**
Written Format	Alias=s
Value Range	s is an alias with the length not exceeding 20 characters. It does not accept the digits '0' ~ '9' as the initial character. The value being null indicates no alias is specified. The default value is null.
Description	Allocates aliases to voice segments.
Example	Alias=D0

3.2.2.3 CodecFormat

Configuration Item	**CodecFormat**
Section	**[SegNo=x]**
Written Format	CodecFormat=n

Value Range	n=6: A-Law (default)； n=7: μ-Law； n=17: IMA ADPCM。
Description	Sets the voice CODEC.
Note	All preload voice segments should have the same CODEC.

3.2.2.4 StartOffset

Configuration Item	**StartOffset**
Section	**[SegNo=x]**
Written Format	StartOffset=n
Value Range	n: The start position offset of the voice segment in the file, calculated by byte, with the default value of 0. While using ADPCM format, it should be set to the multiple of 256.
Description	Sets the position among voice data in the file where the voice segment is counted from. In the standard wav file, the value being 0, the segment is counted from the first data byte following the file header; in the non-header file, the segment is counted from the top of the file.

3.2.2.5 Length

Configuration Item	**Length**
Section	**[SegNo=x]**
Written Format	Length=n
Value Range	n: The length of the voice segment, calculated by byte, with the default value of -1.
Description	Sets the data length of this voice segment. The conversion ratio between the data length and the time length varies on the CODEC. See the SynCTI Supported CODECs section in Chapter 1 for details.
Note	● If this parameter is set greater than the length of data from the configuration item StartOffset to the end of the file, the length of the voice segment actually loaded is just the value of StartOffset; ● Length being set to -1 indicates the loaded voice segment fills the space from lStartPos to the end of the file.

3.3 SynIPRecorder Slaver Configuration File record_slaver.ini (SynIPR only)

Select ‘Full’ or ‘Special’ installation mode to well install the SynCTI driver. Then the installation program will generate a configuration file named record_slaver.ini under the installation directory, which is used to store the configuration items for the Slaver of SynIPRecorder. The developers can edit or modify this configuration file by using text editing software.

This configuration file is divided into sections. Each section contains one or several items. The name of the section lies within a pair of square brackets and each line of text following the section name is a configuration item. The effective range of a section starts from the second row of the current section and ends with the row above the next section. All configuration items in this file are written in the format of ‘x=y’: on the left side of the sign ‘=’ is the name of the configuration item while on the right side is the content.

3.3.1 Essential Configuration Items for SynIPRecorder in Slaver

3.3.1.1 ActiveConnect

Configuration Item	**ActiveConnect**
Section	**[SlaverConfig]**
Written Format	ActiveConnect=m
Value Range	m: m is a number indicating the connection mode of Slaver, with the default value of 1. Range of value: m=0: Passive connection mode. In this mode, the Slaver will not connect the Master actively, and multiple Slavers can be connected with one Master. The configurations SlaverListenIP and SlaverListenPort are valid while the configurations SlaverIp, SlaverPort, MasterIp and MasterPort are invalid. m=1: Active connection mode. In this mode the Slaver will connect the Master actively but the Master cannot connect the Slaver. The configurations SlaverIp, SlaverPort, MasterIp and MasterPort are valid while the configuration items SlaverListenIP and SlaverListenPort are invalid.
Description	Sets the connection mode of the Slaver.

3.3.1.2 SlaverIP

Configuration Item	**SlaverIP**
Section	**[SlaverConfig]**
Written Format	SlaverIP =m
Value Range	m: m is a character string in the form of IPv4 address. The default value is 127.0.0.1.
Description	IP address in the Slaver end.

3.3.1.3 SlaverPort

Configuration Item	**SlaverPort**
Section	**[SlaverConfig]**
Written Format	SlaverPort =m
Value Range	m: m is a number with the default value of 9885.
Description	Port in the Slaver end.

3.3.1.4 MasterIP

Configuration Item	**MasterIP**
Section	**[SlaverConfig]**
Written Format	MasterIP =m
Value Range	m: m is a character string in the form of IPv4 address. The default value is 127.0.0.1.
Description	Host IP address of SynIPRecorder Master

3.3.1.5 MasterPort

Configuration Item	**MasterPort**
Section	**[SlaverConfig]**
Written Format	MasterPort =m
Value Range	m: m is a number with the default value of 9888.
Description	Host monitoring port of SynIPRecorder Master.

3.3.1.6 SlaverListenIP

Configuration Item	**SlaverListenIP**
Section	**[SlaverConfig]**
Written Format	SlaverListenIP =m

Value Range	m: m is a character string in the form of IPv4 address. The default value is 127.0.0.1.
Description	The monitoring IP address of the Slaver in passive connection mode.

3.3.1.7 SlaverListenPort

Configuration Item	**SlaverListenPort**
Section	**[SlaverConfig]**
Written Format	SlaverListenPort=m
Value Range	m: m is a number with the default value of 9887.
Description	Monitoring port of the Slaver in passive connection mode.

3.3.1.8 RecStereo

Configuration Item	**RecStereo**
Section	**[SlaverConfig]**
Written Format	RecStereo=m
Value Range	m: m is a number 0 or 1, and the default value is 0. When it is set to 1, the stereo recording feature is enabled.
Description	IPR stereo recording switch, only supporting A-Law and μ-Law file recording at present.

3.3.2 Advanced Universal Configuration Items for SynIPRecorder in Slaver

3.3.2.1 KeepAliveTime

Configuration Item	**KeepAliveTime**
Section	**[SlaverConfig]**
Written Format	KeepAliveTime =m
Value Range	m: m is a number with the default value of 3.
Description	Alive time of the Slaver end and the Master end in second. It indicates how often the Slaver communicates with the Master to keep the connection automatically. When disconnection is detected, the Slaver will automatically try to link the Master in every m seconds. When m=0, the Slaver will not try to link the Master automatically in case of disconnection.

3.3.2.2 RTPPortRange

Configuration Item	**RTPPortRange**
Section	**[SlaverConfig]**
Written Format	RTPPortRange =m,n
Value Range	m: m is a number with the default value of 6000. it indicates the starting port for receiving RTP data. n: n is a number with the default value of 10000. it indicates the ending port for receiving RTP data.
Description	Port range for RTP data reception.

3.4 HMP Client Configuration File HMPCodec.ini (SynHMP only)

3.4.1 Essential Configuration Items for HMP Client

3.4.1.1 NICNum

Configuration Item	**NICNum**
Section	**[HMPCodec]**
Written Format	NICNum =n

Value Range	n: Sets the network port amount of the Host where HMP Client locates. Range of value: 0<n≤8, with the default value of 1.
Description	Sets one single machine where HMP Client locates to support multiple network ports. For HMP Server, that is equivalent to that multiple HMP Clients run on one Host. There are some bandwidth restrictions for a single network port to process RTP package. So if HMP Client needs to process more RTP resource, and the Host it locates has more than one network cards, to configure multiple network ports is necessary.

3.4.1.2 LocalIP

Configuration Item	**LocalIP**
Section	**[HMPCodec]**
Written Format	LocalIP[n] =m
Value Range	n: n is the No. of the Host network port. Range of value: 0~ NICNum -1. m: m is a character string in the form of IPv4 address. By default, m=127.0.0.1. If it is multiple network cards, that is, the value of NICNum is larger than 1, the value of m must be the IP address corresponds to the network card.
Description	IP address of the host where HMP Client locates, used to receive RTP from HMP Server and meanwhile transmit RTP data to HMP Server so as to realize the RTP receiving and transmitting between HMP Server and HMP Client.

3.4.1.3 RemoteIP

Configuration Item	**RemoteIP**
Section	**[HMPCodec]**
Written Format	RemoteIP[n] =m
Value Range	n: n is the No. of the Host network port. Range of value: 0~ NICNum -1. m: m is a character string in the form of IPv4 address. By default, m=127.0.0.1. If it is multiple network cards, that is, the value of NICNum is larger than 1, the value of m must be the IP address corresponds to the network card.
Description	IP address of the host where HMP server locates.

3.4.1.4 LocalPort

Configuration Item	**LocalPort**
Section	**[HMPCodec]**
Written Format	LocalPort[n] =m
Value Range	n: n is the No. of the Host network port. Range of value: 0~ NICNum -1. m: Port number, with the default value of 5050.
Description	Port of the host where HMP Client locates.

3.4.1.5 RemotePort

Configuration Item	**RemotePort**
Section	**[HMPCodec]**
Written Format	n: n is the No. of the Host network port. Range of value: 0~ NICNum -1. RemotePort[n] =m
Value Range	m: Port number, with the default value of 5051.
Description	Port of the host where HMP server locates.

3.4.1.6 LogType

Configuration Item	**LogType**
Section	**[HMPCodec]**
Written Format	LogType =m

Value Range	m=0, Not output (default value of Linux OS) m=1, Output to the remote end (default value of Windows OS); m=2, Output to HMPCodecx.log.
Description	Sets the log output mode.

3.4.1.7 LogLocation

Configuration Item	**LogLocation**
Section	**[HMPCodec]**
Written Format	LogLocation =m
Value Range	m is the path to save the HMP Client log file. The default setting is the current directory of the application.
Description	Sets the path to save the HMP Client logs.

3.4.1.8 MaxRtpThread

Configuration Item	**MaxRtpThread**
Section	**[HMPCodec]**
Written Format	MaxRtpThread =m
Value Range	m: Sets the thread amount of HMP Client for RTP CODEC. Range of value: 0<m≤64, with the default value of 4.
Description	More threads are needed for RTP CODEC and forwarding if HMP Client processes more RTP channels. You can assign the treads appropriately according to the amount of the CPU cores and RTP channels.

4 SS7 (CTI Series)

4.1 Basic Concepts

4.1.1 OPC & DPC

Signaling Point (SP) is a node in a signaling network that either originates and receives signaling messages, or transfers signaling messages from one signaling link to another, or both. The signaling point that originates messages is called Originating Point Code (OPC), set in the configuration item OPC; the signaling point that receives messages is called Destination Point Code (DPC). Both OPC and DPC are configured in the file Ss7Server.ini for the SS7 server.

There exist two encoding formats for signaling points: International Signaling Point Code (ISPC) Format and National Signaling Point Code (NSPC) Format in China. As to the international signaling point code, CCITT suggests using the14-bit format. See below for the grouping of the ISPC bits.

3bit	8bit	3bit
ZONE	AREA	POINT

The national signaling point code in China uses the 24-bit format, grouped as follows.

8bit	8bit	8bit
MAIN AREA	SUBAREA	POINT

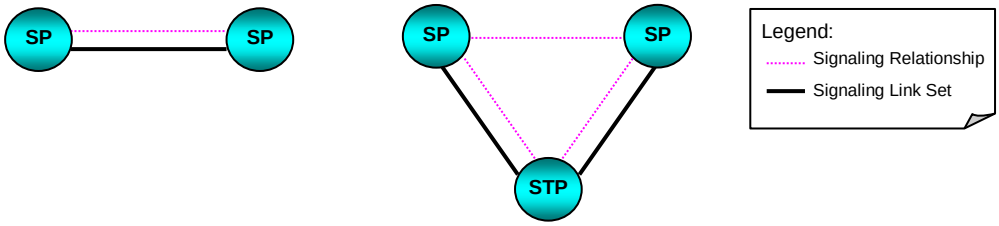
The Synway SHD Series boards support both International Signaling Point Code (ISPC) Format and National Signaling Point Code (NSPC) Format in China, configurable via the configuration item SpCodeLen.

4.1.2 Associated Mode & Quasi-associated Mode

Directly connecting the signaling links between two signaling points to transmit the inbetween signaling messages is called Associated Mode.

Connecting two or more than two signaling links serially via one or more than one signaling transport points to transmit signaling messages, provided the path of signaling messages through the signaling network is predetermined and fixed within a certain period of time, is called Quasi-associated Mode.

These two concepts are vividly illustrated below.



(a) Associated Mode

(b) Quasi-associated Mode

The SynCTI driver supports both the associated mode and the quasi-associated mode, which can be set via the configuration items MaxDPC/DPC and MaxUP_DPC/UP_DPC. See the SS7 Server Configuration section in this chapter for more information.

4.1.3 Signaling Link & Signaling Link Set

The link used to transmit signaling messages between two signaling points is called Signaling Link; a group of links used to connect two signaling points directly constitute a signaling link set.

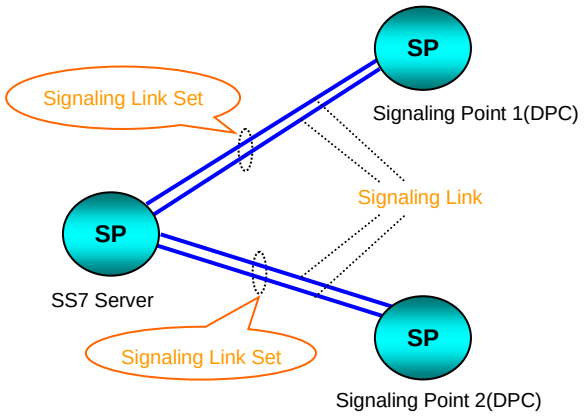
The SynCTI driver supports up to 48 signaling link sets, configurable via the configuration items MaxLinkSet and LinkSet. Each signaling link set supports up to 16 64kbps signaling links, which can be set via the configuration items MaxSs7Pcm and Ss7PcmLink. While 16 signaling links are included in a signaling link set, they will work in a load sharing mode.

The SS7 server is allowed to connect with up to 48 signaling points or signaling transport points.

4.2 SS7 Application System Based on Synway Board

The SynCTI driver assigns the SS7 service to the application by the SS7 server. The application program serves as the Client program, without limitation on the quantity and physical environment. It is acceptable that one or more application programs run in one or more computers, or run with the SS7 server in a same computer.

