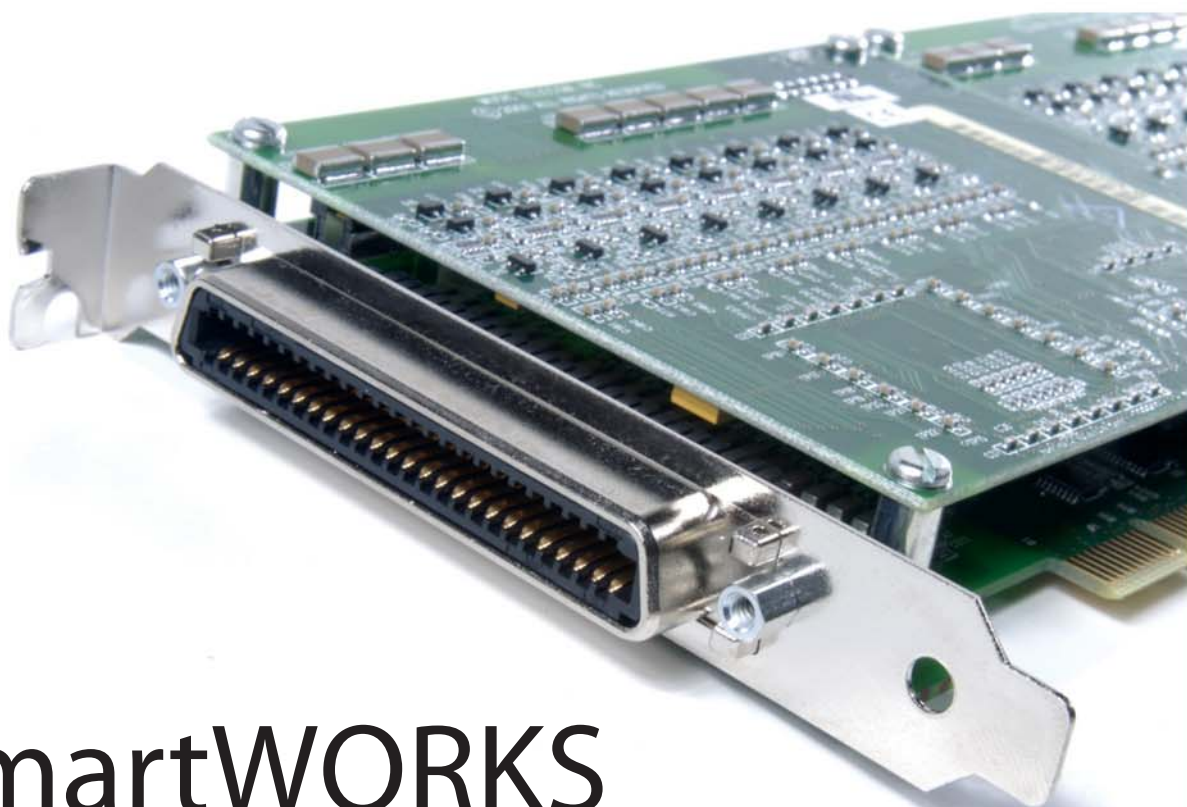




SmartWORKS Function Reference Library v. 5.2.0

AudioCodes USA
www.audiocodes.com/blades
27 World's Fair Drive, NJ · 08873
T: 732-469-0880 · F: 732-469-2298

404-0001-007 · Build 090828.01 REVB



Table Of Contents

Welcome	1
Legal Notice	2
Purpose of this Documentation	2
Release Update History	4
Document Version Control	7
Contacting Audio Codes	8
Technical Support	8
Sales and General Information	8
Mailing Address—USA	8
Function Reference Library	9
API's Presented Alphabetically with Product Support	10
APIs Presented Logically	23
Misc. Functions	24
Error & Get Description Functions	25
Network Interface Alarms & Statistics	30
Framer Mode and ABCD Signaling Functions	48
Audio Jack Functions	61
Event Reporting Functions	72
Channel Event Control	73
Board Event Control	90
System Event Control	97
System Functions	101
System Start Up Functions	102
System Configuration Functions	105
System Information Functions	109
Sync Host/Board Time	117
System Functions - Obsolete	119
Board Control and Configuration Functions	120
Board Control	121
Board Information Functions	125
Board Identification & OEM Functions	133
Board Configuration	141
Board Clock Source Functions	149
Board Firmware Functions	162
Board Functions - Obsolete APIs	168
Channel Control and Information Functions	179
Opening and Closing Channels	180
Controlling Background Functions	185
Channel Numbering (GCI) Functions	191
Channel Information Functions	194
Channel Statistics Functions	204
Terminate cards (T1 RBS, ISDN) functions	210
Terminate cards (ISDN) Functions	217
Passive cards (ISDN) Functions	222
Channel Functions	233
Gain/Volume Functions	266
HIGHPASS Filter Functions	293
Mux/MVIP bus Routines	304
Play/Record Functions	310
Record Functions	320

Table of Contents

Function Reference Library · II



Stream Functions	326
Structure Functions	340
Activity Detect (ACTD) Control	365
Cpm Functions	387
Tone Functions - Alert Tones	423
Tone Functions - Play Tones	430
Termination/Monitor Functions	434
CT Bus Functions	443
Loop Voltage / Loop Current / Ring Detection Status Functions	467
Summation Functions	477
Trunk Specific APIs	497
Wave Header Utility	502
SmartWORKS PCM Specific APIs	508
IPX (VoIP) Functions.	520
Configuration APIs	521
Station Manager APIs	533
Network Statistics	546
Target Recording APIs	550
None	552
None	553
None	554
None	555
Media Session APIs	556
Media Forwarding APIs	562
Port Status and Statistics APIs	572

Event Code Library 579

Event Codes 580

D-Channel Event Codes and PBX support (NGX Only) 587

MT_EVENT 587

Event Codes Presented Alphabetically 593

Appendix A i

MT_IO_CONTROL i

The MT_IO_CONTROL Data Structure ii

General Termination Control iii

Signal Termination Control iii

Digit Termination Control vii

Miscellaneous Control Data viii

Play/Record Specific Data ix

Data Portion ix

Chapter 1

Welcome

Legal Notice

No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written consent of Audio Codes USA, Inc.

Copyright © 2000 - 2008 Audio Codes USA, Inc. All rights reserved.

Audio Codes, and the Audio Codes logo are trademarks or registered trademarks of Audio Codes, Inc.

Microsoft Windows is a registered trademarks of Microsoft Corporation.

All other trademarks or registered trademarks are the property of their respective companies.

Audio Codes reserves the right to make changes to its products and specifications at any time in order to improve on performance, manufacturing, or reliability. Information furnished by Audio Codes is believed to be accurate. No responsibility is assumed by Audio Codes for the use of said information, nor for any infringement of patents or of other third party rights that may result from said use. No license is granted by implication or otherwise under any patent or patent rights of any Ai-Technology Group.

Purpose of this Documentation

All information in this document was previously found in the SmartWORKS Developer's Guide. The SmartWORKS Function Reference Library contains all API prototype examples including data structures, plus all event codes.

An introduction to the SmartWORKS SDK, all developer's notes along with theoretical or logical discussions of APIs are available in the SmartWORKS Developer's Guide.

This document applies to the following Audio Codes products:

NOTE: Lead free boards are referenced by weight.

Product Name	Part Number	Weight	Status
SmartWORKS VR3209	910-0303-002		Retired
SmartWORKS VR6409	910-0321-001		Retired
SmartWORKS DP3209	910-0308-002	255 g	Released
SmartWORKS DP6409	910-0324-001	280 g	Released
SmartWORKS DP3209-eh	910-0703-001	245 g	Released (special order only)
SmartWORKS DP6409-eh	910-0703-002	270 g	Released (special order only)
SmartWORKS NGX800	910-0314-001	204g	Released
SmartWORKS NGX1600	910-0314-002	286 g	Released
SmartWORKS NGX2400	910-0314-003	366 g	Released

Product Name	Part Number	Weight	Status
SmartWORKS MX80 (Expansion)	910-0315-001	68 g	Released
SmartWORKS MX80A (Expansion)	910-1315-001	60 g	Released
SmartWORKS NGX800-eh	910-0700-001	208 g	Released (special order only)
SmartWORKS NGX1600-eh	910-0700-002	269 g	Released (special order only)
SmartWORKS NGX2400-eh	910-0700-003	208 g	Released (special order only)
SmartWORKS LD 101	910-0805-001	130 g	Retired
SmartWORKS LD 409	910-0801-001	165 g	Released
SmartWORKS LD 409H	910-0807-001		Released
SmartWORKS LD 809	910-0802-001	280 g	Released
SmartWORKS LD 809X	910-0808-001	385 g	Released
SmartWORKS LD 1609	910-0803-001	490 g	Released
SmartWORKS LD 2409	910-0804-001	605 g	Released
SmartWORKS LD 809-eh	910-0701-001	355 g	Released (special order only)
SmartWORKS LD 1609-eh	910-0701-002	460 g	Released (special order only)
SmartWORKS LD 2409-eh	910-0701-003	575 g	Released (special order only)
SmartWORKS DT6409TE	910-0323-002	275 g	Released
SmartWORKS DT3209TE	910-0325-002		Released
SmartWORKS DT3209TE-eh	910-0704-001	240 g	Released (special order only)
SmartWORKS DT6409TE-eh	910-0704-002	265 g	Released (special order only)
SmartWORKS PCM 3209	910-0330-001		Retired
SmartWORKS PCM 6409	910-0329-001	270 g	Released (special order only)
SmartWORKS PCM 3209-eh	910-0702-001	265 g	Released (special order only)
SmartWORKS PCM 6409-eh	910-0702-002	240 g	Released (special order only)
SmartWORKS IPX-C	910-0331-007	250 g	Released

NOTE: Retired boards can no longer be purchased, but are still supported by the SmartWORKS software.

This documentation is intended for the developer of CTI application software. This manual assumes the reader is fairly proficient in standard C++ programming, computer telephony and voice processing.

Release Update History

SMARTWORKS 3.11

IPX PBX Integrations: NEAX 2400 and Nortel Call Control support, H,,323, NGX PBX Integrations: Mitel ICP 3300 and Harris 2020, New Features: Japanese Caller ID (LD only), TCP re-ordering (IPX), Link Status and Network statistics (IPX)

SMARTWORKS 3.10

IPX PBX Integrations: NEAX 2400 Dchannel support, NGX PBX Integrations: Ascom Ascotel (Vox & D-Channel) support, Siemens AC WIN D-Channel. New Features: pciExpress boards, Caller ID per NTT Telephone Interface Service Edition 5, MAC address has been added as station information (IPX only).

SMARTWORKS 3.9

IPX PBX Integrations: Siemens HiPath 4000, Alcatel OMNI PCX Enterprise 6.0, SIP. NGX PBX Integrations: Rockwell Spectrum, Tadiran Coral, Aastra. New Features: Enhanced IPX/IPX-C performance, required license with IPX-C, Energy Tagging, one codeset 5 information element when tapping ISDN, multi-point BRI support with the NGX.

SMARTWORKS 3.8

New board, IPX - C. Added DPNSS support on the SmartWORKS DP. Improvements to MSI install. PBX Integrations - NGX: New Panasonic Multi-point phone support, Alcatel OmniPCX beta DChannel support, Siemens Realitis iSDT(2W) Beta Vox support, Siemens Realitis DTI(4W) Beta Vox support. New IPX Integrations: Beta SIP support.

SMARTWORKS 3.7

New PBX integrations with the SmartWORKS IPX: Avaya (Call Control), Ericsson (Call Control), Nortel (DChannel). New APIs used with the IPX:

MTIpDChannelEventFilteringControl() and **MTIpDChannelEventFilteringStatus()**.

The functionality of the SmartWORKS IPX has been improved, refer to the release notes. Documentation changes: New book - *SmartWORKS Function Reference Library*.

SMARTWORKS 3.6

Beta 2 support for the IPX board. New API, **MTBoardGetCustomSwitchSetting()**. The following support has been added on the SmartWORKS NGX: Beta support of Intertel (Vox and Dchannel), Philips Sophos is supported with call control events similar to NGX BRI event reporting, Nortel Meridian; 3820 & 3310 series phones are now supported (Vox and Dchannel), Panasonic TDA 50; multi-point support has been added with KXT-7600 series phones, NEC NEAX2400; F revision line card support has been added with NEC I series phones, Astraa (EADS) Matra; MC420E series phones are now supported (Vox and Dchannel), eOn eQueue.

SMARTWORKS 3.5

Added support for SmartWORKS PCM and LD809X boards. Beta support for new PBXs: Panasonic (Vox and Dchannel), EADS 4-wire support, and a new line card is supported with the Ericsson MD110.

SMARTWORKS 3.4

D-channel support has been added for the following PBXs - LG Starex, Toshiba Strata Dk, Toshiba Strata CTX

SMARTWORKS 3.3

Enhanced CPM, improved buffer handling with the MT_EVENT structure, Voice/ Answering Machine detection, G.726 MSB first.

SMARTWORKS 3.2

Secondary input control for Activity detection and DTMF tone detection, Plug and Play, board clock synchronization, Flash firmware using APIs, and MSI support. Beta release of LD409H and the LD2409.

SMARTWORKS 3.0

Windows NT is no longer supported. Windows 2003 Server 32 Bit support has been added.

SMARTWORKS 2.10.0

SmartTERM DT6409TE, and DT 3209TE plus SmartTAP LD809 introduced to product line. Features added: multi-processor support, board identification, a signal profiling utility - SmartWORKS Profiler. See Release Notes for more information.

[SMARTWORKS 2.9.0](#)

Beta Release of LD 409. Support added for media format G.723.1. Support for terminate ISDN (SmartTerm DT). See Release Notes for more information.

[SMARTWORKS 2.7.0](#)

Minor changes. See release notes.

[SMARTWORKS 2.6.0](#)

Added D-Channel support for SmartTAP NGX

[SMARTWORKS 2.4.2](#)

2nd Beta release of SmartTAP NGX

[SMARTWORKS 2.4.0](#)

Beta release of SmartTAP NGX. G.729A support added (VR6409, PT1609).

[SMARTWORKS 2.3.7](#)

Alpha and Early Adopter release of SmartTAP NGX

[SMARTWORKS 2.3.6](#)

Early Adopter release of NFAS functionality

[SMARTWORKS 2.3.5](#)

Beta release of Windows 2000 WDM driver

Early Adopter release of H.100 functionality

[SMARTWORKS 2.3.4](#)

Beta release of SmartTERM AT and DT

[SMARTWORKS 2.3.2](#)

Release of SmartTAP PT series

[SMARTWORKS 2.3.0](#)

Beta release of SmartTAP PT series

[SMARTWORKS 2.2.0](#)

Release of SmartTAP DP series

[SMARTWORKS 2.0.0](#)

Release of SmartDSP VR series

Document Version Control

The following has been added to this document since the last release:

TABLE 1: VERSION CONTROL

Page	Description
REV A	
events chapter	Added EVT_DONGLE_ADDED, EVT_DONGLE_REMOVED and updated EVT_QUOTA_EXCEEDED
REV B	
	Clarified <i>MTSetGain()</i> and <i>MTSetAGC()</i> by stating these functions only apply to primary side if mixing is disabled.

Contacting Audio Codes

Your feedback is important to maintain and improve the quality of our products. Use the information below to request technical assistance, make general inquiries, or to provide comments.

TECHNICAL SUPPORT

For programming, installation, or configuration assistance, use the following contact methods:

- Call technical support at 732.469.0880 or call toll free in the USA at 800.648.3647.
- For technical support log onto our online help system. Be sure to include a detailed description of the problem along with PC configuration, Audio Codes hardware, driver versions, firmware versions, a sample program that demonstrates the issue, and any other pertinent information.

To request an online help account please contact technical support at blade-support@audiocodes.com.

SALES AND GENERAL INFORMATION

For sales and general information, use the following contact methods:

- Call us at 732.469.0880 or toll free from the USA at 800.648.3647.
- Fax us at 732.469.2298.
- E-mail us at bladesinfo@audiocodes.com.
- Visit our web site at www.audiocodes.com/blades.

MAILING ADDRESS—USA

Ship packages or send certified mail to us at the following address:

Audio Codes USA, Inc.

27 World's Fair Drive

Somerset, NJ 08873

Chapter 2

Function Reference Library

API's Presented Alphabetically with Product Support

Below is a table showing which SmartWORKS hardware products support the many SmartWORKS API functions

TABLE 2: API FUNCTION STATUS AND PRODUCT SUPPORT

API Function Status Legend: F=Fully supported · O=To Be Obsolete · P=Pending future release, beta · *= Board Supported	Status	SmartWORKS DP	SmartWORKS DT	SmartWORKS NGX	SmartWORKS LD	SmartWORKS PCM	IPX IPX-C HPX
MakeWaveGSMHeader()	F	*	*	*	*	*	*
MakeWavePCMHeader()	F	*	*	*	*	*	*
MTAddSummationInput()	F			*	*		
MTAGCCControl() - see MTChInputAGCCControl()	F	*	*	*	*	*	
MTAGCStatus()	F	*	*	*	*	*	
MTAJListen()	F			*	*		
MTAJTalk()	F			*	*		
MTArmCTAlarm()	F	*	*	*	*	*	
MTArmFramerAlarm()	F	*	*	*		*	
MTArmMasterAlarm()	F	*	*	*	*	*	
MTAVCCControl()	F	*	*	*	*	*	
MTAVCStatus()	F	*	*	*	*	*	
MTBlinkBoard()	F	*	*	*	*	*	*
MTBoardConfig()	F	*	*	*	*	*	*
MTBoardGetConfig()	F	*	*	*	*	*	*
MTBoardGetCustomSwitchSetting()	F			*	*		
MTBoardGetLicensedSessions()	F						*
MTBoardGetTerminationImpedance()	F					*	
MTBoardPCMGetSignalPolarity()	F					*	
MTBoardPCMSetSignalPolarity()	F					*	
MTBoardPCMTrunkFrameSyncControl()	F					*	
MTBoardPCMTrunkFrameSyncStatus()	F					*	
MTBoardPCMTrunkSlotTxControl()	F					*	
MTBoardPCMTrunkSlotTxStatus()	F					*	
MTBoardSetConfig()	F	*	*	*	*	*	*

TABLE 2: API FUNCTION STATUS AND PRODUCT SUPPORT (CONTINUED)

API Function Status Legend: F=Fully supported · O=To Be Obsolete · P=Pending future release, beta · *= Board Supported	Status	SmartWORKS DP	SmartWORKS DT	SmartWORKS NGX	SmartWORKS LD	SmartWORKS PCM	IPX/IPX-C/HPX
MTBoardSetIdleCoding()	F					*	
MTBoardSetTerminationImpedance()	F					*	
MTCalculateFilterRange()	F	*	*	*	*	*	
MTCallString()	F				*		
MTCallStringAsync()	F				*		
MTCallStringEx()	F				*		
MTCC_ConnectRequest()	F		*				
MTCC_ConnectResp()	F		*				
MTCC_DiscRequest()	F		*				
MTCC_FacilityRequest()	F		*				
MTCC_GetStatusByCh()	F	*					
MTCC_GetStatusByRef()	F	*					
MTCC_ResumeRequest()	F		*				
MTCC_SuspendRequest()	F		*				
MTCC_UserInfoRequest()	F		*				
MTChCallerIDControl()	F				*		
MTChCallerIDStatus()	F				*		
MTChClearUsrSignalParams()	F	*	*	*	*	*	
MTChCPMControl()	F	*	*	*	*	*	
MTChCPMStatus()	F	*	*	*	*	*	
MTChCPMSignalControl()	F	*	*	*	*	*	
MTChCPMSignalStatus()	F	*	*	*	*	*	
MTChGetCallerIDParams()	F				*		
MTChGetCPMSignalParams()	F	*	*	*	*	*	
MTChGetUsrSignalParams()	F	*	*	*	*	*	
MTChGetVoiceDetectParams()	F	*	*	*	*	*	
MTChInputACTDControl()	F	*	*	*	*	*	
MTChInputACTDStatus()	F	*	*	*	*	*	

TABLE 2: API FUNCTION STATUS AND PRODUCT SUPPORT (CONTINUED)

API Function Status Legend: F=Fully supported · O=To Be Obsolete · P=Pending future release, beta · *= Board Supported	Status	SmartWORKS DP	SmartWORKS DT	SmartWORKS NGX	SmartWORKS LD	SmartWORKS PCM	IPX IPX-C/HPX
MTChInputAGCControl()	F	*	*	*	*	*	
MTChInputAGCStatus()	F	*	*	*	*	*	
MTChInputGetACTDDetectPower()	F	*	*	*	*	*	
MTChInputGetACTDParams()	F	*	*	*	*	*	
MTChInputGetAGC()	F	*	*	*	*	*	
MTChInputGetDTMFDetectParams()	F	*	*	*	*	*	
MTChInputGetGain()	F	*	*	*	*	*	
MTChInputGetToneDetectParams()	F	*	*	*	*	*	
MTChInputSetACTDParams()	F	*	*	*	*	*	
MTChInputSetAGC()	F	*	*	*	*	*	
MTChInputSetDTMFDetectParams()	F	*	*	*	*	*	
MTChInputSetGain()	F	*	*	*	*	*	
MTChInputSetToneDetectParams()	F	*	*	*	*	*	
MTChInputStepGain()	F	*	*	*	*	*	
MTChInputToneDetectControl()	F	*	*	*	*	*	
MTChInputToneDetectStatus()	F	*	*	*	*	*	
MTChResetCPMSignalParams()	F	*	*	*	*	*	
MTChSetCallerIDParams()	F				*		
MTChSetCPMSignalParams	F	*	*	*	*	*	
MTChSetUsrSignalParams()	F	*	*	*	*	*	
MTChSetVoiceDetectParams()	F	*	*	*	*	*	
MTChUsrSignalControl()	F	*	*	*	*	*	
MTChUsrSignalStatus()	F	*	*	*	*	*	
MTChVoiceDetectControl()	F	*	*	*	*	*	
MTChVoiceDetectStatus()	F	*	*	*	*	*	
MTClearBoardEventCallback()	F	*	*	*	*	*	
MTClearDTMFDigits()	F	*	*	*	*	*	
MTClearEventCallback()	F	*	*	*	*	*	*
MTClearFramerStatistic()	F	*	*	*		*	

TABLE 2: API FUNCTION STATUS AND PRODUCT SUPPORT (CONTINUED)

API Function Status Legend: F=Fully supported · O=To Be Obsolete · P=Pending future release, beta · *= Board Supported	Status	SmartWORKS DP	SmartWORKS DT	SmartWORKS NGX	SmartWORKS LD	SmartWORKS PCM	IPX/IPX-C/HPX
MTClearMFTones()	F	*	*	*	*	*	
MTClearPriorityEventCallback()	F	*	*	*	*	*	
MTClearUsrTone() -see MTClearUsrToneParams()	O	*	*	*	*	*	
MTClearUsrToneParams()	P	*	*	*	*	*	
MTCloseBoard()	F	*	*	*	*	*	
MTCloseChannel()	F	*	*	*	*	*	
MTDialString()	F	*	*	*	*	*	
MTDialStringAsync()	F	*	*	*	*	*	
MTDialStringEx()	F	*	*	*	*	*	
MTDisableCallerID()	F	*	*	*	*	*	
MTDisableFramerABCDSignalingEvent()	F	*	*				
MTDisableMixing()	F	*	*	*	*	*	
MTDisarmCTAlarm()	F	*	*	*	*	*	
MTDisarmFramerAlarm()	F	*	*	*		*	
MTEnableCallerID()	F	*	*	*	*	*	
MTEnableFramerABCDSignalingEvent()	F	*	*				
MTEnableMixing()	F	*	*	*	*	*	
MTEventControl()	F	*	*	*	*	*	*
MTFlushCallerID()	F	*	*	*	*	*	
MTFlushEvents()	F	*	*	*	*	*	*
MTGetAdapterDescription()	F	*	*	*	*	*	
MTGetAdapterEEPROMConfig()	F	*	*	*	*	*	
MTGetAdapterInfo()	F	*	*	*	*	*	
MTGetAdapterStatusDescription()	F	*	*	*	*	*	
MTGetAdapterXInfo()	F	*	*	*	*	*	
MTGetAGC() - see MTChInputGetAGC()	F	*	*	*	*	*	
MTGetAJListen()	F			*	*		
MTGetAJMap()	F			*	*		

TABLE 2: API FUNCTION STATUS AND PRODUCT SUPPORT (CONTINUED)

API Function Status Legend: F=Fully supported · O=To Be Obsolete · P=Pending future release, beta · *= Board Supported	Status	SmartWORKS DP	SmartWORKS DT	SmartWORKS NGX	SmartWORKS LD	SmartWORKS PCM	IPX/IPX-C/HPX
MTGetAJTalk()	F			*	*		
MTGetAlertToneParams()	F	*	*	*	*	*	
MTGetAVC()	F	*	*	*	*	*	
MTGetBoardAssemblyInfo()	F	*	*	*	*	*	
MTGetBoardIDSetting()	F			*			
MTGetBoardOEMInfo()	F	*	*	*	*	*	
MTGetBoardOpenStatus()	F	*	*	*	*	*	
MTGetBoardStatistics()	F	*	*	*	*	*	
MTGetCallerID()	F	*	*	*	*	*	
MTGetCallerIDStatus()	F	*	*	*	*	*	
MTGetCallStatus()	F	*	*	*	*	*	
MTGetCCStatistics()	F	*					
MTGetChannelEvent()	F	*	*	*	*	*	
MTGetChannelInfo()	F	*	*	*	*	*	
MTGetChannelOpenStatus()	F	*	*	*	*	*	
MTGetChannelOwner()	F	*	*	*	*	*	
MTGetChannelStatus()	F	*	*	*	*	*	
MTGetCpmToneParams()	P	*	*	*	*	*	
MTGetCTAlarmStatus()	F	*	*	*	*	*	*
MTGetCTMasterClock()	F	*	*	*	*	*	
MTGetCTNetrefSource()	F	*	*	*	*	*	
MTGetCTRoute()	F		*		*	*	
MTGetCTSlaveClockMode()	F	*	*	*	*	*	
MTGetCTSlotConnection()	F	*	*	*	*	*	
MTGetReturnCodeDescription()	F	*	*	*	*	*	*
MTGetDataLinkStatus()	F	*					
MTGetDigits()	F	*	*	*	*	*	
MTGetDigitsAsync()	F	*	*	*	*	*	
MTGetDigitsEx()	F	*	*	*	*	*	

TABLE 2: API FUNCTION STATUS AND PRODUCT SUPPORT (CONTINUED)

API Function Status Legend: F=Fully supported · O=To Be Obsolete · P=Pending future release, beta · *= Board Supported	Status	SmartWORKS DP	SmartWORKS DT	SmartWORKS NGX	SmartWORKS LD	SmartWORKS PCM	IPX IPX-C/HPX
MTGetDLLVersion()	F	*	*	*	*	*	*
MTGetDTMFDetectParams() - see MTChInputGetDTMFDetectParams()	O	*	*	*	*	*	
MTGetE1FramerStatistic()	F	*	*				
MTGetECControl()	F		*		*		
MTGetECParams()	F		*		*		
MTGetECFreezeAdaptation()	F		*		*		
MTGetEventCodeDescription()	F	*	*	*	*	*	
MTGetEventFilters()	F	*	*	*	*	*	
MTGetExtension()	P						*
MTGetFramerABCDSignalingEventStatus()	F	*	*				
MTGetFramerAlarmArmStatus()	F	*	*	*		*	
MTGetFramerAlarmStatus()	F	*	*			*	
MTGetFramerLoopbackMode()	F		*				
MTGetFramerOutput	F	*	*	*		*	
MTGetFraming()	F	*	*			*	
MTGetGain() - see MTChInputGetGain()	F	*	*	*	*	*	
MTGetGCI()	F	*	*	*	*	*	
MTGetGCIMap()	F	*	*	*	*	*	
MTGetHookState()	F				*		
MTGetInputs()	F	*	*	*	*	*	
MTGetLastErrorA()	F	*	*	*	*	*	
MTGetLastErrorW()	F	*	*	*	*	*	
MTGetLastTerm()	F	*	*	*	*	*	
MTGetLCDetectState	F				*		
MTGetLVDetectState()	F						
MTGetLVDetectValue()	F				*		
MTGetLVParams()	F				*		
MTGetMasterAlarmStatus()	F	*	*	*	*	*	

TABLE 2: API FUNCTION STATUS AND PRODUCT SUPPORT (CONTINUED)

API Function Status Legend: F=Fully supported · O=To Be Obsolete · P=Pending future release, beta · *= Board Supported	Status	SmartWORKS DP	SmartWORKS DT	SmartWORKS NGX	SmartWORKS LD	SmartWORKS PCM	IPX IPX-C/HPX
MTGetMixingStatus()	F	*	*	*	*	*	
MTGetMoni()	F	*	*	*	*	*	
MTGetNET()	F	*	*	*		*	
MTGetNGXFramerStatistic()	F			*			
MTGetOutput()	F	*	*	*	*	*	
MTGetOutputSource()	F	*	*	*	*	*	
MTGetPHYStatistics()	F	*	*				
MTGetPSTNParams()	F	*	*	*	*	*	
MTGetReverseLoopPolarity()	F				*		
MTGetRingDetectState()	F				*		
MTGetStationMacAddress()	F						*
MTGetStationMacAddressEx()	F						*
MTGetStatistics()	F	*	*	*	*	*	
MTGetStreamingConfig()	F	*	*	*	*	*	
MTGetSummationAVC()	F			*	*		
MTGetSummationGain()	F			*	*		
MTGetSummationInfo()	F			*	*		
MTGetSummationInputs()	F			*	*		
MTGetSummationOutput()	F			*	*		
MTGetSummationVolume()	F			*	*		
MTGetSystemInfo()	F	*	*	*	*	*	
MTGetT1FramerStatistic()	F	*	*				
MTGetTerm()	F	*	*	*	*	*	
MTGetTimeSlotRxSignaling()	F	*	*			*	
MTGetTimeSlotTxSignaling()	F		*			*	
MTGetToneDetectParams() - MTChInputGetToneDetectParams()	O	*	*	*	*	*	
MTGetToneGenParams()	F	*	*	*	*	*	
MTGetUsrToneParams()	P	*	*	*	*	*	

TABLE 2: API FUNCTION STATUS AND PRODUCT SUPPORT (CONTINUED)

API Function Status Legend: F=Fully supported · O=To Be Obsolete · P=Pending future release, beta · *= Board Supported	Status	SmartWORKS DP	SmartWORKS DT	SmartWORKS NGX	SmartWORKS LD	SmartWORKS PCM	IPX IPX-C/HPX
MTGetVersion()	F	*	*	*	*	*	*
MTHighpassFilterControl()	F	*	*	*	*	*	
MTHighpassFilterStatus()	F	*	*	*	*	*	
MTIpCCEventControl()	F						*
MTIpCCEventStatus()	F						*
MTIpClearPassiveNetworkTransportStatistic()	F						*
MTIpClearNetworkPortStatistics()	F						*
MTIpClearStationCallStatistics()	F						*
MTIpCreateMediaSession()	F						*
MTIpDChannelEventControl()	F						*
MTIpDChannelEventFilteringControl()	F						*
MTIpDChannelEventFilteringStatus()	F						*
MTIpDChannelEventStatus()	F						*
MTIpDeleteMediaSession()	F						*
MTIpDisableSignalingProtocol()	P						*
MTIpEnableSignalingProtocol()	F						*
MTIpGetLicenseInfo()	F						*
MTIpGetMediaRoutingParams()	F						*
MTIpGetMediaSessionInfo()	F						*
MTIpGetNetworkPortStatistics	F						*
MTIpGetNetworkPortStatus()	F						*
MTIpGetPassiveNetworkTransportStatistics()	F						*
MTIpGetSessionMediaDest()	F						*
MTIpGetStationCount()	F						
MTIpGetStationList()	F						
MTIpGetStationParams()	F						*
MTIpGetStationCallStatistics()	F						*
MTIpLoadLicense()	F						*
MTIpSetMediaRoutingParams()	F						

TABLE 2: API FUNCTION STATUS AND PRODUCT SUPPORT (CONTINUED)

API Function Status Legend: F=Fully supported · O=To Be Obsolete · P=Pending future release, beta · *= Board Supported	Status	SmartWORKS DP	SmartWORKS DT	SmartWORKS NGX	SmartWORKS LD	SmartWORKS PCM	IPX IPX-C/HPX
MTIpSetSessionMediaDest()	F						
MTIpSignalingProtocolStatus()	F						*
MTLoadSignalProfile()	F	*	*	*	*	*	
MTOffHook()	F				*		
MTOnHook()	F				*		
MTOpenBoard()	F	*	*	*	*		
MTOpenChannel()	F	*	*	*	*	*	
MTPlayBuffer()	F	*	*	*	*	*	
MTPlayBufferAsync()	F	*	*	*	*	*	
MTPlayBufferEx()	F	*	*	*	*	*	
MTPlayDevice()	F	*	*	*	*	*	
MTPlayDeviceAsync()	F	*	*	*	*	*	
MTPlayDeviceEx()	F	*	*	*	*	*	
MTPlayFile()	F	*	*	*	*	*	
MTPlayFileAsync()	F	*	*	*	*	*	
MTPlayFileEx()	F	*	*	*	*	*	
MTPlayIndex()	F	*	*	*	*	*	
MTPlayIndexEx()	F	*	*	*	*	*	
MTPlayIndexAsync()	F	*	*	*	*	*	
MTPlayTone()	F	*	*	*	*	*	
MTPlayToneAsync()	F	*	*	*	*	*	
MTPlayToneEx()	F	*	*	*	*	*	
MTPutBoardEvent()	F	*	*	*	*	*	
MTPutChannelEvent()	F	*	*	*	*	*	
MTReadDigit()	F	*	*	*	*	*	
MTReadDTMFTone()	F	*	*	*	*	*	
MTReadMFTone()	F	*	*	*	*	*	
MTRecBuffer()	F	*	*	*	*	*	
MTRecBufferAsync()	F	*	*	*	*	*	

TABLE 2: API FUNCTION STATUS AND PRODUCT SUPPORT (CONTINUED)

API Function Status Legend: F=Fully supported · O=To Be Obsolete · P=Pending future release, beta · *= Board Supported	Status	SmartWORKS DP	SmartWORKS DT	SmartWORKS NGX	SmartWORKS LD	SmartWORKS PCM	IPX/IPX-C/HPX
MTRecBufferEx()	F	*	*	*	*	*	
MTRecDevice()	F	*	*	*	*	*	
MTRecDeviceAsync()	F	*	*	*	*	*	
MTRecDeviceEx()	F	*	*	*	*	*	
MTRecFile()	F	*	*	*	*	*	
MTRecFileAsync()	F	*	*	*	*	*	
MTRecFileEx()	F	*	*	*	*	*	
MTRemoveSummationInput()	F			*	*		
MTResetAJListen()	F			*	*		
MTResetAJTalk()	F			*	*		
MTResetBoardStatistics()	F	*	*	*	*	*	
MTResetCpmToneParams()	P	*	*	*	*	*	
MTResetCTRoute()	F		*		*	*	
MTResetFramerLoopbackMode()	F		*				
MTResetInputs()	F	*	*	*	*	*	
MTResetOutput()	F	*	*	*	*	*	
MTResetStatistics()	F	*	*	*	*	*	
MTResetSummation()	F			*	*		
MTResyncTime()	F	*	*	*	*	*	
MTReturnEventBuffer()	F	*	*	*		*	
MTSetACTDParams() - see MTChInputSetACTDParams()	F	*	*	*	*	*	
MTSetAdapterEEPROMConfig()	F	*	*	*	*	*	
MTSetAdapterToDefault()	F	*	*	*	*	*	
MTSetAGC() - see MTChInputSetAGC()	F	*	*	*	*	*	
MTSetAlertToneParams()	F	*	*	*	*	*	
MTSetAVC()	F	*	*	*	*	*	
MTSetBoardEventCallback()	F	*	*	*	*	*	
MTSetChannelToDefault()	F	*	*	*	*	*	

TABLE 2: API FUNCTION STATUS AND PRODUCT SUPPORT (CONTINUED)

API Function Status Legend: F=Fully supported · O=To Be Obsolete · P=Pending future release, beta · *= Board Supported	Status	SmartWORKS DP	SmartWORKS DT	SmartWORKS NGX	SmartWORKS LD	SmartWORKS PCM	IPX IPX-C/HPX
MTSetCpmToneParams()	P	*	*	*	*	*	
MTSetCTMasterClock()	F	*	*	*	*	*	
MTSetCTNetrefSource()	F	*	*	*	*	*	
MTSetCTRoute()	F		*		*	*	
MTSetCTSlaveClockMode()	F	*	*	*	*	*	
MTSetDeviceIO()	F	*	*	*	*	*	
MTSetDTMFDetectParams() - see MTChInputSetDTMFDetectParams()	F	*	*	*	*	*	
MTSetECControl()	F		*		*		
MTSetECParams()	F		*		*		
MTSetECFreezeAdaptation()	F		*		*		
MTSetEventCallback()	F	*	*	*	*	*	
MTSetEventFilters()	F	*	*	*	*	*	
MTSetFramerLoopbackMode()	F		*				
MTSetFramerOutput()	F	*	*	*		*	
MTSetGain() - see MTChInputSetGain()	F	*	*	*	*	*	
MTSetInputs()	F	*	*	*	*	*	
MTSetLVParams()	F				*		
MTSetMoni()	F	*	*	*	*	*	
MTSetNET()	F	*	*	*		*	
MTSetOutput()	F	*	*	*	*	*	
MTSetOutputSource()	F	*	*	*	*	*	
MTSetPriorityEventCallback()	F	*	*	*	*	*	
MTSetPSTNParams()	F	*	*	*	*	*	
MTSetReverseLoopPolarity()	F				*		
MTSetStreamingConfig()	F	*	*	*	*	*	
MTSetSummationAVC()	F			*	*		
MTSetSummationGain()	F			*	*		
MTSetSummationVolume()	F			*	*		

TABLE 2: API FUNCTION STATUS AND PRODUCT SUPPORT (CONTINUED)

API Function Status Legend: F=Fully supported · O=To Be Obsolete · P=Pending future release, beta · *= Board Supported	Status	SmartWORKS DP	SmartWORKS DT	SmartWORKS NGX	SmartWORKS LD	SmartWORKS PCM	IPX IPX-C/HPX
MTSetTerm()	F	*	*	*	*	*	
MTSetTimeSlotTxSignaling()	F		*				
MTSetToneDetectParams()	F	*	*	*	*	*	
MTSetToneGenParams()	F	*	*	*	*	*	
MTSetUsrToneParams()	P	*	*	*	*	*	
MTSetVolume()	F	*	*	*	*	*	
MTStartAlertTone()	F	*	*	*	*	*	
MTStartCurrentFunction()	F	*	*	*	*	*	
MTStartMUX()	F	*	*	*	*	*	
MTStartMUXOnChannel()	F	*	*	*	*	*	
MTStartStreaming()	F	*	*	*	*	*	
MTStartSummation()	F			*	*		
MTStepGain() - see MTChInputStepGain()	F	*	*	*	*	*	
MTStepVolume()	F	*	*	*	*	*	
MTStopAlertTone()	F	*	*	*	*	*	
MTStopBlinkBoard()	F	*	*	*	*	*	
MTStopChannel()	F	*	*	*	*	*	
MTStopCurrentFunction()	F	*	*	*	*	*	
MTStopMUX()	F	*	*	*	*	*	
MTStopMUXOnChannel()	F	*	*	*	*	*	
MTStopStreaming()	F	*	*	*	*	*	
MTStopSummation()	F			*	*		
MTStreamBufIn()	F	*	*	*	*	*	
MTStreamBufOut()	F	*	*	*	*	*	
MTStreamBufPause()	F	*	*	*	*	*	
MTStreamBufResume()	F	*	*	*	*	*	
MTSummationAVCControl()	F			*	*		
MTSummationAVCStatus()	F			*	*		
MTSysClearEventCallback()	F	*	*	*	*	*	*

TABLE 2: API FUNCTION STATUS AND PRODUCT SUPPORT (CONTINUED)

API Function Status Legend: F=Fully supported · O=To Be Obsolete · P=Pending future release, beta · *= Board Supported	Status	SmartWORKS DP	SmartWORKS DT	SmartWORKS NGX	SmartWORKS LD	SmartWORKS PCM	IPX IPX-C/HPX
MTSysGetVersion()	F	*	*	*	*	*	*
MTSysGetConfig()	F	*	*	*	*	*	*
MTSysSetEventCallback()	F	*	*	*	*	*	
MTSysSetConfig()	F	*	*	*	*	*	*
MTSysShutdown()	F	*	*	*	*	*	
MTSysStartup()	F	*	*	*	*	*	*
MTSysWaitForEvent()	F	*	*	*	*	*	*
MTWaitForAdapterEvent()	F	*	*	*	*	*	
MTWaitForChannelEvent()	F	*	*	*	*	*	
MTWink()	F				*		
MTWFGetUpdatingStatus	F	*	*		*	*	
MTWFInit()	F	*	*		*	*	
MTWFUninit()	F	*	*		*	*	
MTWFUpdateFirmware	F	*	*		*	*	

APIs Presented Logically

The following section, organized in logical order, provides a detailed description of each API.

MISC. FUNCTIONS

The following section defines SmartWORKS APIs that have been defined as miscellaneous.

ERROR & GET DESCRIPTION FUNCTIONS

The following section describes APIs which are used to obtain board errors, and descriptions for events and return codes.

MTGetLastErrorA()

Description

MTGetLastErrorA() is a function used to retrieve the last error code and associated ASCII message in the system. The first parameter points to the buffer to take the last error code. The second parameter contains the size of the ASCII string buffer when passed and the length of the ASCII message when returned. The third parameter points to the ASCII string buffer to take the actual message associated with the error code. The retrieved ASCII string is always null terminated and the length of the returned string replaces the size of the error message buffer when the API returns.

Prototype

MT_RESULT MTGetLastErrorA (

```
const PMT_RESULT    pExtError,
const PUSHORT      nErrBufSize,
const PSTR          pErrBuf,
);
```

Parameters

pExtError	Pointer to error code buffer
nErrBufSize	Pointer to size of error message buffer
pErrBuf	Pointer to error message buffer

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_DATA_TRUNCATED	Data copied but truncated; the buffer size set by the user application is not large enough for this API request. <i>FlagDataTruncated</i> field of the MTSYS_CONFIG or the MTSYS_CONFIGURATION data structure must be enabled (1) to receive this return code, otherwise MT_RET_OK is used, and the truncated data is passed to the user application
MT_RET_NOT_AVAILABLE	Driver is not loaded

Events

None

Scenario

Normally, **MTGetLastError()** is mapped as **MTGetLastErrorA()** to use an ASCII string for displaying messages. The user application can define UNICODE before including [NTIAPI.h](#) to use a unicode string instead of ASCII.

MTGetLastErrorW()

Description

MTGetLastErrorW() is a function that is used to retrieve the last error code and its associated message occurred in the system. It works the same as **MTGetLastErrorA()** except that the error message is put into UNICODE string format.

Prototype

MT_RESULT GetLastErrorW(

```
const PMT_RESULT    pExtError,  
const PUSHORT      nErrBufSize,  
const PWSTR         pErrBuf,  
);
```

Parameters

pExtError	Pointer to error code buffer
nErrBufSize	Pointer to size of error message buffer
pErrBuf	Pointer to error message buffer

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_DATA_TRUNCATED	Data copied but truncated; the buffer size set by the user application is not large enough for this API request. <i>Flag-DataTruncated</i> field of the MTSYS_CONFIG or the MTSYS_CONFIGURATION data structure must be enabled (1) to receive this return code, otherwise MT_RET_OK is used, and the truncated data is passed to the user application
MT_RET_NOT_AVAILABLE	Driver is not loaded

Events

None

Scenario

Normally, **MTGetLastError()** is mapped as **MTGetLastErrorA()** to use an ASCII string for displaying messages. The user application can define UNICODE before including [NTIAPI.h](#) to use **MTGetLastErrorW()**, the unicode version, instead of ASCII.

MTGetReturnCodeDescription()

Description

For valid error codes, returns the descriptive macro name for the error code, as found in the SDK file "NTiErr.h".

Prototype

void MTGetReturnCodeDescription (

```
const MT_RESULT      ReturnCode,  
const PCHAR          pDescription,  
const int             nLength,  
);
```

Parameters

ReturnCode	The value generated for any MT_RET message. For example 0x0.
pDescription	pointer to the NULL terminated description
nLength	length of the description buffer

Return Codes

None

Events

None

MTGetEventCodeDescription()

Description

Retrieves the text description for any event code.

Prototype

void MTGetEventCodeDescription (

```
const ULONG      EventCode,  
const PCHAR      pDescription,  
const int        nLength  
);
```

Parameters

EventCode	The hex value generated for any event code. For example 0x01.
pDescription	pointer to the NULL terminated description
nLength	length of the description buffer

Return Codes

None

Events

None

NETWORK INTERFACE ALARMS & STATISTICS

The following section defines the APIs used for obtaining network interface alarms and statistics.

MTArmFramerAlarm()

Description

MTArmFramerAlarm() arms the alarm watch for framer error reporting, otherwise framer error alarm goes unreported. When alarmed, an error is only reported once. The application must rearm the alarm so the next error can be reported.

The board index is 0 for the first SmartWORKS board within the platform.

The framer index is 0 for the first framer. A framer alarm API (**MTGetFramerAlarmStatus()**) specifies which framer is to retrieve the arm status.