The SS7 server supports the connectivity of the local signaling point with other signaling points, providing the signaling service for the digital trunks which are compliant with China SS7 protocol. The relationship among the signaling link, the signaling link set, DPC/OPC and the signaling route is shown in the figure below.



Features of this SS7 server:

- A digital trunk can transmit both signaling messages and voice information simultanously. Usually TS16 is used to transport signaling messages while other 30 time slots (TS 1~15 and TS 17~31) to transmit voice information.
- Each signaling link set contains up to 16 64kbps signaling links which work in the load sharing mode. The load management is automatically performed by the server itself.
- The local signaling point is allowed to connect with up to 48 SP or STP. In direction connection, however, only 1 signaling link set (i.e. 1 signaling route) can be used between the local point and the other signaling points.
- Uses the C/S setup which is available via the distributed signaling link/client connectivity supported by the TCP/IP protocol. Network cards are not necessary for a single computer system (the server and the on-board signaling links are involved in a same computer).

- If the SS7 server has physical signaling links in it, it will automatically load the SynCTI driver for the SHD Series boards upon the start. **Note: In such case, the voice time slots on the digital trunk cannot be used as voice paths.**

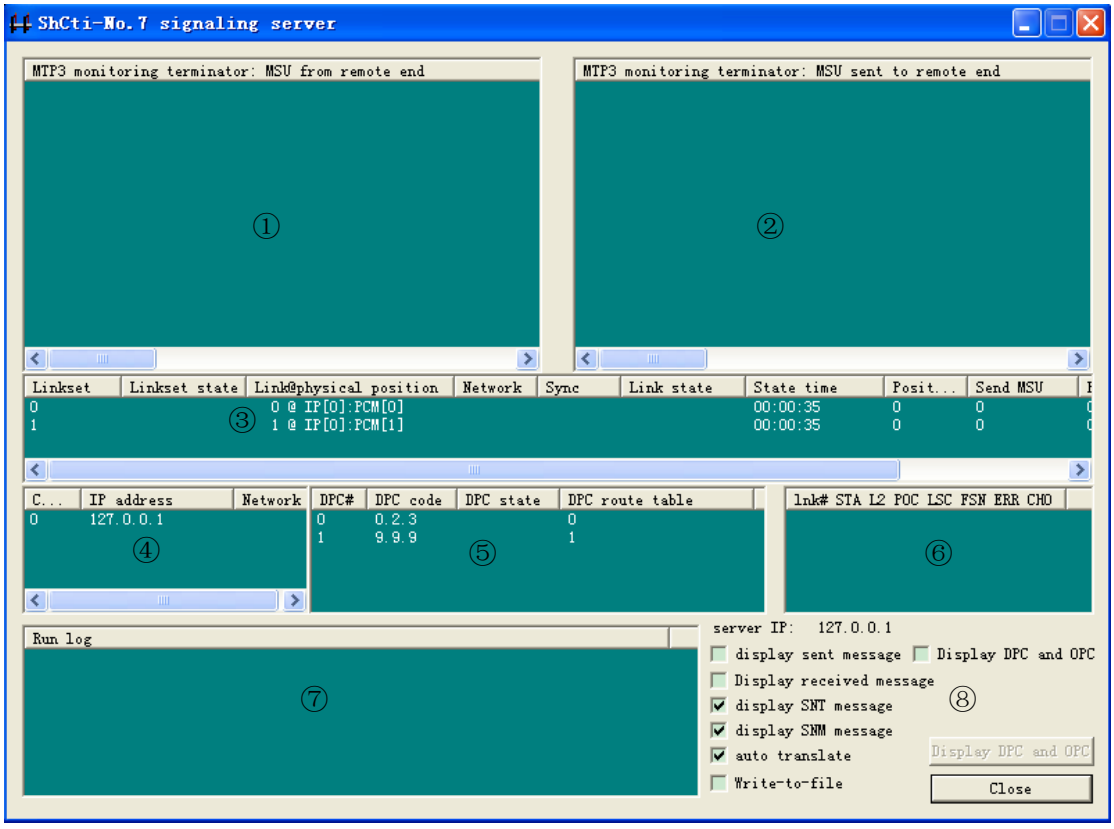
The SS7 server includes the following files.

Ss7Server.ini	Configuration file for the SS7 server
Ss7Monitor.exe	Main program of the SS7 server
Ss7Server.dll	The scheduling component for SS7 signaling
Mtp3.dll	The MTP3 component in the SS7 server
TcpServer.dll	SS7 Server-to-Client Communication Component 1
MmfServer.dll	SS7 Server-to-Client Communication Component 2

4.3 SS7 Server Configuration

The executive program of the SS7 server is Ss7Monitor.exe, and the configuration file it uses is Ss7server.ini.

Run Ss7Monitor.exe and the following interface appears:



There are seven status bars and a button selection area in the interface.

1. ①②: Receive/transmit message list

This program displays the received and sent messages respectively in the above windows ① and ②. However, what kind of messages shall be displayed are determined by which options are selected in the button selection area ⑧:

- Display sent message: Display the message sent to the remote end only if ticked.
- Display received message: Display the message received from the remote end only if ticked.
- Display SNT message: Display the SNT message when ticked
- Display SNM message: Display the SNM message when ticked

- Auto translate: Automatically translate the received/sent message when ticked; otherwise display the received/sent data originally in the hexadecimal format.
- Write-to-file: Output the displayed message to the file with the name 'msu_rcv + number' or 'msu_send + number' only if ticked.
- Display DPC and OPC: Display DPC and OPC only if ticked

When "Auto translate" is ticked, all receive/transmit messages are usually in the following format:

Time Total number Signaling link number# SIO Content

For the TUP message, SIO is just 'TUP' (0x84), followed by the message content. It is usually in the following format:

Title code CIC=PCM:TS Message body

2. ③: Linkset/signaling link information

This window shows the information about signaling links and link sets.

- Linkset: Linkset number, from Section [LinkSetInfo] in the configuration file Ss7Server.ini.
- Linkset state: Working state of linkset, including 'Service On' and 'Service Interrupt'. A signaling linkset will go into the state 'Service On' as long as a link in it is at the state of 'Service On'.
- Link@physical position: Signaling link number and its physical position. For example, '0 @ IP[0]:PCM[0]' means the physical position of Link 0 in this gateway is the E1 with the local PCM numbered 0 on Client 0. This example comes from the configuration file Ss7Server.ini, Section [Ss7PcmLinkInfo].
- Network: Whether the signaling link is registered to the program, including two states 'Connected' and 'Disconnected' (or no display). The signaling link can be used normally only in the state of 'Connected'.
- Sync: Basic frame Synchronization (Timeslot 0), including two states 'Synchronous' and 'Asynchronous'. The signaling link can be used only in the state of 'Synchronous'.
- State time: A time period since the last time the signaling link enters into the state of 'Service On'.
- Position: How many times of positioning that occurs on the signaling link after the program starts.
- Send MSU: Total number of messages sent on the signaling link after the program starts.
- Receive MSU: Total number of messages received on the signaling link after the program starts.
- Switchover MSU: Total number of messages switched over on the signaling link after the program starts.

3. ④: Client Information

This window displays the client IP address and the connection state which are from the configuration file Ss7Server.ini, Section [Ss7ClientInfo].

- Client: Client number
- IP address: IP address of the client
- Network: Whether the client has been successfully connected to the program

4. ⑤: DPC Information

This window displays the DPC information configured by the program which is from the configuration file Ss7Server.ini, Section [DPCInfo].

- DPC#: DPC number which starts from 0
- DPC code: Destination point code, allocated by the central office
- DPC state: Indicates whether the route to this signaling point is available, involving two states 'Service On' and 'Service Interrupt'. The message can be sent to the DPC only when the route to the signaling point is at the state of 'Service On'. The DPC will turn into the state of 'Service On' as long as one of the linksets reaching the DPC is at the state of 'Service On'.
- DPC route table: Route to the DPC, i.e. linkset number.

5. ⑥: Link information

This status bar displays the detailed information on the state of all signaling links, usually used for searching the cause for service interrupt on a signaling link.

Link#	STA	L2	POC	LSC	FSN	ERR	CHO
LINK NUMBER	Link States 0-6	Link Failure Causes (interrupt)	Processor Failures 0-3	Live Communication Server Service 0-1	Forward Sequence Number	spare	spare
	0: uploaded but not started	0: normal	0: normal	0: service is unavailable			
	1: service interrupt	1: BSNR illegal	1: the local end processor failure	1: service is available			
	2: initial positioning	2: FIBR illegal	2: the remote end processor failure				
	3: positioned/ ready	3: T2 timeout	3: both ends processor failure				
	4: positioned/ not ready	4: T6 timeout, the remote end busy					
	5: service on	5: L3 sends a command to stop					
	6: processor failure	6: signaling error rate too high					
		7: during the course of initial positioning, fail to enter a normal position					
		8: Timer 1 timeout					
		9: positioned					

		and ready, receive the interrupt signal of the remote end					
		10: positioned but not ready, receive the interrupt signal of the remote end					
		11: in the state of Service On, receive the interrupt signal of the remote end					
		12: in a processor failure, receive the interrupt signal of the remote end					

6. ⑦: Run log

This log records all MTP3 commands and all error information that pops up during the operation.

This status bar displays the log records generated after the program starts.

4.3.1 Setting Signaling Point Code

4.3.1.1 OPC

Configuration Item	OPC
Section	**[Ss7SystemConfig]**
Written Format	OPC=a.b.c
Value Range	A: Main area code, represented by decimal digits; B: Subarea code, represented by decimal digits; C: Point code, represented by decimal digits.
Description	Sets the signaling point code for the SS7 server which is usually allocated by the central office.
Note	It is an essentially configured item
Example	OPC=1.2.13

4.3.1.2 SpCodeLen

Configuration Item	SpCodeLen
Section	[Ss7SystemConfig]
Written Format	SpCodeLen=n
Value Range	n: Signaling point code format n=24 (default): Uses National Signaling Point Code Format in China n=14: Uses International Signaling Point Code Format
Description	Sets the signaling point code format.
Note	Users in China don't need to configure this item.

4.3.1.3 TEJ

Configuration Item	TEJ
Section	[Ss7SystemConfig]
Written Format	TEJ=n
Value Range	n: Signaling mode of SS7 server n=0 (default): E1 n=1: T1
Description	Sets the signaling mode of SS7 server.
Note	This configuration item requires SynCTI Ver.5.4.0.0 or above.

4.3.2 Setting IP Address

4.3.2.1 ServerIP

Refer to **SecondServerIP**

4.3.2.2 SecondServerIP

Configuration Item	ServerIP SecondServerIP
Section	[Ss7SystemConfig]
Written Format	ServerIP=a.b.c.d SecondServerIP=a.b.c.d。
Value Range	a.b.c.d: IP address. The default value of ServerIP is 127.0.0.1.
Description	Sets the IP address of the SS7 server: ServerIP is used to set the IP address for the SS7 server itself while SecondServerIP is used to set the IP address for the back-up server.
Note	There is no need to set SecondServerIP if only one server is used. In case SecondServerIP is configured, the SS7 server must be started in the warm back-up mode.

4.3.2.3 Port

Configuration Item	Port
Section	[Ss7SystemConfig]
Written Format	Port=n
Value Range	n: Port number, with the default value of 5150.
Description	Sets the monitoring port number of the SS7 sever.
Note	There is no need to configure this item if only one server is used.

4.3.3 Setting Operating Mode and Parameters for MTP3 Layer

4.3.3.1 ConfigMtp3AsSTP

Configuration Item	ConfigMtp3AsSTP
Section	[Ss7SystemConfig]

Written Format	ConfigMtp3AsSTP=m
Value Range	m=0: Signaling point (default). The driver will check whether the DPC of a signaling link set conforms to its OPC at the MTP3 layer. m=1: Signaling transport point. The driver will not examine the DPC at the MTP3 layer.
Description	Sets the operating mode of MTP3 in the SS7 server.

4.3.3.2 SendSNT

Configuration Item	**SendSNT**
Section	**[Ss7SystemConfig]**
Written Format	SendSNT=n
Value Range	n=0: not send (default) n=1: send
Description	Sets parameters for MTP3: whether to regularly send Signaling Link Test Message (SLTM) to the remote PBX.

4.3.3.3 SubServicefield

Configuration Item	**SubServicefield**
Section	**[Ss7SystemConfig]**
Written Format	SubServicefield=0xn
Value Range	n: The subservice code, represented by hexadecimal digits. The 4 lower bits (Bit3~Bit0 are respectively marked as DCBA): DC: Network Indicator 00: International Network 01:Spare International Network 10: National Network 11: Spare National Network BA: Spare. The 4 higher bits (Bit7~Bit4 are respectively marked as DCBA): DC: Spare BA: Testing Message Type 00: Spare 01: SNT message 10: MTNS message 11: Spare The default value of n is 0x18.
Description	sets the SS7 subservice code.
Example	SubServicefield=0x18
Note	The 4 higher bits are valid for SynCTI Ver 5.4.0.0 or above.

4.3.3.4 SubServicefield

Configuration Item	**SubServicefield**
Section	**[LinkSetInfo]**
Written Format	SubServicefield[k]=0xn

Value Range	k: The linkset number. n: The subservice code, represented by hexadecimal digits. The 4 lower bits being valid (Bit3~Bit0 are respectively marked as DCBA): DC: Network Indicator 00: International Network 01:Spare International Network 10: National Network 11: Spare National Network BA: Spare. The 4 higher bits (Bit7~Bit4 are respectively marked as DCBA): DC: Spare BA: Testing Message Type 00: Spare 01: SNT message 10: MTNS message 11: Spare The default value of n is 0x18.
Description	Sets the SS7 subservice code in the specified link set.
Example	SubServicefield[0]=0x18 SubServicefield[1]=0x18
Note	The 4 higher bits are valid for SynCTI Ver 5.4.0.0 or above.

4.3.4 Setting Client Information

4.3.4.1 MaxSs7Client

Refer to **IP**

4.3.4.2 IP

Configuration Item	**MaxSs7Client** **IP**
Section	**[Ss7ClientInfo]**
Written Format	MaxSs7Client=M IP[m]=a.b.c.d
Value Range	M: The total number of client computers that use SS7 servers, with the default value of 0. The maximum value of M is 10. If M＞0, the configuration item IP must be displayed for M times. M: The logical client number, numbered from 0, 0≤m<M. a.b.c.d: The IP address of Client m.
Description	MaxSs7Client is used to set the total number of client computers that use SS7 signaling, while IP is used to set the IP address of each client computer point by point. This configuration item can be set to 1 in a single computer system.
Example	If the SS7 server is running with the client in a same computer: **[Ss7ClientInfo]** MaxSs7Client=1 IP[0]=127.0.0.1 If 2 client computers are connected to the SS7 server: **[Ss7ClientInfo]** MaxSs7Client=2 IP[0]=203.203.203.200 IP[1]= 203.203.203.201

4.3.5 Setting Physical Position of Signaling Link

4.3.5.1 MaxSs7Pcm

Refer to **Ss7PcmLink**

4.3.5.2 Ss7PcmLink

Configuration Item	MaxSs7Pcm Ss7PcmLink
Section	[Ss7PcmLinkInfo]
Written Format	MaxSs7Pcm=M Ss7PcmLink[m]=IP[n],LocalPCM[k]
Value Range	M: The total number of signaling links. The SS7 server supports up to 48 signaling link sets, each of which supports up to 16 signaling links. Therefore, the range of M is 0<M≤768; m: The logical signaling link number in the server (numbered from 0), 0≤m<M; n: The logical number for the client computer which is connected to Signaling Link m. This logical number is defined in the configuration item IP; k: The logical number on the client computer for Signaling Link m in the SS7 server. k can be set via the configuration items TotalPcm and Pcm in the file ShConfig.ini.
Description	MaxSs7Pcm is used to set the total number of 64kbps signaling links, while Ss7PcmLink is to set the physical position on the client computer of a signaling link in the SS7 server.
Example	On the assumption that the application system contains one SS7 server, two client computers and two signaling links in total, if the 1st link is physically located at the 1st client computer, with the local number of 0, the 2nd link is at the 2nd client computer, with the local number of 3, relative configuration items are set as follows. **[Ss7ClientInfo]** MaxSs7Client=2 IP[0]=203.203.203.200 IP[1]= 203.203.203.201 **[Ss7PcmLinkInfo]** MaxSs7Pcm=2 Ss7PcmLink[0]=IP[0],LocalPCM[0] Ss7PcmLink[1]=IP[1],LocalPCM[3] If both the two signaling links are at the 1st client computer and their local numbers are respectively 0 and 1, the set values of MaxSs7Pcm and Ss7PcmLink shall be: Ss7PcmLink[0]=IP[0],LocalPCM[0] Ss7PcmLink[1]=IP[0],LocalPCM[1]

4.3.6 Setting Signaling Link Set

4.3.6.1 MaxLinkSet

Refer to **LinkSet**

4.3.6.2 LinkSet

Configuration Item	MaxLinkSet LinkSet
Section	[LinkSetInfo]

Written Format	MaxLinkSet=N Format 1: LinkSet[n]=Ss7PcmLink[i] (only one signaling link involved in the link set) Format 2: LinkSet[n]=Ss7PcmLink[i]+ … +Ss7PcmLink[j] (two or more than two signaling links involved in the link set) Format 3: LinkSet[n]=Ss7PcmLink[i],a.b.c (only one signaling link involved in the link set) Format 4: LinkSet[n]=Ss7PcmLink[i]+ … +Ss7PcmLink[j],a.b.c (two or more than two signaling links involved in the link set)
Value Range	N: The total number of signaling link sets in the server, 1≤N≤48. N determines how many LinkSet the driver will read; n: The logical serial number of the signaling link set, numbered from 0, 0≤n<N; i, ,j: The logical number in the sever for those signaling links involved in Signaling Link Set n, specified in the configuration item Ss7PcmLink. Each link set includes up to 16 signaling links; a.b.c: The OPC allocated to Signaling Link Set n.
Description	MaxLinkSet is used to set the total number of signaling link sets connected to the local server; LinkSet is to configure which signaling links compose the link set.
Note	● Format 1 and Format 3 are used for the situation when only one signaling link is involved in the link set; Format 2 and Format 4 are used for the situation when there are two or more than two signaling links in the link set; ● Format 3 and Format 4 are applicable to the situation that involves different OPCs; ● The OPC of the signaling link set is preferentially specified by the configuration item LinkSet[n]. If not designated in LinkSet, it is set by the configuration item OPC.
Example	MaxLinkSet=2 LinkSet[0]=Ss7PcmLink[1],1.1.1 LinkSet[1]=Ss7PcmLink[2]+ Ss7PcmLink[0],1.1.2

4.3.7 Setting DPC

4.3.7.1 MaxDPC

Refer to **DPC**

4.3.7.2 DPC

Configuration Item	**MaxDPC** **DPC**
Section	**[DPCInfo]**
Written Format	MaxDPC=N DPC[n]=x.y.z,LinkSet[k]
Value Range	N: The total number of DPC, 0≤N≤48; n: The logical DPC number, numbered from 0, 0≤n<N; x.y.z: The signaling point code for DPC n (represented by decimal digits): x - Main Area, y - Subarea. z is the signaling point code allocated by the central office. k: The logical number of the signaling link set which connects the SS7 server with DPC n. This number is specified in the configuration items MaxLinkSet and LinkSet.
Description	MaxDPC is used to set the total number of DPC (SP or STP) connected to the SS7 server; DPC is to set the signaling point code and the route information for the DPC.
Note	● If N≥1, DPC must be configured for N times, numbered from n=0. ● This configuration item is only applicable to the associated mode which you can set via the configuration items MaxUP_DPC and UP_DPC.
Example	MaxDPC=2 DPC[0]=1.1.8,LinkSet[0] DPC[1]=1.1.9,LinkSet[1]

4.3.8 Setting DPC for User Layer Messages

4.3.8.1 MaxUP_DPC

Refer to **UP_DPC**

4.3.8.2 UP_DPC

Configuration Item	MaxUP_DPC UP_DPC
Section	[UP_DPCInfo]
Written Format	MaxUP_DPC=N Format 1: UP_DPC[n]=x.y.z,LinkSet[i] Format 2: UP_DPC[n]= x.y.z,LinkSet[i]+LinkSet[j] Format 3: UP_DPC[n]= x.y.z,LinkSet[i]+LinkSet[j],m
Value Range	N: The number of UP_DPC, 0≤N≤48; n: The logical UP_DPC number, numbered from 0, 0≤n<N; x.y.z: The signaling point code for UP_DPC n (represented by decimal digits): x - Main Area, y - Subarea. z is the signaling point code allocated by the central office. i,j: The logical number of the signaling link set which connects the SS7 server with UP_DPC n. This number is specified in the configuration items MaxLinkSet and LinkSet. m: In case there are 2 signaling link sets between the SS7 server and UP_DPC n, m is to specify the way how the server uses Link Set i and Link Set j. m=0: The load sharing mode (default, and in such case, Format 2 and Format 3 makes no difference for your use) m=1: Signaling Link Set i serves as the normal route while Signaling Link Set j works as the alternate route.
Description	MaxUP_DPC is used to set the total number of SP which connect to the SS7 server via STP. Its value determines how many UP_DPC[n] the driver will read. UP_DPC is used to set the signaling point code and the route information for SP n.
Note	● If N≥1, UP_DPC must be configured for N times, numbered from n=0. ● This configuration item is applicable to the associated mode, the quasi-associated mode, and the hybrid use of these two modes. ● If this item is used in the associated mode, it should comply to the parameters of configuration items MaxDPC and DPC. That is, UP_DPC and its corresponding DPC should point to a same SP. ● The format used to set UP_DPC is determined by the total number of signaling link sets between the SS7 server and UP_DPC n. ✧ Only 1 link set: use Format 1 ✧ 2 link sets available: use Format 2 or Format 3
Example	MaxUP_DPC=2 UP_DPC[0]=1.2.8,LinkSet[0] UP_DPC[1]=1.2.9,LinkSet[1]

4.3.9 Setting Route to Distribute TUP/ISUP Messages

4.3.9.1 UP_DPC

Refer to **DPC**

4.3.9.2 CIC_PCM

Refer to **DPC**

4.3.9.3 DPC

Configuration Item	UP_DPC CIC_PCM DPC
Section	[TUPRouter] / [ISUPRouter]
Written Format	Format 1: UP_DPC[i],CIC_PCM[p]=IP[j],LocalPCM[k] //used for TUP/ISUP protocol Format 2: DPC[i],CIC_PCM[p]=IP[j],LocalPCM[k] // used for TUP/ISUP protocol
Value Range	i: The logical DPC/UP_DPC number. Format 1 specifies the use of UP_DPC, in which case the value of the configuration item MaxUP_DPC must be set greater than 0. It is used in the associated mode or quasi-associated mode. Format 2 specifies the use of DPC, in which case the configuration item MaxUP_DPC may either be set to 0 or not be set. It is only applicable to the associated mode. p: The digital trunk number in the CIC field, and the value is obtained by dividing the initial CIC number from the central office by 32, not necessarily numbered from 0. j: The logical number for the client computer's IP address. See description on configuration items MaxSs7Client and IP for more information. k: The logical digital trunk number on the client computer. See description on configuration items TotalPcm and Pcm for more information.
Description	UP_DPC, CIC_PCM are used to set the route to distribute TUP/ISUP messages. How the SS7 server distributes TUP/ISUP Messages is described as follows. ✧ Every time when the SS7 server receives a TUP/ISUP message from UP_DPC / DPC i, it will acquire the digital trunk number p from the CIC field. If the matched route can be found for this TUP/ISUP message according to the route set by the configuration items UP_DPC, CIC_PCM, the server will change the digital trunk number p to k in the CIC field and forward this TUP message to Client j; otherwise, this TUP/ISUP message will be discarded. ✧ Every time when the SS7 server receives a TUP/ISUP message from Client j, it will acquire the digital trunk number k from the CIC field. If the matched route can be found for this TUP/ISUP message according to the route set by the configuration items UP_DPC, CIC_PCM, the server will change the digital trunk number k to p in the CIC field and forward this TUP message to UP_DPC / DPC i; otherwise, the server will discard the TUP/ISUP message.
Note	● The TUP protocol uses the **[TUPRouter]** section while the ISUP protocol uses the **[ISUPRouter]** section; ● If the [UP_DPCInfo] section is not set or MaxUP_DPC=0, what shall be used is DPC, CIC_PCM: DPC[i],CIC_PCM[p]=IP[j],LocalPCM[k]; ● If the client uses the TUP/ISUP protocol provided by the driver, this configuration item must be set; ● This configuration item shall be used to set all digital trunks allocated by the central office one by one.
Example	[ISUPRouter] DPC[0]:CIC_PCM[0]=IP[0],LocalPCM[0] DPC[1]:CIC_PCM[1]=IP[0],LocalPCM[1]

4.3.10 Setting Way to Start SS7 Server

4.3.10.1 ConfigAsGateway

Configuration Item	ConfigAsGateway
Section	[Monitor]
Written Format	ConfigAsGateway=n
Value Range	n=0: no (default) n=1: yes

Description	Determines whether to load the SynCTI driver at the start of the SS7 server. If no Synway boards are installed in the server, this configuration item shall be set to 0; if the SHD Series boards are installed for signaling links, this item must be set to 1, and in such case the SS7 server will automatically load the SynCTI driver when it is started. When loading the SynCTI driver, the SS7 server uses two configuration files named ShConfig.ini and ShIndex.ini. The server will search both files under the current directory, and then under the SynCTI driver installation directory. If they are not found, the server will fail to start and return error information. Refer to Chapter 4 for the configuration of ShConfig.ini and ShIndex.ini.
Note	If this configuration item is set to 1, other application programs (if available) running in the server are not allowed to load the SynCTI driver.

4.3.11 Setting Display Ability of SS7 Server

4.3.11.1 RcvMsuListMaxItem

Refer to TxMsuListMaxItem.

4.3.11.2 TxMsuListMaxItem

Configuration Item	**RcvMsuListMaxItem** **TxMsuListMaxItem**
Section	**[Monitor]**

Written Format	RcvMsuListMaxItem=n //used for messages from the client to the server
	TxMsuListMaxItem=m //used for messages from the SP/STP to the server

Value Range	n,m: The maximum number of the messages allowed to be displayed, with the default value of 50.
Description	The SS7 server can display the messages from the SP/STP or the client on the screen. The configuration item is used to set the maximum number of the messages allowed to be displayed. TxMsuListMaxItem is used to set the maximum number for the messages from the client to the server while RcvMsuListMaxItem is to set the maximum number for the messages from the SP/STP to the server.

4.3.12 Special Configuration Item for Windows SS7 Server

4.3.12.1 EnDisplayL32Msu

Configuration Item	**EnDisplayL32Msu**
Section	**[Monitor]**
Written Format	EnDisplayL32Msu=n
Value Range	n=0: disable (default) n=1: enable
Description	Display sent message.

4.3.12.2 EnDisplayL23Msu

Configuration Item	**EnDisplayL23Msu**
Section	**[Monitor]**
Written Format	EnDisplayL23Msu=n
Value Range	n=0: disable (default) n=1: enable
Description	Display received message.

4.3.12.3 ShowSNT

Configuration Item	ShowSNT
Section	[Monitor]
Written Format	ShowSNT=n
Value Range	n=0: disable n=1: enable (default)
Description	Display SNT message.