The possible framer alarm enables are:

T1 Framer Alarm Enables:

ALARM_LOS	for Loss of Signal alarm
ALARM_LOF	for Loss of Frame alarm
ALARM_AIS	for Alarm Indication Signal alarm
ALARM_YELLOW	for YELLOW alarm (Remote Alarm Indication - RAI)

E1 Framer Alarm Enables:

ALARM_LOS	for Loss of Signaling Multiframe Alignment alarm
ALARM_LOSMF	for Loss of Signaling Multiframe Alignment alarm enable. This alarm is not used when configured for Basic E1.
ALARM_LOCRCMF	for Loss of CRC Multiframe Alignment alarm. This alarm is not used when configured for Basic E1.
ALARM_TS16RAI	for Time Slot 16 Remote Alarm Indication alarm. This alarm is not used when configured for ISDN or DASS2.
ALARM_YELLOW	for YELLOW alarm (Remote Alarm Indication - RAI)

NGX Framer Alarm Enables:

ALARM_LOS	for Loss of Signaling Multiframe Alignment alarm

PCM Framer Alarm Enables:

ALARM_LOSYNC	for Loss of Synchronization on PCM boards

NOTE: A definition of each alarm is available where the API **MTGetFramerAlarmStatus() is defined.**

Prototype**MT_RESULT MTArmFramerAlarm(**

```

const BOARD      nBoard,
const USHORT     nFramer,
const ULONG      AlarmEnable
);

```

Parameters

nBoard	board index
nFramer	framer index - valid ranges DP board: 0-3 DT board: 0-1 NGX board: 0-23 (24 channel card)
AlarmEnable	framer alarm enable

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_INVALID_BOARD	Board number is out of range
MT_RET_BOARD_NOT_OPENED	Board device not opened for access
MT_RET_CHANNEL_NOT_OPENED	This return value only applies to the NGX where there is one framer per channel. This is returned to the user if this API is invoked on a closed channel.
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_API_THREAD_PROTECTED	Another thread is currently accessing this API

Events

```

EVT_LOS_ALARM
EVT_LOF_ALARM
EVT_AIS_ALARM
EVT_YELLOW_ALARM
EVT_LOSMF_ALARM
EVT_LOCRCMF_ALARM
EVT_TS16RAI_ALARM
EVT_LOSYNC

```

See Also [MTGetFramerAlarmArmStatus\(\)](#)

MTDisarmFramerAlarm()

Description

MTDisarmFramerAlarm() disarms the alarm watch for framer error reporting. When alarmed, an error is only reported once. The application must rearm the alarm so the next error can be reported. The board index is 0 for the first SmartWORKS board within the platform.
The framer index is 0 for the first framer.
A framer alarm API specifies which framer is to retrieve the arm status.

The possible framer alarm enables are

T1 Framer Alarm Enables:

ALARM_LOS	for Loss of Signal alarm
ALARM_LOF	for Loss of Frame alarm
ALARM_AIS	for Alarm Indication Signal alarm
ALARM_YELLOW	for YELLOW alarm

E1 Framer Alarm Enables

ALARM _LOSMF	for Loss of Signaling Multiframe Alignment alarm
ALARM _LOCRCMF	for Loss of CRC Multiframe Alignment alarm
ALARM _TS16RAI	for Time Slot 16 Remote Alarm Indication alarm
ALARM_LOS	Loss of Signal alarm
ALARM_YELLOW	for YELLOW alarm

NGX Framer Alarm Enables:

ALARM _LOS	for Loss of Signaling Multiframe Alignment alarm

Prototype

MT_RESULT MTDisarmFramerAlarm(

```
const BOARD      nBoard,
const USHORT     nFramer,
const ULONG      AlarmEnable
);
```

Parameters

nBoard	board index
nFramer	framer index - valid ranges DP board: 0-3 DT board: 0-1 NGX board: 0-23 (24 channel card)
AlarmEnable	framer alarm enable

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_INVALID_BOARD	Board number is out of range
MT_RET_BOARD_NOT_OPENED	Board device not opened for access
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_API_THREAD_PROTECTED	Another thread is currently accessing this API

Events

EVT_LOS_ALARM
EVT_LOF_ALARM
EVT_AIS_ALARM
EVT_YELLOW_ALARM
EVT_LOSMF_ALARM
EVT_LOCRCMF_ALARM
EVT_TS16RAI_ALARM

MTGetFramerAlarmArmStatus()

Description

Retrieves whether the specified alarm watch is armed.

Prototype

MT_RESULT MTGetFramerAlarmArmStatus(

```
const BOARD      nBoard,
const USHORT     nFramer,
const ULONG      AlarmEnable,
ULONG * const    pAlarmEnable
);
```

Parameters

nBoard	board index
nFramer	framer index - valid ranges DP board: 0-3 DT board: 0-1 NGX board: 0-23 (24 channel card)
AlarmEnable	framer alarm enable
pAlarmEnable	pointer to framer alarm arm status buffer Framer alarm arm status will be retrieved into the provided buffer in the bit order as the enable bit defined for framer alarm enable. Bit with value 1 is for alarm armed, bit with value 0 is for alarm not armed.

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_PARAM	Invalide argument passed to function
MT_RET_INVALID_BOARD	Board number is out of range
MT_RET_BOARD_NOT_OPENED	Board device not opened for access
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_API_THREAD_PROTECTE D	Another thread is currently accessing this API

Events

None

See Also: [MTArmFramerAlarm\(\)](#)

MTGetFramerAlarmStatus()

Description

Retrieves framer alarms. **MTArmFramerAlarm()** must be used in cooperation with this API. Alarms will not be reported unless **MTArmFramerAlarm()** has been used the arm the alarm.

A non-zero value in field of LOS_Alarm, LOF_Alarm, AIS_Alarm, or YELLOW_Alarm indicates the framer has detected that alarm and the alarm condition has not cleared.

Prototype

MT_RESULT MTGetFramerAlarmStatus(

```
const BOARD      nBoard,
const USHORT     nFramer,
ULONG * const    pAlarmEnable
);
```

Parameters

nBoard	board index
nFramer	framer index - valid ranges DP board: 0-3 DT board: 0-1 NGX board: 0-23 (24 channel card)
pAlarmEnable	pointer to framer alarm arm status buffer Framer alarm arm status will be retrieved into the provided buffer in the bit order as the enable bit defined for framer alarm enable. Bit with value 1 is for alarm armed, bit with value 0 is for alarm not armed.

Structure

In T1 mode, framer status MT_FRAMER_STATUS is defined as follows:

TABLE 3: MT_FRAMER_STATUS - T1

Type	Name	Purpose
UCHAR	nFramer	Framer index
UCHAR	LOS_Alarm	LOS alarm detected and is still present. This may indicate that the incorrect signal is being received (wrong configuration) or the trunk is unplugged.
UCHAR	LOF_Alarm	LOF alarm detected. The framer alignment of the Rx signal is incorrect possibly due to a weak signal, or bit errors. This alarm is also present while the framer is synching upon start-up and clears after 30 seconds.

TABLE 3: MT_FRAMER_STATUS - T1

Type	Name	Purpose
UCHAR	AIS_Alarm	Alarm Indication Signal (AIS) alarm detected. Transmitted to the NI upon a loss of originating signal or a sever disruption.
UCHAR	YELLOW_Alarm	Yellow alarm, also known as the Remote Alarm Indicator (RAI). Communicates a loss of signal or an out of frame to the far end.
UCHAR	SMFAE	Not applicable, E1 only
UCHAR	SMFCRC4E	Not applicable, E1 only
UCHAR	TS16RAI_alarm	Not applicable, E1 only

In E1 mode, framer status MT_FRAMER_STATUS is defined as follows:

TABLE 4: MT_FRAMER_STATUS - E1

Type	Name	Purpose
UCHAR	nFramer	Framer index
UCHAR	LOS_Alarm	LOS alarm detected and is still present. This may indicate that the incorrect signal is being received (wrong configuration) or the trunk is unplugged.
UCHAR	LOF_Alarm	not applicable
UCHAR	AIS_Alarm	not applicable
UCHAR	Yellow_Alarm	Yellow alarm, also known as the Remote Alarm Indicator (RAI). This data is obtained from Bit 3 of Timeslot 0 (odd frames only). Communicates a loss of signal or an out of frame to the far end.
UCHAR	SMFAE	Indicates signaling MF alignment error. The received alignment bits (bits 1-7 of even frames, timeslot 0) are not aligned to the local system. This field is not used when configured for Basic E1.
UCHAR	SMFCRC4E	Indicates a CRC-4 MF (Cycle Redundancy) error - the occurrence of a received set of check bits that differ from the locally generated code. The E Bits (Bit 1 of Frames 13 and 15 of Timeslot 0) is set to 1 to indicate an alarm state. Each E bit represents one sub-frame.This field is not used when configured for Basic E1.
UCHAR	TS16RAI_alarm	Timeslot 16 RAI indicator. A Remote Alarm Indicator (RA) indicating a remote side signaling multi-frame error. Bit 6 of Timeslot 16 is set to 1 (alarm state) when signaling bits (bits 0-3 of framer 0 TS 16) are not aligned. This field is not valid when configured for ISDN DASS2 and the data should be ignored.

In NGX mode, framer status MT_FRAMER_STATUS is defined as follows:

TABLE 5: MT_FRAMER_STATUS - NGX

Type	Name	Purpose
UCHAR	nFramer	N/A
UCHAR	LOS_Alarm	LOS alarm detected and is still present. This may indicate that the incorrect signal is being received (wrong configuration) or the trunk is unplugged.
UCHAR	LOF_Alarm	N/A
UCHAR	AIS_Alarm	N/A
UCHAR	YELLOW_Alarm	N/A
UCHAR	LOSMF_Alarm	N/A
UCHAR	LOCRC4MF_alarm	N/A
UCHAR	TS16RAI_alarm	N/A

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_INVALID_BOARD	Board number is out of range
MT_RET_BOARD_NOT_OPENED	Board device not opened for access
MT_RET_NOT_AVAILABLE	Driver is not loaded

Events

None

See Also: [MTArmFramerAlarm\(\)](#)

MTGetT1FramerStatistic()

Description

Gets error statistics for the specified framer.
In T1 mode, accumulation of bit errors, framing bit errors, OOF and line code violations is provided for performance monitoring.
The board index is 0 for the first SmartWORKS board within the platform. The framer index is 0 for the first framer.

Prototype

MT_RESULT MTGetT1FramerStatistic(

```
const BOARD          nBoard,
const USHORT         nFramer,
PMT_T1_FRAMER_STAT const pFramerStat,
PULONG const        pLength
);
```

Parameters

nBoard	board index
nFramer	framer index
pFramerStat	pointer to framer statistic buffer
pLength	Set by the user application, this is a pointer to the size of the given buffer for the MT_T1_FRAMER_STAT structure.

Structure

The framer status MT_T1_FRAMER_STAT is defined as follows:

TABLE 6: MT_T1_FRAMER_STAT

Type	Name	Purpose
ULONG	FrameErr;	Framing bit error In T1 mode, a framing bit error is defined as an Fe-bit error when using ESF, and a framing bit error when using SF.
ULONG	OutofFrameErr;	Out of frame error In T1 mode, OutOfFrameErr indicates the number out of frame alignment events that occurred during the previous accumulation interval. The count is incremented each time a severely erred framing event forces a reframe.
ULONG	BitErr;	Bit errors In T1 mode, BitErr contains the number of bit error events that occurred during the previous accumulation interval. A bit error event is defined as a CRC-6 error when using ESF, a framing bit error when using SF.

TABLE 6: MT_T1_FRAMER_STAT

Type	Name	Purpose
ULONG	LineCode_ZeroRunErr;	Line Code Violation count LineCode_ZeroRunErr bits indicate the number of LCV error events that occurred during the previous accumulation interval. An LCV event is defined as the occurrence of a bipolar violation or excessive zeros.

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_INVALID_BOARD	Board number is out of range
MT_RET_BOARD_NOT_OPENED	Board device not opened for access
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_API_THREAD_PROTECTED	Another thread is currently accessing this API

Events

None

MTGetE1FramerStatistic()

Description

Gets error statistics for the specified framer. In E1 mode, accumulation of line code errors, frame alignment signal (FAS) bit errors, far end block error and line code violations is provided for performance monitoring.

The board index is 0 for the first SmartWORKS board within the platform.

The framer index is 0 for the first framer.

Prototype

MT_RESULT MTGetE1FramerStatistic(

```
const BOARD          nBoard,
const USHORT         nFramer,
PMT_E1_FRAMER_STAT const pFramerStat,
PULONG const        pLength
);
```

Parameters

nBoard	board index
nFramer	framer index
pFramerStat	pointer to framer statistic buffer
pLength	Set by the user application, this is a pointer to the size of the given buffer for the MT_E1_FRAMER_STAT structure.

Structure

For E1 mode, the framer status MT_E1_FRAMER_STAT is defined as follows:

TABLE 7: MT_E1_FRAMER_STAT

Type	Name	Purpose
ULONG	FASErr;	Framing bit error In E1 mode, the count is either number of FAS (frame alignment signal) bits.
ULONG	FarEndBlockErr;	Far end block errors (FEBE) In E1 mode, FarEndBlockErr indicates the number of far end block error events that occurred during the previous accumulation interval. FEBE counts are suppressed when the E1 framer has lost frame alignment.

TABLE 7: MT_E1_FRAMER_STAT

Type	Name	Purpose
ULONG	CRCErr;	In E1 mode, CRCErr indicates the number of CRC error events that occurred during the previous accumulation interval. CRC error events are suppressed when the framer is out of CRC-4 multiframe alignment.
ULONG	LineCode_ZeroRunErr;	Line Code Violation count LineCode_ZeroRunErr bits indicate the number of LCV error events that occurred during the previous accumulation interval. An LCV event is defined as the occurrence of a bipolar violation or excessive zeros.

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_INVALID_BOARD	Board number is out of range
MT_RET_BOARD_NOT_OPENED	Board device not opened for access
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_API_THREAD_PROTECTED	Another thread is currently accessing this API

Events

None

MTGetNGXFramerStatistic()

Description

Retrieves various network interface statistics that have accumulated on the specified framer either since the system was started or since the last call of **MTClearFramerStatistic()**. The statistics include cumulative errors on the PBX signal, cumulative errors on the Phone signal, cumulative count of synchronization loss errors, the amplitude of the PBX signal in volts, the amplitude of the Phone signal in volts, the amplitude of the Noise level in volts and the current clipping status for the selected framer.

Prototype

MT_RESULT MTGetNGXFramerStatistic(

```
const BOARD          nBoard,
const USHORT         nFramer,
PMT_NTTE_FRAMER_STAT const pFramerStat,
PULONG              pLength
);
```

Parameters

nBoard	Board number
nFramer	Framer index (0-23)
pFramerStat	Pointer to an MT_NTTE_FRAMER_STAT structure that will be filled in by the function
pLength	Set by the user application, this is a pointer to the size of the given buffer for the MT_NTTE_FRAMER_STAT structure

Structure

TABLE 8: MT_NTTE_FRAMER_STAT

Type	Name	Purpose
ULONG	NTerr*	PBX error. Indicates packets from PBX with bad data and listed as errors. These can result from random bad packets, or bad line conditions.
ULONG	TErr*	phone error. Indicates packets from the phone with bad data and listed as error. These can result from random bad packets or bad line conditions.
ULONG	SyncErr*	Indicates a complete loss of signal
float	NTAmpl	PBX amplitude in volts. This is not an error indication, rather it is a true reading of current voltage levels from the board's perspective

TABLE 8: MT_NTTE_FRAMER_STAT

Type	Name	Purpose
float	TEAmpl	Phone amplitude in volts. This is not an error indication, rather it is a true reading of current voltage levels.
float	Noise	Noise amplitude in volts. Recommended noise amplitude should be below 80 mv. If the noise is above this value, verify cable lengths, taps, and punchdown connections.
UCHAR	SignalClipStatus	Indicates clipping status since last call to API (status clears after each API call). A returned value of 0 indicates normal conditions. If the clipping value is high, change gain. If there are clipping errors then this will impact the other readings, first clear these errors then get NGX Framer Statistics again to verify the other readings.

* Only these fields cleared when the **MTClearFramerStatistics()** API is called.

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_BOARD	Board number is out of range
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_TIMEOUT	Wait for event timed out
MT_RET_BUSY	Board is busy
MT_RET_DATA_TRUNCATED	Data copied but truncated; the buffer size set by the user application is not large enough for this API request. <i>Flag-DataTruncated</i> field of the MTSYS_CONFIG or the MTSYS_CONFIGURATION data structure must be enabled (1) to receive this return code, otherwise MT_RET_OK is used, and the truncated data is passed to the user application
MT_RET_API_THREAD_PROTECTED	Another thread is currently accessing this API

Events

None

MTGetFraming()

Description

Gets a framing format for a specified framer on the specified board.

Prototype

MT_RESULT MTGetFraming (

```
PMT_FRAMER_STATE const    pFraming
);
```

Parameter

pFraming

Parameter pFramer is a pointer to structure MT_FRAMER_STATE which contains a field for board index (MT_FRAMER_STATE.nBoard) and framer index (MT_FRAMER_STATE.nFramer) on the specified board.

Structure

The FRAMER_STATE structure is defined as follows:

TABLE 9: MT_FRAMER_STATE

Type	Name	Purpose
UCHAR	nBoard	Board index. The board index is 0 for the first SmartWORKS board within the platform
UCHAR	nFramer	Framer index. The framer index is 0 for the first framer on board
UCHAR	State	Framer state: FRAMER_RESET FRAMER_T1_PASSIVE FRAMER_T1_TERMINATE FRAMER_E1_PASSIVE FRAMER_E1_TERMINATE
UCHAR	reserved	
UCHAR	FramingFormat	FF_E1_G704 - default FF_E1_CRC4 FF_E1_UNFRAMED FF_T1_SF FF_T1_ESF FF_T1_UNFRAMED
UCHAR	LineCoding	LC_AMI - default LC_E1_HDB3 LC_T1_B8ZS
UCHAR	LBO	Possible values for MT_FRAMER_STATE.LBO are defined in the NtiData.h file.
UCHAR	ZCS	Zero-code suppression: ZCS_NONE - default ZCS_GTE ZCS_DDS ZCS_BELL

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_INVALID_BOARD	Board number is out of range
MT_RET_BOARD_NOT_OPENED	Board device not opened for access
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_TIMEOUT	Wait for event timed out

MTClearFramerStatistic()

Description

Clears error statistics for the specified framer.

In T1 mode, accumulation of CRC-6 errors, framing bit errors, OOF and line code violations is provided for performance monitoring.

In E1 mode, accumulation of CRC-4 errors, frame alignment signal (FAS) bit errors, far end block error and line code violations is provided for performance monitoring.

The board index is 0 for the first SmartWORKS board within the platform. The framer index is 0 for the first framer.

The MT_FRAMER_STAT structure is defined where the

MTGetT1FramerStatistic() and the **MTGetE1FramerStatistic()** APIs are explained.

When **MTClearFramerStatistic()** is called on the SmartWORKS NGX, the MT_NTTE_FRAMER_STAT structure is used. This structure is defined where the **MTGetNGXFramerStatistic()** is explained. Note that only the following elements are cleared: NTErr, TEErr, and SyncErr.

Prototype

MT_RESULT MTClearFramerStatistic(

```
const BOARD      nboard,  
const USHORT     nFramer  
);
```

Parameters

nboard	board index
nFramer	framer index

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_PARAM	invalid argument passed to function
MT_RET_INVALID_BOARD	Board number is out of range
MT_RET_BOARD_NOT_OPENED	Board device not opened for access
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_API_THREAD_PROTECTED	Another thread is currently accessing this API

Events

None

FRAMER MODE AND ABCD SIGNALING FUNCTIONS

The following section defines APIs which monitor/control board framers.

MTSetFramerLoopbackMode()

Description

The **MTSetFramerLoopbackMode()** function will place the designated channel in one of several different data loopback modes. This application is used to test framer operation on any digital network (ISDN, E1, T1). *This function is designed only for synchronous operation on DT channels.*

The desired mode may be selected by passing a valid mode ID in the *LoopbackMode* parameter. Valid modes are defined as:

FRAMER_LOOPBACK_NONE - framer is not in loopback mode

FRAMER_NETWORK_LINE_LOOPBACK – Places the framer associated with the channel in loopback mode between the framer and the central office. This is useful for testing the digital trunk.

FRAMER_NETWORK_PAYLOAD_LOOPBACK - This is useful for testing framer clock operation.

The data read out of RX-FIFO is timed to the transmitter clock, and the transmit frame alignment indication is used to synchronize the output frame alignment. The transmit frame alignment is arbitrary. Note that because the transmit and receive streams are not superframe aligned, any robbed-bit signaling in the receive stream will not fall in the correct frame once looped back and that transmit robbed-bit signaling will overwrite the looped back data if signaling insertion is enabled.

FRAMER_SYSTEM_DIGITAL_LOOPBACK – Places the framer associated with the channel in loopback mode between the framer and the on-board voice resource. This is useful for testing framer operation with the board's DSP resources.

Prototype

MT_RESULT MTSetFramerLoopbackMode(

```
const BOARD      nBoard,
const USHORT     nFramer,
const USHORT     LoopbackMode
);
```

Parameters

nBoard	Board index number
nFramer	Framer number
LoopbackMode	Valid mode Id. See description above.

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_INVALID_BOARD	Board number is out of range
MT_RET_BOARD_NOT_OPENED	Board not opened for access
MT_RET_NOT_AVAILABLE	Driver is not loaded

Events

None

MTGetFramerLoopbackMode()

Description

The **MTGetFramerLoopbackMode()** function retrieves the current framer loopback mode. *This function is designed only for synchronous operation on DT channels.*

Valid modes are defined as:

FRAMER_LOOPBACK_NONE - framer is not in loopback mode

FRAMER_NETWORK_LINE_LOOPBACK – Places the framer associated with the channel in loopback mode between the framer and the central office. This is useful for testing the digital trunk.

FRAMER_NETWORK_PAYLOAD_LOOPBACK - This is useful for testing framer clock operation. The data read out of RX-FIFO is timed to the transmitter clock, and the transmit frame alignment indication is used to synchronize the output frame alignment. The transmit frame alignment is arbitrary. Note that because the transmit and receive streams are not superframe aligned, any robbed-bit signaling in the receive stream will not fall in the correct frame once looped back and that transmit robbed-bit signaling will overwrite the looped back data if signaling insertion is enabled.

FRAMER_SYSTEM_DIGITAL_LOOPBACK – Places the framer associated with the channel in loopback mode between the framer and the on-board voice resource. This is useful for testing framer operation with the board's DSP resources.

Prototype

MT_RESULT MTGetFramerLoopbackMode(

```
const BOARD      nBoard,  
const USHORT     nFramer,  
const pUSHORT    pLoopbackMode  
);
```

Parameters

nBoard	Board index number
nFramer	Framer number
pLoopbackMode	Pointer to the buffer containing the LoopbackMode information

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_INVALID_BOARD	Board number is out of range
MT_RET_BOARD_NOT_OPENED	Board not opened for access
MT_RET_NOT_AVAILABLE	Driver is not loaded

Events

None

MTResetFramerLoopbackMode()

Description

The **MTResetFramerLoopbackMode()** function disables loopback mode on the designated framer (FRAMER_LOOPBACK_NONE). Data already on the Encode and Decode queues will not be flushed by this function. *This function is designed only for synchronous operation on DT channels.*

Prototype

MT_RESULT MTResetFramerLoopbackMode(

```
const BOARD      nBoard,  
const USHORT     nFramer  
);
```

Parameters

nBoard	Board index number
nFramer	Framer number

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_INVALID_BOARD	Board number is out of range
MT_RET_BOARD_NOT_OPENED	Board not opened for access
MT_RET_NOT_AVAILABLE	Driver is not loaded

Events

None

MTEnableFramerABCDSignalingEvent()

Description

This function controls ABCD signaling change through events EVT_INCOMING_SIGCHANGE and EVT_OUTGOING_SIGCHANGE. The actual ABCD bits are kept in the MT_EVENT.SubReason field. The actual timeslot where the signaling change occurred is kept in the MT_EVENT.XtraInfo field. Signaling bits A, B, C, & D are contained as bits 3, 2, 1, and 0 respectively. When T1 with SF framing mode is selected only two bits are used, therefore bits A=C and bits B=D. ABCD signaling change reporting is disabled by default.

Prototype

MT_RESULT MTEnableFramerABCDSignalingEvent(

```
const BOARD      nboard,
const USHORT     nframer
);
```

Parameters

nboard	board index: 0 for first board
nframer	framer index: 0-3 for the DP, and 0-1 for the DT.

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_PARAM	Framer index error, invalid parameter
MT_RET_INVALID_BOARD	invalid board type
MT_RET_BOARD_NOT_OPENED	Board device not opened for access
MT_RET_NOT_AVAILABLE	Driver is not loaded

Events

EVT_INCOMING_SIGCHANGE
EVT_OUTGOING_SIGCHANGE

Scenario

ABCD signaling events are reported as CHANNEL events. Use **MTWaitForChannelEvent()** or **MTSetEventCallback()** to retrieve ABCD signaling events after an ABCD signaling change is reported. See Also: [MTWaitForChannelEvent\(\)](#), [MTSetEventCallback\(\)](#)

CODE EXAMPLE**MTEnableFramerABCDSignalingEvent()**

```
handleMT(MTSetEventCallback(channel, (MTCALLBACK)ChannelEventCallBack, 0), "MTSetEventCallback() channel %d", channel);
```

```
if(handleMT(MTEnableFramerABCDSignalingEvent(board, framer), "MTEnableFramerABCDSignalingEvent() board %d, framer %d", board, framer))
{
    wait = true;
    while(wait)
        Sleep(20);
}
```

```
handleMT(MTClearEventCallback(channel), "MTClearEventCallback() channel %d", channel);
```

```
handleMT(MTSysShutdown(), "MTSysShutdown()");
```

A complete code example is available on the AudioCodes website:

<http://www.audiocodes.com/blades/support.htm>

MTDisableFramerABCDSignalingEvent()

Description

Disables the queuing and reporting of ABCD signaling changes. ABCD signaling change reporting is disabled by default.

Prototype

MT_RESULT MTDisableFramerABCDSignalingEvent(

```
const BOARD      nBoard,  
const USHORT     nframer  
);
```

Parameters

nBoard	board index: 0 for first board
nframer	framer index: 0-3 for the DP, and 0-1 for the DT

Return Codes

MT_RET_OK	No Error
MT_RET_INVALID_PARAM	Board index or framer index error (invalid parameter)
MT_RET_INVALID_BOARD	Invalid board type
MT_RET_BOARD_NOT_OPENED	Board device not opened for access
MT_RET_NOT_AVAILABLE	Driver is not loaded

Events

None

MTGetFramerABCDSignalingEventStatus()

Description

Retrieves the status of the queuing and reporting of ABCD signaling changes.

Prototype

MT_RESULT MTGetFramerABCDSignalingEventStatus (

```
const BOARD      nBoard,  
const USHORT     nFramer,  
PUSHORT const    fEnabled  
);
```

Parameters

nBoard	board index: 0 for first board, -1 for all boards
nFramer	framer index: 0-3 for the DP and 0-1 for the DT
fEnabled	status flag: 1=enabled, 0=disabled

Return Codes

MT_RET_OK	No Error
MT_RET_INVALID_PARAM	Board index or framer index error (invalid parameter)
MT_RET_INVALID_BOARD	Invalid board type
MT_RET_BOARD_NOT_OPENED	Board device not opened for access
MT_RET_NOT_AVAILABLE	Driver is not loaded

Events

None.

MTGetTimeSlotRxSignaling()

Description

MTGetTimeSlotRxSignaling() retrieves the current ABCD signaling information on the specified time slot on the specified framer of the specified board.

The board index is 0 for the first SmartWORKS board within the platform. The framer index is 0 for the first framer.

Prototype

MT_RESULT MTGetTimeSlotRxSignaling() (

```
const BOARD          nBoard,
const USHORT         nFramer,
const ULONG          nTimeSlot,
PMT_TS_SIGNALING const pSignaling
);
```

Parameters

nBoard	board index
nFramer	framer index from 0 to 3 for the DP, and 0-1 for the DT
nTimeSlot	time slot index from 1 to 24 (T1) or 1 to 30 (E1)
pSignaling	pointer to MT_TS_SIGNALING structure

Structure

Structure MT_TS_SIGNALING is defined as follows:

MT_TS_SIGNALING

Type	Name	Purpose
UCHAR	Framer	Framer index
UCHAR	TimeSlot	Time slot index
UCHAR	Signal	Signaling bits: ABCD bits: 0x01 - D bit 0x02 - C bit 0x04 - B bit 0x08 - A bit NOTE: In T1/SF framer mode, bits A=C and bits B=D
//UCHAR	PrevSignal*	Previous signaling
//ULONG	ElapsedTime*	Duration (in 10ms unit) between Signal and PrevSignal

*The data in the fields PrevSignal and Elapsed Time are not passed into this structure. Instead, the information is passed into the MT_EVENT structure when the events

EVT_OUTGOING_SIGCHANGE and EVT_INCOMING_SIGCHANGE are generated.

Return Codes

MT_RET_OK	No Error
MT_RET_INVALID_PARAM	Error in board type, or NULL pointer (invalid parameter)
MT_RET_INVALID_BOARD	Board number is out of range
MT_RET_BOARD_NOT_OPENED	Board device not opened for access
MT_RET_NOT_AVAILABLE	Driver is not loaded

Events

None

CODE EXAMPLE

Code Example MTGetTimeslotRxSignaling()

```
ZeroMemory(&signaling,size);

if(handleMT(MTGetTimeSlotRxSignaling(board,frame,slot,&signaling),"MTGetTimeSlotRxSignaling() board %d, frame %d, slot %d",board,frame,slot))
{
    UCHAR frameNo = signaling.Framer;
    UCHAR slotNo = signaling.TimeSlot;
    UCHAR signal = signaling.Signal;
    cout << "Frame = " << (INT)frameNo << endl
    << "Time slot = " << (INT)slotNo << endl
    << "Signal = " << (INT)signal << endl <<
endl;
}

handleMT(MTSysShutdown(),"MTSysShutdown()");
```

A complete code example is available on the AudioCodes website:

<http://www.audiocodes.com/blades/support.htm>

MTSetTimeSlotTxSignaling()

This API is used with terminate boards only

Description

Sets the transmit signaling information on the specified time slot
MTSetTimeSlotRxSignaling() sets the transmit ABCD signaling information on a specified time slot on a specified framer of a board.

The board index is 0 for the first NTI board within the platform. The framer index is 0 for the first framer. The time slot index is 1 for the first T1 time slot, 0 for the first E1 time slot.

Signal bits for the T1 robbed bit signaling are defined as:

Value	Bit Designation
0x01	D bit
0x02	C bit
0x04	B bit
0x08	A bit

NOTE: In T1 mode with ESF selected, all four bits are transmitted. Valid signal values range from 0x0 to 0xF. In T1 mode with SF (D4) selected, A is the same as C and B is the same as D. There are only four valid signal values: 0x0, 0x5, 0xA and 0xF.

Prototype

MT_RESULT MTSetTimeSlotTxSignaling (

```
const BOARD      nBoard,
const USHORT     nFramer,
const ULONG      nTimeslot,
const USHORT     Signaling
);
```

Parameters

nBoard	board index
nFramer	framer index: 0 to 1 for DT and 0-3 for the DP
nTimeslot	time slot index: 1 to 24 for T1, 0 to 30 for E1(0 and 16 are reserved)
Signaling	the signal to transmit

Return codes

MT_RET_OK	No error
MT_RET_INVALID_PARAM	Bad index or pointer (invalid parameter)
MT_RET_API_THREAD_PROTECTED	Another thread is currently accessing this API
MT_RET_NOT_AVAILABLE	Driver is not loaded

Events

None

CODE EXAMPLE**Code Example MTSetTimeslotTxSignaling()**

```
ZeroMemory(&signaling, size);

if(handleMT(MTGetTimeSlotTxSignaling(board, frame, slot, &signaling), "MTGetTimeSlotTxSignaling() board %d, frame %d, slot %d", board, frame, slot))
{
    UCHAR frameNo = signaling.Framer;
    UCHAR slotNo = signaling.TimeSlot;
    UCHAR signal = signaling.Signal;
    cout << "Frame = " << (INT)frameNo << endl
    << "Time slot = " << (INT)slotNo << endl
    << "Signal = " << (INT)signal << endl <<
endl;
}
```

A complete code example is available on the AudioCodes website:

<http://www.audiocodes.com/blades/support.htm>

MTGetTimeSlotTxSignaling()

This API is used with terminate boards only

Description

MTGetTimeSlotTxSignaling() retrieves the current transmit ABCD signaling information on the specified time slot on the specified framer of the specified board. Board index is 0 for the first NTI board within the platform. Framer index is 0 for the first framer. Time slot index is 1 for the first T1 time slot, 0 for the first E1 time slot.

Prototype

MT_RESULT MTGetTimeSlotTxSignaling(

```
const BOARD          nBoard,
const USHORT         nFramer,
const ULONG          nTimeslot,
PMT_TS_SIGNALING const pSignaling
);
```

Parameters

nBoard	board index
nFramer	framer index: 0 to 3 for DP, 0 to 1 for DT
nTimeslot	time slot index: 1 to 24 for T1, 1 to 30 for E1
pSignaling	pointer to signaling structure

Structure

Structure MT_TS_SIGNALING is defined as follows :

MT_TS_SIGNALING

Type	Name	Purpose
UCHAR	Framer	Framer index
UCHAR	TimeSlot	Time slot index
UCHAR	Signal	Signaling bits: ABCD bits: 0x01 - D bit 0x02 - C bit 0x04 - B bit 0x08 - A bit NOTE: In T1/SF framing mode, bits A=C and bits B=D
UCHAR	PrevSignal	Previous signaling
ULONG	ElapsedTime	Duration (in 10ms unit) between Signal and PrevSignal

AUDIO JACK FUNCTIONS

The following section defines the APIs used to control the Audio Jack.

MTGetAJMap()

Description

MTGetAJMap() locates the specified audio jack and returns the index of the board that the audio jack resides on.

Prototype

MT_RESULT MTGetAJMap(

```
const int          iAudioJack,  
PBOARD const      pBoard  
);
```

Parameters

iAudioJack	audio jack index 0-15, : 0 for the first audio jack
pBoard	Address to take the returned board index

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_PARAM	Bad audio jack index or PBOARD pointer (invalid parameter)
MT_RET_BUSY	On channel busy
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_NOT_AVAILABLE	Driver is not loaded

Events

None

MTAJTalk()

Description

MTAJTalk() sets the audio jack to transmit its output to the specified CT stream and time slot. In MVIP mode, selecting a forward direction uses the first eight (8) MVIP streams for the 256 MVIP time slots. Selecting a reverse direction uses the second eight (8) MVIP streams for the 256 MVIP time slots. The MVIP time slot index is from 0 to 255.

Scenario

The audio jack can talk or listen on either forward or reversed MVIP streams. The VR and PT channels talk on reversed MVIP streams and listen on forward MVIP streams.

For example, for a channel to receive input from an audio jack through MVIP time slot 5, the following is advised:

- 1) Set a route for the channel to listen on MVIP time slot 5
- 2) Set the audio jack to talk on MVIP time slot 5 with a forward direction

Prototype

MT_RESULT MTAJTalk (

```
const int          iAudioJack,
const int          iStream,
const int          iSlot
);
```

Parameters

iAudioJack	audio jack index 0-15, 0 for the first audio jack
iStream	Index of TDM stream, stream direction: under MVIP: 0-7 for forward, 8-15 for reverse. Under H.100: 0 - 31
iSlot	index of TDM slot on a stream; MVIP time slot: 0 to 31, H.100: see description of MTAJListen

Return Codes

MT_RET_OK	No error
MT_RET_MUX_OFF	Multiplexer system was not started
MT_RET_BOARD_NOT_OPENED	Board device not opened for access
MT_RET_INVALID_PARAM	One bad index, direction, or time slot parameters (invalid parameter)
MT_RET_BUSY	CT stream and time slot taken
MT_RET_INVALID_CHANNEL	Audio jack index is out of range
MT_RET_NOT_AVAILABLE	Driver is not loaded

Events

None

MTAJListen()

Description

MTAJListen() sets the audio jack to receive its input from the specified CT bus time slot. When the CT bus type is MVIP, the first eight (8) MVIP streams (0 - 7) are on the forward direction which uses 256 time slots. The next eight (8) streams (8 - 15) are on the reverse direction which also use 256 time slots. The MVIP time slot index is from 0 to 255.

If the CT bus type is H.100, the number of timeslots available per streams 0 - 15 depends on which H.100 stream speed is selected in the SmartControl panel.

At 2048 KHz, 32 timeslots per stream are available.

At 4096 KHz, 64 timeslots per stream are available.

At 8192 KHz, 128 timeslots per stream are available.

Scenario

The audio jack can talk or listen on either forward or reversed MVIP streams. VR and PT channels talk on reversed MVIP streams and listen on forward MVIP streams.

For example, in order for a channel to transmit its output to an audio jack through MVIP time slot 5, the following is advised:

- 1)Set a route for the channel to talk on MVIP time slot 5
- 2)Set the audio jack to listen on MVIP time slot 5 with a reverse direction

Prototype

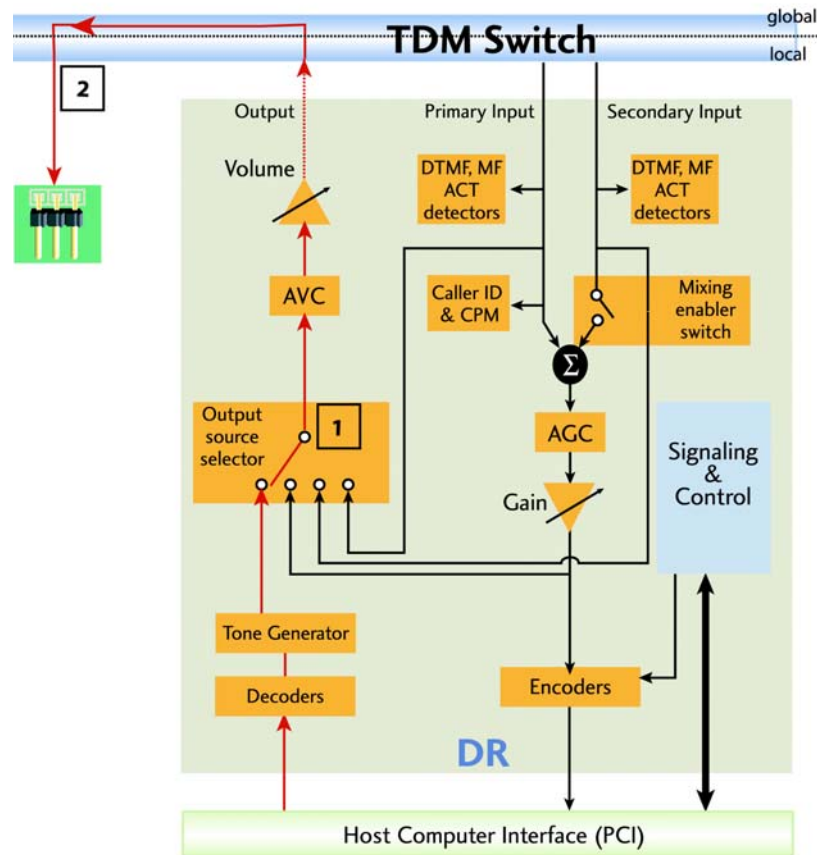
MT_RESULT MTAJListen(

```
const int          iAudioJack,
const int          iStream,
const int          iSlot
);
```

Parameters

iAudioJack	audio jack index 0-15, 0 for the first audio jackIndex
iStream	from code: Index of TDM stream: 0 - 15 for MVIP, 0 - 31 for H.100
iSlot	from code: Index of TDM slot per stream: 0 - 31 for MVIP, H.100: see description above

Playback

**STEP 1:****A. Playback File**

MTSetOutputSource - Set Source and Assign Timeslot for Playback

Source = CHANNEL_DEVICE (playback)

nChannel = GCI index

MTSetOutput - Connect Channel Output to CT Bus

nChannel = GCI index

iStream = stream number

iSlot = time slot number on bus

STEP 2:

Set Audio Jack to Receive from specified CT Bus timeslot number for desired input.

MTAJListen

iAudioJack = audio jack index

iStream = Index of TDM stream

iSlot = Index of TDM slot on a stream

If connection to Audio Jack needs to be broken, use **MTRresetAJListen**

NOTE: The Audio jack can only receive from 1 timeslot at a time. If the connection from the DR to the TDM must be broken (use **MTRresetOutput** or **MTRresetCTRoute** to break connection).

NOTE when using a board with Tone Generation capabilities:

It is important to understand that the channel used for playback is no longer has an output connection to the Network Interface (NI). As a result, tone generation will no longer work. It is recommended to only use this channel for playback.

Return Codes

MT_RET_OK	No error
MT_RET_MUX_OFF	Multiplexer system was not started
MT_RET_BOARD_NOT_OPENED	Board device not opened for access
MT_RET_INVALID_PARAM	One bad index, direction, or time slot parameters (invalid parameter)
MT_RET_BUSY	CT stream and time slot taken
MT_RET_INVALID_CHANNEL	Audio jack index is out of range
MT_RET_NOT_AVAILABLE	Driver is not loaded

Events

None

MTGetAJTalk()

Description

MTGetAJTalk() retrieves the time slot information which a specified audio jack is transmitting onto.

The returned direction code may have the following values:

0	forward direction
1	reverse direction
-1	not connected

The returned time slot index may have the following values:

0 to 255	connected time slot index
-1	not connected

Prototype

MT_RESULT MTGetAJTalk(

```
const int          iAudioJack,
int * const        pStream,
int * const        pSlot
);
```

Parameters

iAudioJack	audio jack index: 0 for the first audio jack
pStream	Address for TDM stream index: 0 - 15 for MVIP, -1 for not connected
pSlot	Address for TDM slot index: 0 - 31 for MVIP, -1 for not connected

Return Codes

MT_RET_OK	No error
MT_RET_MUX_OFF	Multiplexer system was not started
MT_RET_BOARD_NOT_OPENED	Board device not opened for access
MT_RET_INVALID_PARAM	One bad index, direction, or time slot parameters (invalid parameter)
MT_RET_BUSY	On channel busy
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_NOT_AVAILABLE	Driver is not loaded

Events

None

MTGetAJListen()

Description

MTGetAJListen() retrieves the time slot information from which the specified audio jack is receiving.

The returned stream may have the following values:

0 - 15	MVIP bus selected
0 - 31	H.100 bus selected

The returned time slot index may have the following values:

0 - 31	time slot index when MVIP is selected
0 - 127	time slot index when H.100 is selected
-1	not connected

Prototype

MT_RESULT MTGetAJListen(

const int	iAudioJack,
int * const	pStream,
int * const	pSlot
);	

Parameters

iAudioJack	audio jack index: 0 for the first audio jack
pStream	Address for TDM stream index: 0 - 15 for MVIP, -1 for not connected
pSlot	Address for TDM slot index: 0 - 31 for MVIP, -1 for not connected

Return Codes

MT_RET_OK	No error
MT_RET_MUX_OFF	Multiplexer system was not started
MT_RET_BOARD_NOT_OPENED	Board device not opened for access
MT_RET_INVALID_PARAM	One bad index, direction, or time slot parameters
MT_RET_BUSY	On channel busy
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_NOT_AVAILABLE	Driver is not loaded

Events

None

MTRResetAJTalk()

Description

MTRResetAJTalk() clears the previously set talk, if any, on the specified audio jack.

Prototype

MT_RESULT MTRResetAJTalk(

```
const int          iAudioJack
);
```

Parameters

iAudioJack	audio jack index 0-15: 0 for the first audio jack

Return Codes

MT_RET_OK	No error
MT_RET_MUX_OFF	Multiplexer system was not started
MT_RET_BOARD_NOT_OPENED	Board device not opened for access
MT_RET_INVALID_PARAM	On bad index (invalid parameter)
MT_RET_BUSY	On channel busy
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_NOT_AVAILABLE	Driver is not loaded

Events

None

MTResetAJListen()

Description

MTResetAJListen() clears the previously set LISTEN, if any, on the audio jack.

Prototype

```
MT_RESULT MTResetAJListen(  
    const int          iAudioJack  
);
```

Parameters

iAudioJack	audio jack index: 0 for the first audio jack

Return Codes

MT_RET_OK	No error
MT_RET_MUX_OFF	Multiplexer system was not started
MT_RET_BOARD_NOT_OPENED	Board device not opened for access
MT_RET_INVALID_PARAM	Bad index, direction pointer, or time slot pointer
MT_RET_BUSY	On channel busy
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_NOT_AVAILABLE	Driver is not loaded

Events

None

EVENT REPORTING FUNCTIONS

The following section describes the SmartWORKS APIs used to obtain/control system, board and channel events.

CHANNEL EVENT CONTROL

MTGetEventFilters()

Description

MTGetEventFilters() gets the current channel line event bit enable and the ring count to report an incoming event. The enable bits are defined in **MTSetEventFilters()** API.

Prototype

MT_RESULT MTGetEventFilters (

```
const CHANNEL      nChannel,
const PUSHORT      pEventEnable,
const PUSHORT      pRings
);
```

Parameters

nChannel	channel number
pEventEnable	pointer to event bit enable
pRings	pointer to number of rings for EVT_RINGS

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_CHANNEL_NOT_OPENED	Occurs when a channel is not yet opened

Events

None

CODE EXAMPLE

MTGetEventFilters() Code Example

```
result = MTGetEventFilters(channel,&events,&rings);
        if(result == MT_RET_OK)
        {
            printf("MTGetEventFilters()
successful...\n");
            printf("Sum of events enabled: %u\nNo. of
rings: %u\n",events,rings);
        }
```

A complete code example is available on the AudioCodes website:

<http://www.audiocodes.com/blades/support.htm>

MTSetEventFilters()

Description

MTSetEventFilters() sets the filter enable bits used for phone line event monitoring and reporting. These bits are set on a per channel basis. The default setting is 0 - all off.

Prototype

MT_RESULT MTSetEventFilters (

```
const CHANNEL      nChannel,
const PUSHORT      EventEnable,
const PUSHORT      pRings
);
```

Parameters

nChannel	channel number
EventEnable	event bits that have been enabled
Rings	the number of rings used to trigger the event EVT_RINGS

The following bits can be set.

Filter Setting	Bit	Event
SE_LCURRENT_CHANGE	0X0001	EVT_LOOP_DROP, EVT_LOOP_ON (SmartWORKS LD)
SE_RINGS	0X0002	EVT_RINGS (incoming ring detected)
SE_ACT	0X0004	EVT_MON_ACTIVITY, EVT_MAX_ACTIVITY
SE_SIL	0X0008	EVT_MON_SILENCE, EVT_MAX_SILENCE
SE_OFFH	0X0010	EVT_OFFHOOK
SE_ONH	0X0020	EVT_ONHOOK
	0X0100	EVT_REVERSAL
SE_LVOLTAGE_CHANGE	0X0200	EVT_LVOLTAGEABOVEORBELOW, EVT_LVOLTAGEBELOW, EVT_LVOLTAGEABOVE (SmartWORKS LD)

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_CHANNEL_NOT_OPENED	Occurs when a channel is not yet opened

Events

None

CODE EXAMPLE**MTSetEventFilters() Code Example**

```
result = MTSetEventFilters(channel,events,rings);
        if(result == MT_RET_OK)
        {
            printf("MTSetEventFilters()
successful...\n");
            printf("Setting...\nSum of events enabled:
%u\nNo. of rings: %u\n",events,rings);
        }
```

A complete code example is available on the AudioCodes website:

<http://www.audiocodes.com/blades/support.htm>

MTEventControl()

Description

MTEventControl() enables or disables the event queuing operation for all the channels in the system. It is channel independent. The following possible actions are available

TABLE 10: EVENT QUEUING ACTIONS

Action	Value	Usage
MT_G EVT_ENABLE	1	Enable event queue for all channels
MT_G EVT_DISABLE	2	Disable event queue for all channels
MT_G EVT_FLUSHQUEUE	3	Flush event queue for all channels

Prototype

```
MT_RESULT MTEventControl (
    const USHORT          ACTION
);
```

Parameter

ACTION action to perform

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_NOT_AVAILABLE	Driver is not loaded

Events

None

MTFlushEvents()

Description

MTFlushEvents() clears all events queued for a channel.