4.3.12.4 ShowSNM

Configuration Item	ShowSNM
Section	[Monitor]
Written Format	ShowSNM=n
Value Range	n=0: disable n=1: enable(default)
Description	Display SNM message.

4.3.12.5 AutoTranslate

Configuration Item	AutoTranslate
Section	[Monitor]
Written Format	AutoTranslate=n
Value Range	n=0: disable (default) n=1: enable
Description	Enables the auto translate feature.

4.3.12.6 WriteToFile

Configuration Item	WriteToFile
Section	[Monitor]
Written Format	WriteToFile=n
Value Range	n=0: disable (default) n=1: enable
Description	Write the displayed message to file.

4.3.12.7 ShowDPCOPC

Configuration Item	ShowDPCOPC
Section	[Monitor]
Written Format	ShowDPCOPC=n
Value Range	n=0: disable (default) n=1: enable
Description	Display DPC and OPC.

4.3.12.8 LogToPcap

Configuration Item	LogToPcap
Section	[Monitor]
Written Format	LogToPcap=n
Value Range	n=0: disable (default) n=1: enable
Description	Sets whether to output the PCAP file which is converted from the MSU file while outputting a log file.

4.3.12.9 HeartTimeOut

Configuration Item	HeartTimeOut
Section	[Ss7SystemConfig]
Written Format	HeartTimeOut=n
Value Range	n>0, calculated by s, with the default value of 60.
Description	Sets the heartbeat timeout value for the connection of the PC client and the server under Windows operating system. In case the application system is busy, this parameter should be set to a value as large as possible to avoid the active disconnection.

4.3.12.10 HandleWithoutCic

Configuration Item	HandleWithoutCic
Section	[Ss7SystemConfig]
Written Format	HandleWithoutCic=b
Value Range	b=0: No (default); b=1: Yes.
Description	Sets whether to get the decoding messages by the function SsmGetDecodeSs7Msu in case there is no CIC set in SS7.

4.3.12.11 OutputTimerStateToLog

Configuration Item	OutputTimerStateToLog
Section	[Monitor]
Written Format	OutputTimerStateToLog =b
Value Range	b=0: No; b=1: Yes (default).
Description	Sets whether to output the internal time information of the SS7 server to the log file.

4.3.13 Special Configuration Item for Linux SS7 Server

4.3.13.1 EnDisplayL32Msu

Configuration Item	EnDisplayL32Msu
Section	[Monitor]
Written Format	EnDisplayL32Msu=n
Value Range	n=0: disable (default) n=1: enable
Description	Display sent message.

4.3.13.2 EnDisplayL23Msu

Configuration Item	EnDisplayL23Msu
Section	[Monitor]
Written Format	EnDisplayL23Msu=n
Value Range	n=0: disable (default) n=1: enable
Description	Display received message.

4.3.13.3 ShowMSU

Configuration Item	ShowMSU
Section	[Monitor]
Written Format	ShowMSU=n
Value Range	n=0: disable (default) n=1: enable

Description	Records the MSU messages to msu.0.log.

4.3.13.4 ShowStatus

Configuration Item	ShowStatus
Section	[Monitor]
Written Format	ShowStatus=n
Value Range	n=0: disable (default) n=1: enable
Description	Records the Link messages to Ss7RunInfo.0.log.

4.3.13.5 ShowSNT

Configuration Item	ShowSNT
Section	[Monitor]
Written Format	ShowSNT=n
Value Range	n=0: disable n=1: enable (default)
Description	Enables the SNT recording feature.

4.3.13.6 ShowSNM

Configuration Item	ShowSNM
Section	[Monitor]
Written Format	ShowSNM=n
Value Range	n=0: disable n=1: enable (default)
Description	Enables the SNM recording feature.

4.3.13.7 AutoTranslate

Configuration Item	AutoTranslate
Section	[Monitor]
Written Format	AutoTranslate=n
Value Range	n=0: disable (default) n=1: enable
Description	Enables the MSU translation feature.

4.3.13.8 LogFileSize

Configuration Item	LogFileSize
Section	[Monitor]
Written Format	LogFileSize=n
Value Range	n>0 and n is an integer; the default value is 50
Description	Sets the size of msu.x.log, calculated by M. If the file size is larger than the set value of this configuration item, the message will be sent to the next file. The total number of files is determined by the configuration item TotalLogFiles.

4.3.13.9 TotalLogFiles

Configuration Item	TotalLogFiles
Section	[Monitor]
Written Format	TotalLogFiles=n
Value Range	n>0 and n is an integer; the default value is 5
Description	Sets the maximum number (n+1) of msu.x.log files. When the number of files exceeds the set value of this configuration item, the earliest file will be overwritten.

4.3.13.10 LogPath

Configuration Item	LogPath
Section	[Monitor]
Written Format	LogPath=S
Value Range	S indicates the saving path of the log file. It should be less than 256 characters and its default value is /var/log/ss7log.
Description	Sets the saving path of msu log, pcap log and Ss7RunInfo log. It must be an existing path.

4.3.13.11 LogToPcap

Configuration Item	LogToPcap
Section	[Monitor]
Written Format	LogToPcap=n
Value Range	n=0: disable (default) n=1: enable
Description	Records the msu message to PCAP file.

4.3.13.12 WriteToFile

Configuration Item	WriteToFile
Section	[Monitor]
Written Format	WriteToFile=n
Value Range	n=0: disable (default) n=1: enable
Description	Write the displayed message to msu file.

4.3.13.13 ShowDPCOPC

Configuration Item	ShowDPCOPC
Section	[Monitor]
Written Format	ShowDPCOPC=n
Value Range	n=0: disable (default) n=1: enable
Description	Display DPC and OPC.

4.3.14 One Ss7Monitor Supporting Multiple Clients

For a digital board using SS7(TUP/ISUP), both the Test program (client application) and the Ss7Monitor program should be started. Ss7Monitor.exe is responsible for data transmission at MTP1-3 the bottom layer of SS7.

4.3.14.1 Clients and Server on a Same Machine

Using the configuration item MultiCardMultiProcess, you can set to determine whether the SynCTI driver supports multiprocessing with multi boards or not. If you set MultiCardMultiProcess to 1, more than one Test program (client application) is allowed to run in the machine.

See the specific example below for details:

Suppose there were two SHD-120D-CT/PCI boards inserted in a machine and the configuration item MultiCardMultiProcess was set to 1. Then three files would be generated, including two independent shconfig.ini files and one ss7server.ini file. First configure the self-loop ISUP call as usual. Then modify some configuration files as follows:

1. In Windows:

ss7server.ini before modification:

[Ss7ClientInfo]
MaxSs7Client=2
IP[0] = 127.0.0.1
IP[1] = 127.0.0.1
……
ss7server.ini after modification:
[Ss7ClientInfo]
MaxSs7Client=2
IP[0] = 127.0.0.1-0
IP[1] = 127.0.0.1-1
……

Shconfig.ini of Client[0] before modification:
[SS7]
Ss7ServerIP = 127.0.0.1
LocalIP = 127.0.0.1
After modification:
[SS7]
Ss7ServerIP = 127.0.0.1
LocalIP = 127.0.0.1-0

Shconfig.ini of Client[1] before modification:
[SS7]
Ss7ServerIP = 127.0.0.1
LocalIP = 127.0.0.1
After modification:
[SS7]
Ss7ServerIP = 127.0.0.1
LocalIP = 127.0.0.1-1
2. In Linux
ss7server.ini after modification:
.....
[Ss7ClientInfo]
MaxSs7Client=2
IP[0]=127.0.0.1-0
IP[1]=127.0.0.1-1
.....

Shconfig.ini of Client[0] after modification：
.....
[SS7]
Ss7ServerIP=127.0.0.1:5150-0
LocalIP=127.0.0.1-0
.....

Shconfig.ini of Client[1] after modification:

.....

[SS7]

Ss7ServerIP=127.0.0.1:5150-1

LocalIP=127.0.0.1-1

.....

Note: In Linux, it is required to add ":Port-ClientID" behind the IP address set in the configuration item Ss7ServerIP under the [SS7] section of the shconfig.ini file. For example, 127.0.0.1:5150-1.Modify and save the above configurations. Then start ss7monitor.exe and two Test programs.

4.3.14.2 Clients and Server on Different Machines

Please refer to 4.4.1.3 Multi-PC system for exact application and configuration.

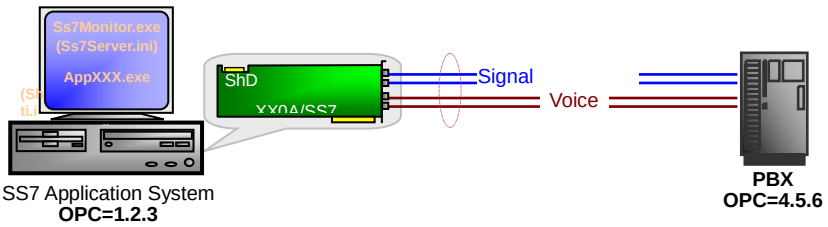
Note: In Linux, it is required to add ":Port-ClientID" behind the IP address set in the configuration item Ss7ServerIP under the [SS7] section of the shconfig.ini file. For example, 127.0.0.1:5150-1.

4.4 Application and Configuration Instance for SS7 Server

4.4.1 Single OPC/Single DPC

4.4.1.1 Single-PC-Single-Board System

The figure below shows the smallest system composed of a computer and one SHD Series board installed on it.



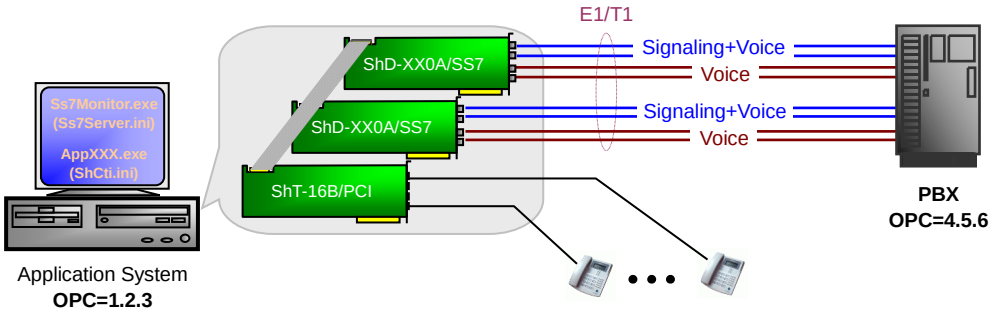
Take the TUP protocol for example. Provided the board used is SHD-60A-CT/SS7/PCI, if as shown above, on the 1st digital trunk, TS16 serves as the signaling link while other 30 time slots work as voice paths, on the 2nd digital trunk, all unused times slots except TS 16 serve as voice paths, the system is configured as follows:

Ss7Server.ini	ShConfig.ini

[Ss7SystemConfig]
 	
OPC=1.2.3
ServerIP=127.0.0.1
 	
[Ss7ClientInfo]
 	
MaxSs7Client=1
IP[0]=127.0.0.1
 	
[Ss7PcmLinkInfo]
 	
MaxSs7Pcm=1
Ss7PcmLink[0]=IP[0],LocalPCM[0]
	……
	[SS7]
	
	Ss7ServerIP=127.0.0.1 //the IP address of the SS7 server
	LocalIP=127.0.0.1 //the IP address of the local PC
	
	[PcmInfo]
	
	TotalPcm=2
	Pcm[0]=0,0
	Pcm[1]=0,1
	
	[BoardId=0]
	
	……

[LinkSetInfo]
 	
MaxLinkSet=1
LinkSet[0]=Ss7PcmLink[0]
 	
[DPCInfo]
 	
MaxDPC=1
DPC[0]=4.5.6,LinkSet[0]
 	
[TUPRouter]
 	
DPC[0]:CIC_PCM[0]=IP[0],LocalPCM[0]
DPC[0]:CIC_PCM[1]=IP[0],LocalPCM[1]
	PcmNumber=2
	
	PcmSSx[0]=7 //SS7
	PcmClockMode[0]=0 //master clock, line synchronization
	Ss7SignalingTS[0]=16 //TS16 used as the signaling time slot
	UseTS16AsCircuit[0]=0
	
	PcmSSx[1]=7 //SS7
	PcmClockMode[1]=2 //slave clock
	Ss7SignalingTS[1]=16 //TS16 used as the signaling time slot
	UseTS16AsCircuit[1]=1

4.4.1.2 Single-PC-Multi-Board System

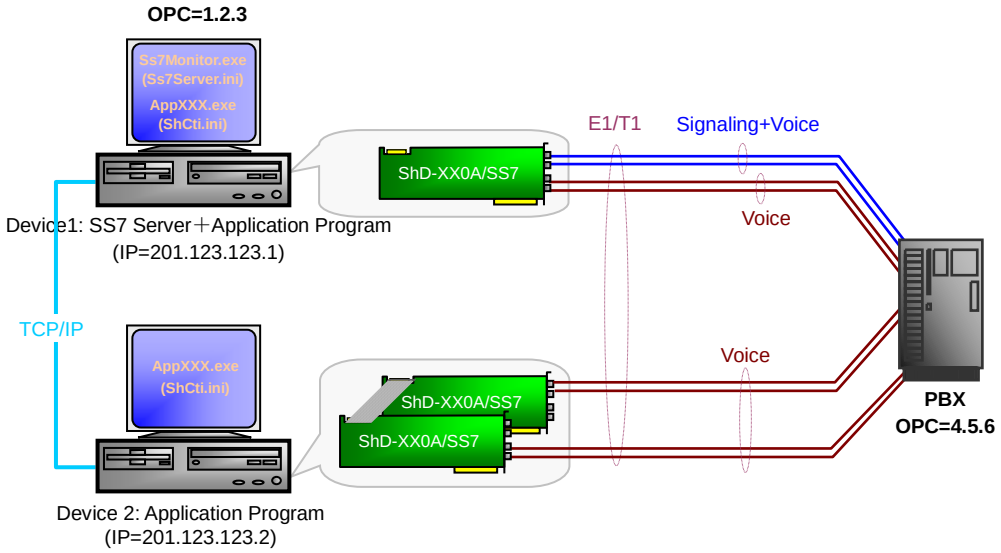


Take the TUP protocol for example. Assuming that the system contains 4 E1/T1 links (established via 2 boards) and 1 16-channel station board: as to both the 1st and the 2nd boards, the 1st PCM is the link to transport both signaling and voices while the 2nd PCM serves as the pure voice path, it is configured as follows.