Prototype

MT_RESULT MTFlushEvents (

 const CHANNEL nChannel
);

Parameters

nChannel	Channel Number

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_CHANNEL_NOT_OPENED	Occurs when a channel is not yet opened
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_NOT_AVAILABLE	Driver is not loaded

Events

None

MTSetEventCallback()

Description

MTSetEventCallback() provides a function pointer that will be called when an event is on the event queue. The function described in MTCALLBACK field is registered for every channel available to the application. The callback function can be declared in the user application as illustrated in the example below:

```
CINTERFACE void MTCALLBACK(GCI, int, PMT_EVENT, LPARAM);
CINTERFACE void EventCallback(
    GCI iCh, int EventCount, int, PMT_EVENT pEvent, LPARAM
    lParam) {}
```

If a user wishes to set all channels with **MTSetEventCallback()**, the GCI parameter must be configured as follows:

If your GCI starting index (configured in Smart Control) is set to 0, set the GCI parameter to -1

If your GCI starting index is set to 1, set the GCI parameter to 0

When a callback function is invoked, the function does not need to use **MTWaitForChannelEvent()** to retrieve the queued events. Events are passed in the array pointed to by the PMT_EVENT parameter. User applications should not release the memory for the event array.

Scenario

MTClearEventCallback() works as a synchronous function. Therefore, once callback is initiated, it does not release its lock on the user application until a response has been received. Invoking **MTClearEventCallback()** from a callback function would enter this control path into a deadlock. To prevent this the return MT_RET_API_THREAD_PROTECTED message is generated.

Prototype

MT_RESULT MTSetEventCallback (

```
const CHANNEL      nChannel,
MTCALLBACK const   Callback,
const LPARAM       lParam
);
```

Parameters

nChannel	channel number
Callback	pointer to callback function
lParam	user parameter to be returned on call-back

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_CHANNEL	Channel number is out of range

MT_RET_CHANNEL_NOT_OPENED

Channel device not opened for
access

MT_RET_INVALID_PARAM

Invalid argument passed to
function

MT_RET_NOT_AVAILABLE

Driver is not loaded

Events

None

MTClearEventCallback()

Description

MTClearEventCallback() clears the callback function that was previously registered for the specified channel(s).

Scenario

Event callback cannot be cleared by registering a NULL callback function. Use this API function to clear the registered callback function.

This API works as a synchronous function. Therefore, once callback is initiated, it does not release its lock on the user application until a response has been received. Invoking **MTClearEventCallback()** from a callback function would enter this control path into a deadlock. To prevent this the return MT_RET_API_THREAD_PROTECTED message is generated.

Prototype

MT_RESULT MTClearEventCallback (

```
const channel      nChannel
);
```

Parameter

nChannel	channel number

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_CHANNEL_NOT_OPENED	Occurs when a channel is not yet opened
MT_RET_API_THREAD_PROTECTED	this API has called a function that is already locked by another application

Events

None

MTSetPriorityEventCallback()

Description

MTSetPriorityEventCallback() provides a function point that will be called when the specified event occurs on the specified channel. All pending events, including the specified trigger event, will be passed to the user application through a callback function parameter. The callback function can be declared or defined in the user application as illustrated below. If set through **MTSetEventCallback()** a general event callback function for a channel and then set through **MTSetPriorityEventCallback()** a priority event callback function for the same channel, then when the specified channel receives the event specified through **MTSetPriorityEventCallback()** function, the priority event callback function will be invoked. Other events will invoke the general event callback function.

```
CINTERFACE void MTCALLBACK(GCI, int, PMT_EVENT, LPARAM);
CINTERFACE void PriorityEventCallback(
GCI iCh, int EventCount, int, PMT_EVENT pEvent, LPARAM
lParam) {}
```

Parameter iCh refers to the channel on which the event(s) are queued; parameter EventCount refers to the number of events pointed to by the MT_EVENT parameter. Parameter PMT_EVENT points to information of EventCount event(s) in the array of structure type MT_EVENT. These events are no longer in the channel queue for later retrieval. The event specified by EventCode is the last event in the event array, since it was the last event to be placed in the event queue.

When called, the priority callback function should process all events pointed to by an event (i.e. PMT_EVENT).

To process more than one event as a priority, this API must be called once for each event.

Prototype

MT_RESULT MTSetPriorityEventCallback (

```
const CHANNEL      nChannel,
const LONG         EventCode,
MTCALLBACK const   Callback,
const LPARAM       lParam
);
```

Parameters

nChannel	channel number
EventCode	event code that will invoke callback
Callback	pointer to callback function
lParam	User event to be returned on callback

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_PARAM	Invalid argument passed to function of NULL pointer, event code on GCI number
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_BOARD_NOT_OPENED	Board device not opened for access

Events

Currently the SmartWORKS SDK only registers one priority callback for each channel, hence, only one event code is monitored.

CODE EXAMPLE

MTSetPriorityEventCallback() Code Example

```
result =
MTSetPriorityEventCallback(channel, EVT_CALLID_STOP, (MTCALLBACK) PriorityEventCa
llback, myParam);

        if(result == MT_RET_OK)
        {
            printf("MTSetPriorityEventCallback()
successfully registered on all %d channels...\n", totalChannels);
        }
```

A complete code example is available on the AudioCodes website:

<http://www.audiocodes.com/blades/support.htm>

MTClearPriorityEventCallback()

Description

MTClearPriorityEventCallback() clears the callback function that was previously registered for the specified channel(s) for the specified event. Currently, event code is not checked.

To clear more than one event, this API must be called once for each event.

Scenario

Priority callback cannot be cleared by registering a NULL callback function. Use this API function to clear the registered callback function.

Prototype

MT_RESULT MTClearPriorityEventCallback(

```
const CHANNEL      nChannel,  
const LONG         EventCode  
);
```

Parameters

nChannel	channel number
EventCode	event code that will invoke callback

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_CHANNEL_NOT_OPENED	Occurs when a channel is not yet opened
MT_RET_API_THREAD_PROTECTED	Another thread is currently accessing this API

Events

None

MTGetChannelEvent

This API was formerly called **MTGetEvent()**.

Description

MTGetChannelEvent() retrieves the first event out of the event queues for all the channels. Each channel has its own event queue. This API will return whenever it finds any event for any channel. Should there be events on both channel 1 and channel 10 at the same time, the event on channel 1 is returned. Should there be no event in the system, this function awaits the arrival of the next new event on any channel or the expiration of the specified waiting period.

The event structure is defined as:

LARGE_INTEGER	TimeStamp
ULONG	UserStatus
ULONG	EventCode
ULONG	SubReason
ULONG	XtraInfo
ULONG	Board
ULONG	Channel
ULONG	PtnBuffer
PVOID	DataLength

The [Event Codes](#) structure is explained in the next chapter.

Prototype

MT_RESULT MTGetChannelEvent (

```
const DWORD          dwTimeoutMs,
PMT_EVENT const      pMTEvent,
PULONG const         pLength
);
```

Parameters

dwTimeoutMs	Maximum waiting time in (ms), this value must be at least 20 ms, -1 equals no timeout
pMTEvent	pointer to event buffer
pLength	Set by the user application, this is a pointer to the size of the given buffer for the MT_EVENT structure.

Return Codes

MT_RET_OK	No error
MT_TIMEOUT	Wait for event timed out

MT_RET_DATA_TRUNCATED	Data copied but truncated; the buffer size set by the user application is not large enough for this API request. <i>FlagDataTruncated</i> field of the MTSYS_CONFIG or the MTSYS_CONFIGURATION data structure must be enabled (1) to receive this return code, otherwise MT_RET_OK is used, and the truncated data is passed to the user application
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_NOT_AVAILABLE	Driver is not loaded

Events

None

MTWaitForChannelEvent()

This API was formerly called ***MTWaitForEvent()***

Description

Checks channel event queue and returns the next event on the channel queue before the specified timeout. This function is typically used in a "message pump" thread to periodically check on the status of an active channel. A timeout of -1 can be used when it's necessary to wait until an event occurs.

Scenario

Currently, the SDK has one thread per all callbacks and one thread per all completion callbacks. This is ok for a single processor host. However, when the SDK is applied to a multiple processor host, user application can take advantage of polling and implement its own callback per thread per resource. This would dramatically increase the operating system time slices for all callbacks.

Developer's Note

The *pLength* parameter is defined as a pointer to a long word. When called, the user sets the size of the buffer they provide, when returned, the SDK puts in the size of valid data it retrieved. Therefore, when TIMEOUT, the "size" will be set to zero to indicate that SDK retrieved no valid data. This is why the *pLength* parameter is defined as a pointer to a long word so that both sides can read/write into it.

Prototype

MT_RESULT MTWaitForChannelEvent (

```
const CHANNEL      nChannel,
const DWORD        dwTimeoutMs,
PMT_EVENT const    pMTEvent,
PULONG const       pLength
);
```

Parameters

nChannel	channel number
dwTimeoutMs	Maximum waiting time in (ms), this value must be at least 20 ms, -1 equals no timeout.
pMTEvent	pointer to MT_EVENT structure
pLength	Set by the user application, this is a pointer to the size of the given buffer for the MT_EVENT structure.

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_INVALID_CHANNEL	Channel number is out of range

MT_RET_CHANNEL_NOT_OPENED	Channel device not opened for access
MT_TIMEOUT	Wait for event timed out
MT_RET_DATA_TRUNCATED	Data copied but truncated; the buffer size set by the user application is not large enough for this API request. <i>FlagDataTruncated</i> field of the MTSYS_CONFIG or the MTSYS_CONFIGURATION data structure must be enabled (1) to receive this return code, otherwise MT_RET_OK is used, and the truncated data is passed to the user application
MT_RET_NOT_AVAILABLE	Driver is not loaded

Events

None

MTPutChannelEvent()

This API was formerly called **MTPutEvent()**

Description

MTPutChannelEvent() places an application specified event into the event queue of the specified channel.

Scenario

If an event callback function is registered for the specified channel, the callback function will be invoked with the newly added event. Otherwise, this event can be retrieved through polling using **MTWaitForChannelEvent()** or **MTGetChannelEvent()**.

The [Event Codes](#) structure is explained in the next chapter.

Prototype

MT_RESULT MTPutChannelEvent(

```
const CHANNEL      nChannel,
PMT_EVENT const    pMTEvent,
PULONG const       pLength
);
```

Parameters

nChannel	channel number
pMTEvent	pointer to a event buffer
pLength	Set by the user application, this is a pointer to the size of the given buffer for the MT_EVENT structure.

Return Codes

MT_RET_OK	No error
MT_RET_QFULL	On event queue full
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_CHANNEL_NOT_OPENED	Channel device not opened for access
MT_RET_NOT_AVAILABLE	Driver is not loaded

Events

None

BOARD EVENT CONTROL

MTPutBoardEvent()

Description

Puts an event to the board.

Prototype

MT_RESULT MTPutBoardEvent (

```
const BOARD      nBoard,  
PMT_EVENT const  pMTEvent,  
PULONG const     pLength  
);
```

Parameters

nBoard	Board Number
pMTEvent	pointer to the PMT_EVENT structure
pLength	Set by the user application, this is a pointer to the size of the given buffer for the MT_EVENT structure.

Return Codes

MT_RET_BOARD_NOT_OPENED	Board not opened for access

Events

None

MTSetBoardEventCallback()

Description

MTSetBoardEventCallback() provides a function pointer that will be called when an event is on the board event queue. When the BOARD parameter is equal to -1, the function described in MTCALLBACK field is registered for every board available to the application. The callback function can be declared in the user application as illustrated in the example below:

```
CINTERFACE void MTCALLBACK(USHORT, int, PMT_EVENT, LPARAM);
CINTERFACE void EventCallback(
    BOARD iBrd, int EventCount, int, PMT_EVENT pEvent,
    LPARAM lParam) {}
```

Parameter *iBrd* refers to the board on which the event(s) are queued; parameter *EventCount* refers to the number of events returned with this callback. Parameter *pEvent* points to the event array. Parameter *lParam* is passed as specified in function **MTSetBoardEventCallback()**.

When a callback function is invoked, the function does not need to use **MTWaitForAdapterEvent()** to retrieve the queued events. Events are passed in the array pointed to by the PMT_EVENT parameter. User applications should not release the memory for the event array.

Scenario

MTSetBoardEventCallback() behaves the same as **MTSetEventCallback()**.

MTClearBoardEventCallback() works as a synchronous function. Therefore, once callback is initiated, it does not release its lock on the user application until a response has been received. Invoking **MTClearBoardEventCallback()** from a callback function would enter this control path into a deadlock. To prevent this the return MT_RET_API_THREAD_PROTECTED message is generated.

Prototype

MT_RESULT MTSetBoardEventCallback (

```
const BOARD      nBoard,
MTCALLBACK const Callback,
const LPARAM     lParam
);
```

Parameters

nBoard	board number, 0 for the 1st board, -1 for all boards
Callback	pointer to callback function
lParam	user parameter to be returned on call-back

Return Codes

MT_RET_OK	No error
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_INVALID_BOARD	Board number is out of range
MT_RET_BOARD_NOT_OPENED	Board device not opened for access

Events

None

MTClearBoardEventCallback()

Description

This function disables the board event callback previously set up through the **MTSetBoardEventCallback()** function. Board events will have to be polled through **MTWaitForAdapterEvent()**.

This API works as a synchronous function. Therefore, once callback is initiated, it does not release it's lock on the user application until a response has been received. Invoking **MTClearBoardEventCallback()** from a callback function would enter this control path into a deadlock. To prevent this the return **MT_RET_API_THREAD_PROTECTED** message is generated.

Prototype

MT_RESULT MTClearBoardEventCallback(

```
const BOARD      nBoard
);
```

Parameters

nBoard	board index: 0 for the first board, -1 for all boards

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_BOARD	Board number is out of range
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_BOARD_NOT_OPENED	Board device not opened for access
MT_RET_API_THREAD_PROTECTED	Another thread is currently accessing this API

Events

None

MTWaitForAdapterEvent()

The API was formerly called **MTWaitForBoardEvent()**

Description

Checks board event queue and returns the next event on the queue. This function is typically used in a "message pump" thread to periodically check on the status of an active board.

The event structure MT_EVENT is defined as:

LARGE_INTEGER	TimeStamp
ULONG	EventCode
ULONG	SubReason
ULONG	XtraInfo
ULONG	FuncCode
SHORT	Board
SHORT	Channel
PVOID	PtnBuffer
ULONG	DataLength

Scenario

Currently, the SDK has one thread per all callbacks and one thread per all completion callbacks. This is ok for a single processor host. However, when the SDK is applied to a multiple processor host, user application can take advantage of polling and implement its own callback per thread per resource. This would dramatically increase the operating system time slices for all callbacks.

Prototype

MT_RESULT MTWaitForAdapterEvent (

```
const BOARD      nBoard,
const DWORD      dwTimeoutMs,
PMT_EVENT const  pMTEvent,
PULONG const     pLength
);
```

Parameters

nBoard	board index, 0 for the 1st board
dwTimeoutMs	Maximum waiting time in (ms), this value must be at least 20 ms, -1 equals no timeout
pMTEvent	pointer to MT_EVENT structure
pLength	Set by the user application, this is a pointer to the size of the given buffer for the MT_EVENT structure.

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_INVALID_BOARD	Board number is out of range

MT_RET_BOARD_NOT_OPENED	Board device not opened for access
MT_TIMEOUT	Wait for event timed out
MT_RET_DATA_TRUNCATED	Data copied but truncated; the buffer size set by the user application is not large enough for this API request. <i>FlagDataTruncated</i> field of the MTSYS_CONFIG or the MTSYS_CONFIGURATION data structure must be enabled (1) to receive this return code, otherwise MT_RET_OK is used, and the truncated data is passed to the user application
MT_RET_NOT_AVAILABLE	Driver is not loaded

Events

None

See Also: *MTWaitForChannelEvent()*

SYSTEM EVENT CONTROL

MTSysSetEventCallback()

Description

MTSysSetEventCallback() provides a function pointer that will be called when an event is on the system event queue.

When a callback function is invoked, the function does not need to use **MTSysWaitForEvent()** to retrieve the queued events. Events are passed in the array pointed to by the MTCALLBACK parameter. User applications should not release the memory for the event array.

MTCALLBACK and LPARAM provide the DLL the address of the API completion routine and the user expected parameter for invoking the completion routine

This API is only used to retrieve system events, such as events generated when using **MTResyncTime()**.

MTSysClearEventCallback() works as a synchronous function. Therefore, once callback is initiated, it does not release it's lock on the user application until a response has been received. Invoking **MTSysClearEventCallback()** from a callback function would enter this control path into a deadlock. To prevent this the return MT_RET_API_THREAD_PROTECTED message is generated.

Prototype

MT_RESULT MTSysSetEventCallback (

MTCALLBACK	pCallback,
LPARAM	lParam
);	

Parameter

pCallback	used as resource indexing, set to -1 to indicate the resource as SYSTEM.
lParam	user defined parameter

Return Codes

MT_RET_OK	No error

Events

None

MTSysClearEventCallback()

Description

MTSysClearEventCallback() clears the callback function that was previously registered for the system.

This API is used when using the callback function for retrieving system events such as those generated by the **MTSyncTime()** API.

This API works as a synchronous function. Therefore, once callback is initiated, it does not release its lock on the user application until a response has been received. Invoking **MTSysClearEventCallback()** from a callback function would enter this control path into a deadlock. To prevent this the return **MT_RET_API_THREAD_PROTECTED** message is generated.

Prototype

```
MT_RESULT MTSysClearEventCallback (  
                                VOID  
);
```

Parameter

no parameter

Return Codes

MT_RET_OK	No error
MT_RET_API_THREAD_PROTECTED	Another thread is currently accessing this API

Events

None

MTSysWaitForEvent()

Description

Checks system event queue and returns the next event on the queue before the specified timeout. This function is typically used in a "message pump" thread to periodically check on the status. A timeout of -1 can be used when it's necessary to wait until an event occurs.

Scenario

Currently, the SDK has one thread per all callbacks and one thread per all completion callbacks. This is ok for a single processor host. However, when the SDK is applied to a multiple processor host, user application can take advantage of polling and implement its own callback per thread per resource. This would dramatically increase the operating system time slices for all callbacks.

Prototype

MT_RESULT MTSysWaitForEvent (

```
const DWORD          dwTimeoutMs,  
PMT_EVENT const     pMTEvent,  
PULONG const        pLength  
);
```

Parameters

dwTimeoutMs	Maximum waiting time in (ms), this value must be at least 20 ms, -1 equals no timeout.
pMTEvent	pointer to MT_EVENT structure
pLength	Set by the user application, this is a pointer to the size of the given buffer for the MT_EVENT structure.

Return Codes

MT_RET_OK	No error
MT_TIMEOUT	Wait for event timed out

Events

None

SYSTEM FUNCTIONS

These APIs provide the developer with a method of setting or checking registry, or the NtiDrv.conf file when running Linux, parameters without using the control panel applet (SmartControl). These APIs can also be used as configuration check/prevention.

NOTE - The NTI driver and DLL rely on system and adapter level parameters at load time to configure their behavior. When the configuration is changed, both the driver and the DLL must be reloaded for the new configuration to take effect.

SYSTEM START UP FUNCTIONS

MTSmartWorksLib_Init()

Description

This API is one of two developed to manage board restart during the DLL shutdown operation. This function requires a manual setting to enable the dynamic configuration mode allowing program control of board configuration and reset. (use of Windows Device Manager is no longer required after board configuration settings have been made).

Activating the Dynamic Initialization feature is controlled by the following registry key:

Key name = LibInitStyle

Key path = HKLM/SYSTEM/CurrentControlSet/Services/Ntidrv/

Parameters

Key type = DWORD

Prototype

int MTSmartWorksLib_Init (

VOID

);

Return Codes

MT_RET_OK

No error

MT_RET_INTERNAL_ERROR

Failure

MT_RET_UNAVAILABLE

Feature not supported
(dynamic mode not enabled)

MTSmartWorksLib_Close()

Description

This API is one of two developed to manage board restart during the DLL shutdown operation. When Dynamic Initialization mode is enabled, this function restarts the board and applies new board configuration settings, when "Warm Start" mode is enabled.

Works in conjunction with *MTSmartWorksLib_Init()* function has been invoked.

Dynamic Initialization feature is controlled by a registry key - See description of *MTSmartWorkLib_Init()* for more information.

Prototype

int MTSmartWorksLib_Close(

```
const BOOL                fWarmStart  
);
```

Parameters

```
fWarmStart                0 = disable, 1 = 1 Enabled, when warm  
                           start is enabled, board is restarted with  
                           new configuration settings applied  
);
```

Return Codes

MT_RET_OK	No error
MT_RET_INTERNAL_ERROR	Failure
MT_RET_UNAVAILABLE	Feature not supported (dynamic mode not enabled)

SYSTEM CONFIGURATION FUNCTIONS

MTSysSetConfig()

This API replaces **MTSetSystemConfig()** as of release 3.6.1. New application development should use this API.

Description

Sets system configuration into the Windows registry or the NtiDrv.conf file when running Linux.

The NTI driver and DLL rely on system and adapter level parameters at load time to configure their behavior. When the configuration is changed, both the driver and the DLL must be reloaded for the new configuration to take effect.

Prototype

int MTSysSetConfig (

```
const MTSYS_CONFIGURATION const    pConfig,
PULONG const                       pLength
);
```

Parameters

Config	pointer to the system config structure
pLength	Set by the user application, this is a pointer to the size of the given buffer for the MTSYS_CONFIGURATION structure

Structure

The MTSYS_CONFIGURATION structure is defined as follows:

TABLE 11: MTSYS_CONFIGURATION - GENERAL SYSTEM PARAMETERS

Type	Name	Description
ULONG	SDKCompiledWith	SDK version the application is compiled with, 0x02070000, i.e. ver 2.7.0.0
ULONG	APILogCount	Max. number of event-log entry allowed, set to 100 during installation, 0 for no limitation
ULONG	SDKLogLevel	[not supported yet]
ULONG	APITraceLevel	[not supported yet] API_TRACE_ERROR_RET 0x00000001, Trace return code != (MT_RET_OK or MT_RET_IO_PENDING) API_TRACE_CALLBACK 0x00000002, Trace callback only API_TRACE_ALLAPI 0x00000004, Trace all API calls API_TRACE_ALL 0xFFFFFFFF, trace on
ULONG	SDKChannel	Max. number of channel allowed, set to MAX_CH_DEVICES when SDK is installed

TABLE 11: MTSYS_CONFIGURATION - GENERAL SYSTEM PARAMETERS

Type	Name	Description
UCHAR	GCIStartingIndex	0 or 1, set to 0 when SDK is installed
UCHAR	HaltOnStartupError	[0 or 1, not supported yet]
UCHAR	FlagDataTruncated	When set to enable bit, SmartWORKS DLL returns MT_RET_DATA_TRUNCATED, otherwise the DLL returns MT_RET_OK and passes over truncated data (set to 0, disabled, when SDK is installed)
UCHAR	SteppedRun	[Not implemented yet]
UCHAR	SysRes[4]	Reserved

TABLE 12: MTSYS_CONFIG - CT BUS PARAMETERS

Type	Name	Description
UCHAR	H100StreamSpeed	H100 stream speed of CT_2MHz, CT_4MHz, or CT_8MHz
UCHAR	BusSegmentation	0 for no bus segmentation supported
USHORT	MVIPStartSlot	Starting MVIP timeslot: 0 to 511
USHORT	MVIPTotalSlot	Max. number of MVIP slot allowed: 0 to 512
UCHAR	SysRes2[2]	Reserved.

Return Codes

All parameters will be range checked before updating the registry or .conf file. This API returns 0 on success. Otherwise, the API returns the index of the error field of structure MTSYS_CONFIGURATION.

MTSysGetConfig()

This API replaces the **MTGetSystemConfig()** function (v3.6.1). New application development should use this API.

Description

Use this API to get system configuration from the registry or the NtiDrv.conf file when running Linux.

For registry values that are not configured, the default values will be returned.

Prototype

int MTSysGetConfig (

```
PMTSYS_CONFIGURATION const    pConfig,  
PULONG const                  pLength  
);
```

Parameters

pConfig	pointer to the system config structure
pLength	Set by the user application, this is a pointer to the size of the given buffer for the MTSYS_CONFIG structure

Return Codes

MT_RET_OK	No error
MT_RET_TIMEOUT	Wait for event timed out
MT_RET_DATA_TRUNCATED	Data copied but truncated; the buffer size set by the user application is not large enough for this API request. <i>FlagDataTruncated</i> field of the MTSYS_CONFIG or the MTSYS_CONFIGURATION data structure must be enabled (1) to receive this return code, otherwise MT_RET_OK is used, and the truncated data is passed to the user application
MT_RET_NOT_ENOUGH_DATA	The amount of collected data does not satisfy request with timeout set to 0

SYSTEM INFORMATION FUNCTIONS

MTGetSystemInfo()

This API was formerly called **MTGetSysInfo()**

Description

MTGetSystemInfo() fills the provided MTSYS_INFO structure with information about the system status, number of boards, number of channels, and the MUX operation information of the system.

Prototype

MT_RESULT MTGetSystemInfo (

```
PMTSYS_INFO const      pInfo,
PULONG const           pLength
);
```

Parameters

pInfo	pointer to MTSYS_INFO structure to store information
pLength	Set by the user application, this is a pointer to the size of the given buffer for the MTSYS_INFO structure

Structure.

TABLE 13: MTSYS_INFO STRUCTURE

Type	Name3	Purpose
ULONG	Status	System status: 1 = SYSTEM_INITIALIZED 2 = SYSTEM_STARTED 3 = SYSTEM_SHUTDOWN
UCHAR	res	Number of summation functions in the system
UCHAR	NumAudios	Number of Audio Jacks in system
USHORT	NumBoards	Number of boards in the system
USHORT	NumChans	Number of channels in the system
USHORT	StartChan	Starting internal MUX channel
UCHAR	StartBoard	Starting internal MUX board
UCHAR	MuxBoards	Total number of MUX boards
USHORT	MuxSystem	Type of MUX system: 1 = MUX_NONE, Board with NO MUX found 2 = MUX_H100, Board with H100 found 4 = MUX_MVIP, Board with MVIP found
USHORT	MuxMVIPSlot	Starting MVIP slot
USHORT	MuxMVIPTotalSlot	Total MVIP slots
USHORT	MuxChans	Total number of MUX channels
UCHAR	MuxGroups	Total number of MUX groups (this field is not used)
UCHAR	GCIStartingIndex	GCI Starting index: 0 or 1

TABLE 13: MTSYS_INFO STRUCTURE (CONTINUED)

Type	Name3	Purpose
CHAR	H100StreamSpeed	H.100 Data Rate for 1st 16 streams: 0 = CT_2MH, CT stream speed of 2 MHz 1 = CT_4MHz, CT stream speed of 4 MHz 2 = CT_8MH, CT stream speed of 8 MHz
CHAR	MVIPStreamSpeed	MVIP Data Rate for the 1st 16 streams: 0
CHAR	CTBusSegmented	CT Bus segmentation: 0 for not allowed
CHAR	NumMVIPMasterBrd	Number of MVIP master board
CHAR	NumH100MasterABrd	Number of H.100 master A board
CHAR	NumH100MasterBBrd	Number of H.100 master B board
CHAR	NumSec8kNetrefBrd	Number of Sec8k/Netref clock sourcing board
SHORT	NumDILoaded	The NtiDrv DLL currently loaded
USHORT	NumSummations	The number of summation functions in the system. This field is currently not used.

Table Key:

Status	Current system status of the following: SYSTEM_INITIALIZED SYSTEM_STARTED SYSTEM_SHUTDOWN
DigiRate	The current digitizing rate in bits per second.
NumAudios	Number of audio jacks.
NumBoards	The number of boards in the local system.
NumChans	The number of channels in the local system.
Mode	The mode, always 1 for enhanced mode.
StartChan	The starting channel of the local MUX system in a multiple-PC global system.
StartBoard	The starting board number of the local MUX system in a multiple-PC global system.
MuxBoards	The number of boards in the global system.
MuxSystem	The type of MUX system. It can contain the values of MUX_NONE or MUX_MVIP.
MuxMVIPSlot	Starting MVIP slot assigned.
MuxMVIPTotalSlot	Total MVIP slots assigned
MuxGroups	The number of groups that are available to the group conference functions can be found in this parameter. (this field is not used).

MuxChans	The number of channels in the global system.
GCIStartingIndex	GCI Starting index: 0 or 1
H100StreamSpeed	H.100 Data Rate for first 16 streams: 0, 1 or 2
MVIPStreamSpeed	MVIP Data Rate for the first 16 streams: 0
CTBusSegmented	CT Bus segmentation: 0 for not allowed
NumMVIPMasterBrd	MVIP master board number
NumH100MasterABrd	H.100 master A board number
NumH100MasterBBrd	H.100 master B board number
NumSec8kNetrefBrd	Sec8k/Netref clock sourcing board number

Return Codes

MT_RET_OK	no error
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_DATA_TRUNCATED	Data copied but truncated; the buffer size set by the user application is not large enough for this API request. <i>FlagDataTruncated</i> field of the MTSYS_CONFIG or the MTSYS_CONFIGURATION data structure must be enabled (1) to receive this return code, otherwise MT_RET_OK is used, and the truncated data is passed to the user application

Events

None

CODE EXAMPLE

MTGetSystemInfo() Code Example

```
ZeroMemory(&sysinfo, size);

result = MTGetSystemInfo(&sysinfo, &size);
if(result == MT_RET_OK)
{
    printf("MTGetSystemInfo() successful...\n");
    USHORT totalBoards = sysinfo.NumBoards;
    USHORT totalChannels = sysinfo.NumChans;
    UCHAR gciStart = sysinfo.GCIStartingIndex;
    printf("Total number of boards: %u\nTotal
number of channels: %u\nGCI: %d\n", totalBoards, totalChannels, gciStart);
}
```

A complete code example is available on the AudioCodes website:

<http://www.audiocodes.com/blades/support.htm>

See also: [MTGetAdapterInfo\(\)](#)

MTGetVersion()

Description

MTGetVersion() returns the version of the SmartWORKS driver and on board firmware. The versions are for the first board. Use **MTGetAdapterInfo()** to get the versions for each board. The version information is defined as MT_VERSION. The SmartWORKS software version number is composed of two 32-bit version long integers.

Prototype

MT_RESULT MTGetVersion (

```
PMT_VERSION const      pDriverVersion,
PMT_VERSION const      pFWVersion
);
```

Parameters

pDriverVersion	stores version of the host driver
pFWVersion	stores firmware version of the first board in target platform

Structure.

TABLE 14: MT_VERSION

Type	Name	Function
UCHAR	Major	Major version number
UCHAR	Minor	Minor version number
USHORT	Internal	Internal number for special purposes such as debugging, etc. For a formal release, this number should be 0.
USHORT	Build	Build number
UCHAR	Release	Release code, 'A' for Alpha, 'B' for Beta, ' ' (one blank space) for release
UCHAR	Feature	Feature code: 'C' for Check build, 'D' for Debug build, ' ' (one blank space) for release build

Return Codes

MT_RET_OK	no error
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_INVALID_PARAM	Invalid argument passed to function.

Events

None

MTSysGetVersion()

Description

This API retrieves the version of SDK components at run time. If a component is missing from the target PC, the version info for it should be all zeros. This applies to the Linux operating system where the MdC, BRI and DCDecode versions are not displayed.

Prototype

MT_RESULT MTSysGetVersion (

```
PMTSDK_VERSION      pSDKVersion,
PULONG              pLength
);
```

Parameters

pSDKVersion	pointer to the structure
pLength	pointer to the length of the structure

Structure.

TABLE 15: MT_VERSION

Name	Function
MT_VERSION	VerKernelDriver
MT_VERSION	VerAgentService
MT_VERSION	VerNtiDrvDll
MT_VERSION	VerMdcDll
MT_VERSION	VerDcDecodeDll
MT_VERSION	VerDBRIDecodeDll

Return Codes

MT_RET_OK	no error
MT_RET_INVALID_PARAM	Invalid argument passed to function. Use MTGetLastError() to check this error.

Events

None

MTGetDLLVersion()

This API will be obsolete as of Sept. 2005. Use **MTSysGetVersion()** instead.

Description

MTGetDLLVersion() is a function that is used to retrieve the version information associated with the NtiDrv.dll file.

Prototype

```
MT_RESULT MTGetDLLVersion (
    PMT_VERSION const    pDLLVersion
);
```

Parameters

pDLLVersion	Pointer to version information buffer

Return Codes

MT_RET_OK	no error
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_INVALID_PARAM	Invalid argument passed to function.

Events

None

CODE EXAMPLE

MTGetDLLVersion() Code Example

```
ZeroMemory(&version, size);

result = MTGetDLLVersion(&version);
if(result == MT_RET_OK)
{
    printf("MTGetDLLVersion() successful...\n");
    UCHAR major = version.Major;
    UCHAR minor = version.Minor;
    USHORT internal = version.Internal;
    USHORT build = version.Build;
    UCHAR release = version.Release;
    UCHAR feature = version.Feature;
    printf("%02d.%02d.%02d\nBuild %d\nRelease %d\nFeature %d\n", major, minor, internal, build, release, feature);
}
```

A complete code example is available on the AudioCodes website:

<http://www.audiocodes.com/blades/support.htm>

SYNC HOST/BOARD TIME

MTResyncTime()

Description

By default, the timestamp is synced when a board/channel is opened for the very first time. Use this API to re-synchronize the time between the host PC and SmartWORKs board without shutting down the NTI application. If the host PC's time is modified this API is called to resync the time so that event timestamps are adjusted.

Two events are generated when synchronization is complete. One event is generated with the old timestamp, and the second is generated with the new timestamp. Both events are system events. These events must be called using the system event callback APIs - **MTSysSetEventCallback()** and **MTSysWaitforEvent()**.

This API resynchronizes the host time with all SmartWORKs board(s) in the platform provided that the application has access through either the board or a channel. In the case of an NGX card, whose board access is excluded from this resynchronization capability, the application requires access of one NGX channel per NGX board.

If access to a SmartWORKs board is not present, SDK assumes that time resynchronization is not required on this board.

If multiple applications are running, such as one application per channel, this API needs to be called from each and every application.

Prototype

```
MT_RESULT MTResyncTime (  
                                VOID  
);
```

Parameters

no parameters are required

Return Codes

MT_RET_OK	no error
MT_RET_NOT_APPLICABLE	the application does not have access to any board or channel resource
MT_RET_TIMEOUT	Wait for event timed out

Events

EVT_SYS_SYNCTIME_OLD
EVT_SYS_SYNCTIME_NEW

SYSTEM FUNCTIONS - OBSOLETE

The following functions are obsolete, but have been maintained in the SDK for backwards compatibility.

BOARD CONTROL AND CONFIGURATION FUNCTIONS

BOARD CONTROL

MTOpenBoard()

Description

MTOpenBoard() opens the specified board for access. The channels will remain closed. This supports up to 64 applications per board.

To open all boards and channels use the function **MTSysStartup()**.

Prototype

MT_RESULT MTOpenBoard(

const BOARD nboard
);

Parameters

nboard board number

Return Codes

MT_RET_OK	No error
MT_RET_NOT_STARTED	On failure (most likely due to the resource having been already grabbed by another application)
MT_RET_INVALID_BOARD	Board number is out of range
MT_RET_MALFUNCTION	Hardware failed during initialization
MT_RET_BOARD_INUSE	On failure (most likely due to the resource having been already grabbed by another application)
MT_RET_DEVICE_NOTFOUND	Voice interface device was not found
MT_RET_INTERNAL_ERROR	Internal error in DLL
MT_RET_NOT_AVAILABLE	Driver is not loaded

Events

None

See Also: [MTSysStartup\(\)](#)

MTCloseBoard()

Description

MTCloseBoard() closes the specified board for further access. Should the user application need to release all resources (boards, channels, TDM timeslots, etc.), it is suggested to do so with **MTSysShutdown()**. Calling of this function is not necessary unless only a specified board is to be closed.

Prototype

MT_RESULT MTCloseBoard(

const BOARD nboard
);

Parameters

nboard board number

Return Codes

MT_RET_OK	No error
MT_RET_NOT_STARTED	On failure (most likely due to the resource having been already grabbed by another application)
MT_RET_INVALID_BOARD	Board number is out of range
MT_RET_BOARD_NOT_OPENED	Board device not opened for access
MT_RET_NOT_AVAILABLE	Driver is not loaded

Events

None

See Also: [MTSysShutdown\(\)](#)

MTGetBoardOpenStatus()

Description

This API returns the ownership of the specified board. A board needs to be opened before any access or control can be performed on it by the application. This function returns a boolean value to the user:

TRUE-Board is opened, i.e. owned by the caller application

FALSE-Board is not opened by the caller application

Scenario

A board can only be opened by one application at one time. When a board is opened and held by one application, attempting to open the same board will fail. However, APIs such as **MTGetAdapterInfo()**, **MTGetBoardOEMInfo()**, **MTGetBoardAssemblyInfo()** still return board information regardless of the board open/close ownership.

Prototype

MT_RESULT MTGetBoardOpenStatus(

```
const BOARD          nboard,  
BOOLEAN * const      pOpen  
);
```

Parameters

nboard	Board Index starting with 0
pOpen	Pointer to the open statu
	Possible values returned
	TRUE-Board is opened, i.e. owned by the caller application
	FALSE-Board is not opened by the caller application

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_BOARD	Board number is out of range
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_NOT_AVAILABLE	Driver is not loaded

Events

None

See Also: [MTGetChannelOpenStatus\(\)](#)

BOARD INFORMATION FUNCTIONS

MTGetAdapterInfo()

This API was previously called *MTGetBoardInfo()*

Description

MTGetAdapterInfo() fills the MTADAPTER_INFO structure with information about the specified board. NOTE: SmartWORKS boards are indexed beginning with 0. The total number of boards in the system can be retrieved through **MTGetSystemInfo()** function.

This function can be called for both local and external boards. The board numbers for global boards can be calculated similarly to the calculation of global channels:

$$\text{local\_board} = \text{global\_board} - \text{MTSYS\_INFO.startBoard.}$$

For external boards, only the channels parameter will contain valid information.

Developer’s Note

A note about the *TDMSignaling* field is required. The driver and the DLL each has its own scheme of identifying whether A-Law or Mu-Law is being used.

The DLL uses:

1 = Mu-Law

2 = A-Law

This is followed by the *TDMSignaling* field within the MTADAPTER_INFO structure (returned by the **MTGetAdapterInfo()** API function).

However, the driver uses:

0 = Mu-Law

1 = A-Law

The *TDMEncoding* field within the MTADAPTER_CONFIG structure uses this scheme. This is the structure that is read from and written to by the **MTSetAdapterConfig()** and **MTGetAdapterConfig()** functions, respectively.

Prototype

MT_RESULT MTGetAdapterInfo (

const BOARD	nboard,
PMTADAPTER_INFO const	pBoardInfo,
PULONG const	pLength
);	

Parameters

nboard	board number
pBoardInfo	pointer to MTADAPTER_INFO structure
pLength	Set by the user application, this is a pointer to the size of the given buffer for the MTADAPTER_INFO structure.

Structure

The MTADAPTER_INFO structure is defined as follows:

TABLE 16: MTADAPTER_INFO

Type	Name	Description
ULONG	Status	The status of the board. To retrieve a ASCII description of the board status use MTGetAdapterStatusDescription() .
ULONG	BoardNumber	Board index number with 0 for the first board
MT_VERSION	FWVersion	This is the DSP on-board firmware version. See “MT_VERSION” on page 94 for the definition of version information
ULONG	BaseAddress	I/O base address
ULONG	Interrupt	Interrupt number
ULONG	StartChannel	Logical number of the first on board channel
ULONG	Channels	Number of channels on the board
ULONG	Option1	bit 0: WAV bit 1: MVIP bit 2: Volume/Pitch bit 3: MF bit 4: Silence Compression
ULONG	Option2	bit 0: Caller ID bit 1: Conference bit 2: Call Logging bit 4: FAX bit 6: Pulse detection
ULONG	Digirate	Digitizing rate
ULONG	Bandwidth	Channel bandwidth
ULONG	PCISlot	PCI slot number
ULONG	PCIBusNumber	PCI bus number
ULONG	BoardType	Board Type of DUAL_T1_PASSIVE_HI_CARD, etc. All board types are defined in the NtiData.h file.
UCHAR	BusType	MUX_H100, MUX_MVIP, or MUX_NONE
UCHAR	MasterMode	Board Master Mode of the following: 0: slave board 1: MVIP master, or H100 Master A 2: H100 Master B
UCHAR	MasterClock	Master clock source reference value of CLOCK_SOURCE_OSC, and etc. All board clock reference codes are defined in the NtiData.h file.
UCHAR	Sec8kNetrefClock	MVIP Sec8K or H100 Netref clock output 0: Disabled, output tri-stated 1, 3 - 4: Type of clock reference of CLOCK_SOURCE_OSC, etc. All board clock reference codes are defined in the NtiData.h file.
UCHAR	NumSummations	Number of summation resource on board, if any
UCHAR	StartSummation	Index of the first on board summation resource
UCHAR	NumAudios	Number of audio jack resource on board, if any

TABLE 16: MTADAPTER_INFO

Type	Name	Description
UCHAR	StartAudio	Index of the first on board audio jack resource
UCHAR	TDMSignaling	CT TDM bus signaling: ULAWPXD for u-law (defined as 1), ALAWPXD for A-law (defined as 2)
UCHAR	ConfigurationID	Adapter Configuration Index (i.e. SwitchIndex)
UCHAR	res[3]	Reserved
MT_VERSION	FWVersionExpected	0x02040b05 is Version 2.4.11.5, i.e. major version 2, minor version 4, point of release version 11, and build version 5.

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_INVALID_BOARD	Board number is out of range
MT_RET_NOT_AVAILABLE	Driver is not loaded

Events

None

CODE EXAMPLE

MTGetAdapterInfo() Code Example

```
ZeroMemory(&adapterinfo, size);

MTGetAdapterInfo(board, &adapterinfo, &size);

successful...\n");

printf("MTGetAdapterInfo()

ULONG boardNo = adapterinfo.BoardNumber;
ULONG status = adapterinfo.Status;
MT_VERSION version = adapterinfo.FWVersion;
printf("Board: %u\nFirmware: %d.%d.%d Build

%d %c
%c\n", boardNo, version.Major, version.Minor, version.Internal, version.Build, versi
on.Release, version.Feature);

printf("Status: ");
switch(status)
{
```

A complete code example is available on the AudioCodes website:

<http://www.audiocodes.com/blades/support.htm>

See also: [MTGetSystemInfo\(\)](#)

MTGetAdapterXInfo()

Description

MTADAPTER_INFO contains general information whereas MTADAPTER_XINFO contains information for protocol/variant settings for digital cards (DP/DT).

The user application must have the ownership of the specified board.

Prototype

MT_RESULT MTGetAdapterXInfo(

```
const BOARD          nBoard,
PMTBOARD_XINFO const pBoardXInfo,
PULONG const         pLength
);
```

Parameters

nBoard	board index, 0 for the 1st board
pBoardXInfo	pointer to the PMT_BOARD_XINFO structure
pLength	Set by the user application, this is a pointer to the size of the given buffer for the MTADAPTER_XINFO structure.

Structure

TABLE 17: MT_ADAPTER_XINFO

Type	Name	Function
ULONG	BoardNumber	Board index in SDK, i.e. SystemIndex, 0 for the 1st board
USHORT	NumFramers	Number of local framers: 0 - 4
UCHAR	FramerSignaling[4]	Signaling for the max. of 4 framers. All signaling values are defined in the NtiData.h file.
UCHAR	FramerProtocol[4]	Protocol variant for the max. of 4 framers. All protocol values are defined in the NtiData.h file.
USHORT	NumTrunks	Number of trunks on board: 0 - 2
UCHAR	TrunkChannelCount[2]	Number of channels on each trunk
UCHAR	TrunkSignaling[2]	Signaling for the max. of 2 trunks
UCHAR	TrunkProtocol[2]	Protocol variant for the max. of 2 trunks. All values are defined in the NtiData.h file.
UCHAR	TrunkInterfaceSide[2]	Terminal/Network side support for 2 trunks: 0 = SUPPORT_TE, Terminal side support 1 = SUPPORT_NT, Network side support
UCHAR	TrunkNFASId[2]	Signaling NFAS Group Index

Return Codes

MT_RET_OK	No error
MT_RET_DATA_TRUNCATED	Data copied but truncated; the buffer size set by the user application is not large enough for this API request. <i>Flag-DataTruncated</i> field of the MTSYS_CONFIG or the MTSYS_CONFIGURATION data structure must be enabled (1) to receive this return code, otherwise MT_RET_OK is used, and the truncated data is passed to the user application
MT_RET_NOT_AVAILABLE	Driver is not loaded

Events

None

MTGetAdapterStatusDescription()

Description

MTGetAdapterStatusDescription() retrieves the text description that can be used to explain the value returned by the MTADAPTER_INFO.Status field.

Prototype

MT_RESULT MTGetAdapterStatusDescription (

```
const ULONG           StatusCode,  
const PCHAR           pDescription,  
const int             nLength  
);
```

Parameters

StatusCode	status code passed into the MTADAPTER_INFO.Status field. For example: 0x05.
pDescription	pointer for the NULL terminated description
nLength	length of the description buffer

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_PARAM	Invalid argument passed to function

Events

None

MTGetAdapterDescription()

Description

When called, **MTGetaAdapterDescription()** will return an ASCII description of the board in a system. A call to **MTGetAdapterInfo()** is required to use this API to obtain the first parameter, BoardType.

Prototype

void MTGetAdapterDescription (

```
const ULONG          BoardType,  
const PCHAR          pDescription,  
const int             nLength  
);
```

Parameters

BoardType	BoardType:DP6409, DP3209, etc.
pDescription	Pointer for the NULL terminated description
nLength	Length of the description buffer

CODE EXAMPLE

MTGetAdapterDescription() Code Example

```
// need to call MTGetAdapterInfo() before calling MTGetAdapterDescription()  
ZeroMemory(&adapterinfo, adaptersize);  
result =  
MTGetAdapterInfo(board, &adapterinfo, &adaptersize);  
if(result != MT_RET_OK)  
{  
    printf("MTGetAdapterInfo() failed...\n");  
    printf("- Cannot get board  
description...\n");  
    return 1;  
}  
else  
{  
    ZeroMemory(&description, descsize);  
  
    MTGetAdapterDescription(adapterinfo.BoardType, description, descsize);  
    printf("MTGetAdapterDescription()  
successful...\n");  
    printf("Board type: %u\nBoard description:  
%s\n", adapterinfo.BoardType, description);  
}
```

A complete code example is available on the AudioCodes website:

<http://www.audiocodes.com/blades/support.htm>

See Also: [MTGetAdapterInfo\(\)](#)

BOARD IDENTIFICATION & OEM FUNCTIONS

MTBlinkBoard()

Description

MTBlinkBoard() causes CR17 LED on the specified board to blink its board index plus one 10 times so that the user can match a physical board with its board number. There is a 2 second delay between each blinking of the index plus one.
For board 0, **MTBlinkBoard()** causes the CR-17 to blink once for 10 times, with a 2 second delay between each blink. For board 1, **MTBlinkBoard()** causes the CR17 to blink twice for 10 times.

Prototype

```
MT_RESULT MTBlinkBoard (  
    const BOARD          nboard  
);
```

Parameters

nboard	board number

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_BOARD	Board number is out of range
MT_RET_BOARD_NOT_OPENED	Board device not opened for access
MT_RET_NOT_AVAILABLE	Driver is not loaded

Events

None

CODE EXAMPLE

MTBlinkBoard() Code Example

```
result = MTBlinkBoard(board);  
if(result == MT_RET_OK)  
{  
    printf("MTBlinkBoard() successful...\n");  
    printf("CR17 LED on board %d will blink %02d  
time(s)\n",board,board + 1);  
}
```

A complete code example is available on the AudioCodes website:

<http://www.audiocodes.com/blades/support.htm>

MTStopBlinkBoard()

Description

MTBlinkBoard() causes CR17 LED on the specified board to blink its board index plus one 10 times so that the user can match a physical board with its board number. There is a 2 second delay between each blinking of the index plus one.
For board 0, **MTBlinkBoard()** causes the CR-17 to blink once for 10 times, with a 2 second delay between each blink. For board 1, **MTBlinkBoard()** causes the CR17 to blink twice for 10 times.

Prototype

```
MT_RESULT MTStopBlinkBoard (  
    const BOARD          nboard  
);
```

Parameters

nboard	board number

Return Code

MT_RET_OK	No error
MT_RET_INVALID_BOARD	Board number is out of range
MT_RET_BOARD_NOT_OPENED	Board device not opened for access
MT_RET_NOT_AVAILABLE	Driver is not loaded

Events

None

MTBoardGetCustomSwitchSetting()

Description

MTBoardGetCustomSwitchSetting() returns the board’s switch setting. The Board’s switch is set using the board’s thumb wheel. Currently, this API is only supported for the SmartWORKS NGX and the SmartWORKS LD 2409, 1609, and 809X. **This function is only supported on LD boards greater than REV A.** If this function is attempted on an LD board that does not have a custom switch then MT_RET_OK is returned with a 255 value.

The board does not have to be restarted for the switch setting to take effect - the switch can be changed “on the fly” and this function will return the correct value.

This function is supported on the LD boards as of release 3.8.

This function can be used on NGX boards as of release

Prototype

```
MT_RESULT MTBoardGetCustomSwitchSetting (  
    const BOARD          nBrd;  
    int * const           pCurrentSwitchSetting  
);
```

Parameters

nBoard	board number
pCurrentSwitchSetting	A pointer to an integer parameter which returns the board’s current switch setting

Return Codes

MT_RET_OK	No error and switch setting is returned. If this function is attempted on an LD board that does not have a custom switch then a 255 value is returned.
MT_RET_INVALID_PARAM	Invalid argument passed to function (board number is not valid)
MT_RET_NOT_AVAILABLE	Function not supported by the board
MT_RET_TIMEOUT	Board is unavailable
MT_RET_THREAD_PROTECTED	Function unable to complete in expected time
MT_RET_BOARD_NOT_OPENED	Board is not opened for access

Events

None

MTGetBoardAssemblyInfo()

This API was formerly called **MTGetBoardSerialInfo()**

Description

MTGetBoardAssemblyInfo() is used to retrieve the serial information that resides on the board's EEPROM. The application may read but not alter this information.

Prototype

MT_RESULT MTGetBoardAssemblyInfo(

```
const BOARD          nBoard,
PSERIAL_INFO const  pBoard_SerialInfo,
PULONG const        pLength
);
```

Parameters

nBoard	board number
pBoard_SerialInfo	pointer to SERIAL_INFO structure
pLength	Set by the user application, this is a pointer to the size of the given buffer for the SERIAL_INFO structure.

Structure

TABLE 18: SERIAL_INFO

Type	Name	Description
CHAR	AssemblyNumber	NULL terminated string, maximum size 16 bytes
CHAR	HWRevision	NULL terminated string of hardware revision number, maximum size 4 bytes
CHAR	SerialNumber	NULL terminated string of board serial number, maximum size 8 bytes
CHAR	AssemblyCode	NULL terminated string of assembly code, maximum size 4 bytes
CHAR	DateCode	NULL terminated string of manufacturing date, maximum size 8 bytes

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_INVALID_BOARD	Board number is out of range

MT_RET_DATA_TRUNCATED	Data copied but truncated; the buffer size set by the user application is not large enough for this API request. <i>Flag-DataTruncated</i> field of the MTSYS_CONFIG or the MTSYS_CONFIGURATION data structure must be enabled (1) to receive this return code, otherwise MT_RET_OK is used, and the truncated data is passed to the user application
MT_RET_NOT_AVAILABLE	Driver is not loaded

Events

None

MTSetAdapterEEPROMConfig()

Description

This API allows users to populate a data field for board identification.

Prototype

int MTSetAdapterEEPROMConfig (

```
const BOARD          nBoard,  
PCHAR const          pEEPROMConfig,  
PULONG const         pLength  
):
```

Parameters

nBoard	number of the specified board
pEEPROMConfig	pointer to the user data. This field is limited to 128 bytes.
pLength	Set by the user application, this is a pointer to the size of the given buffer for the EEPROMConfig parameter

NOTE: For backwards compatability with older AudioCodes products, the LENGTH parameter for Set APIs allows the SDK to set up to the specified size. The rest of parameters are not changed. MT_RET_OK is returned.

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_BOARD	Board number is out of range
MT_RET_BOARD_NOT_OPENED	Board is not opened for access
MT_RET_INVALID_PARAM	Invalid argument passed to function

NOTE: When using MTSetAdapterEEPROMConfig(), if the user passes information that is longer than 128 bytes, the data will be truncated. When using MTGetAdapterEEPROMConfig(), the application returns the value, with a pLength of 128 bytes.

MTGetAdapterEEPROMConfig()

Description

MTGetAdapterEEPROMConfig() gets the user populated data field for board identification. This API obtains the user specified value.

Prototype

int MTGetAdapterEEPROMConfig (

const BOARD	nBoard,
PCHAR const	pEEPROMConfig,
PULONG const	pLength
):	

Parameters

nBoard	number of the specified board
pEEPROMConfig	pointer to the EEPROMConfig parameter used to get the board ID.
pLength	Set by the user application, this is a pointer to the size of the given buffer for the EEPROMConfig parameter.

NOTE: For backwards compatability with older AudioCodes products, the **LENGTH** parameter for Set APIs allows the SDK to set up to the specified size. The rest of parameters are not changed. **MT_RET_OK** is returned.

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_BOARD	Board number is out of range
MT_RET_BOARD_NOT_OPENED	Board is not opened for access
MT_RET_INVALID_PARAM	Invalid argument passed to function

NOTE: When using **MTSetAdapterEEPROMConfig()**, if the user passes information that is longer than 128 bytes, the data will be truncated. When using **MTGetAdapterEEPROMConfig()**, the application returns the value, with a **pLength** of 128 bytes.

BOARD CONFIGURATION

MTBoardSetConfig()

This API replaces *MTSetAdapterConfig()* (v3.6.1). New application development should use this function.

The data structure MTADAPTER_CONFIGURATION used with *MTBoardSetConfig()* contains more fields than the MTADAPTER_CONFIG data structure used with *MTSetAdapterConfig()*.

Description

The NTI driver and DLL rely on system and adapter level parameters at load time to configure their behavior. When the configuration is changed, both the driver and the DLL must be reloaded for the new configuration to take effect. There are two ways to reload the driver and DLL:

Stop/Start the board drivers using the device manager or the command line program if running Linux.

Prototype

```
int MTBoardSetConfig (  
    const PMTADAPTER_CONFIGURATION    pConfig,  
    const  
    PULONG const                      pLength  
);
```

Parameters

pConfig	pointer to the board configuration structure
pLength	Set by the user application, this is a pointer to the size of the given buffer for the MTADAPTER_CONFIGURATION structure

Structure

The MTADAPTER_CONFIGURATION structure is defined as follows:

TABLE 19: MTADAPTER_CONFIGURATION

Type	Name	Description	Error Value
ULONG	SystemIndex	Adapter index on PCI bus presented by OS, 0 - (MAX_BRD_DEVICES-1)	1
ULONG	AdapterType	This field is read only: DP3209, DP6409, and etc. UNKNOWN_CARD for no board in index 'SystemIndex'	2
		General Adapter Parameters	
UCHAR	MasterMode	MODE_SLAVE(default), MODE_MASTER, or MODE_MASTER_B. NOTE: This field can also be set using the <i>MTSetCTMasterClock()</i> function.	3
UCHAR	MasterClock	Master clock source reference value, default, CLOCK_SOURCE_OSC. NOTE: This field can also be set using the <i>MTSetCTMasterClock()</i> function.	4

TABLE 19: MTADAPTER_CONFIGURATION

Type	Name	Description	Error Value
UCHAR	Sec8kNetrefClock	MVIP Sec8K or H100 Netref clock output, default, SPRM_SOURCE_DISABLE	5
UCHAR	PresentationPreference	Board ordering preference for the associated adapter: Range: 0..MAX_BRD_DEVICES (default) Not implemented for Linux yet	6
		General CT Bus Parameters	
ULONG	CTBusType	MUX_MVIP MUX_H100 MUX_NONE(default)	7
ULONG	TDMEncoding	ULAWPXD for u-law, ALAWPXD for A-law	8
ULONG	CTBusSegmentIndex	Bus segment index the associated adapter is on	9
		Interface Parameters	
ULONG	SMSize	Shared memory size in bytes, min MIN_SMSIZE, default 8K	10
		NGX Parameters Index 0 for base card, 1 for the 1st daughter card, 2 for the 2nd daughter card	
ULONG	CTBusTermination	0 for disable, others for enable	11
ULONG	PBXType[MAX_DC]	PBX supported. Refer to the NtiData.h file for a complete list of valid values.	12 - base card 13 - DC 1 14 - DC 2
ULONG	DChOption[MAX_DC]	0 for D-channel disabled	15 - base card 16 - DC 1 17 - DC 2
ULONG	Termination[MAX_DC]	0 for Hi-Z, 1 for 120 ohm	18 - base card 19 - DC 1 20 - DC 2
		DT/DP Trunk Parameters Index 0 for 1st trunk and etc.	
ULONG	E1Option	T1_OPTION for T1, E1_OPTION for E1	21
ULONG	Framing[MAX_TRUNKS]	FRM_E1_G704(default), FRM_T1_SF, etc.	22-25
ULONG	LineCoding[MAX_TRUNKS]	LC_AMI(default), LC_E1_HDB3, or LC_T1_B8ZS	26-29
ULONG	T1LineBuildOut[MAX_TRUNKS]	LBO_T1_15DB(default), etc.	30-33
ULONG	E1LineBuildOut[MAX_TRUNKS]	LBO_E1_120OHM(default) or LBO_E1_75OHM	34-37
ULONG	ZeroCodeSupression[MAX_TRUNKS]	ZCS_NONE(default), ZCS_GTE, etc. Refer to the NtiData.h file for a complete list of valid values defined for the MT_FRAMER_STATE.ZCS field.	38-41

TABLE 19: MTADAPTER_CONFIGURATION

Type	Name	Description	Error Value
ULONG	ProtocolSignaling[MAX_TRUNKS]	SIGNALING_NONE SIGNALING_ISDN SIGNALING_NFAS SIGNALING_RBS (T1 only on SmartWORKS DT)	42-45
ULONG	ProtocolVariant[MAX_TRUNKS]	E1 & T1: (example PROTOCOL_ISDN_ETS300 national 2 At&t 5ESS DMS 100 NTT Austel 1 Q.SiG RBS (T1 only): (example PROTOCOL_RBS_LOOP_FXO) E&M Wink E&M Immediate Loop_FXS Loop_FXO	46-49
ULONG	ISDNInterfaceSide[MAX_TRUNKS]	SUPPORT_TE(default) or SUPPORT_NT	50-53
ULONG	RBSMaxDigits[MAX_TRUNKS]	Default 10	54-57
ULONG	RBSInterDigitTime[MAX_TRUNKS]	Min 10. Default 3000, in unit of ms	58-61
ULONG	RBSDialingDelay[MAX_TRUNKS]	Min 300, Default 1000, in unit of ms	62-65
ULONG	NFASIndex[MAX_TRUNKS]	0 to (MAX_NFAS - 1)	66-69
		LD Board	
ULONG	OffhookImpedance	FCC_600 = FCC_600, FCC 600 ohms resistive ETSI_270 = ETSI_270, ETSI 270 ohm+750 ohm 0.15 micro-F AUSTRALIA_220 = AUSTRALIA_220, Australia 220 ohm+680 ohm 0.12 micro-F CHINA_200 = CHINA_200, China 200 ohm+680 ohm 0.1 micro-F	70
		IPX Board	Error Value
	MT_IPCONFIG	IpNetParams[MAX_IP_NET_PORTS] An array of settings for the three ports on the IPX. The MT_IPCONFIG structure is defined in the next table in this section.	1
ULONG	DefaultGtwBootp	If DHCP is used on this network, this field is used to enable DHCP on the IPX (transmitting port only) MT_DHCP or MT_STATIC.	
ULONG	DefaultGtwSrcInterface	The network interface from which to retrieve DHCP information (NOTE: The network inter- face identified here, does not have to have DHCP enabled, typically this will be set to '0' as this is the port used for transmitting data).	

TABLE 19: MTADAPTER_CONFIGURATION

Type	Name	Description	Error Value
ULONG	DefaultGateway	Used when DHCP is not used to retrieve the IP address of the default gateway. This field contains the IP Address of the default gateway. This board parameter is only used by the single port on the IPX used for transmitting to the network.	
ULONG	eth0Bootp	Enables DHCP on interface 0. MT_DHCP or MT_STATIC.	
ULONG	eth1Bootp	Enables DHCP on interface 1. MT_DHCP or MT_STATIC.	
ULONG	eth2Bootp	Enables DHCP on interface 2. MT_DHCP or MT_STATIC.	
ULONG	DNSBootp	Enables the automatic retrieval of the IP Addresses of the Preferred and Alternate DNS servers. MT_DHCP or MT_STATIC. <i>(This field reserved for future use).</i>	
ULONG	DNSSrcInterface	The network interface from which to retrieve DNS information when DHCP is enabled for DNS. (NOTE: The network interface identified here, does not have to have DHCP enabled, typically this will be set to '0' as this is the port used for transmitting data). <i>(This field reserved for future use).</i>	
ULONG	preferredDNSAddr	Preferred DNS server <i>(This field reserved for future use).</i>	
ULONG	alternateDNSAddr	Alternate DNS server <i>(This field reserved for future use).</i>	
ULONG	VLANEnable	If all VoIP traffic is available on a single VLAN, this field can be enabled to configure the IPX to only process packets from a single VLAN. MT_PASSIVE_VLAN_ENABLE, MT_PASSIVE_VLAN_DISABLE	
ULONG	VLANId	The ID of the only VLAN that the IPX should monitor.	

TABLE 20: MT_IPCONFIG STRUCTURE

Type	Name	Description
ULONG	IPAddress	IP Address for this port
ULONG	SubNetMask	SubNet Mask for this port
ULONG	DefaultGatewayAddr	The IP address of the default Gateway (this field is maintained for backwards compatibility, but is no longer read into the IPX.

Return Codes

All parameters will be range checked before updating the registry. This API returns 0 on success. Otherwise, the API returns the index of the error field of structure MTADAPTER_CONFIGURATION (documented in table above). No system level check is performed in this API. The checking of configuration conflict, if any, between boards is not within the scope of this API.

MTBoardGetConfig()

This function replaces **MTGetAdapterConfig()** (v3.6.0). New application development should use this function. If **MTBoardSetConfig()** is used to modify board values, then this API *must* be used to retrieve board settings.

Description

Use this API to get current adapter configuration from the registry or the NtiDrv.conf file when running Linux. The adapter index should be in field MTADAPTER_CONFIGURATION.SystemIndex before calling this API.

The MTADAPER_CONFIGURATION structure is defined where the API **MTBoardSetConfig()** is explained.

For registry values that are not configured, the default values will be returned

Prototype

int MTBoardGetConfig (

PMTADAPTER_CONFIGURATION const	pConfig,
PULONG const	pLength
);	

Parameters

pConfig	pointer to the PMTADAPTER_CONFIGURATION structure
pLength	Set by the user application, this is a pointer to the size of the given buffer for the MTADAPTER_CONFIGURATION structure

Return Codes

This API returns 0 on success. For registry values that are not configured, the default values will be returned.

MT_RET_OK	No error
MT_RET_TIMEOUT	Wait for event timed out
MT_RET_DATA_TRUNCATED	Data copied but truncated; the buffer size set by the user application is not large enough for this API request. <i>FlagDataTruncated</i> field of the MTSYS_CONFIG or the MTSYS_CONFIGURATION data structure must be enabled (1) to receive this return code, otherwise MT_RET_OK is used, and the truncated data is passed to the user application
MT_RET_NOT_ENOUGH_DATA	The amount of collected data does not satisfy request with timeout set to 0

MTBoardGetLicensedSessions()

Description

Retrieve the number of sessions licensed on a specific IPX-C board within the system. This API is synchronous only.

NOTE: If using this legacy IPX products, users can monitor and forward media for a maximum of 64 stations (64 concurrent media sessions).

NOTE: If license file installation fails, the board default is 8 sessions.

Prototype

int MTBoardGetLicensedSessions (

const BOARD	nBoard,
PULONG const	pNumSessions
);	

Parameters

nBoard	Board Id
pNumSessions	Pointer to a buffer to receive the number of sessions

Return Codes

.	
MT_RET_OK	No error
MT_RET_INVALID_BOARD	board number is out of range
MT_RET_BOARD_NOT_OPENED	board is not opened for access
MT_RET_NOT_AVAILABLE	board does not support this request
MT_RET_INVALID_PARAMETER	pNumSessions pointer is invalid or NULL

Events

None

BOARD CLOCK SOURCE FUNCTIONS

MTSetCTMasterClock()

Description

MTSetCTMasterClock() changes the master mode and clock source settings for MVIP/H.100 bus to any of the following: MASTER_A, MASTER_B, or SLAVE for H100 MASTER or SLAVE for MVIP

To change the Master/Slave configuration of an existing system, the following rules must be applied:

1. All boards must be open (*MTSysStartup()*)
2. The Slave board must be set to Master first
3. Once a new Master board has been set, then users can change the original Master board to a Slave (if a new Master has not been defined, then MT_RET_BUSY is returned to user application which is trying to set Slave to this board).

Prototype

```
MT_RESULT MTSetCTMasterClock (  
    const BOARD      nBoard,  
    const int         BusType,  
    const int         mMaster,  
    const USHORT      ClockSource  
);
```

Parameters

nBoard	board index number
BusType	MUX_MVIP or MUX_H100
mMaster	H.100: MODE_MASTER_A, MODE_MASTER_B, or MODE_SLAVE MVIP: MODE_MASTER or MODE_SLAVE
ClockSource	clock source for the specified master, ignored if MODE_SLAVE specified Possible ClockSource values are: CLOCK_SOURCE_OSC CLOCK_SOURCE_CT_NETREF CLOCK_SOURCE_SEC8K CLOCK_SOURCE_CT_H100 CLOCK_SOURCE_NET1 CLOCK_SOURCE_NET2 CLOCK_SOURCE_NET3

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_INVALID_BOARD	Board number is out of range
MT_RET_BOARD_NOT_OPENED	Board device not opened for access
MT_RET_NOT_AVAILABLE	Driver is not loaded (board is not visible to DLL)
MT_RET_INVALID_BUSTYPE	Failure to specify MUX_MVIP or MUX_H100
MT_RET_API_THREAD_PROTECTED	Another thread is currently accessing this API

Events**None****CODE EXAMPLE****MTSetCTMasterClock() Code Example**

```

result = MTSetCTMasterClock(board,bustype, master, clock);
        if(result == MT_RET_OK)
        {
            printf("MTSetCTMasterClock()
successful...\n");
            printf("Setting...\nBus type to:
MUX_MVIP\nMaster mode to: MODE_MASTER\nClock source to: NET1\n");
        }

```

A complete code example is available on the AudioCodes website:

<http://www.audiocodes.com/blades/support.htm>

MTGetCTMasterClock()

Description

Retrieves the master mode (Master or Slave for MVIP, MasterA, MasterB or Slave for H.100) and clock source settings of the MVIP/H.100 bus

Prototype

MT_RESULT MTGetCTMasterClock(

const BOARDnBoard,

const intBusType ,

const intpMaster,

const USHORTpClockSource

);

Parameters

nBoard	board index number
BusType	MUX_MVIP or MUX_H100
pMaster	buffer for the returned master setting, MODE_SLAVE, MODE_MASTER_A, or MODE_MASTER_B
pClockSource	buffer for the returned clock source setting

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_INVALID_BOARD	Board number is out of range
MT_RET_BOARD_NOT_OPENED	Board device not opened for access
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_INVALID_BUSTYPE	Failure to specify MUX_MVIP or MUX_H100
MT_RET_DATA_TRUNCATED	Data copied but truncated; the buffer size set by the user application is not large enough for this API request. <i>Flag-DataTruncated</i> field of the MTSYS_CONFIG or the MTSYS_CONFIGURATION data structure must be enabled (1) to receive this return code, otherwise MT_RET_OK is used, and the truncated data is passed to the user application
MT_RET_API_THREAD_PROTECTED	Another thread is currently accessing this API

Events

None

CODE EXAMPLE**MTGetCTMasterClock() Code Example**

```

result = MTGetCTMasterClock(board,bustype,&master,&clock);
    if(result == MT_RET_OK)
    {
        printf("MTGetCTMasterClock()

successful...\n");

        printf("Bus type: MUX_MVIP\n");
        printf("Master mode: ");
        switch(master) // for MVIP setup only
        {
            case MODE_SLAVE:
                printf("Slave\n");
                break;
            case MODE_MASTER:
                printf("Master\n");
                break;
        }
        printf("Clock source: ");
        switch(clock) // for MVIP setup only
        {
            case CLOCK_SOURCE_OSC:
                printf("OSC\n");
                break;
            case CLOCK_SOURCE_SEC8K:
                printf("SEC8K\n");
                break;
            case CLOCK_SOURCE_NET1:
                printf("NET1\n");
                break;
            case CLOCK_SOURCE_NET2:
                printf("NET2\n");
                break;
            case CLOCK_SOURCE_NET3:
                printf("NET3\n");
                break;
        }
    }

```

A complete code example is available on the AudioCodes website:

<http://www.audiocodes.com/blades/support.htm>

MTSetCTNetrefSource()

This API was formerly called *MTSetSlaveSEC8KClockSource()*

Description

Sets the CT_NETREF_1 pin clock source on an H.100 bus or the SEC8K pin clock source on an MVIP bus. A board must be in slave mode to use this API with a CT bus. Only one slave can drive this signal per system unless you enable bus segmentation.

Prototype

MT_RESULT MTSetCTNetrefSource (

```
const BOARD      nBoard,
const int        BusType,
const USHORT     ClockSource
);
```

Parameters

nBoard	board index number
BusType	MUX_MVIP or MUX_H100
ClockSource	Possible ClockSource values are:
	CLOCK_SOURCE_OSC 1
	CLOCK_SOURCE_CT_NETREF 2 MUX_H100 only
	CLOCK_SOURCE_SEC8K 2 MUX_MVIP only
	CLOCK_SOURCE_CT_H100 3 MUX_H100 only
	CLOCK_SOURCE_NET1 4
	CLOCK_SOURCE_NET2 5
	CLOCK_SOURCE_NET3 6

Board support:

LD, VR, NGX, DT, DP, AT, PT	CLOCK_SOURCE_OSC
NGX	CLOCK_SOURCE_CT_NETREF
NGX	CLOCK_SOURCE_SEC8K
NGX	CLOCK_SOURCE_CT_H100
NGX, DT, DP	CLOCK_SOURCE_NET1
NGX	CLOCK_SOURCE_NET2
NGX	CLOCK_SOURCE_NET3

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_INVALID_BOARD	Board number is out of range
MT_RET_BOARD_NOT_OPENED	Board device not opened for access
MT_RET_NOT_AVAILABLE	Driver is not loaded (board is not visible to DLL)
MT_RET_INVALID_BUSTYPE	Failure to specify MUX_MVIP or MUX_H100
MT_RET_INVALID_BUSTYPE	Invalid bus type was specified
MT_RET_API_THREAD_PROTECTED	Another thread is currently accessing this API

Events

None

MTGetCTNetrefSource()

This API was formerly called *MTGetSlaveSEC8KClockSource()*

Description

MTGetCTNetrefSource() retrieves the Netref clock source setting (H.100) and SEC8K clock source setting (MVIP).

Prototype

MT_RESULT MTGetCTNetrefSource(

const BOARD nBoard,
const int BusType ,
USHORT * const pClockSource
);

Parameters

nBoard	board index number
BusType	MUX_MVIP or MUX_H100
pClockSource	buffer for the returned NetRef setting

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_INVALID_BOARD	Board number is out of range
MT_RET_BOARD_NOT_OPENED	Board device not opened for access
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_API_THREAD_PROTECTED	Another thread is currently accessing this API

Events

None

MTSetCTSlaveClockMode()

Description

MTSetCTSlaveClockMode() sets the clock source (Master Clock A or Master Clock B) of a specified board in H.100 bus mode. It also enables/disables auto-switching to an alternate clock if the selected clock is invalid.

Prototype

MT_RESULT MTSetCTSlaveClockMode(

```
const BOARD      nBoard,  
const int        BusType,  
const USHORT     Clock,  
const BOOLEAN    fAutoSwitch  
);
```

Parameters

nBoard	board index number
BusType	MUX_H100
Clock	0 for clock A (default), 1 for clock B, MODE_CLOCK_A or MODE_CLOCK_B
fAutoSwitch	true (default) to enable auto switching to the alternate clock if the selected clock is invalid

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_PARAM	Error in clock parameter (invalid argument passed to function)
MT_RET_INVALID_BOARD	Board number is out of range
MT_RET_BOARD_NOT_OPENED	Board device not opened for access
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_INVALID_BUSTYPE	Invalid bus type was specified
MT_RET_API_THREAD_PROTECTED	Another thread is currently accessing this API

Events

None

MTGetCTSlaveClockMode()

Description

MTGetCTSlaveClockMode() retrieves the clock source (from Master A or Master B) and auto-switching setting of a specified board in H.100 slave mode.

Prototype

```
MT_RESULT MTGetCTSlaveClockMode(  
    const BOARD          nBoard,  
    const int             BusType,  
    PUSHORT const        pClock,  
    BOOLEAN * const       pAutoSwitch  
);
```

Parameters

nBoard	board index number
BusType	MUX_H100, MUX_MVIP
pClock	buffer for the returned clock currently selected
pAutoSwitch	buffer for the returned auto-switching setting

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_INVALID_BOARD	Board number is out of range
MT_RET_BOARD_NOT_OPENED	Board device not opened for access
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_INVALID_BUSTYPE	Invalid bus type was specified
MT_RET_API_THREAD_PROTECTED	Another thread is currently accessing this API

Events

None

MTGetNET()

Description

Retrieves the source of recovered received clock.

The NET index starts from 1. For SmartWORKS cards, the maximum NET index is 2; for NGX cards, the maximum NET index is 3.

The framer index starts from 0 to 3 for the four possible framers for each line. The SmartWORKS DP has four (4) framers, SmartWORKS DT has two (2) framers, and the SmartWORKS NGX has either eight(8), sixteen (16) or twenty-four (24) depending on the number of daughter cards.

For NET index 1, **MTSetNET()** sets the source of the recovered clock for the specified board from:

Framer 0	line 1 NE side - incoming (recommended)
Framer 1	line 1 CP side - outgoing
Framer 2	line 2 NE side - incoming
Framer 3	line 2 CP side - outgoing

For NET index 2, **MTSetNET()** sets the source of the recovered clock for the specified board from:

Framer 0	line 1 NE side - incoming
Framer 1	line 1 CP side - outgoing
Framer 2	line 2 NE side - incoming (recommended)
Framer 3	line 2 CP side - outgoing

Prototype

```
MT_RESULT MTGetNET(
    const BOARD      nBoard,
    const USHORT     nNET,
    USHORT * const   pFramer
);
```

Parameters

nBoard	board index
nNET	NET index
pFramer	pointer to framer index

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_PARAM	Error in board type, or NULL pointer (invalid argument passed to function)
MT_RET_INVALID_BOARD	Board number is out of range
MT_RET_BOARD_NOT_OPENED	Board device not opened for access
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_API_THREAD_PROTECTED	Another thread is currently accessing this API

Events

None

MTSetNET()

Description

Sets the source of recovered received clock when the clock source

The NET index starts from 1.
For SmartWORKS cards, the maximum NET index is 2; for NGX cards, the maximum NET index is 3.

The framer index starts from 0 to 3 for the four possible framers for each line. The SmartWORKS DP has four (4) framers, SmartWORKS DT has two (2) framers, and the SmartWORKS NGX has either eight(8), sixteen (16) or twenty-four (24) depending on the number of daughter cards.

The NGX base card is NET1, the daughter cards are NET2 and NET3 respectively.

Prototype

MT_RESULT MTSetNET(

```
const BOARD      nBoard,  
const USHORT     nNET,  
const USHORT     nFramer  
);
```

Parameters

nBoard	board index
nNET	NET index
nFramer	framer index (Valid ranges for SmartWORKS DP is 0-3, SmartWORKS DT is 0-1, valid ranges for SmartWORKS NGX is 0-7 -see description below)

For SmartWORKS DP

MTSetNET() sets for the specified board the source of the recovered received clock from:

Framer 0	line 1 NE side - incoming (recommended)
Framer 1	line 1 CP side - outgoing
Framer 2	line 2 NE side - incoming
Framer 3	line 2 CP side - outgoing

For SmartWORKS NGX

MTSetNET() sets the source of the recovered received clock for the specified board from:

Framer 0	Line 1 of the NGX base or daughter card
Framer 1	Line 2 of the NGX base or daughter card
Framer 2	Line 3 of the NGX base or daughter card
Framer 3	Line 4 of the NGX base or daughter card
Framer 4	Line 5 of the NGX base or daughter card

Framer 5	Line 6 of the NGX base or daughter card
Framer 6	Line 7 of the NGX base or daughter card
Framer 7	Line 8 of the NGX base or daughter card

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_PARAM	Error in board index, board type, or NULL pointer (invalid argument)
MT_RET_INVALID_BOARD	Board number is out of range
MT_RET_BOARD_NOT_OPENED	Board device not opened for access
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_API_THREAD_PROTECTED	Another thread is currently accessing this API

Events**None**

BOARD FIRMWARE FUNCTIONS

MTWFInit()

Description

Initializes the firmware load process.

Prototype

```
MT_RESULT MTWFInit (  
    VOID  
);
```

Parameters

no parameter required

Return Codes

MT_RET_OK	no error
MT_RET_LIB_LOAD_FAILED	the driver (NtiDRV.DLL) load failed

Events

None

MTWFUpdateFirmware()

Description

This function loads the firmware. The file name is specified as a function parameter.

If the parameter “CompletionRoutine” is NULL, this function behaves as synchronous function. It returns MT_RET_OK when the update is completed. If “CompletionRoutine” is not NULL, this becomes a background function that returns “MT_RET_IO_PENDING” with no error. The completion routine is invoked when the firmware load is completed or any error occurs.

NOTE: This function must have exclusive rights to the board. For best results, each application that access the board should invoke *MTCloseBoard()*.

Prototype

```
MT_RESULT MTWFUpdateFirmware(  
    const BOARD      nBoard,  
    PCSTR            pFileName,  
    MTWFCOMPLETE     CompletionRoutine,  
    LPARAM           IParam  
);
```

Parameters

nBoard	board number
pFileName	the complete path and name of the firmware file that must be loaded (NULL terminated)
CompletionRoutine	the address of the callback function
IParam	a user defined parameter to echo back to the completion routine

Return Codes

MT_RET_OK	no error
MT_RET_IO_PENDING	Background I/O operation is in progress or queued
MT_RET_INVALID_BOARD	Board number is out of range
MT_RET_INVALID_FILENAME	the function is unable to locate the firmware file
MT_RET_INVALID_BINFILE	image file version cannot be retrieved
MT_RET_BOARD_INCOMPATIBLE_BINFILE	the firmware file does not match the board type (sub system id in file name does not match with the sub system id from MTGetAdepterDescription() for corresponding SmartWORKS hardware
MT_RET_LIB_LOAD_FAILED	the driver (NtiDRV.DLL) load failed

MT_RET_BOARD_INUSE

another application is currently accessing the board, or the board is currently in use

MT_RET_NO_MEM

An operation could not be completed due to insufficient memory or an error occurred while attempting to allocate memory

Events

None

MTWFGetUpdatingStatus()

Description

This function retrieves the percentage of written flash, specified by a percentage. This API is only used when **MTWFUpdateFirmware()** is operating as background function.

Prototype

MT_RESULT MTWFGetUpdatingStatus (

```
const BOARD          nBoard,  
PULONG const         percentage  
);
```

Parameters

nBoard	board number
percentage	a pointer for retrieving the percentage of the written flash

Return Codes

MT_RET_OK	no error
MT_RET_LIB_LOAD_FAILED	the driver (NtiDRV.DLL) load failed
MT_RET_INVALID_BOARD	Board number is out of range

Events

None

MTWFUninit()

Description

This is cleanup function. It is strongly recommended to call this function when users complete their operation with "NTIWF.DLL". This function will unload the "NTIDRV.DLL" loaded by this "NTIWF.DLL"

Prototype

```
MT_RESULT MTWFUnit(  
    VOID  
);
```

Parameters

no parameter required

Return Codes

MT_RET_OK	no error

Events

None

BOARD FUNCTIONS - OBSOLETE APIs

This section lists all board functions which are obsoleted, but maintained in the SDK for backwards compatibility.

MTGetBoardOEMInfo()

The API was formerly called **MTGetOEMInfo()**

Description

MTGetBoardOEMInfo() returns the MTOEM_INFO structure with specific OEM information about each board. The contents of the structure are dependent on the OEM.

Prototype

MT_RESULT MTGetBoardOEMInfo (

```
const BOARD          nboard,
PMTOEM_INFO const    pOem_info (PMTOEM_INFO),
PULONG const         pLength
);
```

Parameters

nboard	board number
pOem_info	pointer to the MTOEM_INFO structure
pLength	Set by the user application, this is a pointer to the size of the given buffer for the MTOEM_INFO structure.

Structure

The MTOEM_INFO is defined as the following:

TABLE 21: MTOEM_INFO

Type	Name	Description
CHAR	OEMInfo[OEM_INFO_LEN]	NULL terminated string with OEM information, maximum size 64 bytes
CHAR	Copyright[OEM_INFO_LEN]	NULL terminated string with copyright information, maximum size 64 bytes

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_INVALID_BOARD	Board number is out of range
MT_RET_DATA_TRUNCATED	Data copied but truncated; the buffer size set by the user application is not large enough for this API request. <i>Flag-DataTruncated</i> field of the MTSYS_CONFIG or the MTSYS_CONFIGURATION data structure must be enabled (1) to receive this return code, otherwise MT_RET_OK is used, and the truncated data is passed to the user application
MT_RET_NOT_AVAILABLE	Driver is not loaded

Events

None

CODE EXAMPLE**MTGetBoardOEMInfo() Code Example**

```
ZeroMemory(&oeminfo, size);

result =
MTGetBoardOEMInfo(board, &oeminfo, &size);
if(result == MT_RET_OK)
{
printf("MTGetBoardOEMInfo()

successful...\n");

CHAR *oemInfo = oeminfo.OEMInfo;
CHAR *copyRight = oeminfo.CopyRight;
printf("OEM Info: %s\nCopyRight:

%s\n", oemInfo, copyRight);
}
```

A complete code example is available on the AudioCodes website:

<http://www.audiocodes.com/blades/support.htm>

See also: [MTGetBoardAssemblyInfo\(\)](#)

CODE EXAMPLE**MTGetBoardAssemblyInfo() Code Example**

```
#include <windows.h>
#include <NTAPI.h>
#include <stdio.h>

int main(int argc, char *argv[]) {
    // Variables
    SERIAL_INFO mtSerialInfo;
    ULONG      uLength  = sizeof(mtSerialInfo);
    MT_RESULT  mtResult;
    char       cRetDesc[128];

    //Does the board need to be open?

    mtResult = MTGetBoardAssemblyInfo(0, &mtSerialInfo, &uLength);
    MTGetReturnCodeDescription(mtResult, cRetDesc, sizeof(cRetDesc));

    printf("Board Closed                -- %s\n", cRetDesc);
    printf("MT_SERIAL_INFO.AssemblyNumbe    -- %s\n",
mtSerialInfo.AssemblyNumber);
    printf("MT_SERIAL_INFO.HWRevision          -- %s\n",
mtSerialInfo.HWRevision);
    printf("MT_SERIAL_INFO.SerialNumber        -- %s\n",
mtSerialInfo.SerialNumber);
    printf("MT_SERIAL_INFO.AssemblyCode        -- %s\n",
mtSerialInfo.AssemblyCode);
    printf("MT_SERIAL_INFO.DateCode            -- %s\n",
mtSerialInfo.DateCode);

    return 0;
}
```

```
}
```

```
/***** Output *****/
```

```
Board Closed          -- MT_RET_OK  
MT_SERIAL_INFO.AssemblyNumber -- 152-1024-001  
MT_SERIAL_INFO.HWRRevision  -- C  
MT_SERIAL_INFO.SerialNumber  -- 5941  
MT_SERIAL_INFO.AssemblyCode  -- 07  
MT_SERIAL_INFO.DateCode     -- 0332
```

```
Press any key to continue
```

```
*****/
```

A complete code example is available on the AudioCodes website:

<http://www.audiocodes.com/blades/support.htm>

MTSetAdapterConfig()

This API has been replaced as of v3.6.1. New application development should use the **MTBoardSetConfig()** function.

The data structure MTADAPTER_CONFIGURATION used with **MTBoardSetConfig()** contains more fields than the MTADAPTER_CONFIG data structure.

Description

The NTI driver and DLL rely on system and adapter level parameters at load time to configure their behavior. When the configuration is changed, both the driver and the DLL must be reloaded for the new configuration to take effect. There are two ways to reload the driver and DLL:

Stop/Start the board drivers using the device manager or the command line program when running Linux.

Prototype

int MTSetAdapterConfig (

```
MTADAPTER_CONFIG const      Config,
PULONG const                pLength
);
```

Parameters

Config	pointer to the system config structure
pLength	Set by the user application, this is a pointer to the size of the given buffer for the MTADAPTER_CONFIG structure

NOTE: For backwards compatability with older AudioCodes products, the LENGTH parameter for Set APIs allows the SDK to set up to the specified size. The rest of parameters are not changed. MT_RET_OK is returned.

Structure

The MTADAPTER_CONFIG structure is defined as follows:

TABLE 22: MTADAPTER_CONFIG

Type	Name	Description	Error Value
ULONG	SystemIndex	Adapter index on PCI bus presented by OS, 0 - (MAX_BRD_DEVICES-1)	1
ULONG	AdapterType	This field is read only: DP3209, DP6409, and etc. UNKNOWN_CARD for no board in index 'SystemIndex'	2
		General Adapter Parameters	
UCHAR	MasterMode	MODE_SLAVE(default), MODE_MASTER, or MODE_MASTER_B	3
UCHAR	MasterClock	Master clock source reference value, default, CLOCK_SOURCE_OSC	4
UCHAR	Sec8kNetrefClock	MVIP Sec8K or H100 Netref clock output, default, SPRM_SOURCE_DISABLE	5

TABLE 22: MTADAPTER_CONFIG

Type	Name	Description	Error Value
UCHAR	PresentationPreference	Board ordering preference for the associated adapter: Range: 0..MAX_BRD_DEVICES (default) Not implemented for Linux yet	6
		General CT Bus Parameters	
ULONG	CTBusType	MUX_MVIP MUX_H100 MUX_NONE(default)	7
ULONG	TDMEncoding	ULAWPXD for u-law, ALAWPXD for A-law	8
ULONG	CTBusSegmentIndex	Bus segment index the associated adapter is on	9
		Interface Parameters	
ULONG	SMSize	Shared memory size in bytes, min MIN_SMSIZE, default 8K	10
		NGX Parameters Index 0 for base card, 1 for the 1st daughter card, 2 for the 2nd daughter card	
ULONG	CTBusTermination	0 for disable, others for enable	11
ULONG	PBXType[MAX_DC]	PBX supported. Refer to the NtiData.h file for a complete list of valid values.	12 - base card 13 - DC 1 14 - DC 2
ULONG	DChOption[MAX_DC]	0 for D-channel disabled	15 - base card 16 - DC 1 17 - DC 2
ULONG	Termination[MAX_DC]	0 for Hi-Z, 1 for 120 ohm	18 - base card 19 - DC 1 20 - DC 2
		DT/DP Trunk Parameters Index 0 for 1st trunk and etc.	
ULONG	E1Option	T1_OPTION for T1, E1_OPTION for E1	21
ULONG	Framing[MAX_TRUNKS]	FRM_E1_G704(default), FRM_T1_SF, etc.	22-25
ULONG	LineCoding[MAX_TRUNKS]	LC_AMI(default), LC_E1_HDB3, or LC_T1_B8ZS	26-29
ULONG	T1LineBuildOut[MAX_TRUNKS]	LBO_T1_15DB(default), etc.	30-33
ULONG	E1LineBuildOut[MAX_TRUNKS]	LBO_E1_120OHM(default) or LBO_E1_75OHM	34-37
ULONG	ZeroCodeSupression[MAX_TRUNKS]	ZCS_NONE(default), ZCS_GTE, etc. Refer to the NtiData.h file for a complete list of valid values defined for the MT_FRAMER_STATE.ZCS field.	38-41
ULONG	ProtocolSignaling[MAX_TRUNKS]	SIGNALING_NONE SIGNALING_ISDN SIGNALING_NFAS SIGNALING_RBS (T1 only on SmartWORKS DT)	42-45

TABLE 22: MTADAPTER_CONFIG

Type	Name	Description	Error Value
ULONG	ProtocolVariant[MAX_TRUNKS]	E1 & T1: (example PROTOCOL_ISDN_ETS300 national 2 At&t 5ESS DMS 100 NTT Austel 1 Q.SiG RBS (T1 only): (example PROTOCOL_RBS_LOOP_FXO) E&M Wink E&M Immediate Loop_FXS Loop_FXO	46-49
ULONG	ISDNInterfaceSide[MAX_TRUNKS]	SUPPORT_TE(default) or SUPPORT_NT	50-53
ULONG	RBSMaxDigits[MAX_TRUNKS]	Default 10	54-57
ULONG	RBSInterDigitTime[MAX_TRUNKS]	Min 10. Default 3000, in unit of ms	58-61
ULONG	RBSDialingDelay[MAX_TRUNKS]	Min 300, Default 1000, in unit of ms	62-65
ULONG	NFASIndex[MAX_TRUNKS]	0 to (MAX_NFAS - 1)	66-69
		LD Board	
ULONG	OffhookImpedance	FCC_600 = FCC_600, FCC 600 ohms resistive ETSI_270 = ETSI_270, ETSI 270 ohm+750 ohm 0.15 micro-F AUSTRALIA_220 = AUSTRALIA_220, Australia 220 ohm+680 ohm 0.12 micro-F CHINA_200 = CHINA_200, China 200 ohm+680 ohm 0.1 micro-F	70

Return Codes

All parameters will be range checked before updating the registry. This API returns 0 on success. Otherwise, the API returns the index of the error field of structure MTADAPTER_CONFIG. (documented in table above). No system level check is performed in this API. The checking of configuration conflict, if any, between boards is not within the scope of this API.

MTSetAdapterToDefault()

Description

MTSetAdapterToDefault() returns the board’s clock and master mode setting to the last known values set to the registry or the NtiDrv.conf file when running Linux at runtime. This allows application developers to modify these settings using the API, but return the board to ‘default’ when finished.

Prototype

```
MT_RESULT MTSetAdapterToDefault (  
    const BOARD          nBoard  
);
```

Parameters

nBoard	board number

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_PARAM	Invalid argument passed to function

Events

None

MTSetNET1(), MTSetNET2()*Obsoleted - Use MTSetNet().***Description**

Sets the source of the recovered received clock for the specified board. The board index is 0 for the first SmartWORKS board within the platform.

With a SmartWORKS DP, the framer index goes from 0 to 3 for the four possible framers for each line. On a SmartWORKS NGX, the framer index goes from 0-7 for the eight possible framers on either the NGX base card or the MX80 daughter card(s). See the User's Guide for more information on each SmartWORKS board.