Ss7Server.ini	ShConfig.ini
[Ss7SystemConfig] OPC=1.2.3 ServerIP=127.0.0.1 **[Ss7ClientInfo]** MaxSs7Client=1 IP[0]=127.0.0.1 [**Ss7PcmLinkInfo**] MaxSs7Pcm=2 Ss7PcmLink[0]=IP[0],LocalPCM[0] Ss7PcmLink[1]=IP[0],LocalPCM[2] **[LinkSetInfo]** MaxLinkSet=1 LinkSet[0]=Ss7PcmLink[0]+Ss7PcmLink[1] **[DPCInfo]** MaxDPC=1 DPC[0]=4.5.6,LinkSet[0] **[TUPRouter]** DPC[0]:CIC_PCM[0]=IP[0],LocalPCM[0] DPC[0]:CIC_PCM[1]=IP[0],LocalPCM[1] DPC[0]:CIC_PCM[2]=IP[0],LocalPCM[2] DPC[0]:CIC_PCM[3]=IP[0],LocalPCM[3]	…… **[SS7]** Ss7ServerIP=127.0.0.1 LocalIP=127.0.0.1 **[PcmInfo]** TotalPcm=4 Pcm[0]=0,0 Pcm[1]=0,1 Pcm[2]=1,0 Pcm[3]=1,1 **[BoardId=0]** …… PcmNumber=2 PcmSSx[0]=7 PcmClockMode[0]=0 //master clock, line synchronization PcmSSx[1]=7 PcmClockMode[1]=2 //slave clock **[BoardId=1]** …… PcmNumber=2 PcmSSx[0]=7 PcmClockMode[0]=2 //slave clock PcmSSx[1]=7 PcmClockMode[1]=2 //slave clock

	[BoardId=2] //configure SHT-16B-CT/PCI ……

4.4.1.3 Multi-PC System



Assume that the system contains 2 devices: the SS7 sever and the application program running at one time in Device 1; Device 2 serving as the Client of Device 1. Device 1 covers 2 digital trunks: on the 1st digital trunk, TS16 serves as the signaling link while other 30 time slots work as voice paths; on the 2nd digital trunk, all unused times slots except TS16 serve as voice paths. Device 2 also contains 2 digital trunks, with all of the 30×2 time slots serving as voice paths. Taking the TUP protocol for example, such a system is configured as follows.

- Device 1: Configuration Files for SS7 Sever and Application Program

Ss7Server.ini	ShConfig.ini

[Ss7SystemConfig]
 	
OPC=1.2.3
ServerIP=201.123.123.1
 	
[Ss7ClientInfo]
 	
MaxSs7Client=2
IP[0]= 201.123.123.1
IP[1]= 201.123.123.2
 	
[Ss7PcmLinkInfo]
 	
MaxSs7Pcm=1
Ss7PcmLink[0]=IP[0],LocalPCM[0]
 	
[LinkSetInfo]
 	
MaxLinkSet=1
LinkSet[0]=Ss7PcmLink[0]
 	
[DPCInfo]
 	
MaxDPC=1
DPC[0]=4.5.6,LinkSet[0]
 	
[TUPRouter]
 	
DPC[0]:CIC_PCM[0]=IP[0],LocalPCM[0]
	……
	[SS7]
	
	Ss7ServerIP=201.123.123.1 //the IP address of the SS7 server
	LocalIP=201.123.123.1 //the IP address of the local PC
	……

DPC[0]:CIC_PCM[1]=IP[0],LocalPCM[1] DPC[0]:CIC_PCM[2]=IP[1],LocalPCM[0] DPC[0]:CIC_PCM[3]=IP[1],LocalPCM[1]	

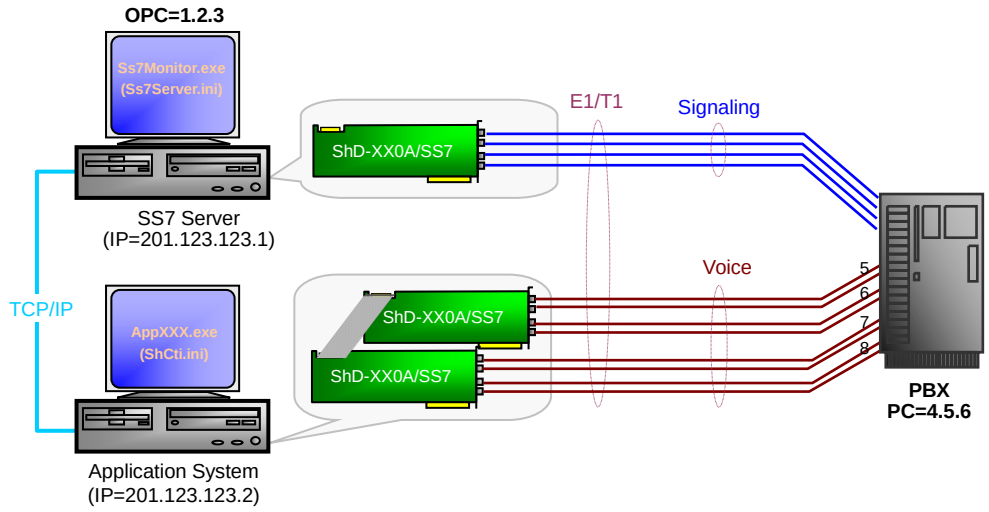
➢ Device 2: Configuration File for Application Program (ShConfig.ini)

……
[SS7]

Ss7ServerIP=201.123.123.1 //the IP address of the SS7 server
LocalIP=201.123.123.2 //the IP address of the local PC
……

4.4.1.4 Multi-PC System with Separate SS7 Server

For high system reliability, we suggest the use of a computer as the separate SS7 server and the run of application programs on other one or more computers as shown below.



In the figure above, Device 1 only processes signaling, which requires not only the SS7 server but also the SynCTI driver loaded for the board only supporting signaling links, while the application system serves as the client. Assuming that the separate SS7 server has 2 digital trunks, with only TS16 being in use, and Device 2 covers 4 digital trunks, consecutively numbered as 5, 6, 7, 8 by the central office in the TUP protocol, the system is configured as follows.

➢ SS7 Server Configuration Files

Ss7Server.ini	ShConfig.ini

[Ss7SystemConfig]
 	
OPC=1.2.3
ServerIP=201.123.123.1 //the IP address of the SS7 server
 	
[Ss7ClientInfo]
 	
MaxSs7Client=2
IP[0]= 201.123.123.1
IP[1]= 201.123.123.2
 	
[Ss7PcmLinkInfo]
 	
MaxSs7Pcm=2
Ss7PcmLink[0]=IP[0],LocalPCM[0]
Ss7PcmLink[1]=IP[0],LocalPCM[1]
	……
	[SS7]
	
	Ss7ServerIP=201.123.123.1 //the IP address of the SS7 server
	LocalIP=201.123.123.1 //the IP address of the client
	……

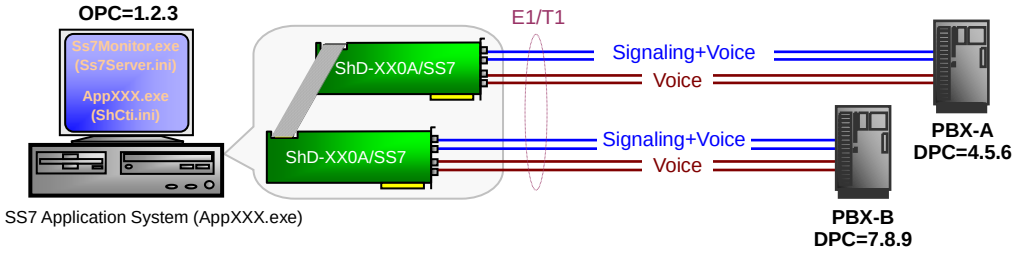
[LinkSetInfo] MaxLinkSet=1 LinkSet[0]=Ss7PcmLink[0]+ Ss7PcmLink[1] **[DPCInfo]** MaxDPC=1 DPC[0]=4.5.6,LinkSet[0] **[TUPRouter]** DPC[0]:CIC_PCM[5]=IP[1],LocalPCM[0] DPC[0]:CIC_PCM[6]=IP[1],LocalPCM[1] DPC[0]:CIC_PCM[7]=IP[1],LocalPCM[2] DPC[0]:CIC_PCM[8]=IP[1],LocalPCM[3] **[Monitor]** ConfigAsGateway=1 //need to load the SynCTI driver	

- Configuration File for Application System (ShConfig.ini)

……
[SS7]

Ss7ServerIP=201.123.123.1 //the IP address of the SS7 server
LocalIP=201.123.123.2 //the IP address of the local PC
……

4.4.2 Single OPC/Multiple DPC



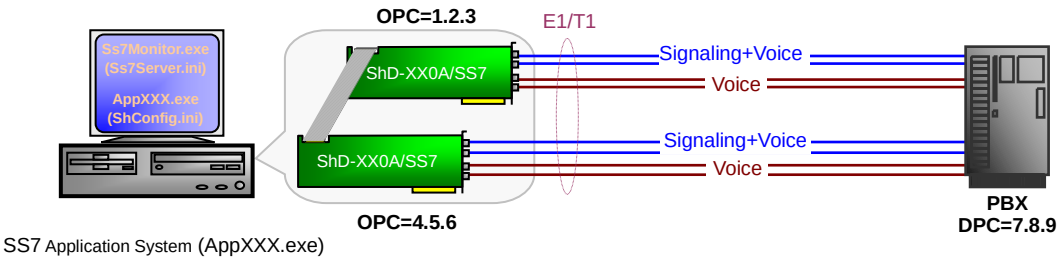
Assume that the application system involves two SHD-60A-CT/PCI/SS7 boards, each of which has the time slots TS 1~15,17~31 on its 2 digital trunks serving as voice paths, TS16 on the 1st digital trunk being the signaling link, and TS16 on the 2nd digital trunk being unused. These 2 boards are respectively connected to 2 PBXes with different DPC. Taking the TUP protocol for example, such a system is configured as follows.

Ss7Server.ini	ShConfig.ini

[Ss7SystemConfig]
 	
OPC=1.2.3
ServerIP=127.0.0.1
 	
[Ss7ClientInfo]
 	
MaxSs7Client=1
IP[0]= 127.0.0.1 //the IP address of the client
 	
[Ss7PcmLinkInfo]
 	
MaxSs7Pcm=2
Ss7PcmLink[0]=IP[0],LocalPCM[0]
Ss7PcmLink[1]=IP[0],LocalPCM[2]
 	
[LinkSetInfo]
 	
MaxLinkSet=2
LinkSet[0]=Ss7PcmLink[0]
LinkSet[1]=Ss7PcmLink[1]
	……
	
	[PcmInfo]
	
	TotalPcm=4
	Pcm[0]=0,0
	Pcm[1]=0,1
	Pcm[2]=1,0
	Pcm[3]=1,1
	
	[BoardId=0]
	
	……
	PcmNumber=2
	
	PcmSSx[0]=7
	PcmSSx[1]=7
	PcmClockMode[0]=0 //master clock, line synchronization
	PcmClockMode[1]=2 //slave clock
	
	[BoardId=1]

[DPCInfo]
MaxDPC=2
DPC[0]=4.5.6,LinkSet[0]
DPC[1]=7.8.9,LinkSet[1]
[TUPRouter]
DPC[0]:CIC_PCM[0]=IP[0],LocalPCM[0]
DPC[0]:CIC_PCM[1]=IP[0],LocalPCM[1]
DPC[1]:CIC_PCM[0]=IP[0],LocalPCM[2]
DPC[1]:CIC_PCM[1]=IP[0],LocalPCM[3]
	……
	PcmNumber=2
	PcmSSx[0]=7
	PcmSSx[1]=7
	PcmClockMode[0]=2 //slave clock
	PcmClockMode[1]=2 //slave clock

4.4.3 SS7 Application System with Multiple OPC



Assume that the application system involves two SHD-60A-CT/PCI/SS7 boards. The 1st PCM serves as the link for both signaling and voices (TS16 is the signaling time slot) while the 2nd PCM works as the pure voice path. These two PCM correspond to a same OPC. Likewise, the 3rd PCM serves as the link for both signaling and voices (TS16 is the signaling time slot) while the 4th PCM works as the pure voice path. These two PCMs correspond to another OPC. Taking the TUP protocol for example, such a system is configured as follows.

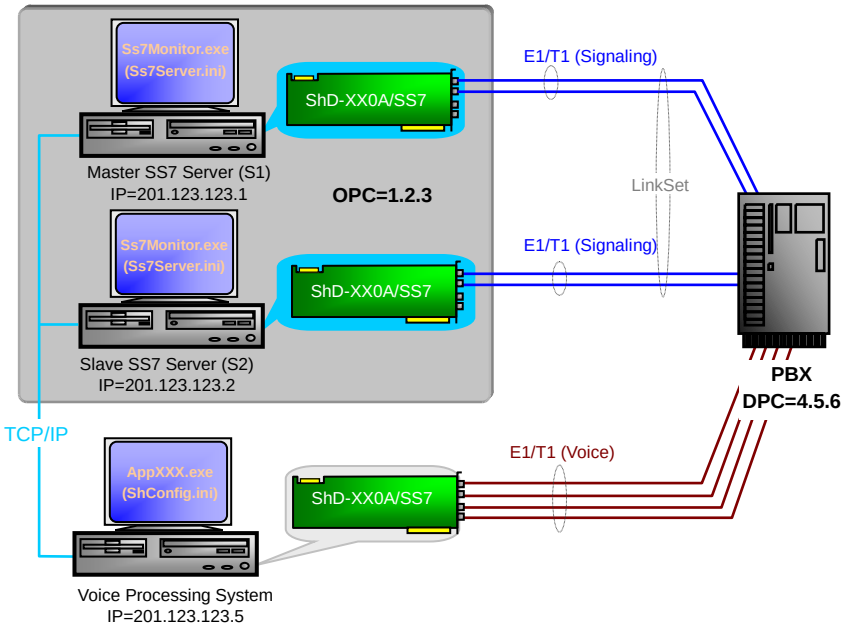
Ss7Server.ini	ShConfig.ini

[Ss7SystemConfig]
OPC=1.2.3
ServerIP=127.0.0.1
[Ss7ClientInfo]
MaxSs7Client=1
IP[0]=127.0.0.1
[Ss7PcmLinkInfo]
MaxSs7Pcm=2
Ss7PcmLink[0]=IP[0],LocalPCM[0]
Ss7PcmLink[1]=IP[0],LocalPCM[2]
[LinkSetInfo]
MaxLinkSet=2
LinkSet[0]=Ss7PcmLink[0],1.2.3
LinkSet[1]=Ss7PcmLink[1],4.5.6
[DPCInfo]
MaxDPC=2
DPC[0]=7.8.9,LinkSet[0]
DPC[1]=7.8.9,LinkSet[1]
[TUPRouter]
DPC[0]:CIC_PCM[0]=IP[0],LocalPCM[0]
DPC[0]:CIC_PCM[1]=IP[0],LocalPCM[1]
DPC[1]:CIC_PCM[0]=IP[0],LocalPCM[2]
DPC[1]:CIC_PCM[1]=IP[0],LocalPCM[3]
	……
	[PcmInfo]
	TotalPcm=4
	Pcm[0]=0,0
	Pcm[1]=0,1
	Pcm[2]=1,0
	Pcm[3]=1,1
	[BoardId=0]
	……
	PcmNumber=2
	PcmSSx[0]=7
	PcmSSx[1]=7
	PcmClockMode[0]=0 //master clock, line synchronization
	PcmClockMode[1]=2 //slave clock
	[BoardId=1]
	……
	PcmNumber=2
	PcmSSx[0]=7
	PcmSSx[1]=7
	PcmClockMode[0]=2 //slave clock
	PcmClockMode[1]=2 //slave clock

4.4.4 High-reliability Application System

The SS7 server developed by Synway supports the warm back-up mode, bringing the carrier-class reliability to the

application system. The figure below illustrates a typical high-reliability application system.



As shown in the figure above, two SS7 servers, each of which has an SHD board installed in it and only uses the signaling time slot on the digital trunk (i.e. does not process voice data), run at one time and share the same point code (OPC=1.2.3), providing signaling services to the voice processing system. One serves as the Master server while the other as the Slave server. Once something wrong happens to the master server, the slave server will automatically take over all signaling services from it.

The rules below shall be followed to run the master and slave servers.

1. Use the first-started SS7 server as the master server and the other as the slave server.
2. While both servers are running well, the slave server will automatically become the master server if the master server fails, and will be the slave server again once the original master server recovers into a normal use.
3. It is allowed to switch manually between the master and slave servers while both servers are running.

The voice processing system consist of one or more computers and the SHD boards installed on them, and acquires signaling services from the server via the local area network (LAN). Therefore, it does not rely on the client computer to run the SS7 server.

- Master SS7 server (referred to as S1 for short) Configuration Files (including Ss7Server.ini and ShConfig.ini)

Ss7Server.ini	ShConfig.ini

[Ss7SystemConfig]
 	
OPC=1.2.3
ServerIP=201.123.123. 1 //the IP address of S1
SecondServerIP=201.123.123.2 //the IP address of S2
……
[Ss7ClientInfo]
 	
MaxSs7Client=3
IP[0]=201.123.123.1 //signaling link processing program
IP[1]=201.123.123.2 //signaling link processing program
IP[2]=201.123.123.3 //the IP address of the client
	……
	[PcmInfo]
	
	TotalPcm=1
	Pcm[0]=0,0
	[BoardId=0]
	
	……
	PcmNumber=1
	PcmSSx[0]=7
	PcmClockMode[0]=0 //master clock, line synchronization
	[SS7]
	
	Ss7ServerIP=201.123.123.1 //the IP address of S1

[Ss7PcmLinkInfo]
 	
MaxSs7Pcm=2
Ss7PcmLink[0]=IP[0],LocalPCM[0]
Ss7PcmLink[1]=IP[1],LocalPCM[0]
 	
[LinkSetInfo]
 	
MaxLinkSet=1
LinkSet[0]=Ss7PcmLink[0]+ Ss7PcmLink[1]
 	
[DPCInfo]
 	
MaxDPC=1
DPC[0]=4.5.6,LinkSet[0]
 	
[TUPRouter] //not to be configured in no use of the TUP protocol
 	
DPC[0]:CIC_PCM[0]=IP[2],LocalPCM[0]
DPC[0]:CIC_PCM[1]=IP[2],LocalPCM[1]
 	
[ISUPRouter] //not to be configured in no use of the ISUP protocol
 	
DPC[0]:CIC_PCM[0]=IP[2],LocalPCM[0]
DPC[0]:CIC_PCM[1]=IP[2],LocalPCM[1]
 	
[Monitor]
 	
ConfigAsGateway=1 //need to load the SynCTI driver
……
	SecondServerIP=201.123.123.2 ///the IP address of S2
	LocalIP= 201.123.123.1 //the IP address of the local PC
	……

➢ Slave SS7 Server (referred to as S2 for short) Configuration Files (including Ss7Server.ini and ShConfig.ini)

Ss7Server.ini	ShConfig.ini

[Ss7SystemConfig]
 	
OPC=1.2.3
ServerIP=201.123.123.2 //the IP address of S2
SecondServerIP=201.123.123.1 //the IP address of S1
……
[Ss7ClientInfo]
MaxSs7Client=3
IP[0]= 201.123.123.1 //signaling link processing program
IP[1]= 201.123.123.2 //signaling link processing program
IP[2]= 201.123.123.3 //the IP address of the client
 	
[Ss7PcmLinkInfo]
 	
MaxSs7Pcm=2
Ss7PcmLink[0]=IP[1],LocalPCM[0]
Ss7PcmLink[1]=IP[0],LocalPCM[0]
 	
[LinkSetInfo]
 	
MaxLinkSet=1
LinkSet[0]=Ss7PcmLink[0]+ Ss7PcmLink[1]
 	
[DPCInfo]
 	
MaxDPC=1
DPC[0]=4.5.6,LinkSet[0]
 	
[TUPRouter] //not to be configured in no use of the TUP protocol
 	
DPC[0]:CIC_PCM[0]=IP[2],LocalPCM[0]
DPC[0]:CIC_PCM[1]=IP[2],LocalPCM[1]
 	
[ISUPRouter] //not to be configured in no use of the
	……
	[PcmInfo]
	
	TotalPcm=1
	Pcm[0]=0,0
	[BoardId=0]
	
	……
	PcmNumber=1
	PcmSSx[0]=7
	PcmClockMode[0]=0 //master clock, line synchronization
	
	[SS7]
	
	Ss7ServerIP=201.123.123.1 //the IP address of S1
	SecondServerIP=201.123.123.2 //the IP address of S2
	LocalIP= 201.123.123.2 //the IP address of the local PC
	……

ISUP protocol
 	
DPC[0]:CIC_PCM[0]=IP[2],LocalPCM[0]
DPC[0]:CIC_PCM[1]=IP[2],LocalPCM[1]
 	
[Monitor]
 	
ConfigAsGateway=1 //need to load the SynCTI driver
……	

- Client Configuration File (ShConfig.ini)

```
……

[PcmInfo]

TotalPcm=1
Pcm[0]=0,0

[BoardId=0]

……
PcmNumber=2
PcmSSx[0]=7
PcmSSx[1]=7
PcmClockMode[0]=0          //master clock, line synchronization
PcmClockMode[1]=2          //slave clock

[SS7]

Ss7ServerIP=201.123.123.1      //the IP address of S1
SecondServerIP=201.123.123.2   //the IP address of S2
LocalIP=201.123.123.3          //the IP address of the local PC
……
```

In Linux, TCP/IP is used by default to transmit the information. If it is required to use SCTP, you shall do the following two operations other than the above configurations.

1. Load the SCTP module of the operating system.

   Comand: modprobe sctp

   In case of a successful loading, type the command ‘lsmod’ and you can see the information about the SCTP module as follows:

   [root@localhost out]# lsmod |grep sctp

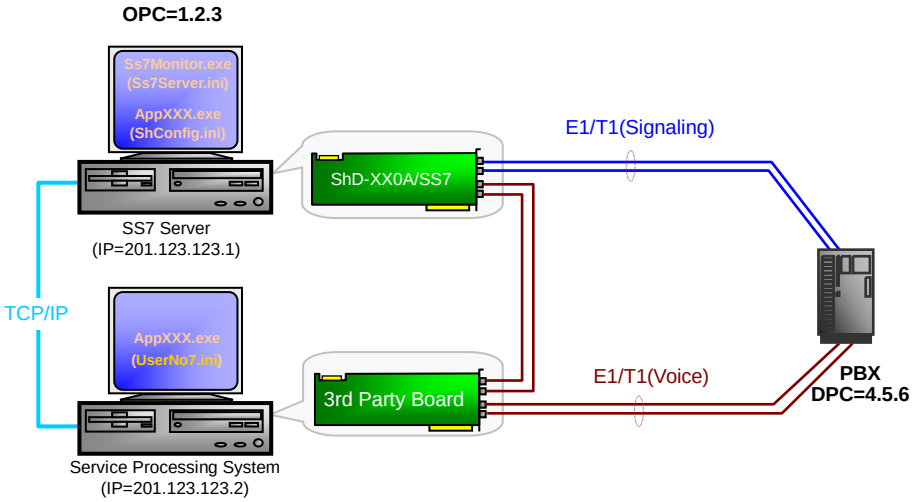
   sctp 153737 1 [unsafe]

2. Modify the configuration file.

   Add UseSctp=1 to the [SS7] section in the ShConfig.ini file

   Add UseSctp=1 to the [Ss7SystemCconfig] section in the Ss7Server.ini file

4.4.5 Supplying SS7 Service for Third-party Board



In the figure above, the SS7 server uses a Synway board to process signaling and the service processing system controls the third-party board. In this way, the signaling messages are acquired from the SS7 server via the TCP/IP protocol. Assuming that the digital trunks connected to the SS7 server have voice time slots, the data on those time slots can be exchanged to another digital trunk interface so as to reach the application system with the help of the Synway board's TDM capability which is enabled via the configuration item LoadShp_a3AsSIU, as illustrated above by dotted lines. Taking the TUP protocol for example, such a system is configured as follows.

- SS7 Server Configuration Files (Ss7Server.ini and ShConfig.ini)

Ss7Server.ini	ShConfig.ini

[Ss7SystemConfig]
OPC=1.2.3
ServerIP=201.123.123.1
[Ss7ClientInfo]
MaxSs7Client=2
IP[0]= 201.123.123.1
IP[1]= 201.123.123.2
[Ss7PcmLinkInfo]
MaxSs7Pcm=1
Ss7PcmLink[0]=IP[0],LocalPCM[0]
[LinkSetInfo]
MaxLinkSet=1
LinkSet[0]=Ss7PcmLink[0]
[DPCInfo]
MaxDPC=1
DPC[0]=4.5.6,LinkSet[0]
[TUPRouter]
DPC[0]:CIC_PCM[0]=IP[1],LocalPCM[0]
[Monitor]
ConfigAsGateway=1 //need to load the SynCTI driver
	……
	[SystemConfig]
	LoadShp_a3AsSIU=1 //specify the time slot to output voice data
	……
	[PcmInfo]
	TotalPcm=1
	Pcm[0]=0,0
	[BoardId=0]
	PcmNumber=1
	PcmSSx[0]=7
	PcmClockMode[0]=0 //master clock, line synchronization

- UserNo7.ini (the configuration file for the application program based on the third-party board which is using the client of the SS7 server SDK)

  [SystemConfig]

TotalPcm=2
DefaultPcmLinkStatus=0x0000
……

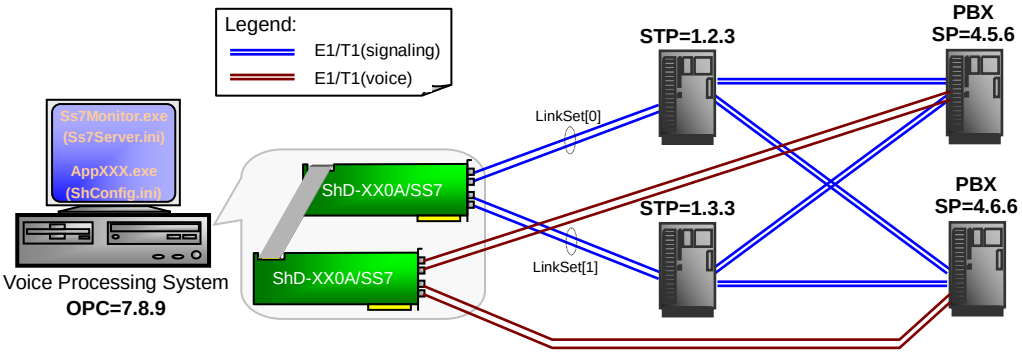
[AppChToPcmTsTable]

TotalAppCh=60
AppCh[0]=0,1..15
AppCh[15]=0,17..31
AppCh[30]=1,1..15
AppCh[45]=1,17..31

[SS7]

Ss7ServerIP=201.123.123.1 //the IP address of the SS7 server
LocalIP=201.123.123.2 //the IP address of the local PC
……

4.4.6 Application System in Quasi-associated Mode



As shown in the figure above, assume that the application system is composed of two SHD-120A/SS7 boards, with the 1st one only processing the signaling on the 4 digital trunks (the logical PCM numbers are respectively 0, 1, 2, 3, and TS16 on each trunk is the signaling time slot) and the 2nd one processing the voice data on the 4 digital trunks. Taking the TUP protocol for example, such a system is configured as follows.