Prototype**MT_RESULT MTSetNET1(****MT_RESULT MTSetNET2(**

```
const BOARD      nBoard,
const USHORT     nFramer
);
```

Parameters

nBoard	board index
nFramer	framer index (Valid ranges for SmartWORKS DP is 0-3, valid ranges for SmartWORKS NGX is 0-7-see description below)

For SmartWORKS DP

MTSetNET1() sets for the specified board the source of the recovered received clock from

Framer 0	line 1 NE side - incoming (recommended)
Framer 1	line 1 CP side - outgoing
Framer 2	line 2 NE side - incoming
Framer 3	line 2 CP side - outgoing

For SmartWORKS NGX

MTSetNET1() (base card) and **MTSetNet2()** (daughter card) set the source of the recovered received clock for the specified board from:

Framer 0	Line 1 of the NGX base or daughter card
Framer 1	Line 2 of the NGX base or daughter card
Framer 2	Line 3 of the NGX base or daughter card
Framer 3	Line 4 of the NGX base or daughter card
Framer 4	Line 5 of the NGX base or daughter card
Framer 5	Line 6 of the NGX base or daughter card
Framer 6	Line 7 of the NGX base or daughter card
Framer 7	Line 8 of the NGX base or daughter card

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_INVALID_BOARD	Board number is out of range

MT_RET_BOARD_NOT_OPENED	Board device not opened for access
MT_RET_NOT_AVAILABLE	Driver is not loaded

Events

None

CHANNEL CONTROL AND INFORMATION FUNCTIONS

OPENING AND CLOSING CHANNELS

MTOpenChannel()

Description

MTOpenChannel() opens the specified channel for access. It is suggested to open all boards and channels by invoking **MTSysStartup()** if the developer does NOT intend to implement multiple applications using the SmartWORKS SDK.

Prototype

```
MT_RESULT MTOpenChannel(  
    const CHANNEL      nChannel  
);
```

Parameters

nChannel	channel number

Return Codes

MT_RET_OK	No error
MT_RET_CHANNEL_INUSE	On failure (most likely due to the resource having been already grabbed by another application)
MT_RET_MALFUNCTION	Hardware failed during initialization
MT_RET_DEVICE_NOTFOUND	Voice interface device was not found
MT_RET_INTERNAL_ERROR	Internal error in DLL
MT_RET_NOT_AVAILABLE	Driver is not loaded

Events

None

See Also: [MTSysStartup\(\)](#)

MTCloseChannel()

Description

MTCloseChannel() closes the specified channel for access.

Should the user application need to release all resources (boards, channels, TDM timeslots, etc.), it is suggested to do so with **MTSysShutdown()**. This function is not used unless a specified channel must be closed.

Prototype

```
MT_RESULT MTCloseChannel(  
    const CHANNEL          nChannel  
);
```

Parameters

nChannel	channel number

Return Code

MT_RET_OK	No error
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_CHANNEL_NOT_OPENED	Channel device not opened for access
MT_RET_NOT_AVAILABLE	Driver is not loaded

Events

None
See Also: [MTSysShutdown\(\)](#)

MTSysStartup()

This API was formerly called **MTStartLink()**

Description

When this API is called, SmartWORKS boards will be opened, all SmartWORKS channels will be opened, board communications between DLL and driver will be opened, and MUX will be enabled.

System status may be retrieved from **MTGetSystemInfo()**. Prior to using **MTSysStartup()**, the value is SYSTEM_INITIALIZED. After using **MTSysStartup()** the value will change to SYSTEM_STARTED. This function must be called before any other API.

Prototype

MT_RESULT MTSysStartup (VOID)

Return Codes

MT_RET_OK	no error
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_NOT_STARTED	No system resource to be started; usually occurs when all resources are in use

Events

None

MTSysShutdown()

This API was formerly called ***MTStopLink()***.

Description

Closes all channels by flushing all current and pending functions on all channels. This closes all communication with driver, and all channels. This system status will change from SYSTEM_STARTED to SYSTEM_SHUTDOWN.

Once this API is used, the channels will no longer respond to any API calls.

Prototypes

MT_RESULT MTSysShutdown (VOID)

Return Codes

MT_RET_OK	no error
MT_RET_NOT_AVAILABLE	Driver is not loaded

Events

None

CONTROLLING BACKGROUND FUNCTIONS

MTStopChannel()

Description

MTStopChannel() stops a single channel without affecting any other channel in the system. This function is useful in halting a record or playback function if, for instance, the system operator is finished recording a new message prompt.

Once completed, this function returns the channel to an idle state. Any background functions queued for this channel are released and not executed.

This function will always return a success return code of MT_RET_OK. If there is a current function, event EVT_STOP will be issued when that function is aborted. After flushing all functions (if any), two EVT_IDLE status events will always be issued to signal the channel state (idle): one for record, the other for playback.

Scenario

If the specified channel is performing both recording and playing simultaneously, two EVT_STOP events will be issued: one for stopping the recording function, the other for stopping the playback of the function.

In addition, if the channel is also performing the **MTGetDigits()** function, a third EVT_STOP event will be issued for stopping digit collection.

Prototype

```
MT_RESULT MTStopChannel (  
    const CHANNEL      nChannel  
);
```

Parameter

nChannel	channel number to be stopped

Return Codes

MT_RET_OK	No error
MT_RET_IDLE	Channel is idle
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_CHANNEL_NOT_OPENED	Channel device not opened for access
MT_RET_NOT_AVAILABLE	Driver is not loaded

Events

EVT_STOP
EVT_IDLE
See Also: [MTStopCurrentFunction\(\)](#)

CODE EXAMPLE**Code Example MTStopChannel()**

```
if(handleMT(MTStopChannel(channel), "MTStopChannel() channel %d", channel))
{
    wait = true;
    while(wait)
        Sleep(20);
}
```

A complete code example is available on the AudioCodes website:

<http://www.audiocodes.com/blades/support.htm>

MTStopCurrentFunction()

Description

MTStopCurrentFunction() terminates only the current function in progress on the specified codec on a channel. Any subsequent functions in the queue will be executed.

To stop all functions in a channel, use **MTStopChannel()**.

The Event EVT_STOP will be issued when the current function is aborted. However, if there is no current function, the MT_RET_IDLE return code will be returned.

Scenario

If the specified channel is performing both recording and playing simultaneously, two EVT_STOP events will be issued: one for stopping the recording function, the other for stopping the playback of the function.

In addition, if the channel is also performing the MTGetDigits() function, a third EVT_STOP event will be issued for stopping digit collection.

Prototype

MT_RESULT MTStopCurrentFunction (

const CHANNEL nChannel,
const ULONG fOperation
);

Parameters

nChannel	Channel Number
fOperation	MT_ENCODE (MT_RECORD), MT_DECODE (MT_PLAY), or MT_COLLECT_DTMF

Return Codes

MT_RET_OK	No error
MT_RET_IDLE	Channel is idle
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_CHANNEL_NOT_OPENED	Channel device not opened for access
MT_RET_NOT_AVAILABLE	Driver is not loaded

Events

EVT_STOP

CODE EXAMPLE**Code Example MTStopCurrentFunction()**

```
if(handleMT(MTStopCurrentFunction(channel,operation), "MTStopCurrentFunction()  
channel %d, operation %d",channel,operation))  
{  
    wait = true;  
    while(wait)  
        Sleep(20);  
}  
  
handleMT(MTClearEventCallback(channel), "MTClearEventCallback() channel  
%d", channel);
```

A complete code example is available on the AudioCodes website:

<http://www.audiocodes.com/blades/support.htm>

MTSetChannelToDefault()

Description

MTSetChannelToDefault() returns channel configuration to default settings.

Prototype

```
MT_RESULT MTSetChannelToDefault (  
    const CHANNEL      nChannel  
);
```

Parameters

nChannel	Channel Number

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_CHANNEL_NOT_OPENED	Occurs when a channel is not yet opened
MT_RET_INVALID_PARAM	Invalid argument passed to function

Events

None

CHANNEL NUMBERING (GCI) FUNCTIONS

MTGetGCI()

Description

All APIs use the GCI as a channel reference. This function retrieves the GCI based on the provided board and on board channel indexes.

Prototype

MT_RESULT MTGetGCI (

const BOARD nBoard,
const CHANNEL nChannelOnBoard,
PCHANNEL const pGCI
);

Parameters

nBoard	board number
nChannelOnBoard	channel number on board
pGCI	pointer to logical channel number (channel GCI)

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_INVALID_BOARD	Board number is out of range
MT_RET_NOT_AVAILABLE	Driver is not loaded

Events

None

CODE EXAMPLE

MTGetGCI()

```
result = MTGetGCI(board, channel, &pChannel);  
if(result == MT_RET_OK)  
{  
    printf("MTGetGCI() successful...\n");  
    printf("GCI: %d\n", pChannel);  
}
```

A complete code example is available on the AudioCodes website:

<http://www.audiocodes.com/blades/support.htm>

MTGetGCIMap()

Description

All APIs (other than IPX APIs) use GCI as a channel reference. This function maps a GCI indexed channel to a board on the system, as well as the channel index on that board. Board type information is also provided for reference. Information about each board can be retrieved through **MTGetAdapterInfo()** function.

Prototype

MT_RESULT MTGetGCIMap (

```
const CHANNEL      nGCI,  
PBOARD const0     pBoard,  
PUSHORT const     pTDMBoardType,  
PCHANNEL const    pChannelOnBoard  
);
```

Parameters

nGCI	logical channel number, (GCI for a channel)
pBoard	pointer for board index, 0 for the first board
pTDMBoardType	pointer for board type
pChannelOnBoard	pointer for on board channel index, 0 for the first channel on board (not affected by user's GCI index setting)

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_NOT_AVAILABLE	Driver is not loaded

Events

None

See Also: [MTGetAdapterInfo\(\)](#)

CHANNEL INFORMATION FUNCTIONS

MTGetChannelOpenStatus()

Description

This API returns whether the channel is opened or closed. A channel needs to be opened before any access or control can be performed on it by the application.

Scenario

A channel can only be opened by one application at one time. When a channel is opened and held by one application, attempting to open the same channel will fail upon reset of the applications in the platform.

Prototype

MT_RESULT MTGetChannelOpenStatus(

```
const CHANNEL      nChannel,  
BOOLEAN * const    pOpen  
);
```

Parameters

nChannel	channel number
pOpen	Pointer to the open status

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_INVALID_CHANNEL	Channel number is out of range. possible values returned inside open status: TRUE - Channel is opened, i.e. owned by the caller application FALSE- Channel is not opened by the caller application

Events

None

See Also: [MTGetBoardOpenStatus\(\)](#)

MTGetChannelStatus()

This API was formerly called *MTGetStatus()*

Description

MTGetChannelStatus() returns detailed, current information about a channel. If no event queue is used, this is the only means to keep track of a channel's operation status.

Prototype

MT_RESULT MTGetChannelStatus (

```
const CHANNEL          nChannel,
PMTCHAN_STATUS const   pChan_status,
PULONG const           pLength
);
```

Parameters

nChannel	channel number
pChan_status	pointer to MTCHAN_STATUS structure to store channel status info
pLength	Set by the user application, this is a pointer to the size of the given buffer for the MTCHAN_STATUS structure

Structure

The MTCHAN_STATUS structure is defined as follows:

TABLE 23: MTCHAN_STATUS

Type	Name	Function
ULONG	EventEnable	Transition event enabled for event reporting
ULONG	Status	Current channel status. All status values are defined in the NtiData.h file.
ULONG	CurrentDecFunction	Current decode function defined with prefix F_
ULONG	CurrentEncFunction	Current encode function defined with prefix F_
ULONG	Bytes	Bytes processed for current task including encode and decode
ULONG	PreviousEvent	Previous termination event
ULONG	RecordedBytes	Previously recorded (encode) bytes
ULONG	PlayedBytes	Previously played (decode) bytes
USHORT	DTMFCount	Number of DTMF digits queued
USHORT	MFCCount	Number of MF (R2 or R1) tones queued
ULONG	Error	Last error code
ULONG	LineStatus	Status of the phone line. All line status values are defined in the NtiData.h file.
ULONG	CpmResult	Last CPM termination type

EventEnable	The application can switch event reporting of specific event on or off by setting a bit with MTSetEventFilters() . Field EventEnable reflects this setting. The bits are defined by SE_... (Refer to the MTSetEventFilters() section).
Status	Status reflects the current API function the specified channel is performing. The values of this field are defined in the table below.

TABLE 24: STATUS

Bit	Define
x0000	TASK_IDLE
x0001	TASK_RECORD
x0002	TASK_PLAY
x0003	TASK_RECPLAY (TASK_RECORD & TASK_PLAY)
x0004	TASK_GETDIGITS
x0008	TASK_OFFHOOK
x0010	TASK_DIALING
x0020	TASK_ONHOOK
x0040	TASK_CPM
x0080	TASK_PARAM
x0100	TASK_WINK
x0200	TASK_PLAYTONE
x0400	TASK_PREREC
x0800	TASK_STREAMING
xffffffff	NOT_OPENED

CurrentDecFunction	Current recording function on decoder
CurrentEncFunction	Current recording function on encoder
Bytes	The current number of bytes that is transferred to or from the channel during play and/or record. In the situation where the channel is performing play and record at the same time, this counter records the combined data count. After completion, this count is retained in the RecordedBytes and PlayedBytes fields.
PreviousEvent	The event that terminated the previous background function.
RecordedBytes	Number of previously recorded bytes
PlayedBytes	Number of previously played bytes
DTMFCount	The number of received DTMF digits queued for this channel.

MFCCount	The number of received MF tones queued for this channel.
Error	If a function terminated because of an error, this parameter holds the error code. This field is cleared after each call to MTGetChannelStatus().
LineStatus	The line status with various status bits assigned explained below.
CpmResult	If a CPM condition terminated a background function, an EVT_CPM_STOP event is generated with a SubReason code that specifies the exact termination. The SubReason code is stored in the CpmResult field. This parameter is cleared after each call to MTGetChannelStatus().

Line Status definitions:

TABLE 25: LINE DEFINITIONS

Bit	Define	Bit set if...
0	LINE_CPM_COMP	In CPMMODE_COMP (enhanced)
1	LINE_POLARITY	Tip-Ring polarity is negative (enhanced)
2	LINE_RINGBACK	CPM detects ringback
3	LINE_ON_HOOK	The line is on-hook
4	LINE_NO_RINGS	No incoming rings are detected
5	LINE_NO_LOOP	No loop current is present
6	LINE_DTMF	DTMF signal is currently present
7	LINE_SILENT	No activity (no signal) is present.
8	LINE_NO_LVOLTAGE_OFFHOOK	No loop voltage offhook present
9	LINE_NO_DIAL	

NOTE: LINE_POLARITY is only valid when the channel is OFF-HOOK. If the channel is off-hook, a '0' indicates normal loop current and a '1' indicates reverse loop current. The SDK sets the LINE_POLARITY bit to '0' or normal loop current, as an initial value. The system will update the LINE_POLARITY bit when the state of the front end changes. For the LD series boards, events that will cause the LINE_POLARITY bit to be updated include: EVT_LVOLTAGE_ABOVE, EVT_LVOLTAGE_BELOW, or EVT_LVOLTAGE_MIDDLE (i.e., EVT_LVOLTAGE_OFFHOOK). In addition, the events EVT_LOOP_ON, EVT_LOOP_DROP, and EVT_LOOP_REVERSE will cause the state of LINE_VOLTAGE to be updated as well.

The LINE_VOLTAGE bit represents the polarity of the loop current, and not the polarity of line voltage (i.e., Tip with respect to Ring).

Return Codes

MT_RET_OK

No error

MT_RET_DATA_TRUNCATED	Data copied but truncated; the buffer size set by the user application is not large enough for this API request. <i>FlagDataTruncated</i> field of the MTSYS_CONFIG or the MTSYS_CONFIGURATION data structure must be enabled (1) to receive this return code, otherwise MT_RET_OK is used, and the truncated data is passed to the user application
MT_RET_CHANNEL_NOT_OPENED	Channel device not opened for access
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_NOT_AVAILABLE	Driver is not loaded

Events

None

CODE EXAMPLE

MTGetChannelStatus() Code Example

```
ZeroMemory(&chanstatus, size);  
  
result =  
MTGetChannelStatus(channel, &chanstatus, &size);  
if(result == MT_RET_OK)  
{  
    printf("MTGetChannelStatus()  
successful...\n");  
  
    ULONG status = chanstatus.Status;  
    printf("Channel status: ");  
    switch(status)  
    {
```

A complete code example is available on the AudioCodes website:

<http://www.audiocodes.com/blades/support.htm>

MTGetChannelInfo()

Description

MTGetChannelInfo() returns detailed information about channel properties and startup status of the board the channel resides on.

Scenario

Unlike **MTGetAdapterInfo()**, **MTGetChannelInfo()** only retrieves the load status of the board the specified channel resides on. Should more information about a board be desired, use **MTGetAdapterInfo()** which retrieves sp, board version, etc. Use **MTGetGCIMap()** to find out the index number of specified channel's board. Use **MTGetChannelStatus()** to determine the runtime status of a channel.

Prototype

MT_RESULT MTGetChannelInfo (

```
const CHANNEL      nChannel,
PMTCHAN_INFO const pChannel_info,
PULONG const       pLength
);
```

Parameters

nChannel	channel number
pChannel_info	pointer to MTCHAN_INFO structure to store
pLength	Set by the user application, this is a pointer to the size of the given buffer for the MTCHAN_INFO structure.

Structure

TABLE 26: MTCHAN_TS STRUCTURE

Type	Name	Function
SHORT	Framer	-1 for no association
SHORT	Timeslot	-1 for no association

TABLE 27: MTCHAN_INFO STRUCTURE

Type	Name	Function
ULONG	Type	Channel type of VR64, VR32, DP, etc. All channel types are defined in the NtiData.h file.
ULONG	Variant	Channel variant such as protocol, phone type, etc.
ULONG	LoadStatus	Board load status from driver. All load status codes are defined in the NtiData.h file.

TABLE 27: MTCHAN_INFO STRUCTURE

Type	Name	Function
ULONG	nAssociatedTS	Number of timeslots associated with this channel
MTCHAN_TS	AssociatedTS[2]	Associated timeslot information. This structure is defined in the NtiData.h file as the following: Framer = -1 for no association Timeslot = -1 for no association
CHAR	BusType	CT bus of MUX_MVIP, MUX_H100, or MUX_NONE
CHAR	StreamSpeed	CT bus data rate for the 1st 16 streams
CHAR	Res1	CT bus signaling: 0 for u-law, 1 for A-law
CHAR	Res2	Reserved

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_DATA_TRUNCATED	Data copied but truncated; the buffer size set by the user application is not large enough for this API request. <i>FlagDataTruncated</i> field of the MTSYS_CONFIG or the MTSYS_CONFIGURATION data structure must be enabled (1) to receive this return code, otherwise MT_RET_OK is used, and the truncated data is passed to the user application
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_NOT_AVAILABLE	Driver is not loaded

Events

None

MTGetChannelOwner()

Description

If the specified channel is opened, the returned *pPid contains the process ID of the access owner of the specified channel. If the channel is currently not opened by any thread or application, the returned *pPid will be 0.

No access ownership of the specified channel is required.

Prototype

MT_RESULT MTGetChannelOwner (

```
const CHANNEL      nChannel,  
int * const        pPid  
);
```

Parameters

nChannel	Channel Number
pPid	pointer for the process ID of the channel owner

Returned Codes

MT_RET_OK	No error
MT_RET_NOT_AVAILABLE	Driver is not loaded

Events

None

CHANNEL STATISTICS FUNCTIONS

MTGetStatistics()

Description

Runtime errors such as encode overflow, decode underflow, and etc. are recorded for each NTI channel. This API allows user application to retrieve these counters.

The user application does not need to have the access rights of the specified channel.

This API must be used in cooperation with **MTResetStatistics()**.

Prototype

MT_RESULT MTGetStatistics (

```
const CHANNEL          nChannel,
PMTRUNTIME_STATISTICS pRunTimeStat,
const
PULONG const          pLength
);
```

Parameters

nChannel	channel number
pRunTimeStat	pointer to the PMTRUNTIME_STATISTICS structure
pLength	Set by the user application, this is a pointer to the size of the given buffer for the PMRUNTIME_STATISTICS structure.

Structure

The following explains the MTRUNTIME_STAT structure:

Indicates Firmware is in trouble:.

Type	Name	Function
ULONG	PanicCode	Panic code for technical support

Indicates the Driver and Firmware are incompatible:

Type	Name	Function
ULONG	UnknownEvent	Unknown event
ULONG	UnknownRsp	Unknown response
ULONG	CmdError	Command error

Indicates that the application fails to retrieve information from the queue::

Type	Name	Function
ULONG	EventQueueOverflow	Channel event queue overflow
ULONG	DTMFQueueOverflow	Channel DTMF queue overflow

Type	Name	Function
ULONG	MFQueueOverflow	Channel MF queue overflow
ULONG	CIDQueueOverflow	Channel Caller ID queue overflow

Indicates the host ran into performance problems::

Type	Name	Function
ULONG	EncodeOverflow	Channel record overflow
ULONG	DecodeUnderflow	Channel playback underflow

Indicates the firmware is having trouble::

Type	Name	Function
ULONG	EncodeTimeout	Channel record timeout
ULONG	DecodeTimeout	Channel playback timeout

For HDLC packets::

Type	Name	Function
ULONG	HDLCStreamError	Firmware detects bad HDLC stream

Reserved field::

Type	Name	Function
ULONG	Reserved[48]	Reserved for SDK internal use

Return Codes

MT_RET_OK

No error

MT_RET_DATA_TRUNCATED

Data copied but truncated; the buffer size set by the user application is not large enough for this API request. *FlagDataTruncated* field of the MTSYS_CONFIG or the MTSYS_CONFIGURATION data structure must be enabled (1) to receive this return code, otherwise MT_RET_OK is used, and the truncated data is passed to the user application

MT_RET_NOT_AVAILABLE

Driver is not loaded

Events

None

MTResetStatistics()

Note - ResetStatistics works only when there is no PanicCode recorded

Description

This API enables the user application to reset the channel runtime statistics.

There is no requirement for the application to have access rights of the specified channel. It is up to the application to maintain this cooperation between threads and the application while collecting runtime statistics for debugging reference.

This API is used in cooperation with **MTGetStatistics()**.

Prototype

```
MT_RESULT MTResetStatistics (  
    const CHANNEL      nChannel  
);
```

Parameter

nChannel	channel number

Return Codes

MT_RET_OK	No error
MT_RET_NOT_AVAILABLE	Driver is not loaded

Events

None

MTGetBoardStatistics()

Description

Runtime errors such as encode overflow, decode underflow, and etc. are recorded for each NTI channel. This API allows user application to retrieve these counters for an entire board.

The user application does not need to have the access rights of the channels.

This API must be used in cooperation with **MTResetStatistics()**.

Prototype

MT_RESULT MTGetBoardStatistics (

const BOARDnBoard,

PMTRUNTIME_STATISTICSpRunTimeStat,

const

PULONG constpLength

);

Parameters

nBoard	board number
pRunTimeStat	pointer to the PMTRUNTIME_STATISTICS structure
pLength	Set by the user application, this is a pointer to the size of the given buffer for the PMRUNTIME_STATISTICS structure.

Return Codes

MT_RET_OK	No error
MT_RET_DATA_TRUNCATED	Data copied but truncated; the buffer size set by the user application is not large enough for this API request. <i>FlagDataTruncated</i> field of the MTSYS_CONFIG or the MTSYS_CONFIGURATION data structure must be enabled (1) to receive this return code, otherwise MT_RET_OK is used, and the truncated data is passed to the user application
MT_RET_NOT_AVAILABLE	Driver is not loaded

Events

None

MTResetBoardStatistics()

NOTE

MTResetBoardStatistics() works only when there is no PanicCode recorded

Description

This API works on the specified board the same way as **MTResetStatistics()** works on the specified channel.

No access ownership of the specified board is required.

Prototype

MT_RESULT MTResetBoardStatistics (

const BOARD nBoard
);

Parameters

nBoard board number

Return Codes

MT_RET_OK	No error
MT_RET_NOT_AVAILABLE	Driver is not loaded

Events

None

TERMINATE CARDS (T1 RBS, ISDN) FUNCTIONS

The following APIs are used for Terminate ISDN (PRI) on the DT cards.

NOTE - It is important to understand that both ISDN and RBS rely on the same Call Control APIs. A discussion is available in the SmartWORKS Developer's Guide.

MTCC_ConnectRequest()

Description

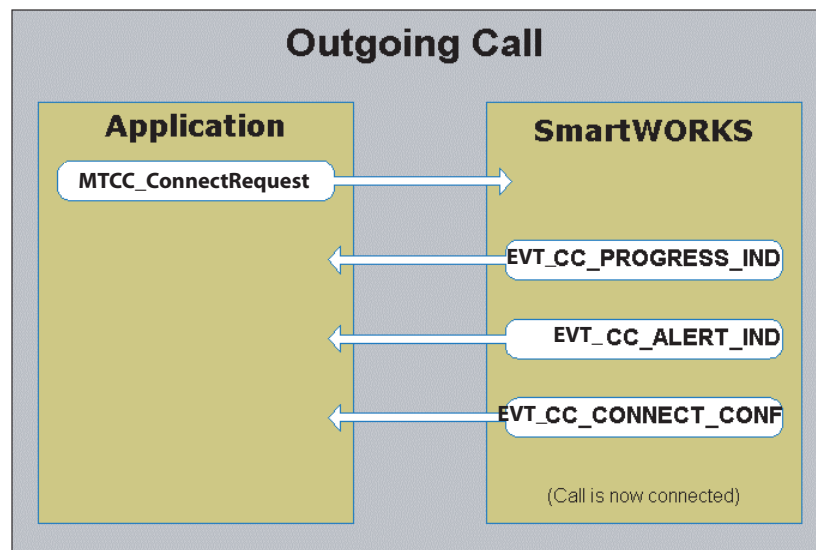
MTCC_ConnectRequest() allows the user application to initiate an outgoing call. The application must allocate a unique Connection ID for the call and format an MT_CC_CONNECT_REQ structure before calling this function. **MTCC_ConnectRequest()** checks the structure content and returns the error code MT_RET_CC_INVALID_REQ if any field is invalid.

Scenarios

Outbound calling number presentation and screening can be controlled using the SmartWORKS API. These options are controlled by the *PresScrInd* field in the MT_CC_CONNECT_REQ structure. Refer to the *Call Processing* section of the SmartWORKS Developer's Guide.

Users can set the user specific protocol used to pass text messages along the ISDN network. All fields are controlled in the MT_CC_U2U structure. **NOTE:** The SmartWORKS API limits the text message to 128 bytes.

Outgoing Call Control shown with call progress indicators.



Prototype

MT_RESULT MTCC_ConnectRequest(

```
const CHANNEL          nChannel,
PMT_CC_CONNECT_REQ const pCallInfo
);
```

Parameters

nChannel channel number

pCallInfo

pointer to MT_CC_CONNECT_REQ structure. This structure, as well as related structures, are defined in the Nti-DataCC.h file.

Return Code

MT_RET_OK	No error
MT_CC_CHANNEL_INUSE	Channel is not idle
MT_CC_NO_DIGITS	No called party number provided
MT_RET_NOT_AVAILABLE	Driver is not loaded

Events

- EVT_CC_PROGRESS_IND - Call progress indication
- EVT_CC_CONNECT_CONF - Call connect confirmation

CODE EXAMPLE

MTCC_ConnectRequest() Code Example

```
/*
                                result =
MTCC_ConnectRequest(channel, &request);
                                if(result == MT_RET_OK)
                                printf("MTCC_ConnectRequest()
successful...\n");
                                else
                                printf("MTCC_ConnectRequest() failed...\n");
                                */
```

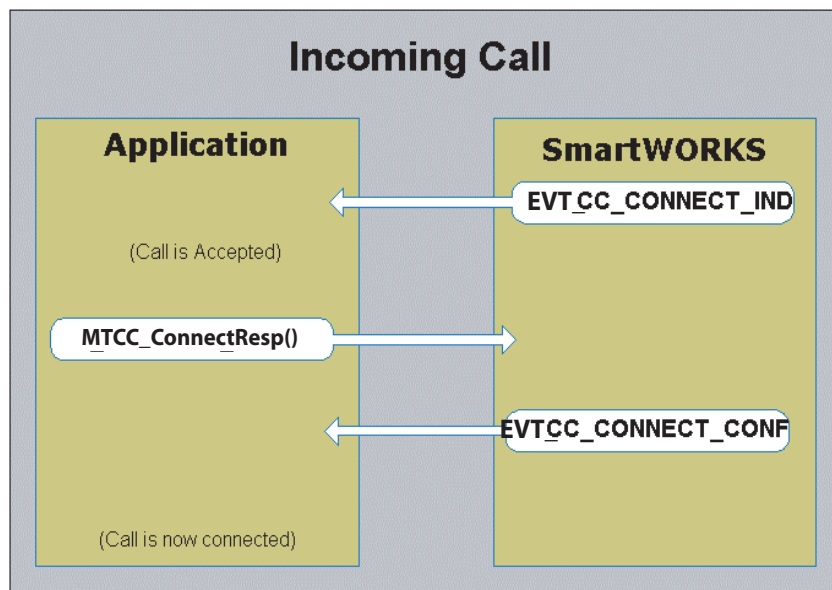
A complete code example is available on the AudioCodes website:
<http://www.audiocodes.com/blades/support.htm>

MTCC_ConnectResp()

Description

MTCC_ConnectResp() allows the user application to accept an incoming call notified by the Call Control event `EVT_CC_CONNECT_IND`. Before accepting the call, the user application must allocate a unique connection ID for this call and associate it with the Call Reference indicated in the `MT_CC_CONNECT_IND` carried by the event `EVT_CC_CONNECT_IND`. **MTCC_ConnectResp()** checks the provided structure and if any field is invalid, it returns the error code `MT_RET_CC_INVALID_REQ`.

Incoming call shown with application responses:



Prototype

MT_RESULT MTCC_ConnectResp(

```

const CHANNEL          nChannel,
PMT_CC_CONNECT_RESP const pCallInfo
);

```

Parameters

nChannel	Channel Number
pCallInfo	Pointer to <code>PMT_CC_CONNECT_RESP</code> structure. This structure, as well as related structures, are defined in the <code>Nti-DataCC.h</code> file.

Return Codes

MT_RET_OK	No error
MT_CC_INVALID_STATE	Channel is not in incoming call state
MT_RET_NOT_AVAILABLE	Driver is not loaded

Events

EV_CC_CONNECT_CONF - Call connect confirmation

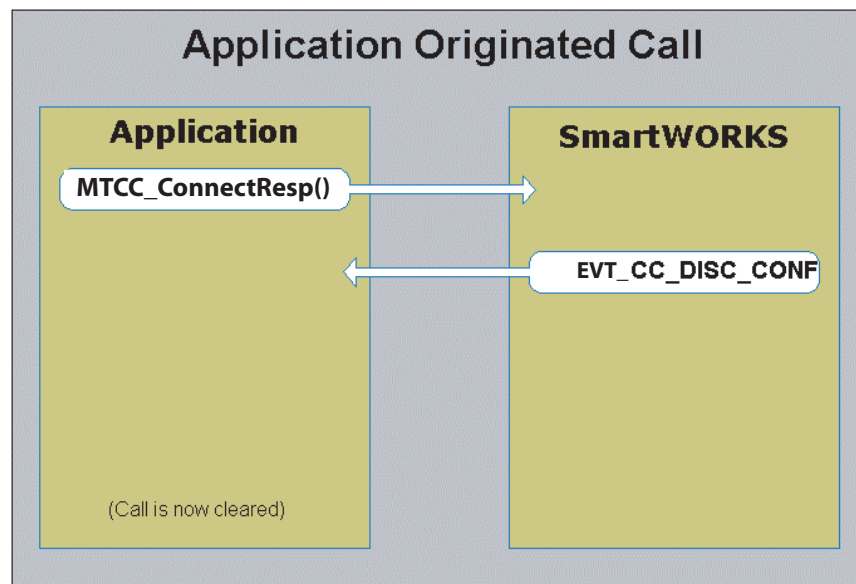
MTCC_DiscRequest()

Description

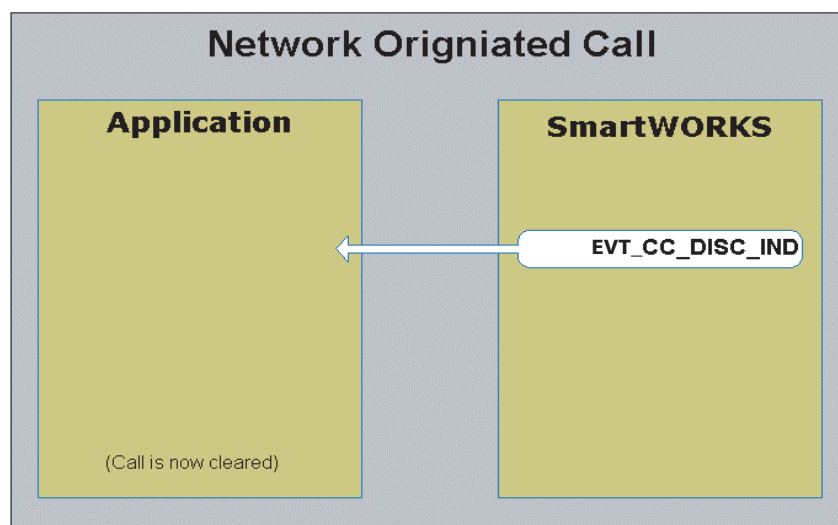
MTCC_DiscRequest() allows the user's application to release an already connected call, to refuse an incoming call or to abort an outgoing call. The application must wait for Call Control event EVT_CC_DISC_CONF to declare the call release complete.

MTCC_DiscRequest() checks the content of the MT_CC_DISC_REQ structure provided and if any field value is invalid, it returns the error code MT_RET_CC_INVALID_REQ

An application initiated call release:



A network initiated call release:



Prototype

MT_RESULT MTCC_DiscRequest(

```
const CHANNEL          nChannel,
PMT_CC_DISC_REQ const  pCallInfo
);
```

Parameters

nChannel	channel number
pCallInfo	pointer to MT_CC_DISC_REQ structure. This structure, as well as related structures, are defined in the NtiDataCC.h file.

Return Code

MT_RET_OK	No error
MT_CC_INVALID_STATE	No call to release
MT_RET_NOT_AVAILABLE	Driver is not loaded

Events

EVT_CC_DISC_CONF - Call release confirmation

TERMINATE CARDS (ISDN) FUNCTIONS

MTCC_FacilityRequest()

Description

This function allows the user application to issue a facility management request. The ISDN supplementary services are mainly provided through this function call. The field Code is used to specify which facility (MT_CC_FAC_HOLD, MT_CC_FAC_RETREIVE, MT_FAC_CALL_FORWARDING,.. etc) . The field Action specifies the action to take (MT_CC_ACTION_ACTIVATE , MT_CC_ACTION_CLEAR or MT_CC_ACTION_ENQUIRY). The other fields are used based on the facility. As result of this request a Facility ISDN messages is sent to the network.

Prototype

MT_RESULT MTCC_FacilityRequest(

```
const CHANNEL          nChannel,
PMT_CC_FACILITY_REQ const pFacilityInfo
);
```

Parameters

nChannel	Channel Number
pFacilityInfo	Pointer to MT_CC_FACILITY_REQ structure. This structure, as well as related structures, are defined in the Nti-DataCC.h file.

Return Codes

MT_RET_OK	No Error
MT_NB_NO_FREE_CONID	if no call resource is available
MT_RET_NOT_AVAILABLE	Driver is not loaded

Events

EVT_CC_FACILITY_CONF - confirms that the request has been executed

MTCC_SuspendRequest()

Description

The user application uses this function to suspend an active call. The application should provide the connection ID and the call reference of the call. A call identity can also be given for the suspended call in order to resume it later.

Prototype

MT_RESULT MTCC_SuspendRequest (

```
const CHANNEL          nChannel,  
PMT_CC_SUSPEND_REQ const pSuspendInfo  
);
```

Parameters

nChannel	Channel Number
pSuspendInfo	Pointer to MT_CC_SUSPEND_REQ structure. This structure, as well as related structures, are defined in the Nti-DataCC.h file.

Return Codes

MT_RET_OK	No Error
MT_NB_INVALID_CONID	if the provided Connection ID is not valid
MT_RET_NOT_AVAILABLE	Driver is not loaded

Events

EVT_CC_SUSPEND_CONF - confirms that the call has been suspended.

MTCC_UserInfoRequest()

Description

This function allows the user application to exchange user to user information. A User information ISDN message (if supported) is sent to the other peer.

Prototype

```
MT_RESULT MTCC_UserInfoRequest (  
    const CHANNEL          nChannel,  
    PMT_CC_USERINFO_REQ const pUserInfo  
);
```

Parameters

nChannel	Channel Number
pUserInfo	Pointer to MT_CC_USERINFO_REQ structure. This structure, as well as related structures, are defined in the Nti-DataCC.h file.

Return Codes

MT_RET_OK	No Error
MT_NB_INVALID_CONID	if the provided Connection ID is not valid
MT_RET_NOT_AVAILABLE	Driver is not loaded

Events

None

MTCC_ResumeRequest()

Description

The application uses this function to resume a suspended call. The application allocates a new connection Id to the call and provides the call identity associated to the suspended call.

Prototype

MT_RESULT MTCC_ResumeRequest (

```
const CHANNEL          nChannel,  
PMT_CC_RESUME_REQ const pResumeInfo  
);
```

Parameters

nChannel	Channel Number
pResumeInfo	Pointer to MT_CC_RESUME_REQ structure. This structure, as well as related structures, are defined in the Nti-DataCC.h file.

Return Codes

MT_RET_OK	No Error
MT_NB_NO_FREE_CONID	if no call resource is available
MT_RET_NOT_AVAILABLE	Driver is not loaded

Events

EVT_CC_RESUME_CONF - confirms that the call has been resumed to active state.

PASSIVE CARDS (ISDN) FUNCTIONS

MTCC_GetStatusByCh()

Description

MTCC_GetStatusByCh() scans all existing ISDN calls to determine which calls are associated with the specified channel number.

MTCC_GetStatusByCh() then returns a complete MT_CC_CALL_INFO for this call. If the channel is free (meaning it's not associated to any ISDN call) this function will return the error code MT_RET_ISDNPT_CHANNEL_NOT_FOUND.

Prototype

MT_RESULT MTCC_GetStatusByCh(

```
const CHANNEL          nChannel,  
PMT_CC_CALL_INFO const pCallInfo,  
PULONG const          pLength  
);
```

Parameters

nChannel	Channel Number
pCallInfo	Pointer MT_CC_CALL_INFO structure. This structure is explained at the beginning of this section.
pLength	Set by the user application, this is a pointer to the size of the given buffer for the MT_CC_CALL_INFO structure.

Return Codes

MT_RET_OK	No error
MT_RET_ISDNPT_CHANNEL_NOT_FOUND	ISDN channel not found
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_CHANNEL_NOT_OPEN	Channel device not opened for access
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_DATA_TRUNCATED	Data copied but truncated; the buffer size set by the user application is not large enough for this API request. <i>Flag-DataTruncated</i> field of the MTSYS_CONFIG or the MTSYS_CONFIGURATION data structure must be enabled (1) to receive this return code, otherwise MT_RET_OK is used, and the truncated data is passed to the user application

Events

None

CODE EXAMPLE**MTCC_GetStatusbyChan() Code Example**

```
ZeroMemory(&callinfo, size);  
  
result =  
MTCC_GetStatusByCh(channel, &callinfo, &size);  
if(result == MT_RET_OK)  
{  
    printf("MTCC_GetStatusByCh()  
successful...\n");  
  
    ULONG callRef = callinfo.CallRef;  
    ULONG callSource = callinfo.CallSource;  
    ULONG callState = callinfo.CallState;  
    printf("Call Reference: %0x\nCall Source:  
%d\nCall State: %d\n", callRef, callSource, callState);  
}
```

A complete code example is available on the AudioCodes website:

<http://www.audiocodes.com/blades/support.htm>

MTCC_GetStatusByRef()

Description

MTCC_GetStatusByRef() returns a complete MT_CC_CALL_INFO for the call specified by the CallRef parameter. If the specified Call reference no longer exists, this function returns the error code MT_RET_ISDNPT_CALL_NOT_FOUND.

Prototype

MT_RESULT MTCC_GetStatusByRef()

```
const ULONG          nCallRef,  
PMT_CC_CALL_INFO const pCallInfo,  
PULONG const        pLength  
);
```

Parameters

nCallRef	ISDN Call reference
pCallInfo	Pointer to MT_CC_CALL_INFO structure. This structure is explained at the beginning of this section.
pLength	Set by the user application, this is a pointer to the size of the given buffer for the MT_CC_CALL_INFO structure.

Return Codes

MT_RET_OK	No error
MT_RET_ISDNPT_CHANNEL_NOT_FOUND	ISDN channel not found, bad call reference
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_DATA_TRUNCATED	Data copied but truncated; the buffer size set by the user application is not large enough for this API request. <i>Flag-DataTruncated</i> field of the MTSYS_CONFIG or the MTSYS_CONFIGURATION data structure must be enabled (1) to receive this return code, otherwise MT_RET_OK is used, and the truncated data is passed to the user application

Events

None

CODE EXAMPLE

MTGetStatusbyRef() Code Example

```
ZeroMemory(&callinfo, size);  
  
result =  
MTCC_GetStatusByRef(callRef, &callinfo, &size);  
if(result == MT_RET_OK)  
{  
    printf("MTCC_GetStatusByRef()  
successful...\n");  
  
    ULONG callSource = callinfo.CallSource;  
    ULONG callState = callinfo.CallState;  
    printf("Call Reference: %0x\nCall Source:  
%d\nCall State: %d\n", callRef, callSource, callState);  
}
```

A complete code example is available on the AudioCodes website:

<http://www.audiocodes.com/blades/support.htm>

MTGetDataLinkStatus()

Description

MTGetDataLinkStatus() returns the data link state for the trunk specified by the board number and trunk number parameters. Data Link Status can be defined as the following:

- MT_DL_UNRESOLVED - not synched up
- MT_DL_ACTIVATED - physical layer has synched up
- MT_DL_ESTABLISHED - data link layer has synched up
- MT_DL_CO_CONNECTING - a Sabme frame was sent from CO. CO is waiting for a reply
- MT_DL_CPE_CONNECTING - a Sabme frame was sent from CPE. CPE is waiting for a reply
- MT_DL_CO_RELEASING - the CO sent a Disconnect(Disc) frame
- MT_DL_CPE_RELEASING - the CPE sent a Disconnect(Disc) frame

Prototype

MT_RESULT MTGetDataLinkStatus(

```
const BOARD      nBoard,  
const USHORT     nTrunk,  
PULONG const     pDLStatus  
);
```

Parameters

nBoard	Board Number
nTrunk	Trunk number within the specified board
pDLStatus	Pointer to the link state

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_NOT_AVAILABLE	Driver is not loaded

Events

None

MTGetDataLinkStatistics()

Description

MTGetDataLinkStatistics() returns the collected Data Link statistics for the trunk specified by the Board Number and Trunk number parameters.
The counts returned correspond to the total number since the channel was last opened. The MT_DL_STATS structure separates both incoming and outgoing data. The structure is returned with incoming data presented first.

Prototype

MT_RESULT MTGetDataLinkStatistics(

```
const BOARD      nBoard,  
const USHORT     nTrunk,  
PMT_DL_STATS const pDLStats,  
PULONG const     pLength  
);
```

Parameters

nBoard	Board Number
nTrunk	Trunk number within the specified board
pDLStats	Pointer to an PMT_DL_STATS structure (defined in the NtiDataCC.h file)
pLength	Set by the user application, this is a pointer to the size of the given buffer for the PMT_DL_STATS structure.

Structure

The MT_DL_STATS structure is defined as the following:
NOTE: The counts returned correspond to the total number since the channel was last opened.

Incoming	
SabmeCnt	the count of Sabme frames received from the outside network
DiscCnt	the count of the disconnect frames sent by the outside network to the local network
RnrCnt	the count of the receive not ready frames that have been sent by the outside network
FrmrCnt	the count of incoming frames rejected. The frame that was received was no good.
Ns_ErrCnt	indicates the number of sequence number errors received on incoming data
CrBitErrCnt	Incoming data was expected from the outside, but came in the format of the local network. This indicates the tapped board is not cabled properly.

Outgoing	
SabmeCnt	the count of Sabme frames received from the local network
DiscCnt	the count of the disconnect frames sent by the local network to the outside network
RnrCnt	the count of the receive not ready frames that have been sent by the local network
FrmrCnt	the count of outgoing frames rejected. The frame that was received was no good.
Ns_ErrCnt	indicates the number of sequence number errors received on outgoing data
CrBitErrCnt	Outgoing data was expected from the local network, but came in the format of the outside network. This indicates the tapped board is not cabled properly.

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_NOT_AVAILABLE	Driver is not loaded

Events

None

MTGetCCStatistics()

Description

MTGetCCStatistics() returns the collected ISDN statistics for the trunk specified by the board number and trunk number parameters.
The counts returned correspond to the total number since the channel was last opened.

Prototype

```
MT_RESULT MTGetCCStatistics(  
    const BOARD          nBoard,  
    const USHORT         nTrunk,  
    PMT_CC_STATS const   pCCStats,  
    PULONG const         pLength  
);
```

Parameters

nBoard	Board Number
nTrunk	Trunk number within the specified board
pCCStats	Pointer to the CC_STATS structure (defined in the NtiDataCC.h file)
pLength	Set by the user application, this is a pointer to the size of the given buffer for the PMT_CC_STATS structure.

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_NOT_AVAILABLE	Driver is not loaded

Events

None

MTGetPHYStatistics()

Description

MTGetPHYStatistics() returns the collected physical layer statistics for the trunk specified by the parameters nBoard and nTrunk.

The counts returned correspond to the total number since the channel was last opened. The MT_PHY_STATS structure separates incoming and outgoing data. Incoming data is presented first.

NOTE: This API works for ISDN only and not on RBS networks.

Prototype

MT_RESULT MTGetPHYStatistics (

```
const BOARD      nBoard,
const USHORT     nTrunk,
PMT_PHY_STATS    pPHYStats, (defined in the NtiDataCC.h
const            file)
PULONG const     pLength
);
```

Parameters

nBoard	Board Number
nTrunk	Trunk Number within the specified Trunk
pPHYStats	a pointer to an MT_PHY_STATS structure
pLength	Set by the user application, this is a pointer to the size of the given buffer for the MT_PHY_STATS structure.

Structure

The MT_PHY_STATS structure is defined as follows:

Incoming

RxOverflowCnt	the count of received overflow errors due to too much incoming traffic on the AudioCodes board
InvalidFramesCnt	the count of invalid frames received from incoming traffic. These may be due to CRC errors or truncated frames. Typically this indicates a problem with the AudioCodes board
ValidFramesCnt	the total count of valid frames received in the incoming direction

Outgoing

RxOverflowCnt	the count of received overflow errors due to too much outgoing traffic on the AudioCodes board

InvalidFramesCnt	the count of invalid frames received from outgoing traffic. These may be due to CRC errors or truncated frames. Typically this indicates a problem with the AudioCodes board
ValidFramesCnt	the total count of valid frames received in the outgoing direction.