Ss7Server.ini	ShConfig.ini

[Ss7SystemConfig]
 	
OPC=7.8.9
ServerIP=127.0.0.1
[Ss7ClientInfo]
 	
MaxSs7Client=1
IP[0]=127.0.0.1
 	
[Ss7PcmLinkInfo]
 	
MaxSs7Pcm=4
Ss7PcmLink[0]=IP[0],LocalPCM[0]
Ss7PcmLink[1]=IP[0],LocalPCM[1]
Ss7PcmLink[2]=IP[0],LocalPCM[2]
Ss7PcmLink[3]=IP[0],LocalPCM[3]
 	
[LinkSetInfo]
 	
MaxLinkSet=2
 	
LinkSet[0]=Ss7PcmLink[0]+ Ss7PcmLink[1]
LinkSet[1]=Ss7PcmLink[2]+ Ss7PcmLink[3]
 	
[DPCInfo]
 	
MaxDPC=2
DPC[0]=1.2.3,LinkSet[0]
	……
	
	[PcmInfo]
	
	TotalPcm=8
	Pcm[0]=0,0
	Pcm[1]=0,1
	Pcm[2]=0,2
	Pcm[3]=0,3
	Pcm[4]=1,0
	Pcm[5]=1,1
	Pcm[6]=1,2
	Pcm[7]=1,3
	
	[BoardId=0]
	
	PcmNumber=4
	PcmSSx[0]=7
	PcmSSx[1]=7
	PcmSSx[2]=7
	PcmSSx[3]=7
	PcmClockMode[0]=0 //master clock, line synchronization
	PcmClockMode[1]=2 //slave clock
	PcmClockMode[2]=2 //slave clock
	PcmClockMode[3]=2 //slave clock
	Ss7SignalingTS[0]=16 //TS16 used as the signaling time slot
	UseTS16AsCircuit[0]=0

DPC[1]=1.3.3,LinkSet[1]

[UP_DPCInfo]

MaxUP_DPC=2
UP_DPC[0]=4.5.6,LinkSet[0]+LinkSet[1] //load sharing
UP_DPC[1]=4.6.6,LinkSet[1]+LinkSet[0],1 //Signaling Link Set 1 serves as the normal route while Signaling Link Set 0 as the alternate route.

[TUPRouter]

UP_DPC[0]:CIC_PCM[1]=IP[0],LocalPCM[4]
UP_DPC[0]:CIC_PCM[2]=IP[0],LocalPCM[5]
UP_DPC[1]:CIC_PCM[1]=IP[0],LocalPCM[6]
UP_DPC[1]:CIC_PCM[2]=IP[0],LocalPCM[7]
……

Ss7SignalingTS[1]=16 //TS16 used as the signaling time slot
UseTS16AsCircuit[1]=0
Ss7SignalingTS[2]=16 //TS16 used as the signaling time slot
UseTS16AsCircuit[2]=0
Ss7SignalingTS[3]=16 //TS16 used as the signaling time slot
UseTS16AsCircuit[3]=0
……

[BoardId=1]

PcmNumber=4
PcmSSx[0]=7
PcmSSx[1]=7
PcmSSx[2]=7
PcmSSx[3]=7
PcmClockMode[0]=2 //slave clock
PcmClockMode[1]=2 //slave clock
PcmClockMode[2]=2 //slave clock
PcmClockMode[3]=2 //slave clock
……

Appendix 1 ISDN Release Cause Information Element

Return Value	Cause	Application
1	Unallocated (unassigned) number	DSS1, ISUP
2	No route to specified transit network	DSS1, ISUP
3	No route to destination	DSS1, ISUP
4	Send special information tone	ISUP
5	Misdialled trunk prefix	ISUP
6	Channel unacceptable	DSS1, ISUP
7	Call awarded and being delivered in an established channel	DSS1, ISUP
8	Preemption	DSS1, ISUP
9	Preemption-circuit reserved for reuse	ISUP
16	Normal call clearing	DSS1, ISUP
17	User busy	DSS1, ISUP
18	No user responding	DSS1, ISUP
19	No answer from user (user alerted)	DSS1, ISUP
20	Subscriber absent	DSS1, ISUP
21	Call rejected	DSS1, ISUP
22	Number changed	DSS1, ISUP
26	Non-selected user clearing	DSS1, ISUP
27	Destination out of order	DSS1, ISUP
28	Invalid number format (address incomplete)	DSS1, ISUP
29	Facility rejected	DSS1, ISUP
30	Response to STATUS ENQUIRY	DSS1
31	Normal, unspecified	DSS1, ISUP
34	No circuit/channel available	DSS1, ISUP
38	Network out of order	DSS1, ISUP
39	Permanent frame mode connection out of service	DSS1
40	Permanent frame mode connection operational	DSS1
41	Temporary failure	DSS1, ISUP
42	Switching equipment congestion	DSS1, ISUP
43	Access information discarded	DSS1, ISUP
44	Requested circuit/channel not available	DSS1, ISUP
46	Precedence call blocked	DSS1, ISUP
47	Resource unavailable, unspecified	DSS1, ISUP
49	Quality of service unavailable	DSS1
50	Requested facility not subscribed	DSS1, ISUP
53	Outgoing calls barred within CUG	DSS1, ISUP

55	Incoming calls barred within CUG	DSS1, ISUP
57	Bearer capability not authorized	DSS1, ISUP
58	Bearer capability not presently available	DSS1, ISUP
62	Inconsistency in designated outgoing access information and subscriber class	DSS1, ISUP
63	Service or option not available, unspecified	DSS1, ISUP
65	Bearer capability not implemented	DSS1, ISUP
66	Channel type not implemented	DSS1
69	Requested facility not implemented	DSS1, ISUP
70	Only restricted digital information bearer capability is available	DSS1, ISUP
79	Service or option not implemented, unspecified	DSS1, ISUP
81	Invalid call reference value	DSS1
82	Identified channel does not exist	DSS1
83	A suspended call exists, but this call identity does not	DSS1
84	Call identity in use	DSS1
85	No call suspended	DSS1
86	Call having the requested call identity has been cleared	DSS1
87	User not member of CUG	DSS1, ISUP
88	Incompatible destination	DSS1, ISUP
90	Non-existent CUG	DSS1, ISUP
91	Invalid transit network selection	DSS1, ISUP
95	Invalid message, unspecified	DSS1, ISUP
96	Mandatory information element is missing	DSS1
97	Message type non-existent or not implemented	DSS1, ISUP
98	Message not compatible with call state or message type non-existent or not implemented	DSS1
99	Information element/parameter non-existent or not implemented	DSS1, ISUP
100	Invalid information element contents	DSS1
101	Message not compatible with call state	DSS1
102	Recovery on timer expiry	DSS1, ISUP
103	Parameter non-existent or not implemented, passed	ISUP
110	Message with unrecognized parameter, discarded	ISUP
111	Protocol error, unspecified	DSS1, ISUP
127	Interworking, unspecified	DSS1, ISUP
128	T303 time out	Synway Define

129	T304 time out	Synway Define
130	T310 time out	Synway Define
131	T308 time out again	Synway Define
132	Network busy	Synway Define
133	Circuit restarted	Synway Define
134	Temporary fault	Synway Define
135	Data link failure	Synway Define
136	Connection after pickup failed	Synway Define
137	Pickup time out	Synway Define

Appendix 2 Optional ISUP Parameters and Descriptions

<table>
<tr><th>Parameter Name</th><th>Code</th><th>Description</th></tr>
<tr><td>Called Party Number</td><td>0x04</td><td>
<table>
<tr><td></td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td></tr>
<tr><td>1</td><td colspan="2">Odd/Even</td><td colspan="6">Nature of Address Indicator</td></tr>
<tr><td>2</td><td colspan="2">INN Indicator</td><td colspan="2">Numbering Plan</td><td colspan="5">Spare</td></tr>
<tr><td>3</td><td colspan="4">2nd Address Signal</td><td colspan="5">1st Address Signal</td></tr>
<tr><td>…</td><td colspan="8">…</td></tr>
<tr><td>n</td><td colspan="4">Filler (If necessary)</td><td colspan="5">nth Address Signal</td></tr>
</table>
1) Odd/Even Indicator
0: Even number of address signals
1: Odd number of address signals
2) Nature of Address Indicator
0000001: Subscriber number
0000011: National number
0000100: International number
3) Internal Network Number Indicator (INN Indicator)
0: Routing to internal network number allowed
1: Routing to internal network number not allowed
4) Numbering Plan Indicator
001: ISDN (Telephony) numbering plan (Recommendations E.164, E.163)
011: Data numbering plan (Recommendation X.164)
100: Telex numbering plan (Recommendation F.164)
5) Address Signal
From the third byte begins the called party number.
6) Filler
In case of an odd number of address signals, the filler code 0000 is inserted after the last address signal.
</td></tr>
</table>

<table>
<tr><td>Calling Party Number</td><td>0x0a</td><td>
<table>
<tr><td></td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td></tr>
<tr><td>1</td><td>Odd/Even</td><td colspan="7">Nature of Address Indicator</td></tr>
<tr><td>2</td><td>Number Incomplete</td><td colspan="2">Numbering Plan Indicator</td><td colspan="2">Presentation Indicator</td><td colspan="2">Screening indicator</td></tr>
<tr><td>3</td><td colspan="3">2nd Address Signal</td><td colspan="5">1st Address Signal</td></tr>
<tr><td>…</td><td colspan="8">…</td></tr>
<tr><td>n</td><td colspan="5">Filler (If necessary)</td><td colspan="3">nth Address Signal</td></tr>
</table>
1) Odd/Even Indicator
0: Even number of address signals
1: Odd number of address signals
2) Nature of Address Indicator
0000001: Subscriber number
0000011: National number
0000100: International number
3) Number Incomplete Indicator (NI)
0: Complete
1: Incomplete
4) Numbering Plan Indicator
001: ISDN (Telephony) numbering plan (Recommendations E.164, E.163)
011: Data numbering plan (Recommendation X.164)
100: Telex numbering plan (Recommendation F.164)
5) Address Presentation Restricted Indicator
00: Presentation allowed
01: Presentation restricted
10: Address not available
6) Screening Indicator
00: User provided, not verified
01: User provided, verified and passed
10: User provided, verified but not passed
11: Network provided
7) Address Signal
From the third byte begins the calling party number.
8) Filler
In case of an odd number of address signals, the filler code 0000 is inserted after the last address signal.
</td></tr>
</table>

<table>
<tr>
<td>Backward Call Indicators (BCI)</td>
<td>0x11</td>
<td>
<table>
<tr><td></td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td></tr>
<tr><td>1</td><td>H</td><td>G</td><td>F</td><td>E</td><td>D</td><td>C</td><td>B</td><td>A</td></tr>
<tr><td>2</td><td>P</td><td>O</td><td>N</td><td>M</td><td>L</td><td>K</td><td>J</td><td>I</td></tr>
</table>
1) BA represents Charge Indicator
00: No indication
01: No Charge
10: Charge
2) DC represents Called Party’s Status Indicator
00: No Indication
01: Subscriber free
10: Connect when free
3) FE represents Called Party’s Category Indicator
00: No Indication
01: Ordinary subscriber
10: Payphone
4) HG represents End-to-end Method Indicator
00: No end-to-end method available (only link-by-link method available)
01: Pass-along method available
10: SCCP method available
11: Pass-along and SCCP methods available
5) I represents Interworking Indicator
0: No interworking encountered
1: Interworking encountered
6) J represents End-to-end Information Indicator
0: No end-to-end information available
1: End-to-end information available
7) K represents ISDN User Part Indicator
0: ISDN user part not used all the way
1: ISDN user part used all the way
8) L represents Holding Indicator
0: Holding not requested
1: Holding requested
9) M represents ISDN Access Indicator
0: Terminating access non-ISDN
1: Terminating access ISDN
10) N represents Echo Control Device Indicator
0: Incoming half-echo control device not included
1: Incoming half-echo control device included
11) PO represents SCCP Method Indicator
00: No indication
01: Connectionless method available
10: Connection oriented method available
11: Connectionless and connection oriented methods available
</td>
</tr>
</table>