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_PARAM	Error in one of the passed parameters
MT_RET_NOT_AVAILABLE	Driver is not loaded

Events

None

CHANNEL FUNCTIONS

MTCallString(), MTCallStringAsync(), MTCallStringEx()

NOTE

If UNICODE is defined SmartWORKS will call the API with the 'W' suffix, if UNICODE is not defined, SmartWORKS will call the API with the 'A' suffix. ie, MTCallStringA or MTCallStringW.

Description

MTCallString() is a background function that automatically calls a number. This function takes the channel off hook, waits for dial tone, dials the number and switches to CPM completion mode to determine if the call is answered.

NOTE: Both Call Progress Monitoring (CPM) and Voice Detection must be enabled before **MTCallString()** can be invoked.

Since **MTCallString()** is a background function, it is controlled with parameters set in the IO_CONTROL structure. Only the CPM_MODE, MaxTime, TermEnable, and MoniEnable fields are used by **MTCallString()**. The CPM_MODE field behaves exactly like the media task settings. Here the user sets whether monitoring and termination conditions are based on channel level settings, task only, or both. The user application is responsible for initializing the IO_CONTROL structure to proper settings. If a zero (0) is passed into the CPM_MODE field then the setting TERM_USE_NONE and MONI_USE_CHANNEL_ONLY is applied.

Since it is a background function, **MTCallString()** stops in the event that a termination condition is met. When a termination condition exists, this background functions stops and the event EVT_CPM_STOP is generated. The SubReason and XtraInfo fields of the MT_EVENT structure indicate the exact CPM termination type (e.g. CPM_BUSY, CPM_NO_ANSWER, CPM_NO_RINGBACK, CPM_CONNECT, CPM_INTERCEPT, CPM_NO_DIAL). The FuncCode field identifies that **MTCallString()** has terminated. When certain line conditions are detected, **MTCallString()** places the channel back ON_HOOK and then the function terminates. If the call remains connected, the user application is responsible for placing the channel back ON_HOOK.

A complete description, including examples of call scenarios is available in the application note: *Managing CPM Events*.

MTCallString() is only supported on ANALOG_VFXO_LS_E types on the SmartWORKS LD card.

MTCallString(), **MTCallStringAsync()**, and **MTCallStringEx()** will actually map to modified versions of themselves depending on whether or not UNICODE is defined for the user application. See "UNICODE Support" for more information about APIs affected by defining UNCODE in a user application.

Asynchronous Call Back

Any API that ends with the suffix Async uses callback as its completion notification method. Each requires a completion routine from the user application. The SmartWORKS function completion routine mechanism is implemented in the same fashion as a WindowsNT system function completion. In addition to the completion routine address, the application also passes a parameter of type LPARAM to the SmartWORKS DLL/Driver, which passes the parameter back, without change, to the LPARAM parameter.

Prototype

MT_RESULT MTCallString (

```
const CHANNEL          nChannel,
PCSTR (A), or PCWSTR(W) pCall_string,
const PMT_IO_CONTROL   ploCtl
);
```

MT_RESULT MTCallStringEx

```
const CHANNEL          nChannel,
PCSTR(A), or PCWSTR(W) pCall_string,
const PMT_IO_CONTROL   ploCtl,
const LPOVERLAPPED     pOverlapped
);
```

MT_RESULT MTCallStringAsync

```
const CHANNEL          nChannel,
PCSTR(A), or PCWSTR(W) pCall_string,
const PMT_IO_CONTROL   ploCtl,
const MTCOMPLETE        pCompletionRoutine,
const LPARAM            lParam
);
```

Parameters

nChannel	Channel number
pCall_string	pointer to a zero terminated dial string, pCall_string
ploCtl	pointer to a The MT_IO_CONTROL Data Structure structure (defined in the previous chapter of this book).
pCompletionRoutine	pointer to completion routine
lParam	parameter to echo back to completion

Return Codes

MT_RET_OK	No error
MT_RET_IO_PENDING	Background I/O operation is in progress or queued
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_CHANNEL_NOT_OPENED	Channel device not opened for access
MT_RET_NOT_AVAILABLE	Driver is not loaded

Events

EVT_CPM_STOP

CODE EXAMPLE**MTCallString() Code Example**

```
result = MTCallString(channel,buffer,&ioctrl);
if(result == MT_RET_OK || result ==
MT_RET_IO_PENDING)
{
    printf("MTCallString() successful...\n");
    BOOL waiting = true;
    while(waiting)
    {
        // wait for event with 10 second timeout
        result =
MTWaitForChannelEvent(channel,10000,&event,&evtsz);
        if(result == MT_RET_OK)
        {
```

A complete code example is available on the AudioCodes website:

<http://www.audiocodes.com/blades/support.htm>

MTDialString(), MTDialStringAsync(), MTDialStringEx()

NOTE

If UNICODE is defined SmartWORKS will call the API with the 'W' suffix, if UNICODE is not defined, SmartWORKS will call the API with the 'A' suffix. ie, MTDialStringA or MTDialStringW.

Description

MTDialString() is a background function that dials the specified string. The default dialing is in DTMF mode and can be switched to MF mode with the M control character. Change to pulse dialing mode with the P control character. The string must be zero terminated. The maximum length of the string buffer is 128 bytes, therefore, the maximum number of string characters is 127. This function does not perform Call Progress Monitoring.

Option '&', and 'H' are supported only on *ANALOG_FXO_LS* and *ANALOG_VFXO_LS_Echannels*. The *MT_IO_CONTROL.MaxTime* field is used for caller to specify the maximum time to wait for option 'W' or 'L'.

MTDialStringEx() and **MTDialStringAsync()** are functionally the same as **MTDialString()**. **MTDialStringEx()** uses overlapped structure to carry the event pointer for DLL to signal the application of the completion of this background function;

MTDialStringAsync() uses a call back function as the completion notification mechanism. The long word user parameter will be passed, without altering, as a parameter to the application provided completion function.

MTDialString(), **MTDialStringAsync()**, and **MTDialStringEx()** will actually map to modified versions of themselves depending on whether or not UNICODE is defined for the user application. See "UNICODE Support" on page 23 for more information about APIs affected by defining UNICODE in a user application.

Asynchronous Call Back

Any API that ends with the suffix Async uses callback as its completion notification method. Each requires a completion routine from the user application to notify completion of the requested API. The SmartWORKS function completion routine mechanism is implemented in the same fashion as a WindowsNT system function completion. In addition to the completion routine address, the application also passes a parameter of type LPARAM to the SmartWORKS DLL/Driver, which passes the parameter back without change.

Prototypes**MT_RESULT MTDialString (**

```

const CHANNEL          nChannel,
const PCSTR (A), or    pDial_string,
const PCWSTR(W)
const PMT_IO_CONTROL   ploCtl
);

```

MT_RESULT MTDialStringEx (

```

const CHANNEL          nChannel,
const PCSTR (A) or     pDial_string,
const PCWSTR(W)
const PMT_IO_CONTROL   ploCtl,
const LPOVERLAPPED     pOverlapped
);

```

MT_RESULT MTDialStringAsync (

```

const CHANNEL          nChannel,
const PCSTR(A) or      pDial_string,
const PCWSTR (W)
const PMT_IO_CONTROL   ploCtl,
const MTCOMPLETE       pCompletionRoutine,
const LPARAM           lParam
);

```

Parameters

nChannel	Channel number to dial out to
pDial_string	Null terminated string of ASCII digits representing the digits to dial. The special digits are defined below.
ploCtl	pointer to a The MT_IO_CONTROL Data Structure structure (defined in the previous chapter of this book). The MT_IO_CONTROL.MaxTime field provides timeout specification for dialing.
pOverlapped	Pointer to NT overlapped structure
pCompleteRoutine	Completion function address
lParam	Long word user parameter

Special Characters

,	Pause (or the character specified through the MT_PSTN.pause_char field)
&	Hook flash (or the character specified through MT_PSTN.flash_char field), supported only ANALOG_VFXO_LS_E (SmartWORKS LD) channels.
W	Wa ANALOG_VFXO_LS_E (SmartWORKS LD) channels. (NOTE: Call Progress Monitoring must be enabled for dial tone detection).

H	Used to go OFF_HOOK if the line is ON_HOOK
L	Wait for loop current detected

Return Codes

MT_RET_OK	No error
MT_RET_IO_PENDING	Background I/O operation is in progress or queued
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_CHANNEL_NOT_OPENED	Channel device not opened for access
MT_RET_NOT_AVAILABLE	Driver is not loaded

Events

EVT_DIALED

CODE EXAMPLE

MTDialString() Code Example

```
result = MTDialString(channel,buffer,&ioctrl);
if(result == MT_RET_OK || result ==
MT_RET_IO_PENDING)
{
printf("MTDialString() successful...\n");
result =
MTWaitForChannelEvent(channel,5000,&event,&evtsize);
if(result != MT_RET_OK)
printf("MTWaitForChannelEvent() timed
out...\n");
}
```

A complete code example is available on the AudioCodes website:

<http://www.audiocodes.com/blades/support.htm>

MTDisableCallerID()

As of release 3.11, this function has been replaced by ***MTChCallerIDControl()***. This function will be maintained for backwards compatibility; however AudioCodes recommends that new application development should use ***MTChCallerIDControl()***.

Description

Channel caller ID detection is disabled. Queued caller ID packets are not removed by this function.

Prototype

```
MT_RESULT MTDisableCallerID (  
    const CHANNEL      nChannel  
);
```

Parameters

nChannel	channel number

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_CHANNEL_NOT_OPENED	Channel device not opened for access
MT_RET_NOT_AVAILABLE	Driver is not loaded

Events

None

MTEnableCallerID()

As of release 3.11, this function has been replaced by **MTChCallerIDControl()**. This function will be maintained for backwards compatibility; however AudioCodes recommends that new application development should use **MTChCallerIDControl()**.

Description

CallerID detection is enabled by default with a sensitivity level set to 0x80. Each channel is able to store up to 15 CallerID packets. Once the queue is full, new CallerID information will be lost.

Each successive call to MTEnableCallerID() will set the detection sensitivity to the level specified. The default value of 0x80 provides moderate sensitivity for typical applications. Reducing the value to 0x02 provides detection of low-level signals but may decrease the noise immunity of the detection algorithm.

Scenario

When caller ID detection is enabled, EVT_CALLID_STOP signals the reception of a well formed 1200 bps FSK data packet. The packet formats for both on-hook and off-hook (call waiting, caller ID) are supported.

Upon reception of event EVT_CALLID_STOP, use MTGetCallerID() to retrieve the data packet.

Prototype

```
MT_RESULT MTEnableCallerID (  
    const CHANNEL      nChannel,  
    const USHORT       nLevel  
);
```

Parameters

nChannel	channel number
nLevel	signal sensitivity level (0x02 - 0x80)

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_CHANNEL_NOT_OPENED	Channel device not opened for access
MT_RET_NOT_AVAILABLE	Driver is not loaded

Events

EVT_CALLID_DROPPED
EVT_CALLID_STOP

MTGetCallerIDStatus()

As of release 3.11, this function has been replaced by **MTChGetCallerIDStatus()**. This function will be maintained for backwards compatibility; however AudioCodes recommends that new application development should use **MTChGetCallerIDStatus()**.

Description

CallerID detection operation state and level is retrieved for the specified channel.

The caller ID detection operation state can be the following values:

- 1 caller ID detection is enabled on the channel
- 0 caller ID detection is disabled on the channel

The caller ID detection signal sensitivity level is between 0x02 and 0x80.

Prototype

MT_RESULT MTGetCallerIDStatus (

```
const CHANNEL      nChannel,  
PUSHORT const     pState,  
PUSHORT const     pLevel  
);
```

Parameters

nChannel	channel number
pState	Operation state: 1 = caller ID is enabled 0 - caller ID is disabled
pLevel	Signal sensitivity level: 0x02-0x80.

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_CHANNEL_NOT_OPENED	Channel device not opened for access
MT_RET_NOT_AVAILABLE	Driver is not loaded

Events

None

MTChSetCallerIDParams()

As of release 3.11, this function replaces by *MTEnableCallerID*).

Description

By default, each channel is configured where North American CallerID detection is enabled with a sensitivity level set to 0x80. Each channel is able to store up to 15 CallerID packets. Once the queue is full, new CallerID information will be lost.

To change Caller ID settings, users can invoke this function and modify any parameters.

Scenario

When caller ID detection is enabled, EVT_CALLID_STOP signals the reception of a well formed data packet. The packet formats for both on-hook and off-hook (call waiting, caller ID) are supported.

Upon reception of event EVT_CALLID_STOP, use *MTGetCallerID*() to retrieve the data packet.

In the event that a CallerID signal is detected, but the checksum validation fails, the user application receives the EVT_CALLID_CSUMERR event but no EVT_CALLID_STOP.

Prototype

MT_RESULT MTChSetCallerIDParams(

```
const CHANNEL      nChannel,
PMT_CIDPARAMS      pCid,
PULONG             pLength
);
```

Parameters

nChannel	channel number
pCid	pointer to the MT_CIDPARAMS data structure
pLength	length of data structure

Structure

Type	Name	Purpose
MT_CID_TYPE	type	select the type of caller ID; CIDTYPE_BELL202, CIDTYPE_NTT
USHORT	level	set the level
MT_CID_TWIST	twist	refer to all values as described in the next table (NOTE: if detected NTT caller ID, then setting a twist level has no effect. Only twist levels ranging from -3 to +7 dB are detected

Twist	Minimum dB	Maximum dB
CIDTWIST_NEG	-10	+3
CIDTWIST_ZERO	-6	+6
CIDTWIST_POS	-3	+10

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_CHANNEL_NOT_OPENED	Channel device not opened for access
MT_RET_NOT_AVAILABLE	Driver is not loaded

MTChGetCallerIDParams()

Description

Returns status and configuration parameters of the CallerID feature.

NOTE: Japanese (NTT) Caller ID has only been tested on the LD boards.

Prototype

MT_RESULT MTChGetCallerIDStatus (

```
const CHANNEL      nChannel,
PMT_CIDPARAMS      pCid,
PULONG             pLength
);
```

Parameters

nChannel	channel number
pCid	pointer to MT_CIDPARAMS data structure
pLength	length of data structure

Structure

Type	Name	Purpose
MT_CID_TYPE	type	select the type of caller ID; CIDTYPE_BELL202, CIDTYPE_NTT
USHORT	level	set the level
MT_CID_TWIST	twist	refer to all values as described in the next table (NOTE: if detected NTT caller ID, then setting a twist level has no effect. Only twist levels ranging from -3 to +7 dB are detected)

Twist	Minimum dB	Maximum dB
CIDTWIST_NEG	-10	+3
CIDTWIST_ZERO	-6	+6
CIDTWIST_POS	-3	+10

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_CHANNEL_NOT_OPENED	Channel device not opened for access
MT_RET_NOT_AVAILABLE	Driver is not loaded

MTChCallerIDControl()

As of release 3.11, this function replaces **MTEnableCallerID()**.

Description

CallerID detection is enabled by default with a sensitivity level set to 0x80. Each channel is able to store up to 15 CallerID packets. Once the queue is full, new CallerID information will be lost.

Use this function to modify the status of CallerID .

Scenario

When Caller ID detection is enabled, EVT_CALLID_STOP signals the reception of a well formed data packet. The packet formats for both on-hook and off-hook Caller ID are supported.

Upon reception of event EVT_CALLID_STOP, use **MTGetCallerID()** to retrieve the data packet.

Prototype

MT_RESULT MTChCallerIDControl (

```
const CHANNEL      nChannel,  
const BOOLEAN      fOn  
);
```

Parameters

nChannel	channel number
fOn	true = 1, false (off) = 0

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_CHANNEL_NOT_OPENED	Channel device not opened for access
MT_RET_NOT_AVAILABLE	Driver is not loaded

Events

EVT_CALLID_DROPPED
EVT_CALLID_STOP
EVT_CALLID_CSUMERR

MTChCallerIDStatus

As of release 3.11, this function replaces *MTGetCallerIDStatus()*.

Description

Obtain the status of the CallerID feature.

Prototype

```
MT_RESULT MTChCallerIDStatus (  
    const CHANNEL      nChannel,  
    const BOOLEAN      fOn  
);
```

Parameters

nChannel	channel number
fOn	true = 1, false (off) = 0

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_CHANNEL_NOT_OPENED	Channel device not opened for access
MT_RET_NOT_AVAILABLE	Driver is not loaded

Events

None

MTGetCallerID()

Description

This function returns the detected caller ID packet data string up to the length provided in the length field. The suggested buffer should be 256 bytes to take the maximum possible ID information. If the buffer is not large enough, the portion of caller ID unable to fit into the buffer is lost.

Each channel can queue up to 15 caller ID data packets. All caller ID information packets are queued in full, i.e. from the packet type byte to the checksum byte.

Scenario

Event EVT_CALLID_STOP displays the length of the received caller ID data packet in the SubReason field. User applications can use this information to allocate the buffer size for **MTGetCallerID()**.

Prototype

MT_RESULT MTGetCallerID (

const CHANNEL	nChannel,
PUSHORT const	pStringLen,
PUCHAR const	pIDString
);	

Parameters

nChannel	Channel Number
pStringLen	length of buffer provided and length of packet on return
pIDString	pointer to buffer for caller ID data string

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_CHANNEL_NOT_OPENED	Channel device not opened for access
MT_RET_NOT_AVAILABLE	Driver is not loaded

Events

None

MTFlushCallerID()

Description

This function removes all queued caller ID packet(s) for the specified channel.

Prototype

```
MT_RESULT MTFlushCallerID (  
    const CHANNEL          nChannel  
);
```

Parameter

nChannel	channel number

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_CHANNEL_NOT_OPENED	Channel device not opened for access
MT_RET_NOT_AVAILABLE	Driver is not loaded

Events

None

MTGetCallStatus()

This API was formerly called **MTGetCallResult()**

Description

MTGetCallStatus() returns the result of the last CPM function termination. When a function terminates with the EVT_CPM_STOP event, the SubReason field of the MT_EVENT structure contains the precise termination reason. This value can also be found in the CpmTerm field of the CALL_RESULT structure.

The following table lists all the values of the SubReason field of the MT_EVENT structure.

Event	Description
CPM_BUSY	Busy signal detected
CPM_NO_ANSWER	No answer after number of rings
CPM_NO_RINGBACK	No ringback after timeout
CPM_CONNECT	Call is answered
CPM_INTERCEPT	Special Intercept Tone (SIT) detected
CPM_BUSY_NORMAL	Regular (slow) busy cadence
CPM_BUSY_TRUNK	Trunk (fast) busy cadence
CPM_CONNECT_FAX	Connect to a FAX machine
CPM_CONNECT_MODEM	Connect to a modem
CPM_CONNECT_MACHINE	Connect to an answering machine
CPM_CONNECT_LOOP	Connect because of loop-drop
CPM_CONNECT_VOICE	Connect to a voice answer
CPM_CONNECT_CADBRK	Connect to a cadence break

The CALL_RESULT structure:

Type	Name	Purpose
ULONG	Status	status value
ULONG	CpmTerm	Last CPM termination
ULONG	offFreq	Percent of signal off-frequency
ULONG	Frequency	Detected frequency
ULONG	HiTime	Longest high period in cadence
ULONG	LoTimeShort	Shortest low period in cadence
ULONG	LoTimeLong	Longest low period in cadence
ULONG	AnswerTime	Duration of answer for CONNECT
ULONG	SubTerm	Termination sub type

Currently only the CpmTerm and SubTerm fields are supported.

Prototype

MT_RESULT MTGetCallStatus (

```
const CHANNEL      nChannel,
const PCALL_RESULT pCall_result,
PULONG const       pLength
);
```

Parameters

nChannel	channel number
pCall_result	pointer to CALL_RESULT structure
pLength	Set by the user application, this is a pointer to the size of the given buffer for the CALL_RESULT structure.

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_CHANNEL_NOT_OPENED	Channel device not opened for access
MT_RET_DATA_TRUNCATED	Data copied but truncated; the buffer size set by the user application is not large enough for this API request. <i>Flag-DataTruncated</i> field of the MTSYS_CONFIG or the MTSYS_CONFIGURATION data structure must be enabled (1) to receive this return code, otherwise MT_RET_OK is used, and the truncated data is passed to the user application
MT_RET_NOT_AVAILABLE	Driver is not loaded

Events

None

CODE EXAMPLE

MTGetCallStatus() Code Example

```
ZeroMemory(&callresult, size);  
  
MTGetCallStatus(channel, &callresult, &size);  
if(result == MT_RET_OK)  
{  
    printf("MTGetCallStatus() successful...\n");  
    ULONG status = callresult.Status;  
    ULONG termination = callresult.CpmTerm;  
    printf("Status: %u\nLast CPM termination:  
%u\n", status, termination);  
}
```

A complete code example is available on the AudioCodes website:
<http://www.audiocodes.com/blades/support.htm>

MTGetDigits(), MTGetDigitsAsync(), MTGetDigitsEx()

Description

MTGetDigits() is a background function that waits for one of the termination conditions as specified by the `MT_IO_CONTROL` structure. Set the buffer parameter of `MT_IO_CONTROL` to point to a buffer where the API will place the received digits. Set the length parameter to the size of the buffer. When this function terminates, the string will be zero terminated, even if no digits are received. If a digit matches a termination condition, all preceding digits plus the terminating digit will be stored to the user buffer. If any other termination event occurs before a digit termination, all received digits are stored into the buffer.

It's possible at the start of this function that a digit termination condition already exists because digits may already be in the queue. The background function transfers the digits and terminates immediately. If this happens, there may still be digits in the queue that were received after the terminating digit.

MTGetDigitsEx() and **MTGetDigitsAsync()** are functionally the same as **MTGetDigits()**. The difference is the applied completion notification mechanism. **MTGetDigits()** specifies no completion notification. The user application receives events, such as `EVT_TERMDIGIT`, `EVT_MAXDIGITS` and etc., based on the termination condition. **MTGetDigitsEx()** uses the Windows overlapped structure to pass into the API the event the application must wait for to complete the request. When DTMF digit collection terminates, the API issues a signal to the user application to examine the digit result in the provided digit buffer.

MTGetDigitsAsync() uses call back notification, where at the termination of a request, the provided completion function will be invoked with the channel index, termination event, and the user provided long word parameter.

NOTE:

When **MTGetDigits()**, **MTGetDigitsAsync()**, or **MTGetDigitsEx()** is called, the event `EVT_DIGIT_PRESSED` will no longer be sent as due to the fact that the digits pressed will be sent to a buffer until the particular API terminates. When the API terminates, a termination event will be sent and the digits in the buffer will be available for use by the user application.

Asynchronous Call Back

Any API that ends with the suffix `Async` uses callback as its completion notification method. Each requires a completion routine from the user application to notify completion of the requested API. The SmartWORKS function completion routine mechanism is implemented in the same fashion as a WindowsNT system function completion. In addition to the completion routine address, the application also passes a parameter of type `LPARAM` to the SmartWORKS DLL/Driver, which, in turn, passes the parameter back without change to the `LPARAM` parameter.

Scenario

Upon using the completion notification mechanism (i.e. **MTGetDigitsAsync()**) the termination event will not only be passed to the application through the invoking of the completion routine, but will also be reported to the event queue if enabled.

Prototype

MT_RESULT MTGetDigits (

```
const CHANNEL          nChannel,
const PMT_IO_CONTROL   ploCtl
);
```

MT_RESULT MTGetDigitsEx (

```
const CHANNEL          nChannel,
const PMT_IO_CONTROL   ploCtl,
const MTCOMPLETE       pCompletionRoutine,
const LPOVERLAPPED     pOverlapped
);
```

MT_RESULT MTGetDigitsAsync (

```
const CHANNEL          nChannel,
const PMT_IO_CONTROL   ploCtl,
const MTCOMPLETE       pCompletionRoutine,
const LPARAM           lParam
);
```

Parameters

nChannel	channel number
ploCtl	pointer to a The MT_IO_CONTROL Data Structure structure (defined in the previous chapter of this book).
pOverlapped	pointer to NT overlapped structure
pCompletionRoutine	completion function address
lParam	pointer to long word user parameter

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_CHANNEL_NOT_OPENED	Channel device not opened for access
MT_RET_IO_PENDING	Background I/O operation is in progress or queued
MT_RET_NOT_AVAILABLE	Driver is not loaded

Events

None

CODE EXAMPLE**MTGetDigits() Code Example**

```
result = MTGetDigits(channel,&ioctrl);
                                     if(result == MT_RET_OK || result ==
MT_RET_IO_PENDING)
                                     {
                                     printf("MTGetDigits() successful...\n");
                                     result =
MTWaitForChannelEvent(channel,5000,&event,&evtsize);
                                     if(result != MT_RET_OK)
                                     printf("MTWaitForChannelEvent() timed
out...\n");
                                     }
```

A complete code example is available on the AudioCodes website:

<http://www.audiocodes.com/blades/support.htm>

MTReadDigit()

Description

MTReadDigit() retrieves the first digit in the DTMF queue to the UCHAR buffer. If the queue is empty, UCHAR buffer is set to 0x00. Otherwise, UCHAR contains one of 16 possible characters: 0 through 9, #, *, and A through D.

Scenario

The digit direction information is lost after the DTMF digit is retrieved through this API call.

Prototype

MT_RESULT MTReadDigit (
 const CHANNEL nChannel,
 const PUCHAR pDigit
);

Parameters

nChannel	channel number
pDigit	pointer to a character that will receive the digit

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_CHANNEL_NOT_OPENED	Channel device not opened for access
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_API_THREAD_PROTECTED	Another thread is currently accessing this API

Events

None

CODE EXAMPLE

MTReadDigit() Code Example

```
result = MTReadDigit(channel,&digit);  
    if(result == MT_RET_OK)  
    {  
        printf("MTReadDigit() successful...\n");  
        if(digit)  
            printf("DTMF digit: %c\n",digit);  
        else  
            printf("No digits currently  
available...\n");  
    }
```

A complete code example is available on the AudioCodes website:

<http://www.audiocodes.com/blades/support.htm>

MTReadDTMFTone()

Description

MTReadDTMFTone() retrieves the first digit in the DTMF queue to the UCHAR digit buffer. If the queue is empty, UCHAR buffer is set to 0x00. Otherwise, UCHAR contains one of 16 possible characters: 0 through 9, #, *, and A through D.

When queued, information about whether the DTMF digit is received on the primary or the secondary channel is also recorded. Digit direction buffer is provided to retrieve the DTMF digit direction info. It could be of the following values:

0	Primary channel
1	Secondary channel

Prototype

MT_RESULT MTReadDTMFTone (

```
const CHANNEL      nChannel,  
const PCHAR        pTone,  
const PCHAR        pDirection  
);
```

Parameters

nChannel	channel number
pTone	pointer to a character DTMF digit buffer
pDirection	pointer to a character DTMF digit direction buffer

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_CHANNEL_NOT_OPENED	Channel device not opened for access
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_API_THREAD_PROTECTED	Another thread is currently accessing this API

Events

None

MTReadMFTone()

Description

MTReadMFTone() retrieves the first tone in the MF queue to the UCHAR tone buffer. If the queue is empty, UCHAR buffer is set to 0x00. Otherwise, UCHAR contains values 0x01 to 0x0f for forward tones 1 to 15 and 0x11 to 0x1f for backward tones 1 to15, respectively.

When queued, information about whether the MF tone is received on the primary or the secondary channel is also recorded. Tone direction buffer is provided to retrieve the MF tone direction info. It could be of the following values:

- 0 Primary channel
- 1 Secondary channel

Prototype

```
MT_RESULT MTReadMFTone (  
    const CHANNEL      nChannel,  
    const PCHAR        pTone,  
    const PCHAR        pDirection  
);
```

Parameters

nChannel	channel number
pTone	pointer to a character MF tone buffer
pDirection	pointer to a character MF tone direction buffer

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_CHANNEL_NOT_OPENED	Channel device not opened for access
MT_RET_NOT_AVAILABLE	Driver is not loaded

Events

None

MTClearDTMFDigits()

This API was previously called ***MTClearDigits()***

Description

MTClearDTMFDigits() flushes the digit queue of all DTMF digits received in the specified channel.

The queue allows a maximum of 256 digits.

Scenario

Before an application needs new digits from the caller, it can flush the queue so it will not use old digits.

Prototype

MT_RESULT MTClearDTMFDigits (

const channel nChannel
);

Parameter

nChannel channel number

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_CHANNEL_NOT_OPENED	Channel device not opened for access
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_API_THREAD_PROTECTED	Another thread is currently accessing this API

Events

None

CODE EXAMPLE

MTClearDTMFDigits() Code Example

```
result = MTClearDTMFDigits(channel);
        if(result == MT_RET_OK)
        {
            printf("MTClearDTMFDigits()
successful...\n");
            printf("DTMF digits flushed on channel
%d\n", channel);
        }
```

A complete code example is available on the AudioCodes website:

<http://www.audiocodes.com/blades/support.htm>

MTClearMFTones()

Description

MTClearMFTones() flushes the MF tone queue of all tones received in the specified channel.

Prototype

```
MT_RESULT MTClearMFTones (  
    const channel          nChannel  
);
```

Parameters

nChannel	Channel Number

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_CHANNEL_NOT_OPENED	Channel device not opened for access
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_API_THREAD_PROTECTED	Another thread is currently accessing this API

Event Codes

None

MTChInputToneDetectControl()

MTEnableMixingDetect() was previously used.

Description

Parameter tone type specifies whether this is to control DTMF queue or MF queue per each input of a channel.

Possible values for queue control are as follows:

Q_DISABLED Tone detection on the specified queue is disabled; no event reporting and tone queuing performed.

Q_FLUSH Tone detection on the specified queue is enabled without tones queued. Tones are reported through channel events.

Q_KEEP Tone detection on the specified queue is enabled with tones queued. Tones are also reported through channel events.

Prototype

MT_RESULT MTChInputToneDetectControl (

```
const CHANNEL      nChannel,
const SHORT        nInput,
const UCHAR        toneType,
const ULONG        queueControl
);
```

Parameter

nChannel	channel number
nInput	Input index: 0=primary,1=secondary
toneType	tone type, TYPE_DTMF or TYPE_MF
queueControl	queue control

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_CHANNEL_NOT_OPENED	Channel device not opened for access
MT_RET_NOT_AVAILABLE	Driver is not loaded

Events

None

See Also: [MTSetToneDetectParams\(\)](#)

MTChInputToneDetectStatus()

Description

Retreives the queue control status of each input per channel.

Possible values for queue control are as follows:

- Q_DISABLED Tone detection on the specified queue is disabled; no event reporting and tone queuing performed.
- Q_FLUSH Tone detection on the specified queue is enabled without tones queued. Tones are reported through channel events.
- Q_KEEP Tone detection on the specified queue is enabled with tones queued. Tones are also reported through channel events.

Prototype

MT_RESULT MTChInputToneDetectStatus (

const CHANNEL nChannel,

const SHORT nInput,

const UCHAR toneType,

PULONG const pQueueStatus

);

Parameter

nChannel	channel number
nInput	Input index: 0=primary,1=secondary
toneType	tone type, TYPE_DTMF or TYPE_MF
pQueueStatus	Pointer to the queue status value

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_CHANNEL_NOT_OPENED	Channel device not opened for access
MT_RET_NOT_AVAILABLE	Driver is not loaded

Events

None

See Also: [MTSetToneDetectParams\(\)](#)

MTSetOutputSource()

Description

MTSetOutputSource() selects the source of channel output from either the decoder (default) or the mixed inputs to the channel.

Possible output source values are:

CHANNEL_DEVICE - The default decoder

AUDIO_STREAM - The mixed input (NT+TE)

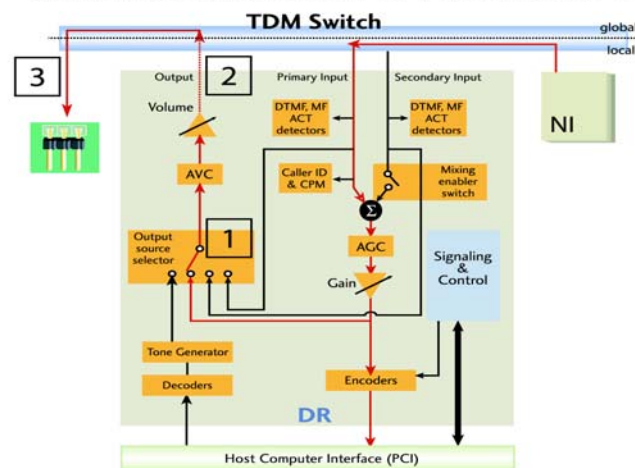
PRIMARY_INPUT - Incoming signal (NT)

SECONDARY_INPUT - Outgoing signal (TE)

The following diagram shows the logical use of the

MTSetOutputSource() API.

Using MTSetOutputSource() for live monitoring while recording



STEP 1: Set Source and Assign Timeslot

MTSetOutputSource

Source = *AUDIO_STREAM* (recommended setting)

CHANNEL_DEVICE

Primary

Secondary

nChannel = GCI index

STEP 2: Connect Channel Output to CT Bus

MTSetOutput

nChannel = GCI index

iStream = stream number

iSlot = time slot number on bus

STEP 3: Set Audio Jack to Receive Input from the CT Bus timeslot number that was specified in STEP 2.

MTAJListen

iAudioJack = audio jack index

iStream = Index of TDM stream

iSlot = Index of TDM slot on a stream

If connection to Audio Jack needs to be broken, use [MTRResetAllListen](#)

To re-establish connection to the NI, the link to theTDM needs to be broken, use [MTRResetOutput](#). To prevent an echo on the line use [MTSetOutputSource](#) and set *Source* back to *Channel_Device*.

NOTE: In this scenario, the channel output is not returned through the Network Interface (NI). This setup also bypasses the tone generator when using boards that support this feature. This is the recommended output method for the SmartWORKS NGX and DP boards.

Prototypes

MT_RESULT MTSetOutputSource(

```
const CHANNEL      nChannel,  
const USHORT      Source  
);
```

Parameters

nChannel	Channel Number
Source	Output Source Value: 0 = CHANNEL_DEVICE 1 = AUDIO_STREAM 2 = PRIMARY_INPUT 3 = SECONDARY_INPUT

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_CHANNEL_NOT_OPENED	Channel device not opened for access
MT_RET_NOT_AVAILABLE	Driver is not loaded

Events

None

MTGetOutputSource()

Description

MTGetOutputSource() selects the source of channel output from either the decoder (default) or the mixed inputs to the channel. The **MTGetOutputSource()** API obtains these settings.

Possible output source values are:

CHANNEL_DEVICE - The default decoder

AUDIO_STREAM - The mixed input (NT+TE)

PRIMARY_INPUT - Incoming signal (NT)

SECONDARY_INPUT - Outgoing signal (ET)

Prototypes

MT_RESULT MTGetOutputSource(

```
const CHANNEL      nChannel,  
const USHORT      pSource  
);
```

Parameters

nChannel	Channel Number
pSource	Pointer to the output source value

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_PARAM	Invalid parameter passed to function
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_CHANNEL_NOT_OPENED	Channel device not opened for access
MT_RET_NOT_AVAILABLE	Driver is not loaded

Events

None

GAIN/VOLUME FUNCTIONS

MTSetAGC()

Description

MTSetAGC() sets the input Automatic Gain Control parameters for both inputs on the channel.

Parameter TMA (target maximum amplitude) ranges from 1 to -45 dBm. The default is 0 dBm.

Parameter AGCR (attack gain change rate) is in unit of -0.00212 dB per millisecond. The default is 400 units, i.e., (-0.00212×400) dB/ms.

Parameter DGCR (decay gain change rate) is in unit of 0.00212 dB per millisecond. The default is 8 units, i.e., (0.00212×8) dB/ms.

Parameter MA (maximum amplification) ranges from 6 dB to 48 dB in 6 dB steps. This parameter must be in multiple of 6 dB. The default is 42 dB.

NOTE: For a complete explanation of gain control refer to the *Automatic Gain Control* section of the SmartWORKS Developer's Guide.

Prototype

MT_RESULT MTSetAGC() (

```
const CHANNEL      nChannel,
const SHORT        TMA,
const SHORT        AGCR,
const SHORT        DGCR,
const SHORT        MA
);
```

Parameters

nChannel	channel number
TMA	Target Maximum Amplitude, 1 to -45 dBm, def= -6 dBm
AGCR	Attack Gain Change Rate, in unit of -0.00212 dB/ms, def=400. Value must be: ≥ 0 Max = 0x7FFF and Min = 1
DGCR	Decay Gain Change Rate, in unit of 0.00212 dB/ms, def=4 Value must be: ≥ 0 Max = 0x7FFF and Min = 1
MA	Maximum Amplification, 6 dB to 48 dB in 6dB steps, def=30 dB

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_CHANNEL_NOT_OPENED	Channel device not opened for access
MT_RET_NOT_AVAILABLE	Driver is not loaded

Events

None

See Also: [MTSetAVC\(\)](#)

MTChInputSetAGC()

Formally known as **MTSetAGC()**.

Description

MTChInputSetAGC() sets the input Automatic Gain Control parameters on a per input basis for each channel.

Parameter TMA (target maximum amplitude) ranges from 1 to -45 dBm. The default is 0 dBm.

Parameter AGCR (attack gain change rate) is in unit of -0.00212 dB per millisecond. The default is 400 units, i.e., (-0.00212 x 400) dB/ms.

Parameter DGCR (decay gain change rate) is in unit of 0.00212 dB per millisecond. The default is 8 units, i.e., (0.00212 x 8) dB/ms.

Parameter MA (maximum amplification) ranges from 6 dB to 48 dB in 6 dB steps. This parameter must be in multiple of 6 dB. The default is 42 dB.

NOTE: For a complete explanation of gain control refer to the *Automatic Gain Control* section of the SmartWORKS Developer's Guide.

Prototype

MT_RESULT MTChInputSetAGC() (

```
const CHANNEL    nChannel,
const SHORT      nInput,
const SHORT      pTMA,
const SHORT      pAGCR,
const SHORT      pDGCR,
const SHORT      pMA
);
```

Parameters

nChannel	channel number
nInput	Input index, 0=primary, 1=secondary
TMA	Target Maximum Amplitude, 1 to -45 dBm, def= -6 dBm
AGCR	Attack Gain Change Rate, in unit of -0.00212 dB/ms, def=400. Value must be: >=0 Max = 0x7FFF and Min = 1
DGCR	Decay Gain Change Rate, in unit of 0.00212 dB/ms, def=4 Value must be: >=0 Max = 0x7FFF and Min = 1
MA	Maximum Amplification, 6 dB to 48 dB in 6dB steps, def=30 dB

Return Codes

MT_RET_OK	No error

MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_CHANNEL_NOT_OPENED	Channel device not opened for access
MT_RET_NOT_AVAILABLE	Driver is not loaded

Events

None
See Also: [MTSetAVC\(\)](#)

MTGetAGC()

Description

MTGetAGC() retrieves the input AGC parameters used by both inputs per channel.

Parameter TMA (target maximum amplitude) ranges from 1 to -45 dBm. The default is 0 dBm.

Parameter AGCR (attack gain change rate) is in unit of -0.00212 dB per millisecond. The default is 400 units, i.e., (-0.00212 x 400) dB/ms.

Parameter DGCR (decay gain change rate) is in unit of 0.00212 dB per millisecond. The default is 8 units, i.e., (0.00212 x 8) dB/ms.

Parameter MA (maximum amplification) ranges from 6 dB to 48 dB in 6 dB steps. This parameter must be in multiple of 6 dB. The default is 42 dB.

NOTE: For a complete explanation of gain control refer to the *Automatic Gain Control* section of the SmartWORKS Developer's Guide.

Prototype

MT_RESULT MTGetAGC (

```
const CHANNEL      nChannel,
PSHORT const      pTMA,
PSHORT const      pAGCR,
PSHORT const      pDGCR,
PSHORT const      pMA
);
```

Parameters

nChannel	channel number
pTMA	Target Maximum Amplitude, 1 to -45 dBm, def= -6 dBm
pAGCR	Attack Gain Change Rate, in unit of -0.00212 dB/ms, def=400 Value must be: >=0
pDGCR	Decay Gain Change Rate, in unit of 0.00212 dB/ms, def=4 Value must be: >=0
pMA	Maximum Amplification, 6 dB to 48 dB in 6dB steps, def= 30 dB

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_CHANNEL_NOT_OPENED	Channel device not opened for access
MT_RET_NOT_AVAILABLE	Driver is not loaded

Events

None

MTChInputGetAGC()

Formally known as *MTGetAGC()*.

Description

MTChInputGetAGC() retrieves the input Automatic Gain Control parameters on a per input basis for each channel.

Parameter TMA (target maximum amplitude) ranges from 1 to -45 dBm. The default is 0 dBm.

Parameter AGCR (attack gain change rate) is in unit of -0.00212 dB per millisecond. The default is 400 units, i.e., (-0.00212 x 400) dB/ms.

Parameter DGCR (decay gain change rate) is in unit of 0.00212 dB per millisecond. The default is 8 units, i.e., (0.00212 x8) dB/ms.

Parameter MA (maximum amplification) ranges from 6 dB to 48 dB in 6 dB steps. This parameter must be in multiple of 6 dB. The default is 42 dB.

NOTE: For a complete explanation of gain control refer to the *Automatic Gain Control* section of the SmartWORKS Developer’s Guide.

Prototype

MT_RESULT MTChInputGetAGC (

const CHANNEL	nChannel,
const SHORT	nInput,
PSHORT const	pTMA,
PSHORT const	pAGCR,
PSHORT const	pDGCR,
PSHORT const	pMA
);	

Parameters

nChannel	channel number
nIndex	Input Index: 0=primary, 1=secondary
pTMA	Target Maximum Amplitude, 1 to -45 dBm, def= -6 dBm
pAGCR	Attack Gain Change Rate, in unit of -0.00212 dB/ms, def=400 Value must be: >=0
pDGCR	Decay Gain Change Rate, in unit of 0.00212 dB/ms, def=4 Value must be: >=0
pMA	Maximum Amplification, 6 dB to 48 dB in 6dB steps, def= 30 dB

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_CHANNEL_NOT_OPENED	Channel device not opened for access
MT_RET_NOT_AVAILABLE	Driver is not loaded

Events

None

MTSetGain()

Description

MTSetGain() sets the gain for both inputs of the specified channel. Valid gain values range from -50 dB to +24dB. By default, the gain is set to 0 dB.

Opening and closing a channel will not affect the gain settings. If one application opens and sets the gain of a channel to 6dB and closes the channel, the channel's gain setting is left at 6dB until an application sets it to a different value.

Setting the gain to -50 dB is equal to muting the channel input.
NOTE: For a complete explanation of gain control refer to the *Automatic Gain Control* section of the SmartWORKS Developer's Guide.

Prototype

```
MT_RESULT MTSetGain (  
    const CHANNEL      nChannel,  
    const SHORT        Gain  
);
```

Parameters

nChannel	channel number
Gain	gain value: -50dB to +24dB. def = 0dB

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_CHANNEL_NOT_OPENED	Channel device not opened for access
MT_RET_NOT_AVAILABLE	Driver is not loaded

Events

None

CODE EXAMPLE

Code Example MTSetGain()

```
handleMT(MTSetGain(channel,gain), "MTSetGain() channel %d",channel);  
  
if(handleMT(MTGetGain(channel,&gain), "MTGetGain() channel %d",channel))  
    cout << "Gain = " << gain << endl << endl;
```

A complete code example is available on the AudioCodes website:
<http://www.audiocodes.com/blades/support.htm>

MTChInputSetGain()

Formally known as **MTSetGain()**.

Description

MTChInputSetGain() sets the input gain of the specified input per channel. Valid gain values range from -50 dB to +24dB. By default, the gain is set to 0 dB.

Opening and closing a channel will not affect the gain settings. If one application opens and sets the gain of a channel input to 6dB and closes the channel, the input's gain setting is left at 6dB until an application sets it to a different value.

Setting the gain to -50 dB is equal to muting the channel input.

NOTE: For a complete explanation of gain control refer to *Automatic Gain Control* section in the SmartWORKS Developer's Guide.

Prototype

MT_RESULT MTChInputSetGain (

```
const CHANNEL      nChannel,
const SHORT        nInput,
const SHORT        Gain
);
```

Parameters

nChannel	channel number
nInput	Input Index: 0=primary, 1=secondary
Gain	gain value: -50dB to +24dB. def = 0dB

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_CHANNEL_NOT_OPENED	Channel device not opened for access
MT_RET_NOT_AVAILABLE	Driver is not loaded

Events

None

MTGetGain()

Description

MTGetGain() gets the current gain of both inputs on the specified channel. Valid gain values range from -50 dB to +24dB. By default, the gain is set to 0 dB.

Opening and closing a channel will not affect the gain settings. If one application opens and sets the gain of a channel to 6dB and closes the channel, the channel's gain setting is left at 6dB until an application sets it to a different value.

Setting the gain to -50 dB is equal to muting the channel input.

NOTE: For a complete explanation of gain control refer to the *Automatic Gain Control* section in the SmartWORKS Developer's Guide.

Prototype

MT_RESULT MTGetGain (

```
const CHANNEL      nChannel,
const PSHORT       pGain
);
```

Parameters

nChannel	channel number
pGain	gain value: -50 dB to +24dB gain < -50 dB equals muting, default = 0dB

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_CHANNEL_NOT_OPENED	Channel device not opened for access
MT_RET_NOT_AVAILABLE	Driver is not loaded

Events

None

CODE EXAMPLE

MTGetGain()

```
if(handleMT(MTGetGain(channel,&gain),"MTGetGain() channel %d",channel))
    cout << "Gain = " << gain << endl << endl;
```

A complete code example is available on the AudioCodes website:

<http://www.audiocodes.com/blades/support.htm>

MTChInputGetGain()

Formally known as **MTGetGain()**.

Description

MTChInputGetGain() gets the current gain of the specified input per channel. Valid gain values range from -50 dB to +24dB. By default, the gain is set to 0 dB.

Opening and closing a channel will not affect the gain settings. If one application opens and sets the gain of the input to 6dB and closes the channel, the input's gain setting is left at 6dB until an application sets it to a different value.

Setting the gain to -50 dB is equal to muting the channel input.

NOTE: For a complete explanation of gain control refer to the *Automatic Gain Control* section in the SmartWORKS Developer's Guide.

Prototype

MT_RESULT MTChInputGetGain (

```
const CHANNEL      nChannel,
const SHORT        nInput,
const PSHORT       pGain
);
```

Parameters

nChannel	channel number
nInput	Input Index: 0=primary, 1=secondary
pGain	gain value: -50 dB to +24dB gain < -50 dB equals muting, default = 0dB

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_CHANNEL_NOT_OPENED	Channel device not opened for access
MT_RET_NOT_AVAILABLE	Driver is not loaded

Events

None

MTStepGain()

Description

Use this API to modify the gain value that has been set with the **MTSetGain()** function. By default gain is set to a default of 0. This API can be used to step the gain up or down by 1dB or reset gain to default of 0.

Step values:	
-1	Adjust gain down by 1 dB
1	Adjust gain up by 1 dB
0	Reset gain to 0 dB

MTSetGain() sets the input gain of the specified channel. Valid gain values range from -50 dB to +24dB. By default, the gain is set to 0 dB.

Opening and closing a channel will not affect the gain settings. If one application opens and sets the gain of a channel to 6dB and closes the channel, the channel's gain setting is left at 6dB until an application sets it to a different value.

Setting the gain to -50 dB is equal to muting the channel input. No minimum value is checked.

NOTE: For a complete explanation of gain control refer to the *Automatic Gain Control* section in the SmartWORKS Developer's Guide.

Prototypes

```
MT_RESULT MTStepGain (  
    const CHANNEL nChannel,  
    const SHORT step  
);
```

Parameters

nChannel	channel number
step	step mode for gain, STEP_DOWN, STEP_RESET, or STEP_UP

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_CHANNEL_NOT_OPENED	Channel device not opened for access
MT_RET_NOT_AVAILABLE	Driver is not loaded

Events

None

MTChInputStepGain()

Formally known as **MTStepGain()**.

Description

Use this API to modify the gain value that has been set with the **MTSetGain()** or **MTChInputSetGain()** function. By default gain is set to a default of 0. This API can be used to step the gain up or down by 1dB or reset gain to default of 0. This API steps the gain level of the specified input per channel.

Step values:

-1	Adjust gain down by 1 dB
1	Adjust gain up by 1 dB
0	Reset gain to 0 dB

MTSetGain() and **MTChInputSetGain()** sets the input gain of the specified channel. Valid gain values range from -50 dB to +24dB. By default, the gain is set to 0 dB.

Opening and closing a channel will not affect the gain settings. If one application opens and sets the gain of a channel's input to 6dB and closes the channel, the gain setting is left at 6dB until an application sets it to a different value.

Setting the gain to -50 dB is equal to muting the channel input. No minimum value is checked.

NOTE: For a complete explanation of gain control refer to the *Automatic Gain Control* section of the SmartWORKS Developer's Guide.

Prototypes

MT_RESULT MTChInputStepGain (

```
const CHANNEL    nChannel,
const SHORT      nInput,
const SHORT      step
);
```

Parameters

nChannel	channel number
nInput	Input Index: 0=primary, 1=secondary
step	step mode for gain, STEP_DOWN, STEP_RESET, or STEP_UP

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_CHANNEL_NOT_OPENED	Channel device not opened for access
MT_RET_NOT_AVAILABLE	Driver is not loaded

Events

None

MTAGCControl()

Description

MTAGCControl() enables or disables the AGC for both inputs of the specified channel. AGC parameters are set through **MTSetAGC()**.

AGC control is disabled by default.

If a channel's AGC is enabled, opening/closing of a channel is not going to change this state.

NOTE: To control the AGC of the two inputs independantly of one another use **MTChInputAGCControl()** and then **MTChInputSetAGC()**.

NOTE: For a complete explanation of gain control refer to the *Automatic Gain Control* section in the SmartWORKS Developer's Guide.

Prototype

MT_RESULT MTAGCControl (

```
const CHANNEL      nChannel,  
const BOOLEAN      fOn  
);
```

Parameters

nChannel	channel number
fOn	TRUE for enable AGC; FALSE for disable AGC

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_CHANNEL_NOT_OPENED	Channel device not opened for access
MT_RET_NOT_AVAILABLE	Driver is not loaded

Events

None

MTChInputAGCControl()

Formally known as *MTAGCControl()*.

Description

MTChInputAGCControl() enables or disables the input Automatic Gain Control of the specified input per channel. When this control option is used, AGC parameters must be set with **MTChInputSetAGC()** instead of **MTSetAGC()**. AGC control is disabled by default.

If a channel's AGC is enabled, opening/closing of a channel is not going to change this state.

NOTE: For a complete explanation of gain control refer to the *Automatic Gain Control* section of the SmartWORKS Developer's Guide.

Prototype

```
MT_RESULT MTChInputAGCControl (  
    const CHANNEL      nChannel,  
    const SHORT        nIndex,  
    const BOOLEAN      fOn  
);
```

Parameters

nChannel	channel number
nIndex	Input Index: 0=primary,1=secondary
fOn	TRUE for enable AGC; FALSE for disable AGC

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_CHANNEL_NOT_OPENED	Channel device not opened for access
MT_RET_NOT_AVAILABLE	Driver is not loaded

Events

None

MTAGCStatus()

Description

MTAGCStatus() retrieves the AGC operating status (TRUE for enabled; FALSE for disabled) for both inputs on a channel. This status check is used when AGC is enabled using **MTAGCControl()**. If the API **MTChInputAGCControl()** was used to enable AGC, then use **MTChInputAGCStatus()**.

If a channel's AGC is enabled, opening/closing of a channel is not going to change this state.

Prototype

MT_RESULT MTAGCStatus (

```
const CHANNEL      nChannel,  
BOOLEAN * const    pOn  
);
```

Parameters

nChannel	channel number
pOn	TRUE for AGC on, FALSE for AGC off

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_CHANNEL_NOT_OPENED	Channel device not opened for access
MT_RET_NOT_AVAILABLE	Driver is not loaded

Events

None

MTChInputAGCStatus()

Formally known as **MTAGCStatus()**.

Description

MTChInputAGCStatus() retrieves the AGC operating status (TRUE for enabled; FALSE for disabled) of each input on the channel.

This status check is used when AGC is enabled using **MTChInputAGCControl()**. If the API **MTAGCControl()** was used to enable AGC, then use **MTAGCStatus()**.

If a channel's AGC is enabled, opening/closing of a channel is not going to change this state.

Prototype

```
MT_RESULT MTChInputAGCStatus (  
    const CHANNEL      nChannel,  
    const SHORT        nInput,  
    BOOLEAN * const    pOn  
);
```

Parameters

nChannel	channel number
nInput	Input Index: 0=primary, 1=secondary
pOn	TRUE for AGC on, FALSE for AGC off

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_CHANNEL_NOT_OPENED	Channel device not opened for access
MT_RET_NOT_AVAILABLE	Driver is not loaded

Events

None

MTGetAVC()

Description

MTGetAVC() retrieves the output AVC parameters.

Parameter TMA (target maximum amplitude) ranges from 1 to -45 dBm. The default is -6 dBm.

Parameter AVCR (attack volume change rate) is in unit of -0.00212 dB per millisecond. The default is 400 units, i.e., (-0.00212 x 400) dB/ms.

Parameter DVCR (decay volume change rate) is in unit of 0.00212 dB per millisecond. The default is 4 units, i.e., (0.00212 x 8) dB/ms.

Parameter MA (maximum amplification) ranges from 6 dB to 48 dB in 6 dB steps. The default is 30 dB.

Prototype

MT_RESULT MTGetAVC (

```
const CHANNEL      nChannel,
PSHORT const      pTMA,
PSHORT const      pAVCR,
PSHORT const      pDVCR,
PSHORT const      pMA
);
```

Parameters

nChannel	channel number
pTMA	pointer to Target Max. Amplitude, 1 to -45 dBm, def=-6 dBm
pAVCR	pointer to Attack Volume Change Rate
pDVCR	pointer to Decay Volume Change Rate
pMA	pointer to Maximum Amplification

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_CHANNEL_NOT_OPENED	Channel device not opened for access
MT_RET_NOT_AVAILABLE	Driver is not loaded

Events

None

MTSetVolume()

Description

MTSetVolume() sets the output volume of the specified channel. Valid ranges for this parameter are from +24 dB to -50 dB. By default, the volume is set to 0 dB. Setting the volume to -50 dB is equal to muting the channel input.

Opening and closing a channel will not affect the volume setting. Therefore, if one application opens and sets the volume of a channel to 6dB and closes the channel, the channel's volume setting will be left at 6dB until (the same or other) application sets it to a different value.

Prototype

MT_RESULT MTSetVolume (
 const CHANNEL nChannel,
 const SHORT Volume
);

Parameters

nChannel	channel number
Volume	valid range: +24 dB to -50dB -50 equals muting def=0dB

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_CHANNEL_NOT_OPENED	Channel device not opened for access
MT_RET_NOT_AVAILABLE	Driver is not loaded

Events

None
See Also: [MTSetGain\(\)](#)

CODE EXAMPLE

Code Example MTSetVolume()

```
handleMT(MTSetVolume(channel,volume), "MTSetVolume() channel %d, volume %d", channel,volume);

if(handleMT(MTGetVolume(channel,&volume), "MTGetVolume() channel %d", channel))
    cout << "Volume = " << volume << endl << endl;
```

A complete code example is available on the AudioCodes website:
<http://www.audiocodes.com/blades/support.htm>

MTGetVolume()

Description

MTGetVolume() gets the current output volume of the specified channel. Valid ranges for this parameter are from +24 dB to –50 dB. By default, the volume is set to 0 dB. Setting the volume to -50 dB is equal to muting the channel input.

Opening and closing a channel will not affect the volume setting. Therefore, if one application opens and sets the volume of a channel to 6dB and closes the channel, the channel's volume setting will be left at 6dB until (the same or other) application sets it to a different value.

Prototype

```
MT_RESULT MTGetVolume (
    const CHANNEL      nChannel,
    const PSHORT       pVolume
);
```

Parameters

nChannel	channel number
pVolume	pointer to volume

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_CHANNEL_NOT_OPENED	Channel device not opened for access
MT_RET_NOT_AVAILABLE	Driver is not loaded

Events

None

CODE EXAMPLE

Code Example MTGetVolume()

```
if (handleMT(MTGetVolume(channel, &volume), "MTGetVolume() channel %d", channel))
    cout << "Volume = " << volume << endl <<
endl;
```

A complete code example is available on the AudioCodes website:

<http://www.audiocodes.com/blades/support.htm>

MTStepVolume()

Description

Step the volume up or down by 1dB or reset volume to default of 0.

Step values:

-1	Adjust volume down by 1 dB
1	Adjust volume up by 1 dB
0	Reset volume to 0 dB

Opening and closing a channel will not affect the volume setting. Therefore, if one application opens and sets the volume of a channel to 6dB and closes the channel, the channel's gain setting will be left at 6dB until (the same or other) application sets it to a different value.

Prototypes

```
MT_RESULT MTStepVolume(  
    const CHANNEL nChannel,  
    const SHORT step  
);
```

Parameters

nChannel	channel number
step	mode for step, STEP_DOWN, STEP_RESET, or STEP_UP

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_CHANNEL_NOT_OPENED	Channel device not opened for access
MT_RET_NOT_AVAILABLE	Driver is not loaded

Events

None

CODE EXAMPLE**Code Example MTStepVolume()**

```
if(handleMT(MTGetVolume(channel,&volume),"MTGetVolume() channel %d",channel))
    cout << "Volume = " << volume << endl <<
endl;

    SHORT step = -1;

    cout << "Setting..." << endl
    << "step = " << step << endl << endl;

handleMT(MTStepVolume(channel,step),"MTStepVolume() channel %d, step
%d",channel,step);

if(handleMT(MTGetVolume(channel,&volume),"MTGetVolume() channel %d",channel))
    cout << "Volume = " << volume << endl <<
endl;
```

A complete code example is available on the AudioCodes website:

<http://www.audiocodes.com/blades/support.htm>

MTAVCControl()

Description

MTAVCControl() enables or disables the output AVC of the specified channel. AVC parameters are set through **MTSetAVC()**.

AVC control is disabled by default.
If a channel's AVC is enabled, opening/closing of a channel is not going to change this state.

Prototype

```
MT_RESULT MTAVCControl (  
    const CHANNEL      nChannel,  
    const BOOLEAN      fOn  
);
```

Parameters

nChannel	channel number
fOn	TRUE for enable AVC; FALSE for disable AVC

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_CHANNEL_NOT_OPENED	Channel device not opened for access
MT_RET_NOT_AVAILABLE	Driver is not loaded

Events

None

MTSetAVC()

Description

MTSetAVC() sets the output AVC parameters.

Parameter TMA (target maximum amplitude) ranges from 1 to -45 dBm. The default is -6 dBm.

Parameter AVCR (attack volume change rate) is in unit of -0.00212 dB per millisecond. The default is 400 units, i.e., (-0.00212×400) dB/ms.

Parameter DVCR (decay volume change rate) is in unit of 0.00212 dB per millisecond. The default is 4 units, i.e., (0.00212×8) dB/ms.

Parameter MA (maximum amplification) ranges from 6 dB to 48 dB in 6 dB steps. This parameter must be in multiple of 6 dB. The default is 30 dB.

Prototype

MT_RESULT MTSetAVC (

```
const CHANNEL      nChannel,
const SHORT        TMA,
const SHORT        AVCR,
const SHORT        DVCR,
const SHORT        MA
);
```

Parameters

nChannel	channel number
TMA	Target Maximum Amplitude, 1 to -45 dBm, def=-6 dBm
AVCR	Attack Volume Change Rate, in unit of -0.00212 dB/ms, def=400 Value must be: ≤ 0
DVCR	Decay Volume Change Rate, in unit of 0.00212 dB/ms, def=4 Value must be: ≤ 0
MA	Maximum Amplification, 6 dB to 48 dB in 6dB steps, def=30 dB

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_CHANNEL_NOT_OPENED	Channel device not opened for access
MT_RET_NOT_AVAILABLE	Driver is not loaded

Events

None

MTAVCStatus()

Description

MTAVCStatus() retrieves the output AVC operating status: TRUE for enabled; FALSE for disabled.

If a channel's AVC is enabled, opening/closing of a channel is not going to change this state.

Prototype

```
MT_RESULT MTAVCStatus (  
    const CHANNEL          nChannel,  
    BOOLEAN * const        pOn  
);
```

Parameters

nChannel	channel number
pOn	TRUE for AVC on, FALSE for AVC off

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_CHANNEL_NOT_OPENED	Channel device not opened for access
MT_RET_NOT_AVAILABLE	Driver is not loaded

Events

None

HIGHPASS FILTER FUNCTIONS

MTHighpassFilterControl()

Description

Setting the input highpass filter on enables secondary and primary PCM streams to pass through a highpass filter before further processing.

By default, the highpass filter is enabled. This API need not be used unless the input highpass filter was disabled previously to the channel.

It is highly recommended that highpass filtering remain enabled, especially when using SmartWORKS DT and SmartWORKS DP boards.

Prototype

MT_RESULT MTHighpassFilterControl (

```
const CHANNEL      nChannel,
const BOOLEAN      fOn
);
```

Parameters

nChannel	channel number
fOn	TRUE to enable the filter, FALSE to disable the filter

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_CHANNEL_NOT_OPENED	Channel device not opened for access

Events

None

MTHighpassFilterStatus()

Description

Retrieves the status of the input high pass filter for the specified channel. See **MTHighpassFilterControl()**.

Prototype

MT_RESULT MTHighpassFilterStatus (

```
const CHANNEL      nChannel,  
BOOLEAN * const    pOn  
);
```

Parameters

nChannel	channel number
pOn	TRUE to enable the filter, FALSE to disable the filter

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_CHANNEL_NOT_OPENED	Channel device not opened for access

Events

None

MEDIA ACCESS WITH APPLICATION PROVIDED DEVICE IO FUNCTIONS

MTSetDeviceIO()

Description

Registers the specified channel's user device I/O routines for record and playback media access through **MTRecDevice()** and **MTPlayDevice()**.

The parameters should contain at least one non-NULL address for either read() or write(). Specification for seek is optional. However, should seek not be defined, MT_IO_CONTROL.FileIndex field should be set to 0 when calling **MTPlayDevice()**, and **MTRecDevice()**. MTDEVICEIO is defined as a boolean function which takes the file handle, buffer pointer, data length, and pointer for returned data length as parameters. A return value of TRUE indicates that the device function has completed the task successfully. A return value of FALSE indicates error in user IO and terminates the API functions such as **MTRecDevice()** and **MTPlayDevice()**.

MTDEVICEIO and MTDEVICESEEK are defined as:

```
#typedef BOOL (*MTDEVICEIO)(PVOID const fHandler, PVOID
                           const pBuf,
                           ULONG const nByteCount, PULONG const
                           pByteAccessed)

#typedef DWORD (*MTDEVICESEEK)(PVOID const fHandler, ULONG
                               const
                               nByteCount, int const fDirection)
```

MTDEVICESEEK is defined as a function which positions the device and returns the position. Parameter fDirection is for the origin of the seek: 0 for the beginning of the device, 1 for the current position of the device, -1 for the end of the device.

Prototype

```
MT_RESULT MTSetDeviceIO (
    const CHANNEL      nChannel,
    const MTDEVICEIO   DeviceRead,
    const MTDEVICEIO   DeviceWrite,
    const MTDEVICESEEK DeviceSeek
);
```

Parameters

nChannel	channel number
DeviceRead	pointer to user device read function
DeviceWrite	pointer to user device write function
DeviceSeek	pointer to user device seek function

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_NOT_AVAILABLE	Driver is not loaded

MT_RET_CHANNEL_NOT_OPENED

Channel device not opened for access

Events

None

MTStartCurrentFunction()

Description

In NTI, all record and playback tasks are controlled through the MT_IO_CONTROL.StartControl field. This API suppresses the start control of START_ACTD, START_LOOPON, and START_LVOLTAGE for the current pending task and forces the task to start immediately.

Note: Currently, the start control is applicable only on record. Should there be start control added for playback in the future, this API will suppress those start controls also.

This API is supported for all recording, on all channels.

Prototype

MT_RESULT MTStartCurrentFunction (

```
const CHANNEL      nChannel,  
const ULONG        fOperation  
);
```

Parameters

nChannel	channel number
fOperation	Operation of MT_ENCODE, MT_DECODE, and/or MT_COLLECT_DTMF

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_CHANNEL_NOT_OPENED	Channel device not opened for access
MT_RET_IDLE	Channel is idle

Events

None

MTPlayDevice(), MTPlayDeviceAsync(), MTPlayDeviceEx()

Description

MTPlayDevice() is a background function that plays data retrieved through user provided device read function. The handle of the user device is passed through MT_IO_CONTROL.FileHandle field.

Asynchronous Call Back
Any API that ends with the suffix Async uses callback as its completion notification method. Each requires a completion routine from the user application to notify completion of the requested API. The SmartWORKS function completion routine is implemented in the same fashion as a WindowsNT system function completion. In addition to the completion routine address, the application also passes a parameter of type LPARAM to the SmartWORKS DLL/Driver, which passes the parameter back without change to the LPARAM parameter.

Prototype

```
MT_RESULT MTPlayDevice (  
    const CHANNEL          nChannel,  
    const PMT_IO_CONTROL   ploCtl  
);  
  
MT_RESULT MTPlayDeviceEx (  
    const CHANNEL          nChannel,  
    const PMT_IO_CONTROL   ploCtl,  
    const LPOVERLAPPED     pOverlapped  
);  
  
MT_RESULT MTPlayDeviceAsync (  
    const CHANNEL          nChannel,  
    const PMT_IO_CONTROL   ploCtl,  
    const MTCOMPLETE       pCompletionRoutine,  
    const LPARAM            lParam  
);
```

Parameters

nChannel	channel number
ploCtl	pointer to a The MT IO CONTROL Data Structure structure (defined in Appendix A)
pOverlapped	pointer to overlapped structure
lParam	user defined parameter
pCompletionRoutine	completion routine address

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_PARAM	Invalid argument passed to function

MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_CHANNEL_NOT_OPENED	Channel device not opened for access
MT_RET_IO_PENDING	Background I/O operation is in progress or queued
MT_RET_BUSY	channel busy
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_INVALID_DEVICE_IO	Invalid Device IO setting
MT_RET_NO_MEM	An operation could not be completed due to insufficient memory or an error occurred while attempting to allocate memory
MT_RET_FORMAT_NOT_SUPPORTED	Media format is not supported for this channel
MT_RET_API_THREAD_PROTECTED	Another thread is currently accessing this API

Events

MT_EVT_DEVICE_READ_ERROR

MT_EVT_DEVICE_SEEK_ERROR

Other termination events

See Also: [MTPlayFile\(\)](#), [MTPlayBuffer\(\)](#)

MTRecDevice(), MTRecDeviceAsync(), MTRecDeviceEx()

Description

MTRecDevice() is a background function that stores the recorded data through user provided device write function. The handle of the user device is passed through MT_IO_CONTROL.FileHandle field.

Prototype

MT_RESULT MTRecDevice (

```
const CHANNEL          nChannel,
const PMT_IO_CONTROL   ploCtl
);
```

MT_RESULT MTRecDeviceEx (

```
const CHANNEL          nChannel,
const PMT_IO_CONTROL   ploCtl,
const LPOVERLAPPED     pOverlapped
):
```

MT_RESULT MTRecDeviceAsync (

```
const CHANNEL          nChannel,
const PMT_IO_CONTROL   ploCtl,
const MTCOMPLETE       pCompletionRoutine,
const LPARAM           lParam
);
```

Parameters

nChannel	channel number
ploCtl	pointer to a The MT_IO_CONTROL Data Structure structure (defined in Appendix A)
pOverlapped	pointer to overlapped structure
lParam	user defined parameter
pCompletionRoutine	completion routine address

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_CHANNEL_NOT_OPENED	Channel device not opened for access
MT_RET_IO_PENDING	Background I/O operation is in progress or queued
MT_RET_BUSY	channel busy
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_INVALID_DEVICE_IO	Invalid Device IO setting

MT_RET_NO_MEM	An operation could not be completed due to insufficient memory or an error occurred while attempting to allocate memory
MT_RET_FORMAT_NOT_SUPPORTED	Media format is not supported for this channel
MT_RET_API_THREAD_PROTECTED	Another thread is currently accessing this API

Events

MT_EVT_DEVICE_WRITE_ERROR

MT_EVT_DEVICE_SEEK_ERROR

Other termination events

See Also: [MTRecFile\(\)](#), [MTRecBuffer\(\)](#)

Mux/MVIP BUS ROUTINES

MTStartMUX()

Description

MTStartMUX() is called to prepare MUX operation before MUX connections can be made. All configurations will be set to default based on the configuration of the TDMCardType and TDMBusType. On successful execution, USHORT pointer contains the total configured channels grabbed by the application.

The default mapping of NI and DR for MUX type MUX_MVIP and MUX_H100 is defined as follows:

NI(n) is mapped with DR(n) as Channel(n); NI(n+1) is mapped with DR(n+1) as Channel(n+1).

Timeslot(16:n) is assigned as receive timeslot for DR(n); timeslot(17:n) is assigned as receive timeslot for NI(n).

Channel(n) is composed by NI(n) receiving on timeslot(17:n), transmitting on timeslot(16:n) and DR(n) receiving on timeslot(16:n) and transmitting on timeslot(17:n).

No cross board mapping of NI and DR.

When calling this function after MUX is already started, return code MT_RET_MUX_ON will be returned and channel numbers will be retrieved if buffer is provided. Error code MT_RET_NO_MUX will be returned if MUX system is not available.

Currently, **MTSysStartup()** will start the MUX as part of system initialization.

Prototype

MT_RESULT MTStartMUX (

```
    USHORT * const      pTotalChans
);
```

Parameters

pTotalChans	Pointer to integer that will be filled with the number of channels in system

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_NO_MUX	Multiplexer is not available %0
MT_RET_MUX_ON	Multiplexer system was already started
MT_TIMEOUT	Wait for event timed out
MT_RET_NOT_AVAILABLE	Driver is not loaded

Events

None

MTStopMUXOnChannel()

Description

MTStopMUXOnChannel() clears the MUX connection on the specified channel without shutting the MUX down, hence leaving MUX on other channels undisturbed.

When calling this function after MUX is stopped the return code MT_RET_MUX_OFF will be returned.

Prototype

```
MT_RESULT MTStopMUXOnChannel (  
    const CHANNEL          nChannel  
);
```

Parameters

nChannel	channel number

Return Codes

MT_RET_OK	No error
MT_RET_MUX_OFF	MUX not started
MT_TIMEOUT	Wait for event timed out
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_CHANNEL_NOT_OPENED	Channel device not opened for access
MT_RET_IDLE	Channel is idle

Events

None

See Also: [MTSysShutdown\(\)](#)

MTStopMUX()

Description

MTStopMUX() clears all existing MUX connections on all channels grabbed by the user application and shuts the MUX down. When calling this function after MUX is stopped, return code MT_RET_MUX_OFF will be returned.

Currently, **MTSysShutdown()** will stop the MUX as a part of system shutdown.

Prototype

MT_RESULT MTStopMUX (
Void
);

Return Codes

MT_RET_OK	No error
MT_RET_MUX_OFF	MUX not started
MT_TIMEOUT	Wait for event timed out
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_IDLE	Channel is idle
MT_RET_NO_MUX	Multiplexer is not available

Events

None
See Also: [MTSysShutdown\(\)](#)

MTGetCTSlotConnection()

Description

This function returns a status table of a group of CT timeslots. The group is specified by the following parameters: StartStream, StartTimeSlot, and pSlotCount. If the requested pSlotCount exceeds the actual available time slots, the number of actual timeslots will be returned.

Prototype

MT_RESULT MTGetCTSlotConnection(

```
const int          BusType,
const int          StartStream,
const int          StartTimeSlot,
int * const        pSlotCount ,
PMT_CT_SLOT const pSlotTable
);
```

Parameter

BusType	MUX_MVIP or MUX_H100
StartStream	Indexed 0 to 15 for MVIP, 31 for H100
StartTimeSlot	Indexed 0 to MAX_MVIP_SLOT-1 or MAX_H100_SLOT-1
pSlotCount	Pointer to count of the MT_CT_SLOT buffer
pSlotTable	Pointer to MT_CT_SLOT buffer

Return Codes

MT_RET_OK	No error
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_INVALID_BOARD	Board number is out of range
MT_RET_BOARD_NOT_OPENED	Board device not opened for access
MT_RET_INVALID_BUSTYPE	Failure to specify MUX_MVIP or MUX_H100

Events

None

Events

None

Events

None

PLAY/RECORD FUNCTIONS

MTPlayBuffer(), MTPlayBufferAsync(), MTPlayBufferEx()

Description

MTPlayBuffer() is a background function that plays back data from an application buffer. The address of the buffer is specified by the buffer parameter in MT_IO_CONTROL. The number of bytes to play must be set in the length parameter. Also, the number of bytes must be divisible by the frame size so that no data will be lost. For example, if the frame size is 320 (16-bit linear voice format) then the buffer length should be 320, 640, 960, 1280, and so on. If the buffer length were set to, let's say 650, then 10 bytes of data will be lost. Below is a table of common voice formats, corresponding frame sizes, and possible buffer lengths. It is not exhaustive, but should illustrate the proper settings for buffer sizes.

Voice Format	Frame Size	Buffer lengths (in bytes)
16-bit linear	320	320, 640, 960, 1280...
PCM	160	160, 320, 480, 640...
MSGSM	65	65, 130, 195, 260...

Prototype

MT_RESULT MTPlayBuffer (

```
const CHANNEL          nChannel,
const PMT_IO_CONTROL   ploCtl
);
```

MT_RESULT MTPlayBufferEx(

```
const CHANNEL          nChannel,
const PMT_IO_CONTROL   ploCtl,
const LPOVERLAPPED     pOverlapped
);
```

Any API that ends with the suffix "Ex" uses the Windows overlapped mechanism as its function completion notification mechanism. **MTPlayBuffer()** uses no notification mechanism. The user application retrieves events from the event queue until it encounters a playback specific event, which indicates completion of the **MTPlayBuffer()** function.

MT_RESULT MTPlayBufferAsync(

```
const CHANNEL          nChannel,
const PMT_IO_CONTROL   ploCtl,
const MTCOMPLETE        pCompletionRoutine,
const LPARAM            lParam
);
```

Parameters

nChannel	channel number of the channel to play to
pIoCtl	pointer to a The MT_IO_CONTROL Data Structure structure (defined in Appendix A)
pOverlapped	pointer to OVERLAPPED structure
pCompletionRoutine	completion routine address
lParam	parameter to echo back

Return Codes

MT_RET_OK	No error
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_CHANNEL_NOT_OPENED	Channel device not opened for access
MT_RET_IO_PENDING	Background I/O operation is in progress or queued
MT_RET_BUSY	On channel busy
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_NO_MEM	An operation could not be completed due to insufficient memory or an error occurred while attempting to allocate memory
MT_RET_FORMAT_NOT_SUPPORTED	Media format is not supported for this channel
MT_RET_API_THREAD_PROTECTED	Another thread is currently accessing this API

Events

- EVT_MAXTIME
- EVT_MAXBYTE
- EVT_MAXDIGIT
- EVT_TERMDIGIT
- EVT_INTERDIGIT

MTPlayFile(), MTPlayFileAsync(), MTPlayFileEx()

NOTE

If UNICODE is defined SmartWORKS will call the API with the 'W' suffix, if UNICODE is not defined, SmartWORKS will call the API with the 'A' suffix. i.e. MTPlayFileA or MTPlayFileW.

Description

MTPlayFile() is a background function that plays data from a file. **MTPlayFile()**, **MTPlayFileAsync()**, and **MTPlayFileEx()** will actually map to modified versions of themselves depending on whether or not UNICODE is defined for the user application. See "UNICODE Support" on page 23 for more information about APIs affected by defining UNICODE in a user application.

Prototype

MT_RESULT MTPlayFile (

```
const CHANNEL          nChannel,
PCSTR -(A)             pFileName,
PCWSTR -(W)
const PMT_IO_CONTROL   ploCtl
);
```

MT_RESULT MTPlayFileEx (

```
const CHANNEL          nChannel,
PCSTR -(A)             pFileName,
PCWSTR -(W)
const PMT_IO_CONTROL   ploCtl,
const LPOVERLAPPED     pOverlapped
);
```

MT_RESULT MTPlayFileAsync (

```
const CHANNEL          nChannel,
PCSTR -(A)             pFileName,
PCWSTR -(W)
const PMT_IO_CONTROL   ploCtl,
const MTCOMPLETE       pCompletionRoutine,
const LPARAM           lParam
);
```

Parameters

nChannel	channel number
pFileName	ASCII file name
ploCtl	pointer to a The MT IO CONTROL Data Structure structure (defined in Appendix A)
pOverlapped	pointer to overlapped structure
lParam	user defined parameter
pCompleteRoutine	completion routine address

Return Codes

MT_RET_OK	No error
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_CHANNEL_NOT_OPENED	Channel device not opened for access
MT_RET_IO_PENDING	Background I/O operation is in progress or queued
MT_RET_BUSY	On channel busy
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_NO_MEM	An operation could not be completed due to insufficient memory or an error occurred while attempting to allocate memory
MT_RET_FORMAT_NOT_SUPPORTED	Media format is not supported for this channel
MT_RET_INVALID_FILENAME	Invalid file name
MT_RET_API_THREAD_PROTECTED	Another thread is currently accessing this API

Events

EVT_EOF
EVT_MAXBYTE
EVT_MAXTIME
EVT_MAXDIGIT
EVT_INTERDIGIT
EVT_TERMDIGIT

See Also: [MTRecFile\(\)](#), [MTPlayBuffer\(\)](#), [MTStreamBufOut\(\)](#), [MTPlayDevice\(\)](#)

CODE EXAMPLE**Code Example MTPlayFile()**

```

handleMT(MTSetEventFilters(channel, filters, rings), "MTSetEventFilters() channel
%d, filters %d, rings %d", channel, filters, rings);

handleMT(MTSetEventCallback(channel, (MTCALLBACK)ChannelEventCallBack, 0), "MTSet
EventCallback() channel %d", channel);

// file to be played
string buffer = "C:\\recordings\\test.pcm";

ZeroMemory(&ioctrl, size);
ioctrl.FileType = MT_PCM_Linear_16bit;

if(handleMT(MTPlayFile(channel, buffer.c_str(), &ioctrl), "MTPlayFile() channel
%d", channel))
{
    wait = true;
    while(wait)
        Sleep(20);
}

handleMT(MTClearEventCallback(channel), "MTClearEventCallback() channel
%d", channel);

handleMT(MTSysShutdown(), "MTSysShutdown()");

```

A complete code example is available on the AudioCodes website:

<http://www.audiocodes.com/blades/support.htm>

CODE EXAMPLE**Code Example MTPlayFileAsync()**

```
if(!handleMT(MTSysStartup(), "MTSysStartup()"))
    exit(-1);

string buffer = "C:\\recordings\\test.pcm";

// file to be played

ZeroMemory(&ioctrl, size);
ioctrl.FileType = MT_PCM_Linear_16bit;

if(handleMT(MTPlayFileAsync(channel, buffer.c_str(), &ioctrl, (MTCOMPLETE)AsyncCa
llBack, 0x1234), "MTPlayFileAsync() channel %d", channel))
{
    wait = true;
    while(wait)
        Sleep(20);
}

handleMT(MTSysShutdown(), "MTSysShutdown()");
```

A complete code example is available on the AudioCodes website:

<http://www.audiocodes.com/blades/support.htm>

MTPlayIndex(), MTPlayIndexEx(), MTPlayIndexAsync()

NOTE

If UNICODE is defined SmartWORKS will call the API with the 'W' suffix, if UNICODE is not defined, SmartWORKS will call the API with the 'A' suffix. i.e. MTPlayIndexA or MTPlayIndexW.

MTPlayIndex(), **MTPlayIndexEx()**, and **MTPlayIndexAsync()** will actually map to modified versions of themselves depending on whether or not UNICODE is defined for the user application. See "UNICODE Support" on page 23 for more information about APIs affected by defining UNICODE in a user application

Description

The parameter PMT_INDEX_TABLE is a pointer to an array of the MT_INDEX_TABLE.

Setting the MT_INDEX_TABLE.Phraseoffset to -1 indicates the end of an array.

Setting the MT_INDEX_TABLE.PhraseLength to 0 indicates a skipping of that index entry.

At the end of sending to decode the voice data specified in each index entry, event EVT_INDEX_PLAYED will be issued. Normal termination also applies to this API. Therefore, at the termination of a normal playback, possible events would be EVT_EOF, EVT_MAXTIME, and EVT_MAXBYTE. EVT_TERMDIGIT and EVT_INTERDIGIT indicate that playback was terminated due to anticipated user interruption.

Prototype

MT_RESULT MTPlayIndex (

```
const CHANNEL          nChannel,
PCSTR - (A)            pFileName,
PCWSTR - (W)
PCMT_INDEX_TABLE      plxTable,
const PMT_IO_CONTROL   ploCtl
);
```

MT_RESULT MTPlayIndexEx (

```
const CHANNEL          nChannel,
PCSTR - (A)            pFileName,
PCWSTR - (W)
PCMT_INDEX_TABLE      plxTable,
const PMT_IO_CONTROL   ploCtl,
const LPOVERLAPPED     pOverlapped
);
```

MT_RESULT MTPlayIndexAsync(

```
const CHANNEL          nChannel,
PCSTR - (A)            pFileName,
PCWSTR - (W)
PCMT_INDEX_TABLE      plxTable,
```

```
const PMT_IO_CONTROL    ploCtl,  
const MTCOMPLETE        pCompletionRoutine,  
const LPARAM            IParam  
);
```

Parameters

nChannel	channel number
pFileName	ASCII file name
plxTable	pointer to an MT_INDEX_TABLE structure that specifies the data to play
ploCtl	pointer to a The MT_IO_CONTROL Data Structure structure (defined in Appendix A)
pOverlapped	pointer to overlapped structure
pCompletionRoutine	completion function address
IParam	application defined parameter

Structure

Structure MT_INDEX_TABLE is defined as follows:

TABLE 28: MT_INDEX_TABLE

Type	Name	Purpose
ULONG	Phraseoffset	offset, from the beginning of the file, to start the play from Set to -1 to indicate the end of the table
ULONG	PhraseLength	Length of data to play. Set to 0 to indicate a skipping of index entry.

Return Codes

MT_RET_OK	No error
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_CHANNEL_NOT_OPENED	Channel device not opened for access
MT_RET_IO_PENDING	Background I/O operation is in progress or queued
MT_RET_BUSY	On channel busy
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_NO_MEM	An operation could not be completed due to insufficient memory or an error occurred while attempting to allocate memory
MT_RET_FORMAT_NOT_SUPPORTED	Media format is not supported for this channel
MT_RET_INVALID_FILENAME	Invalid file name
MT_RET_API_THREAD_PROTECTED	Another thread is currently accessing this API

Events

EVT_INDEX_PLAYED
Other termination events

RECORD FUNCTIONS

MTRecBuffer(), MTRecBufferAsync(), MTRecBufferEx()

Description

MTRecBuffer() is a background function that starts recording to the application buffer. The buffer address is passed in the buffer parameter of the MT_IO_CONTROL structure. The size of the buffer must be set in the length field. Also, the number of bytes must be divisible by the frame size so that no data will be lost. For example, if the frame size is 320 (16-bit linear voice format) then the buffer length should be 320, 640, 960, 1280, and so on. If the buffer length were set to, let's say 650, then 10 bytes of data will be lost. Below is a table of common voice formats, corresponding frame sizes, and possible buffer lengths. It is not exhaustive, but should illustrate the proper settings for buffer sizes.

Voice Format	Frame Size	Buffer lengths (in bytes)
16-bit linear	320	320, 640, 960, 1280...
PCM	160	160, 320, 480, 640...
MSGSM	65	65, 130, 195, 260...

Prototype

MT_RESULT MTRecBuffer (

```
const CHANNEL          nChannel,
const PMT_IO_CONTROL   ploCtl
);
```

MT_RESULT MTRecBufferEx (

```
const CHANNEL          nChannel,
const PMT_IO_CONTROL   ploCtl,
const LPOVERLAPPED     pOverlapped
);
```

MT_RESULT MTRecBufferAsync (

```
const CHANNEL          nChannel,
const PMT_IO_CONTROL   ploCtl,
const MTCOMPLETE       pCompletionRoutine,
const LPARAM           lParam
);
```

Parameters

nChannel	channel to record from
ploCtl	pointer to a The MT IO CONTROL Data Structure structure (defined in Appendix A)

pOverlapped	pointer to overlapped structure
pCompletionRoutine	completion routine address
lParam	parameter to echo back

Return Codes

MT_RET_OK	No error
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_CHANNEL_NOT_OPENED	Channel device not opened for access
MT_RET_IO_PENDING	Background I/O operation is in progress or queued
MT_RET_BUSY	On channel busy
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_NO_MEM	An operation could not be completed due to insufficient memory or an error occurred while attempting to allocate memory
MT_RET_FORMAT_NOT_SUPPORTED	Media format is not supported for this channel
MT_RET_API_THREAD_PROTECTED	Another thread is currently accessing this API

Events

EVT_MAXTIME
EVT_MAXBYTE
EVT_MAXDIGIT
EVT_TERMSILENCE
EVT_TERMDIGIT
See Also: [MTPlayBuffer\(\)](#)

CODE EXAMPLE

Code Example MTRecBufferAsync()

```
ZeroMemory(&event, evtsize);

ZeroMemory(&ioctrl, iosize);
ioctrl.Buffer = buffer;
ioctrl.Length = 1024;
ioctrl.FileType = MT_PCM_Linear_16bit;

if(handleMT(MTRecBufferAsync(channel, &ioctrl, (MTCOMPLETE) AsyncCallback, 0x1234)
, "MTRecBufferAsync() channel %d", channel))
{
    wait = true;
    while(wait)
        Sleep(20);
}
```

A complete code example is available on the AudioCodes website:

<http://www.audiocodes.com/blades/support.htm>

MTRecFile(), MTRecFileAsync(), MTRecFileEx()

NOTE

If UNICODE is defined SmartWORKS will call the API with the 'W' suffix, if UNICODE is not defined, SmartWORKS will call the API with the 'A' suffix. i.e. MTRecFileA or MTRecFileW.

Description

MTRecFile() is a background function that starts recording to a file. If this function terminates because of MaxSilence (as specified by the MT_IO_CONTROL structure), the silence data is discarded. The SILENCE_TRUNCATION bit must be set in MT_IO_CONTROL in the StartControl field. Note that AudioCodes' legacy products had start offsets set by incrementing the offset by the file size plus 1 byte. SmartWORKS does not follow this convention. Instead, offsets under the SmartWORKS API are set by using the exact file size or less than the exact file size if you wish to trim back the recording over the tail of the previous recording.

MTRecFile(), MTRecFileAsync(), and **MTRecFileEx()** will actually map to modified versions of themselves depending on whether or not UNICODE is defined for the user application. See "UNICODE Support" on page 23 for more information about APIs affected by defining UNICODE in a user application.

Scenario

Should this function be called with no control set in MT_IO_CONTROL's structure then the user application will need to apply **MTStopChannel()**, or **MTStopCurrentFunction()** to stop the recording.

Prototype

MT_RESULT MTRecFile (

const CHANNEL	nChannel,
PCSTR - (A)	pFileName,
PCWSTR - (W)	
PMT_IO_CONTROL const	pIoCtl
);	

MT_RESULT MTRecFileEx (

const CHANNEL	nChannel,
PCSTR - (A)	pFileName,
PCWSTR - (W)	
PMT_IO_CONTROL const	pIoCtl,
const LPOVERLAPPED	pOverlapped
);	

MT_RESULT MTRecFileAsync (

const CHANNEL	nChannel,
PCSTR - (A)	pFileName,
PCWSTR - (W)	

```
PMT_IO_CONTROL const    ploCtl,  
const MTCOMPLETE        pCompleteRoutine,  
const LPARAM            IParam  
);
```

Parameters

nChannel	channel number
pFileName	ASCII file name
ploCtl	pointer to a The MT_IO_CONTROL Data Structure structure (defined in Appendix A)
pOverlapped	pointer to overlapped structure
pCompleteRoutine	completion function address
IParam	application defined parameter

Return Codes

MT_RET_OK	No error
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_CHANNEL_NOT_OPENED	Channel device not opened for access
MT_RET_IO_PENDING	Background I/O operation is in progress or queued
MT_RET_BUSY	On channel busy
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_NO_MEM	An operation could not be completed due to insufficient memory or an error occurred while attempting to allocate memory
MT_RET_FORMAT_NOT_SUPPORTED	Media format is not supported for this channel
MT_RET_INVALID_FILENAME	Invalid file name
MT_RET_API_THREAD_PROTECTED	Another thread is currently accessing this API

Events

```
EVT_MAXBYTE  
EVT_MAXTIME  
EVT_MAXDIGIT  
EVT_TERMDIGIT  
EVT_TERMSILENCE  
  
See Also: MTPlayFile\(\)
```


CODE EXAMPLE**Code Example MTRecFile()**

```

handleMT(MTSetEventFilters(channel, filters, rings), "MTSetEventFilters() channel
%d, filters %d, rings %d", channel, filters, rings);

handleMT(MTSetEventCallback(channel, (MTCALLBACK)ChannelEventCallBack, 0), "MTSet
EventCallback() channel %d", channel);

        string buffer =
"C:\\recordings\\testrec.pcm"; // file to be recorded

        ZeroMemory(&ioctrl, size);
        ioctrl.FileType = MT_PCM_Linear_16bit;

if(handleMT(MTRecFile(channel, buffer.c_str(), &ioctrl), "MTRecFile() channel
%d", channel))
    {
        wait = true;
        while(wait)
            Sleep(20);
    }

handleMT(MTClearEventCallback(channel), "MTClearEventCallback() channel
%d", channel);

```

A complete code example is available on the AudioCodes website:

<http://www.audiocodes.com/blades/support.htm>

CODE EXAMPLE**Code Example MTRecFileAsync()**

```

string buffer = "C:\\recordings\\testrec.pcm"; // file to be recorded

        ZeroMemory(&ioctrl, size);
        ioctrl.FileType = MT_PCM_Linear_16bit;

if(handleMT(MTRecFileAsync(channel, buffer.c_str(), &ioctrl, (MTCOMPLETE)AsyncCal
lBack, 0x1234), "MTRecFileAsync() channel %d", channel))
    {
        wait = true;
        while(wait)
            Sleep(20);
    }

```

A complete code example is available on the AudioCodes website:

<http://www.audiocodes.com/blades/support.htm>

STREAM FUNCTIONS

DATA STREAMING

MTSetStreamingConfig()

Description

Configures channel streaming parameters. The default streaming buffer block size is 64K.

Scenario

The streaming buffer size will be applied on the next **MTSartStreaming()** call. If streaming is active, it's buffer size will not be changed.

NOTE: The SmartWORKS DLL limits buffer size: maximum streaming buffer size is set to 1MBytes, minimum cap to 1KBytes.

Prototype

MT_RESULT MTSetStreamingConfig(

const CHANNEL nChannel,
const ULONG BufSize
);

Parameters

nChannel	channel number
BufSize	DLL buffer size, the size of the buffer set by the application for the DLL to buffer into. Default 64K bytes. Valid range: 1Kbytes - 1MBytes

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_CHANNEL_NOT_OPENED	Channel device not opened for access
MT_RET_NOT_AVAILABLE	Driver is not loaded

Events

None

MTGetStreamingConfig()

Description

Gets channel streaming parameters. The default streaming buffer block size is 64K.

Prototype

MT_RESULT MTGetStreamingConfig(

```
const CHANNEL      nChannel,  
const PULONG       pBufSize  
);
```

Parameters

nChannel	channel number
pBufSize	DLL buffer size, the size of the buffer set by the application for the DLL to buffer into. Default 64K bytes

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_CHANNEL_NOT_OPENED	Channel device not opened for access
MT_RET_NOT_AVAILABLE	Driver is not loaded

Events

None

MTStartStreaming()

Description

Currently, encode streaming and decode streaming should be started separately through two calls to MTStartStreaming().

A watermark can be set in the MT_IO_CONTROL.Length field. Once set, event EVT_STREAMIN_WATERMARK or EVT_STREAMOUT_WATERMARK will be issued.

Regardless of the number supplied as the watermark byte count, the DLL rounds it up or down to make it a multiple of the number of frames of the specified format. Alternately, the watermark byte count can be specified as 0 if the user wishes to disable watermark reporting.

TIP: If you are doing real-time monitoring and sending data directly from the event routine to a monitoring routine you can reduce the latency between recording and playing by making the watermark size much smaller. The optimal size depends on the encoding format in use. For example, with MS GSM 6.10 33 bytes of data are recorded each 20 ms. Therefore, a buffer size of 165 bytes would result in data being captured 100 ms after the first byte had been recorded. This would imply a barely perceptible delay between recording and playing. The downside of a smaller watermark is more frequent channel service.