<table>
<tr>
<td>Call Diversion Information</td>
<td>0x36</td>
<td>
<table>
<tr><td></td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td></tr>
<tr><td>1</td><td>H</td><td>G</td><td>F</td><td>E</td><td>D</td><td>C</td><td>B</td><td>A</td></tr>
</table>
1) CBA represents Notification Subscription Options.
000: Unknown
001: Presentation not allowed
010: Presentation allowed with redirection number
011: Presentation allowed without redirection number
2) GFED represents Redirecting Reason
0000: Unknown
0001: User busy
0010: No reply
0011: Unconditional
0100: Deflection during alerting
0101: Deflection immediate response
0110: Mobile subscriber not reachable
</td>
</tr>
<tr>
<td>Calling Party’s Category (CAT)</td>
<td>0x09</td>
<td>
<table>
<tr><td></td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td></tr>
<tr><td>1</td><td colspan="8">Calling Party’s Category</td></tr>
</table>
00001001: National operator (for domestic use)
00001010: Ordinary calling subscriber, used for toll to toll or toll to local office calls
00001011: Calling subscriber with priority
00001100: Data call (voice band data)
00001101: Test call
00001111: Payphone
11111000: Ordinary calling subscriber, used for local office to local office calls
</td>
</tr>
<tr>
<td>Cause Indicators</td>
<td>0x12</td>
<td>
<table>
<tr><td></td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td></tr>
<tr><td>1</td><td>Extension Indicator</td><td colspan="2">Coding Standard</td><td>Spare</td><td colspan="4">Location</td></tr>
<tr><td>2</td><td>Extension Indicator</td><td colspan="7">Cause Value</td></tr>
<tr><td>3</td><td colspan="8" rowspan="3">Diagnostic(s) (if any)</td></tr>
<tr><td>…</td></tr>
<tr><td>n</td></tr>
</table>
1) Extension Indicator
0: Octet continues through the next octet
1: Last octet
2) Coding Standard
00: CCITT standard coding
01: ISO/IEC standard
10: National standard
11: Standard specific to identified location
3) Location
</td>
</tr>
</table>

		0000: User 0001: Private network serving the local user 0010: Public network serving the local user 0011: Transit network 0100: Public network serving the remote user 0101: Private network serving the remote user 0111: International network 1010: Network beyond interworking point 4) Cause Value (Partial) 000 0001: Unallocated number 000 0010: No route to specified transit network 000 0011: No route to destination 000 0100: Send special information tone 000 0101: Misdialed trunk prefix 001 0000: Normal call clearing 001 0001: User busy 001 0010: No user responding 001 0011: No answer from user 001 0101: Call rejected 001 0110: Number changed 001 1011: Destination out of order 001 1100: Invalid number format 001 1101: Facility rejected 001 1111: Normal 010 0010: No circuit available 010 0110: Network out of order 010 1001: Temporary Failure 010 1010: Switching equipment congestion 010 1100: Requested circuit not available 010 1111: Resource unavailable 101 1000: Incompatible destination

<table>
<tr>
<td>Echo Control Information</td>
<td>0x37</td>
<td>
<table>
<tr><td></td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td></tr>
<tr><td>1</td><td>H</td><td>G</td><td>F</td><td>E</td><td>D</td><td>C</td><td>B</td><td>A</td></tr>
</table>
1) BA represents Outgoing Half-echo Control Device Information Indicator
00: No information
01: Outgoing half-echo control device not included
10: Outgoing half-echo control device included
2) DC represents Incoming Half-echo Control Device Information Indicator
00: No information
01: Incoming half-echo control device not included
10: Incoming half-echo control device included
3) FE represents Outgoing Half-echo Control Device Request Indicator
00: No information
01: Outgoing half-echo control device activation request
10: Outgoing half-echo control device deactivation request
4) HG represents Incoming Half-echo Control Device Request Indicator
00: No information
01: Incoming half-echo control device activation request
10: Incoming half-echo control device deactivation request
</td>
</tr>
</table>

<table>
<tr>
<td>Forward Call Indicators</td>
<td>0x07</td>
<td>
<table>
<tr><td></td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td></tr>
<tr><td>1</td><td>H</td><td>G</td><td>F</td><td>E</td><td>D</td><td>C</td><td>B</td><td>A</td></tr>
<tr><td>2</td><td>P</td><td>O</td><td>N</td><td>M</td><td>L</td><td>K</td><td>J</td><td>I</td></tr>
</table>
1) A represents National/International Call Indicator
0: Call to be treated as a national call
1: Call to be treated as an international call
2) CB represents End-to end Method Indicator
00: No end-to end method available
01: Pass-along method available
10: SCCP method available
11: Pass-along and SCCP methods available
3) D represents Interworking Indicator
0: No interworking encountered
1: Interworking encountered
4) E represents End-to-end Information Indicator
0: No end-to-end information available
1: End-to-end information available
5) F represents ISDN User Part Indicator
0: ISDN user part not used all the way
1: ISDN user part used all the way
6) HG represents ISDN User Part Preference Indicator
00: ISDN user part preferred all the way
01: ISDN user part not required all the way
10: ISDN user part required all the way
7) I represents ISDN Access Indicator
0: Originating access non-ISDN
1: Originating access ISDN
8) KJ represents SCCP Method Indicator
00: No indication
01: Connectionless method available
10: Connection oriented method available
11: Connectionless and connection oriented methods available
</td>
</tr>
</table>

<table>
<tr>
<td>Generic Notification Indicator</td>
<td>0x2c</td>
<td>
<table>
<tr><td></td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td></tr>
<tr><td>1</td><td>Extension</td><td colspan="7">Notification Indicator</td></tr>
</table>
1) Extension Indicator
0: Information continues in the next octet
1: Last octet
2) Notification Indicator
0000000: User suspended
0000001: User resumed
0000010: Bearer service change
0000100: Call completion delay
1000010: Conference established
1000011: Conference disconnected
1000100: Other party added
1000101: Isolated
1000110: Reattached
1001000: Other part isolated
1001001: Other part split
1001011: Conference floating
1100000: Call is a waiting call
1101000: Diversion activated
1101001: Call transfer, alerting
1101010: Call transfer, active
1111001: Remote hold
1111010: Remote retrieval
1111011: Call is diverting
</td>
</tr>
</table>

<table>
<tr>
<td>Location Number</td>
<td>0x3f</td>
<td>
<table>
<tr><td></td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td></tr>
<tr><td>1</td><td>Odd/Even</td><td colspan="7">Nature of Address Indicator</td></tr>
<tr><td>2</td><td>INN/NI Indicator</td><td colspan="2">Numbering Plan Indicator</td><td colspan="2">Presentation Indicator</td><td colspan="2">Screening Indicator</td></tr>
<tr><td>3</td><td colspan="4">2nd Address Signal</td><td colspan="4">1st Address Signal</td></tr>
<tr><td>…</td><td colspan="8">…</td></tr>
<tr><td>n</td><td colspan="4">Filler (If necessary)</td><td colspan="4">nth Address Signal</td></tr>
</table>
1) Odd/Even Indicator
0: Even number of address signals
1: Odd number of address signals
2) Nature of Address Indicator
0000011: National number
0000100: International number
3) Internal Network Number Indicator (INN)
0: Routing to internal number allowed
1: Routing to internal number not allowed
4) Numbering Plan Indicator
001: ISDN numbering plan (Recommendation E.164)
011: Data numbering plan (Recommendation X.121)
100: Telex numbering plan (Recommendation F.69)
101: Private numbering plan
5) Presentation Indicator
00: Presentation allowed
01: Presentation restricted
10: Address not available
6) Screening Indicator
01: User provided, verified and passed
11: Network provided
7) Address Signal
From the third byte begins the location number.
8) Filler
In case of an odd number of address signals, the filler code 0000 is inserted after the last address signal.
</td>
</tr>
</table>

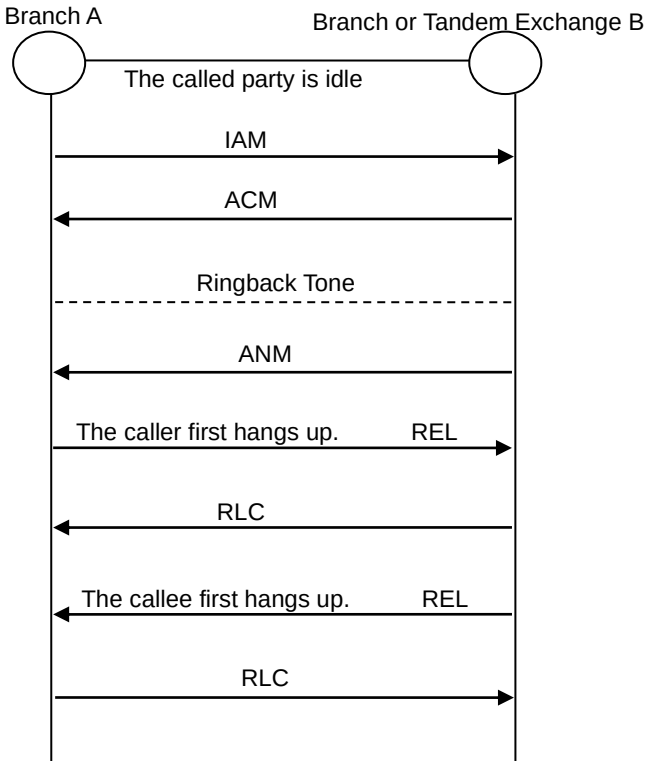
<table>
<tr><td>Nature of Connection Indicators</td><td>0x06</td><td>
<table>
<tr><td></td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td></tr>
<tr><td>1</td><td>H</td><td>G</td><td>F</td><td>E</td><td>D</td><td>C</td><td>B</td><td>A</td></tr>
</table>
1) BA represents Satellite Indicator
00: No satellite circuit in the connection
01: One satellite circuit in the connection
10: Two satellite circuits in the connection
2) DC represents Continuity Check Indicator
00: Continuity check not required
01: Continuity check required on this circuit
10: Continuity check performed on a previous circuit
3) E represents Echo Control Device Indicator
0: Outgoing half-echo control device not included
1: Outgoing half-echo control device included
</td></tr>
<tr><td>Optional Backward Call Indicators</td><td>0x29</td><td>
<table>
<tr><td></td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td></tr>
<tr><td>1</td><td>H</td><td>G</td><td>F</td><td>E</td><td>D</td><td>C</td><td>B</td><td>A</td></tr>
</table>
1) A represents In-band Information Indicator
0: No indication
1: In-band information or appropriate pattern is now available
2) B represents Call Diversion May Occur Indicator
0: No indication
1: Call diversion may occur
3) C represents Simple Segmentation Indicator
0: No additional information will be sent
1: Additional information will be sent in a segmentation message
4) D represents MLPP User Indicator
0: No indication
1: MLPP user
</td></tr>
<tr><td>Optional Forward Call Indicators</td><td>0x08</td><td>
<table>
<tr><td></td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td></tr>
<tr><td>1</td><td>H</td><td>G</td><td>F</td><td>E</td><td>D</td><td>C</td><td>B</td><td>A</td></tr>
</table>
1) BA represents Closed User Group Call Indicator
00: Non-CUG call
10: Closed user group call, outgoing access allowed
11: Closed user group call, outgoing access not allowed
2) C represents Simple Segmentation Indicator
0: No additional information will be sent
1: Additional information will be sent in a segmentation message
3) H represents Connected Line Identity Request Indicator
0: Not Requested
1: Requested
</td></tr>
</table>

<table>
<tr><td>Transit Network Selection</td><td>0x23</td><td>
<table>
<tr><td></td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td></tr>
<tr><td>1</td><td>Odd/Even</td><td colspan="3">Type of Network Identification</td><td colspan="4">Network Identification Plan</td></tr>
<tr><td>2
…
n</td><td colspan="8">Network Identification</td></tr>
</table>
1) Odd/Even Indicator
0: Even number of digits
1: Odd number of digits
2) Type of Network Identification
000: CCITT-standardized identification
010: National network identification
3) Network Identification Plan (For CCITT-standardized identification)
0000: Unknown
0011: Public data network identification code
0110: Public land mobile network identification
</td></tr>
<tr><td>Transmission Medium Requirement</td><td>0x02</td><td>00000000: Speech
00000001: Spare
00000010: 64 kb/s unrestricted
00000011: 3.1KHZ audio
00000100: Alternative: speech (service 2)/ 64kbit/s unrestricted (service 1)
00000101: Alternative: 64kbit/s unrestricted (service 1)/ speech (service 2)
00000110: 64kb/s preferred
00000111: 2*64kb/s unrestricted
00001000: 384 kb/s unrestricted
00001001: Spare
00001010: 1920 kb/s unrestricted</td></tr>
<tr><td>Access Transport</td><td>0x03</td><td></td></tr>
<tr><td>Call History Information</td><td>0x2d</td><td></td></tr>
<tr><td>Circuit Group Supervision Message Type</td><td>0x15</td><td></td></tr>
<tr><td>Circuit State Indicator</td><td>0x26</td><td></td></tr>
<tr><td>Closed User Group Interlock Code</td><td>0x1a</td><td></td></tr>
<tr><td>Connected Number</td><td>0x21</td><td></td></tr>
<tr><td>Continuity Indicators</td><td>0x10</td><td></td></tr>
<tr><td>End of Optional Parameters</td><td>0x00</td><td></td></tr>
<tr><td>Event Information</td><td>0x24</td><td></td></tr>
<tr><td>Facility Indicator</td><td>0x18</td><td></td></tr>
<tr><td>Generic Digits</td><td>0xc1</td><td></td></tr>
<tr><td>Generic Number</td><td>0xc0</td><td></td></tr>
<tr><td>Information Indicators</td><td>0x0f</td><td></td></tr>
</table>

Information Request Indicators	0x0e	
MCID Request Indicators	0x3b	
MCID Response Indicators	0x3c	
Message Compatibility Information	0x38	
Network Specific Facility	0x2f	
Original Called Number	0x28	
Original ISC Point Code	0x2b	
Parameter Compatibility Information	0x39	
Propagation Delay Counter	0x31	
Range and Status	0x16	
Redirecting Number	0x0b	
Redirection Information	0x13	
Redirection Number	0x0c	
Redirection Number Restriction	0x40	
Subsequent Number	0x05	
Suspend/Resume Indicators	0x22	
Signaling Point Code	0x1e	
Transmission Medium Used	0x35	
User Service Information	0x1d	
User Teleservice Information	0x34	
User-to-user Indicators	0x2a	
User-to-user Information	0x20	
Charge Information	0xfe	
Note: For more information, refer to ITU-T Q.763, Signaling System No.7- ISDN User Part Formats and Codes		

Below are two typical scenarios of ISUP signaling connection

1. The called party is idle when you are calling.



2. The called party is busy when you are calling.