NOTE: MaxBytes and MaxTime cannot be used for termination limits when streaming.

NOTE: The SmartWORKS DLL limits buffer size: maximum streaming buffer size is set to 1MBytes, minimum cap to 1KBytes.

Prototype

MT_RESULT MTStartStreaming(

```
const CHANNEL      nChannel,
const USHORT      Action,
const PMT_IO_CONTROL ploCtl
);
```

Parameters

nChannel	channel number
Action	action of the following: MT_STREAM_INPUT MT_STREAM_OUTPUT
ploCtl	pointer to a The MT_IO_CONTROL Data Structure structure (defined in Appendix A)

NOTE: MaxBytes and MaxTime cannot be used for termination limits when streaming.

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_CHANNEL_NOT_OPENED	Channel device not opened for access
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_BUSY	When streaming already started, or when channel is busy
MT_RET_IO_PENDING	Streaming is in startup process (this is the normal return code when the API is successfully called)
MT_RET_NO_MEM	On internal memory shortage
MT_RET_FORMAT_NOT_SUPPORTED	Media format is not supported for this channel
MT_RET_API_THREAD_PROTECTED	Another thread is currently accessing this API

Events

EVT_STREAMIN_START
EVT_STREAMOUT_START
EVT_STREAMIN_DROPPED
EVT_STREAMOUT_EMPTY
EVT_STREAMIN_WATERMARK
EVT_STREAMOUT_WATERMARK

MTStreamBufOut()

Description

Instead of using **MTPlayBuffer()** to play a specified size of voice data, the application can use a stream output function. This enables buffering into the internal queue for playback until the user stops the stream function.

The timeout specifies the waiting time for queue availability. Should timeout be set to 0, this function returns immediately. If there is enough space in the queue to satisfy the requested data length, data will be moved from the user buffer to the queue. Otherwise, the data length will be set to 0 on return. Should the timeout be set to none zero, this function returns when either the specified timeout expires or a data block of the specified size is moved to the queue buffer, whichever happens first. In both cases, the data length field will contain the actual byte count of data been moved from the user buffer to the internal queue. The timeout period is at least 20 milliseconds.

The stream data length should contain the user specified data length and will be set to the actual data moved into the internal buffer on return.

NOTE: The SmartWORKS DLL limits buffer size: maximum streaming buffer size is set to 1MBytes, minimum cap to 1KBytes.

Prototype

MT_RESULT MTStreamBufOut(

const CHANNEL	nChannel,
const DWORD	dwTimeoutMs,
PUCHAR const	pBuf,
PULONG const	pBufSize
);	

Parameters

nChannel	channel number
dwTimeoutMs	timeout (in ms) when queue is full this value must be at least 20 ms. (-1= wait until buffer space is available)
pBuf	pointer to data buffer to feed into play-back streaming
pBufSize	application buffer size

Return Codes

MT_RET_OK	No error
MT_RET_IO_PENDING	Streaming still in startup process
MT_RET_NOT_ENOUGH_BUFSPACE	The amount of free space does not satisfy request with timeout set to 0
MT_RET_SERVICE_NOT_STARTED	Streaming is not started

MT_RET_INTERNAL_ERROR	SDK timed out while queueing decode data
MT_RET_BUSY	Channel is busy
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_CHANNEL_NOT_OPENED	Channel device not opened for access
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_API_THREAD_PROTECTED	Another thread is currently accessing this API

Events

EVT_STREAMOUT_COMP

MTStreamBufIn()

Description

Instead of using **MTRecBuffer()** to record a specified size of voice data, application can use streaming functions. This enables buffering into the internal queue and a stream input function to get voice block one by one until either the application stops the stream function or the specified termination situation occurs. Note that the termination condition is set through **MTStartStreaming()** call.

The timeout specifies the waiting time for data availability. Should timeout be set to 0, this function returns immediately. If there is enough data in the queue to satisfy the requested data length, data will be moved to the user buffer. Otherwise, the data length will be set to 0 on return. Should the timeout be set to a positive number, this function returns when either the specified timeout expires or a data block of the requested size is moved to the user buffer, whichever happens first. In both cases, the data length field will contain the actual byte count of data been moved to the user provided buffer. The timeout period is at least 20 milliseconds. Should the timeout period be set to -1, this function returns when the specified data length is satisfied. Setting timeout period to -1 is equivalent to setting no timeout period.

The stream data length should contain the user requested data length and will be set to the actual data moved into the user buffer on return.

NOTE: The SmartWORKS DLL limits buffer size: maximum streaming buffer size is set to 1MBytes, minimum cap to 1KBytes.

Prototype

MT_RESULT MTStreamBufIn(

const CHANNEL	nChannel,
const DWORD	dwTimeoutMs,
PUCHAR const	pBuf,
PULONG const	pBufSize
);	

Parameters

nChannel	channel number
dwTimeoutMs	timeout (in ms) when queue is empty, this value must be at least 20 ms, (-1= wait until queue data is available)
pBuf	pointer to buffer to take the streaming data
pBufSize	application buffer size

Return Codes

MT_RET_OK	No error
MT_RET_IO_PENDING	Streaming still in startup process
MT_RET_SERVICE_NOT_STARTED	Streaming is not started
MT_RET_INTERNAL_ERROR	SDK timed out while queueing decode data
MT_RET_BUSY	Channel is busy
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_CHANNEL_NOT_OPENED	Channel device not opened for access
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_NOT_ENOUGH_DATA	The amount of collected data does not satisfy request with timeout set to 0
MT_RET_SERVICE_STOPPED	Streaming data is read after termination occurs
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_API_THREAD_PROTECTED	Another thread is currently accessing this API

Events

EVT_STREAMIN_COMP

MTStreamBufPause()

Description

Pause the output streaming. The playback will be paused immediately. Decode data can still be queued so long as there is enough free space in the internal decode streaming buffer. The user application can either stop the decode streaming or resume playback through **MTStreamBufResume()**. This API function does not apply to encode streaming.

Prototype

```
MT_RESULT MTStreamBufPause(  
    const CHANNEL          nChannel  
);
```

Parameters

nChannel channel number

Return Codes

MT_RET_OK	No error
MT_RET_SERVICE_NOT_STARTED	Streaming is not started
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_CHANNEL_NOT_OPENED	Channel device not opened for access
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_IDLE	Decode (playback) streaming is already paused
MT_RET_API_THREAD_PROTECTED	Another thread is currently accessing this API

Events

EVT_STREAMOUT_PAUSED

MTStreamBufResume()

Description

MTStreamBufResume() resumes any paused output streaming on a specified channel.

Prototype

MT_RESULT MTStreamBufResume(

const CHANNEL nChannel
);

Parameters

nChannel channel number

Return Codes

MT_RET_OK	No error
MT_RET_SERVICE_NOT_STARTED	Streaming is not started
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_CHANNEL_NOT_OPENED	Channel device not opened for access
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_BUSY	Decode (playback) streaming is not paused
MT_RET_API_THREAD_PROTECTED	Another thread is currently accessing this API

Events

None

STREAMING - STREAM CONTROL

MTStopStreaming()

Description

Stops the streaming of data.
The parameter Action specifies how it should be stopped.

Stop mode MT_STREAM_STOP_NOW aborts streaming immediately. The process is killed without fetching data from the buffer.

Stop mode MT_STREAM_STOP_DELAY stops streaming after the currently queued streaming buffer function completes. You are required to get all the data using **MTStreamBufln()** since streaming does not technically stop until all the data has been fetched from the buffer.

Prototype

MT_RESULT MTStopStreaming(

```
const CHANNEL      nChannel,
const USHORT      Action,
const USHORT      StopMode
);
```

Parameters

nChannel	channel number
Action	stream action of the following: MT_STREAM_INPUT MT_STREAM_OUTPUT
StopMode	stop mode of: MT_STREAM_STOP_NOW MT_STREAM_STOP_DELAY

Return Codes

MT_RET_OK	No error
MT_RET_SERVICE_NOT_STARTED	Streaming is not started
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_CHANNEL_NOT_OPENED	Channel device not opened for access
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_BUSY	Decode (playback) streaming is not paused
MT_RET_API_THREAD_PROTECTED	Another thread is currently accessing this API

Events

```
EVT_STREAMIN_STOP
EVT_STREAMOUT_STOP
```

STRUCTURE FUNCTIONS

MTGetPSTNParams()

This API was formerly called **MTGetPSTN()**.

Description

MTGetPSTNParams() returns the current PSTN parameter settings. Structure MT_PSTN is detailed in section **MTSetPSTNParams()**.

Prototype

MT_RESULT MTGetPSTNParams (

```
const CHANNEL          nChannel,
PMT_PSTN const         pRegs,
PULONG const          pLength
);
```

Parameters

nChannel	channel number
pRegs	pointer to MT_PSTN structure. This structure is defined on the next pages were MTSetPSTNParams is defined.
pLength	Set by the user application, this is a pointer to the size of the given buffer for the MT_PSTN structure.

NOTE: For backwards compatability with older AudioCodes products, the LENGTH parameter for Set APIs allows the SDK to set up to the specified size. The rest of parameters are not changed. MT_RET_OK is returned.

Return Code

MT_RET_OK	No error
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_CHANNEL_NOT_OPENED	Channel device not opened for access
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_BUSY	Decode (playback) streaming is not paused
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_DATA_TRUNCATED	Data copied but truncated; the buffer size set by the user application is not large enough for this API request. <i>FlagDataTruncated</i> field of the MTSYS_CONFIG or the MTSYS_CONFIGURATION data structure must be enabled (1) to receive this return code, otherwise MT_RET_OK is used, and the truncated data is passed to the user application

Events

None

CODE EXAMPLE**MTGetPSTNParams() Code Example**

```
ZeroMemory(&pstn, size);  
  
result =  
MTGetPSTNParams(channel, &pstn, &size);  
if(result == MT_RET_OK)  
{  
    printf("MTGetPSTNParams() successful...\n");  
    ULONG ringCount = pstn.ring_count;  
    ULONG loopDeglitch = pstn.loop_deglitch;  
    ULONG loopDelay = pstn.loop_delay;  
    printf("Ring count: %u\nLoop deglitch:  
%u\nLoop delay: %u\n", ringCount, loopDeglitch, loopDelay);  
}
```

A complete code example is available on the AudioCodes website:

<http://www.audiocodes.com/blades/support.htm>

MTSetPSTNParams()

This API was formerly called **MTSetPSTN()**

Description

MTSetPSTNParams() sets the PSTN parameters. The telephone signal detection requires some timing parameters. These can be set with the **MTSetPSTNParams()** function using the MT_PSTN structure. Use this structure at any time the system is active*

Prototype

```
MT_RESULT MTSetPSTNParams (
    const CHANNEL      nChannel,
    PMT_PSTN const     pRegs,
    PULONG const       pLength
);
```

Parameter

nChannel	channel number
pRegs	pointer to an MT_PSTN structure
pLength	Set by the user application, this is a pointer to the size of the given buffer for the MT_PSTN structure.

NOTE: For backwards compatability with older AudioCodes products, the LENGTH parameter for Set APIs allows the SDK to set up to the specified size. The rest of parameters are not changed. MT_RET_OK is returned.

Structure

TABLE 29: MT_PSTN

Type	Name	Purpose	Default	Units
USHORT	flash_char	Flash character	&	ascii
USHORT	flash_time	Flash time	500	1 ms
USHORT	wink_delay	Wink delay	150	1ms
USHORT	wink_time	Wink time	150	1ms
USHORT	pause_char	Pause character	,	ascii
USHORT	pause_time	Pause time	2000	1 ms
USHORT	pulse_on	Pulse dialing on-hook length	40	1 ms
USHORT	pulse_off	Pulse dialing off-hook length	60	1 ms
USHORT	pulse_delay_time	Pulse dialing inter-digit delay	1000, valid value: 10-2550	1 ms
USHORT	loop_deglitch	Minimum glitch on or off	200	1 ms
USHORT	loop_delay	Loop event delay	10	1 ms

TABLE 29: MT_PSTN

Type	Name	Purpose	Default	Units
USHORT	<code>ring_on_time</code>	Ring on duration	300	1 ms
USHORT	<code>ring_off_time</code>	Ring off duration	500	1 ms
USHORT	<code>ring_reset</code>	duration (ms) to reset ring count to 0	8000	1 ms
USHORT	<code>ring_count</code>	Ring cycle to report event	1	
USHORT	<code>ring_trigger</code>	Ring detection in leading(0) or trailing(1)	0	flag
USHORT	<code>ring_autoanswer</code>	Auto-answer (0=disable, 1=enable)	0	flag
USHORT	<code>cpm_start_delay</code>	CPM analysis delay after dialing	1000	1 ms
USHORT	<code>offhook_delay</code>	off-hook event delay	500	1 ms
USHORT	<code>min_rings_no_answer</code>	Minimum cycles to report	no answer	4
USHORT	<code>max_sig_time</code>	Maximum continuous signal time	6500	ms
USHORT	<code>max_sil_time</code>	Maximum continuous silence time	40000	ms
USHORT	<code>ring_deglitch</code>	debouncing time for ring detection	200 (320 for AT boards) valid value: 10-2550	ms
USHORT	<code>reserved[3];</code>			

ANALOG_VFXO_LS_E is dedicated for channels on LD809 and LD409.

<code>flash_char</code>	character used to designate hook-flash
<code>flash_time</code>	length of hook-flash
<code>wink_time</code>	wink delay and duration
<code>loop_deglitch</code>	minimum loop glitch time ranged from 10 to 2550 ms
<code>loop_delay</code>	With MTCallString(), the channel dials a number and goes into CPMODE_COMP mode. A delay is started during which a loop current drop will not be reported as connect, but as a regular loop drop. This is to prevent false CPM_CONNECT or EVT_LOOP_STOP messages. This parameter sets the delay.
<code>ring_count</code>	minimum ring cycle to report ring event
<code>ring_trigger</code>	ring detection on leading or trailing signal cycle
<code>cpm_start_delay</code>	the delay time to report CPM event after dialing

offhook_delay	delay time to report the EVT_OFFHOOK event when a channel goes to the off-hook state
min_rings_no_answer	the number of ringbacks or double ringbacks before a no answer is reported
max_sig_time	the maximum continuous signal time before no-ringback is reported. This also sets the activity timeout for CPM_NO_RINGBACK
max_sil_time	the maximum continuous silence time before no-ringback is reported
Ring_deglitch	minimum ring deglitch time ranged from 10 to 2550 ms; applicable only to ANALOG_VDS, ANALOG_FXO_LC, and ANALOG_VF XO_LS_E default values are: 160 ms - LD board 50 ms - PT board 320 ms - AT board

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_CHANNEL_NOT_OPENED	Channel device not opened for access
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_INVALID_PARAM	Invalid argument passed to function

Events

None

CODE EXAMPLE**Code Example MTSetPSTNParams()**

```
handleMT(MTSetPSTNParams(channel,&pstn,&size),"MTSetPSTNParams() channel
%d",channel);

if(handleMT(MTGetPSTNParams(channel,&pstn,&size),"MTGetPSTNParams() channel
%d",channel))

    {
        ULONG ringCount = pstn.ring_count;
        ULONG loopDeglitch = pstn.loop_deglitch;
        ULONG loopDelay = pstn.loop_delay;
        cout << "Ring count = " << ringCount << endl
        << "Loop deglitch = " << loopDeglitch << endl
        << "Loop delay = " << loopDelay << endl <<

endl;
    }
```

A complete code example is available on the AudioCodes website:

<http://www.audiocodes.com/blades/support.htm>

MTSetToneGenParams()

This API was formerly called **MTSetGenRegs()**

Description

MTSetToneGenParams() sets the DTMF/MF tone generation parameters defined in MT_GENREGS structure as follows.

Note: MF is not supported in current version.

Prototype

MT_RESULT MTSetToneGenParams (

```
const CHANNEL      nChannel,
const UCHAR        Type,
PMT_GENREGS const  pRegs,
PULONG const       pLength
);
```

Parameters

nChannel	channel number
Type	tone type: TYPE_DTMF or TYPE_MF
pRegs	pointer to MT_GENREGS structure
pLength	Set by the user application, this is a pointer to the size of the given buffer for the MT_GENREGS structure

NOTE: For backwards compatability with older AudioCodes products, the LENGTH parameter for Set APIs allows the SDK to set up to the specified size. The rest of parameters are not changed. MT_RET_OK is returned.

Structure

NOTE: If any values are set to 0, the application will replace them with the default setting.

TABLE 30: MT_GENREGS - DEFAULT FOR ALL BOARDS

Type	Name	Purpose	Default	Units
USHORT	width	Tone duration time	100	ms
USHORT	gap	Inter-tone duration	50	ms
USHORT	low_ampl	Amplitude for low frequency	10156 (-7dBm)	
USHORT	high_ampl	Amplitude for high frequency	10156 (-7dBm)	

width	The length of a DTMF or MF tone in ms unit. this field cannot be 0
gap*	The inter-digit length between two tones in ms unit

low_ampl The amplitude for tone low frequency. The amplitude of the output sinewave is proportional to the value of this parameter (range 0 to 0x7FFF) with a digital milliwatt corresponding to an amplitude of approximately 0x5900.

high_ampl The amplitude for tone high frequency. See explanation of the low_ampl parameter for more information.

* The inter-tone duration is guaranteed as a minimum gap value. At times, a longer delay between digit generation may be noted.

TABLE 31: MT_GENREGS - DEFAULT FOR AT AND LD BOARDS

Type	Name	Purpose	Default	Units
USHORT	width	Tone duration time	75	ms
USHORT	gap	Inter-tone duration	75	ms
USHORT	low_ampl	Amplitude for low frequency	9500(-7.6dBm)	
USHORT	high_ampl	Amplitude for high frequency	12000(-5.6dBm)	

width The length of a DTMF or MF tone in ms unit. This field cannot = 0.

gap* The inter-digit length between two tones in ms unit

low_ampl The amplitude for tone low frequency

high_ampl The amplitude for tone high frequency

* The inter-tone duration is guaranteed as a minimum gap value. At times, a longer delay between digit generation may be noted.

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_CHANNEL_NOT_OPENED	Channel device not opened for access
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_INVALID_PARAM	Invalid argument passed to function

Events

None

MTGetToneGenParams ()

This API was formerly called **MTGetGenRegs()**

Description

MTGetToneGenParams() returns the DTMF/MF tone generation settings. The tone type indicates which generation structure is to be retrieved, DTMF or MF. Refer to **MTSetToneGenParams()** for details of MT_GENREGS structure which controls the generation of DTMF/MF tones.

Prototype

MT_RESULT MTGetToneGenParams (

```
const CHANNEL      nChannel,
const UCHAR        Type,
PMT_GENREGS const  pRegs,
PULONG const       pLength
);
```

Parameters

nChannel	channel number
Type	tone type; TYPE_DTMF or TYPE_MF
pRegs	pointer to MT_GENREGS structure
pLength	Set by the user application, this is a pointer to the size of the given buffer for the MT_GENREGS structure.

NOTE: For backwards compatability with older AudioCodes products, the LENGTH parameter for Set APIs allows the SDK to set up to the specified size. The rest of parameters are not changed. MT_RET_OK is returned.

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_CHANNEL_NOT_OPENED	Channel device not opened for access
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_DATA_TRUNCATED	Data copied but truncated; the buffer size set by the user application is not large enough for this API request. <i>FlagDataTruncated</i> field of the MTSYS_CONFIG or the MTSYS_CONFIGURATION data structure must be enabled (1) to receive this return code, otherwise MT_RET_OK is used, and the truncated data is passed to the user application

Events

None

MTSetToneDetectParams()

This API was formerly called **MTSetDetectRegs()**

Description

MTSetToneDetectParams() sets the DTMF/MF detection structure. This structure controls the detection of DTMF/MF tones per specification of parameter TYPE_DTMF or TYPE_MF.

All zero parameters are ignored.

This API sets the same tone detection parameters for the primary input of the channel or both inputs if mixing is enabled. To set these parameters on a per input basis use **MTChInputSetToneDetectParams()**.

Scenario

Each channel (primary and secondary) is equipped with a DTMF detector. The detector has been optimized for high immunity to talk-off and the ability to reliably detect short signals (40 ms). All 16 DTMF digits can be detected.

Users have the option of setting detection to occur at the beginning of the signal (leading edge) or after the signal ended (trailing edge). **MTSetToneDetectParams()** API allows user to specify leading or trailing edge detection. Leading edge detection creates a scenario where start of detection is reported as soon as the detection hardware has detected a valid digit. When trailing edge detection is selected, the reporting of detection occurs after the DTMF signal ends.

Start of detection is always reported at the end of the digit or 100 ms after the beginning of the digit, whichever occurs first.

The detector has a dynamic range of -38 dBm to 0 dBm, acceptable twist is set to

8 dB forward and 4 dB reverse, frequency variation is +/- 1.5% for detection and

+/- 2.5% for rejection.

Use this API to configure DTMF and MF detection settings. Use the second parameter to specify whether TYPE_DTMF or TYPE_MF is to be configured.

Prototype

MT_RESULT MTSetToneDetectParams (Parameters)

```
const CHANNEL      nChannel,
const UCHAR        Type,
PMT_DETECTREGS const pRegs,
PULONG const      pLength
);
```

nChannel

channel number

Type

tone type:
0 = TYPE_DTMF
1 = TYPE_MF

pRegs

pointer to MT_DETECTREGS structure

pLength

Set by the user application, this is a
pointer to the size of the given buffer
for the MT_DETECTREGS structure.

NOTE: For backwards compatability with older AudioCodes products, the LENGTH parameter for Set APIs allows the SDK to set up to the specified size. The rest of parameters are not changed. MT_RET_OK is returned.

Structure

Structure MT_DETECTREGS is defined as follows:

TABLE 32: MT_DETECTREGS

Type	Name	Purpose	Default	Units
USHORT	subtype	Tone type specification	MF_R2	
USHORT	trigger	Trailing or leading edge detection	LEADING_EDGE TRAILING_EDGE	flag

subtype

This field specifies the tone types of R1 and R2. This field only applies when tone type is set to TYPE_MF:
Values are:
0 = SUBTYPE_MF_R1 - MF R1 tone detection
all other values = SUBTYPE_MF_R2 - MF R2 tone detection
This parameter is ignored for DTMF tones.

trigger

This flag specifies if a valid DTMF or MF digit is to be reported on the leading edge or the trailing edge of the DTMF signal. If leading edge is selected, the digit will be reported as soon as the detection hardware has detected a valid digit.
Values are:
1 = TRAILING_EDGE - trailing edge detection
2 = LEADING_EDGE - leading edge detection

Return Codes

MT_RET_OK

No error

MT_RET_INVALID_CHANNEL

Channel number is out of range

MT_RET_CHANNEL_NOT_OPENED

Channel device not opened for access

MT_RET_NOT_AVAILABLE

Driver is not loaded

MT_RET_INVALID_PARAM

Invalid argument passed to function

Events

None

MTChInputSetToneDetectParams()

This API was formerly called *MTSetToneDetectParams()*.

Description

MTChInputSetToneDetectParams() sets the DTMF/MF detection structure. This structure controls the detection of DTMF/MF tones per specification of parameter TYPE_DTMF or TYPE_MF.

All zero parameters are ignored.

This API sets the tone detection parameters per each input per channel. One can still use **MTSetToneDetectParams()** to set the same values for both channel inputs.

Scenario

Each channel (primary and secondary) is equipped with a DTMF detector. The detector has been optimized for high immunity to talk-off and the ability to reliably detect short signals (40 ms). All 16 DTMF digits can be detected.

Users have the option of setting detection to occur at the beginning of the signal (leading edge) or after the signal ended (trailing edge). **MTChInputSetToneDetectParams()** API allows user to specify leading or trailing edge detection. Leading edge detection creates a scenario where start of detection is reported as soon as the detection hardware has detected a valid digit. When trailing edge detection is selected, the reporting of detection occurs after the DTMF signal ends.

Start of detection is always reported at the end of the digit or 100 ms after the beginning of the digit, whichever occurs first.

The detector has a dynamic range of -38 dBm to 0 dBm, acceptable twist is set to 8 dB forward and 4 dB reverse, frequency variation is +/- 1.5% for detection and +/- 2.5% for rejection.

Use this API to configure DTMF and MF detection settings. Use the second parameter to specify whether TYPE_DTMF or TYPE_MF is to be configured.

Prototype

MT_RESULT MTChInputSetToneDetectParams (Parameters

const CHANNEL	nChannel,	
const SHORT	nInput,	
const UCHAR	Type,	
PMT_DETECTREGS const	pRegs,	
PULONG const	pLength	
);		
	nChannel	channel number
	nInput	Input index: 0=primary, 1=secondary

Type	tone type: 0 = TYPE_DTMF 1 = TYPE_MF
pRegs	pointer to MT_DETECTREGS structure
pLength	Set by the user application, this is a pointer to the size of the given buffer for the MT_DETECTREGS structure.

NOTE: For backwards compatability with older AudioCodes products, the LENGTH parameter for Set APIs allows the SDK to set up to the specified size. The rest of parameters are not changed. MT_RET_OK is returned.

Structure

Structure MT_DETECTREGS is defined as follows:

TABLE 33: MT_DETECTREGS

Type	Name	Purpose	Default	Units
USHORT	subtype	Tone type specification	MF_R2	
USHORT	trigger	Trailing or leading edge detection	LEADING_EDGE TRAILING_EDGE	flag

subtype	This field specifies the tone types of R1 and R2. This field only applies when tone type is set to TYPE_M: Values are: 0 = SUBTYPE_MF_R1 - MF R1 tone detection all other values = SUBTYPE_MF_R2 - MF R2 tone detection This parameter is ignored for DTMF tones.
trigger	This flag specifies if a valid DTMF or MF digit is to be reported on the leading edge or the trailing edge of the DTMF signal. If leading edge is selected, the digit will be reported as soon as the detection hardware has detected a valid digit. Values are: 1 = TRAILING_EDGE - trailing edge detection 2 = LEADING_EDGE - leading edge detection

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_CHANNEL_NOT_OPENED	Channel device not opened for access
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_INVALID_PARAM	Invalid argument passed to function

Events

None

MTChInputGetToneDetectParams()

This API used to be called **MTGetToneDetectParams()**.

Description

MTChInputGetToneDetectParams() returns the DTMF/MF detection structure. The regstype indicates whether the DTMF or MF detection register is retrieved. This structure controls the detection of DTMF/MF digits. Refer to **MTChInputSetToneDetectParams()** or **MTSetToneDetectParams()** for details.

Prototype

MT_RESULT MTChInputGetToneDetectParams (

```
const CHANNEL      nChannel,
const SHORT        nInput,
UCHAR const        Type,
PMT_DETECTREGS const pRegs,
PULONG const       pLength
);
```

Parameters

nChannel	channel number
nInput	Input index: 0=primary, 1=secondary
Type	tone type, TYPE_DTMF or TYPE_MF
pRegs	pointer to MT_DETECTREGS structure
pLength	Set by the user application, this is a pointer to the size of the given buffer for the MT_DETECTREGSS structure

NOTE: For backwards compatability with older AudioCodes products, the LENGTH parameter for Set APIs allows the SDK to set up to the specified size. The rest of parameters are not changed. MT_RET_OK is returned.

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_CHANNEL_NOT_OPENED	Channel device not opened for access
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_INVALID_PARAM	Invalid argument passed to function

MT_RET_DATA_TRUNCATED

Data copied but truncated; the buffer size set by the user application is not large enough for this API request. *FlagDataTruncated* field of the MTSYS_CONFIG or the MTSYS_CONFIGURATION data structure must be enabled (1) to receive this return code, otherwise MT_RET_OK is used, and the truncated data is passed to the user application

Events

None

MTSetDTMFDetectParams()

Description

This API is only used when the tone detection trigger is set to LEADING_EDGE detection. This is set by using the **MTSetToneDetectParams()** API and set the Trigger field in the MT_DetectRegs structure.

This API sets the parameters which control tone detection and reporting.

Prototype

MT_RESULT MTSetDTMFDetectParams(

```
const CHANNEL          nChannel,
PMT_DTMF_DETECTPARAMS const pParams,
PULONG const          pLength
);
```

Parameters

nChannel	channel number
pParams	pointer to MT_DTMF_DETECTPARAMS structure
pLength	Set by the user application, this is a pointer to the size of the given buffer for the MT_DTMF_DETECTPARAMS structure.

NOTE: For backwards compatability with older AudioCodes products, the LENGTH parameter for Set APIs allows the SDK to set up to the specified size. The rest of parameters are not changed. MT_RET_OK is returned.

Structure

Structure MT_DTMF_DETECTPARAMS is defined as follows:

TABLE 34: MT_DTMF_DETECTPARAMS

Type	Name	Purpose	Default	Units
USHORT	min_detect_time	Min. time to detect a DTMF tone, 40 - 100 ms	40	ms
USHORT	max_report_time	Max. time to report leading edge detection of a DTMF tone. Value must fall within the range of min_detect_time to 100	100	ms

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_CHANNEL_NOT_OPENED	Channel device not opened for access
MT_RET_NOT_AVAILABLE	Driver is not loaded

MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_DATA_TRUNCATED	Data copied but truncated; the buffer size set by the user application is not large enough for this API request. <i>FlagDataTruncated</i> field of the MTSYS_CONFIG or the MTSYS_CONFIGURATION data structure must be enabled (1) to receive this return code, otherwise MT_RET_OK is used, and the truncated data is passed to the user application

Events

None

MTChInputSetDTMFDetectParams()

Formally called **MTSetDTMFDetectParams()**.

Description

This API is only used when the tone detection trigger is set to LEADING_EDGE detection. This is set by using the **MTSetToneDetectParams()** API and set the Trigger field in the MT_DetectRegs structure.

This API sets the parameters which controls tone detection and reporting for the specified input per channel.

Prototype

MT_RESULT MTChInputSetDTMFDetectParams(

```
const CHANNEL          nChannel,
const SHORT            nInput,
PMT_DTMF_DETECTPARAMS const pParams,
PULONG const          pLength
);
```

Parameters

nChannel	channel number
nInput	Input index: 0=primary, 1=secondary
pParams	pointer to MT_DTMF_DETECTPARAMS structure
pLength	Set by the user application, this is a pointer to the size of the given buffer for the MT_DTMF_DETECTPARAMS structure.

NOTE: For backwards compatability with older AudioCodes products, the LENGTH parameter for Set APIs allows the SDK to set up to the specified size. The rest of parameters are not changed. MT_RET_OK is returned.

Structure

Structure MT_DTMF_DETECTPARAMS is defined as follows:

TABLE 35: MT_DTMF_DETECTPARAMS

Type	Name	Purpose	Default	Units
USHORT	min_detect_time	Min. time to detect a DTMF tone, 40 - 100 ms	40	ms
USHORT	max_report_time	Max. time to report leading edge detection of a DTMF tone. Value must fall within the range of min_detect_time to 100	100	ms

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_CHANNEL	Channel number is out of range

MT_RET_CHANNEL_NOT_OPENED	Channel device not opened for access
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_DATA_TRUNCATED	Data copied but truncated; the buffer size set by the user application is not large enough for this API request. <i>FlagDataTruncated</i> field of the MTSYS_CONFIG or the MTSYS_CONFIGURATION data structure must be enabled (1) to receive this return code, otherwise MT_RET_OK is used, and the truncated data is passed to the user application

Events

None

MTChInputGetDTMFDetectParams()

Formally called **MTGetDTMFDetectParams()**.

Description

This API is only used when the tone detection trigger is set to LEADING_EDGE detection. This is set by using the **MTSetToneDetectParams()** API and set the Trigger field in the MT_DetectRegs structure.

This API gets the parameters which control tone detection and reporting on the specified input of each channel.

Prototype

MT_RESULT MTChInputGetDTMFDetectParams (

```
const CHANNEL          nChannel,
const SHORT            nInput,
PMT_DTMF_DETECTPARAMS const pParams,
PULONG const           pLength
);
```

Parameters

nChannel	channel number
nInput	Input Index: 0=primary, 1=secondary
pParams	pointer to MT_DTMF_DETECTPARAMS structure
pLength	Set by the user application, this is a pointer to the size of the given buffer for the MT_DTMF_DETECTPARAMS structure.

NOTE: For backwards compatability with older AudioCodes products, the LENGTH parameter for Set APIs allows the SDK to set up to the specified size. The rest of parameters are not changed. MT_RET_OK is returned.

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_CHANNEL_NOT_OPENED	Channel device not opened for access
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_INVALID_PARAM	Invalid argument passed to function

MT_RET_DATA_TRUNCATED

Data copied but truncated; the buffer size set by the user application is not large enough for this API request. *FlagDataTruncated* field of the MTSYS_CONFIG or the MTSYS_CONFIGURATION data structure must be enabled (1) to receive this return code, otherwise MT_RET_OK is used, and the truncated data is passed to the user application

Events

None

ACTIVITY DETECT (ACTD) CONTROL

MTSetACTDParams()

This API was formerly called *MTSetACTDRegs()*

Description

MTSetACTDParams() sets the silence/activity detection configuration of both inputs on a channel. By default, the primary input is the only input used.

Prototypes

```
MT_RESULT MTSetACTDParams (  
    const CHANNEL          nChannel,  
    PMT_ACTPARAMS const    pActParams,  
    PULONG const           pLength  
);
```

Parameters

nChannel	Channel number
pActParams	Pointer to MT_ACTPARAMS
pLength	Set by the user application, this is a pointer to the size of the given buffer for the MT_ACTPARAMS structure

NOTE: For backwards compatability with older AudioCodes products, the LENGTH parameter for Set APIs allows the SDK to set up to the specified size. The rest of parameters are not changed. MT_RET_OK is returned.

Structure

MT_ACTPARAMS structure is defined as follows:

TABLE 36: MT_ACTPARAMS

Type	Parameter	Description
FLOAT	threshold_low	Silence threshold 0 to -60.0 dBm, default -51.0 dBm
FLOAT	threshold_high	Activity threshold 0 to -60.0 dBm, default -48.0 dBm
USHORT	min_silence	min. silence deglitch time, default 40 ms
USHORT	max_silence	max silence duration, default 30000 ms
USHORT	min_activity	min. activity duration, default 40 ms
USHORT	max_activity	max activity duration, default 10000 ms

threshold_high	Value for Activity High threshold, ranged from 0.0 to -60.0 dBm. Must be greater than or equal to threshold_low.
threshold_low	Value for Activity Low threshold, ranged from 0.0 dBm to -60.0 dBm. Must be less than or equal to threshold_high.

min_silence	This is silence deglitching parameter and, if not zero, specifies the minimum time, that silence have to be present before the EVT_MON_SILENCE event is issued.
min_activity	This is activity deglitching parameter and, if not zero, specifies the minimum time, that activity have to be present before the EVT_MON_ACTIVITY event is issued. Since activity is measured across 20 ms intervals, the min_activity value should be rounded down to 20 ms.
max_silence	This is silence period parameter and, if not zero, specifies the maximum time, that silence have to be present before the EVT_MAX_SILENCE event is issued. Event EVT_MAX_SILENCE will be issued every specified period of max_silence of time until activity is detected.
max_activity	This is activity period parameter and, if not zero, specifies the maximum time, that activity have to be present before the EVT_MAX_ACTIVITY event is issued. Event EVT_MAX_ACTIVITY will be issued every specified period of max_activity of time until silence is detected.

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_CHANNEL_NOT_OPENED	Channel device not opened for access
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_DATA_TRUNCATED	Data copied but truncated; the buffer size set by the user application is not large enough for this API request. <i>FlagDataTruncated</i> field of the MTSYS_CONFIG or the MTSYS_CONFIGURATION data structure must be enabled (1) to receive this return code, otherwise MT_RET_OK is used, and the truncated data is passed to the user application

Events

None

CODE EXAMPLE**MTSetACTDParams() Code Example**

```
ZeroMemory(&actparams, size);

actparams.max_activity = 0;
actparams.max_silence = 0;
actparams.min_activity = 40;
actparams.min_silence = 1500;
actparams.threshold_high = -48;
actparams.threshold_low = -51;
result =
MTSetACTDParams(channel, &actparams, &size);
if(result == MT_RET_OK)
{
printf("MTSetACTDParams() successful...\n");
ULONG maxActivity = actparams.max_activity;
ULONG maxSilence = actparams.max_silence;
ULONG minActivity = actparams.min_activity;
ULONG minSilence = actparams.min_silence;
FLOAT threshHigh = actparams.threshold_high;
printf("Max activity: %u\nMax silence:
%u\nMin activity: %u\nMin silence: %u\nHigh threshold:
%u\n", maxActivity, maxSilence, minActivity, minSilence, threshHigh);
}
```

A complete code example is available on the AudioCodes website:

<http://www.audiocodes.com/blades/support.htm>

MTChInputSetACTDParams()

This API was formerly called ***MTSetACTDRegs()*** or ***MTSetACTDParams()***.

Description

MTChInputSetACTDParams() sets the silence/activity detection configuration on the specified input per channel.

Prototypes

MT_RESULT MTChInputSetACTDParams (

```
const CHANNEL      nChannel,
const SHORT        nInput,
PMT_ACTPARAMS const pActParams,
PULONG const       pLength
);
```

Parameters

nChannel	Channel number
nInput	Input Index: 0=primary, 1=secondary
pActParams	Pointer to MT_ACTPARAMS
pLength	Set by the user application, this is a pointer to the size of the given buffer for the MT_ACTPARAMS structure

NOTE: For backwards compatability with older AudioCodes products, the LENGTH parameter for Set APIs allows the SDK to set up to the specified size. The rest of parameters are not changed. MT_RET_OK is returned.

Structure

MT_ACTPARAMS structure is defined as follows:

TABLE 37: MT_ACTPARAMS

Type	Parameter	Description
FLOAT	threshold_low	Silence threshold 0 to -60.0 dBm, default -51.0 dBm
FLOAT	threshold_high	Activity threshold 0 to -60.0 dBm, default -48.0 dBm
USHORT	min_silence	min. silence deglitch time, default 40 ms
USHORT	max_silence	max silence duration, default 30000 ms
USHORT	min_activity	min. activity duration, default 40 ms
USHORT	max_activity	max activity duration, default 10000 ms

threshold_high Value for Activity High threshold, ranged from 0.0 to -60.0 dBm. **Must be greater than or equal to threshold_low.**

threshold_low Value for Activity Low threshold, ranged from 0.0 dBm to -60.0 dBm. **Must be less than or equal to threshold_high.**

min_silence	This is silence deglitching parameter and, if not zero, specifies the minimum time, that silence has to be present before the EVT_MON_SILENCE event is issued.
min_activity	This is activity deglitching parameter and, if not zero, specifies the minimum time, that activity has to be present before the EVT_MON_ACTIVITY event is issued. Since activity is measured across 20 ms intervals, the min_activity value should be rounded down to 20 ms.
max_silence	This is silence period parameter and, if not zero, specifies the maximum time, that silence have to be present before the EVT_MAX_SILENCE event is issued. Event EVT_MAX_SILENCE will be issued every specified period of max_silence of time until activity is detected.
max_activity	This is activity period parameter and, if not zero, specifies the maximum time, that activity have to be present before the EVT_MAX_ACTIVITY event is issued. Event EVT_MAX_ACTIVITY will be issued every specified period of max_activity of time until silence is detected.

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_CHANNEL_NOT_OPENED	Channel device not opened for access
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_DATA_TRUNCATED	Data copied but truncated; the buffer size set by the user application is not large enough for this API request. <i>FlagDataTruncated</i> field of the MTSYS_CONFIG or the MTSYS_CONFIGURATION data structure must be enabled (1) to receive this return code, otherwise MT_RET_OK is used, and the truncated data is passed to the user application

Events

None

MTChInputGetACTDParams()

This API was formerly called **MTGetACTDRegs()** or **MTGetACTDParams()**.

Description

Gets the silence/activity detection configuration per input on each channel. The structure MT_ACTPARAMS is defined in the previous section where the API **MTSetACTDParams()** or **MTChInputACTDParams()** is defined.

Prototype

MT_RESULT MTChInputGetACTDParams (

```
const CHANNEL      nChannel,
const Input        nInput,
PMT_ACTPARAMS const pActParams,
PULONG const       pLength
);
```

Parameters

nChannel	channel number
nInput	Input Index: 0=primary, 1=secondary
pActParams	pointer to an MT_ACTPARAMS structure
pLength	Set by the user application, this is a pointer to the size of the given buffer for the MT_ACTPARAMS structure

NOTE: For backwards compatability with older AudioCodes products, the LENGTH parameter for Set APIs allows the SDK to set up to the specified size. The rest of parameters are not changed. MT_RET_OK is returned.

Return Code

MT_RET_OK	No error
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_CHANNEL_NOT_OPENED	Channel device not opened for access
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_DATA_TRUNCATED	Data copied but truncated; the buffer size set by the user application is not large enough for this API request. <i>FlagDataTruncated</i> field of the MTSYS_CONFIG or the MTSYS_CONFIGURATION data structure must be enabled (1) to receive this return code, otherwise MT_RET_OK is used, and the truncated data is passed to the user application

Events

None

MTChInputACTDControl()

MTEnableACTD, **MTDisableACTD** and **MTEnableMixingDetect()** were formally used.

Description

This function enables or disables activity detection on the specified input per channel. Activity detection is enabled by default.

Scenario

When activity detection is enabled, only events such as **EVT_MON_SILENCE** and **EVT_MAX_SILENCE** are reported as long as monitoring silence is turned on through **MTSetEventFilters**; events such as **EVT_MON_ACTIVITY** and **EVT_MAX_ACTIVITY** are only reported if monitoring activity is turned on with **MTSetEventFilters()**. If the line is silent, event **EVT_MAX_SILENCE** will only be reported on the first occurrence to prevent flooding the event queue. The same applies to **EVT_MAX_ACTIVITY** event reporting.

Prototype

MT_RESULT MTEnableACTD (

```
const CHANNEL      nChannel,
const SHORT        nInput,
const BOOLEAN      fOn
);
```

Parameter

nChannel	channel number
nInput	Input Index: 0=primary, 1=secondary
fOn	TRUE for enabled, FALSE for disabled

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_CHANNEL_NOT_OPENED	Channel device not opened for access
MT_RET_NOT_AVAILABLE	Driver is not loaded

Events

EVT_MON_ACTIVITY
EVT_MON_SILENCE
EVT_MAX_SILENCE
EVT_MAX_ACTIVITY

See Also: [MTSetMoni\(\)](#), [MTSetACTDParams\(\)](#)

MTChInputACTDStatus()

Description

This API retrieves the ACTD status for the specified channel input.

Prototype

MT_RESULT MTChInputACTDStatus (

```
const CHANNEL      nChannel,  
const SHORT        nInput,  
BOOLEAN * const    Status  
);
```

Parameter

nChannel	channel number
nInput	Input Index: 0=primary, 1=secondary
fOn	TRUE for enabled, FALSE for disabled

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_CHANNEL_NOT_OPENED	Channel device not opened for access
MT_RET_NOT_AVAILABLE	Driver is not loaded

MTChInputGetACTDDetectPower()

Formally known as **MTGetACTDDetectPower()**.

Description

Get current activity power value of the specified input on the incoming line. This value is measured before any gain is applied to the incoming signal. This API is supported for all channels.

Prototype

MT_RESULT MTChInputGetACTDDetectPower (

```
const CHANNEL      nChannel,  
const SHORT        nInput,  
float * const       pPower  
);
```

Parameters

nChannel	Channel Number
nInput	Input Index: 0=primary, 1=secondary
Power	activity in dBm

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_CHANNEL_NOT_OPENED	Channel device not opened for access
MT_RET_NOT_AVAILABLE	Driver is not loaded

Events

EVT_MON_ACTIVITY
EVT_MON_SILENCE

See Also: [MTSetMoni\(\)](#)

MTEnableMixing()

Description

Enables mixing of the second audio data with the primary audio data before the audio data is sent to the encoder.

Scenario

On SmartWORKS DP and SmartWORKS NGX cards, the mixing of the primary and secondary data streams is enabled by default. For the cards to work properly this setting must remain unchanged from its default state.

Prototypes

MT_RESULT MTEnableMixing (

const CHANNEL

nChannel

);

Parameters

nChannel

channel number

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_CHANNEL_NOT_OPENED	Channel device not opened for access
MT_RET_NOT_AVAILABLE	Driver is not loaded

Events

None

MTDisableMixing()

Description

Disables the mixing of the second audio data with the primary audio data before the audio data is sent to the encoder.

Scenario

SmartWORKS DP and SmartWORKS NGX cards require that the secondary stream mixing always be enabled.

Mixing must be disabled when using the SmartWORKS PCM card.

Prototypes

MT_RESULT MTDisableMixing (

```
const CHANNEL      nChannel  
);
```

Parameters

nChannel	channel number

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_CHANNEL_NOT_OPENED	Channel device not opened for access
MT_RET_NOT_AVAILABLE	Driver is not loaded

Events

None

MTGetMixingStatus()

Description

MTGetMixingStatus() retrieves the status of mixing (whether enabled or disabled).

Prototypes

```
MT_RESULT MTGetMixingStatus(  
    const CHANNEL          nChannel,  
    BOOLEAN * const        pOn  
);
```

Parameters

nChannel	channel number
pOn	pointer to take the status: 0 for disabled, others for enabled

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_CHANNEL_NOT_OPENED	Channel device not opened for access
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_INVALID_PARAM	Invalid argument passed to function

Events

None

MTSetECControl()

This API was formerly called **MTSetPxdParams()**.

Description

MTSetECControl() allows the enabling or disabling of echo cancellation for the specified channel. Echo cancellation can also be controlled through **MTSetPxdParams()**, however this API will become obsolete in June 2004. Use **MTGetECControl()** to check the status of echo cancellation on the specified channel.

This API is only supported on terminate boards with 09 DSPs (DT3209, DT6409, LD409, LD809).

Prototype

MT_RESULT MTSetECControl (

```
const CHANNEL      nChannel,
const USHORT      Value
);
```

Parameters

nChannel	channel number
Value	enable = 1; disable = 0 (default)

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_CHANNEL_NOT_OPENED	Channel device not opened for access
MT_RET_NOT_AVAILABLE	This feature is not available on this board
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_API_THREAD_PROTECTED	Another thread is currently accessing this API

Events

None

MTGetECControl()

Description

Use **MTGetECControl()** to check the status of echo cancellation on the specified channel. This API is only supported on terminate boards with 09 DSPs (DT3209, DT6409, LD409, LD809).

Prototype

```
MT_RESULT MTGetECControl (  
    const CHANNEL          nChannel,  
    PULONG const          pValue  
);
```

Parameters

nChannel	channel number
pValue	the status of echo cancellation on the identified channel. 1= enabled, 0=disabled

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_CHANNEL_NOT_OPENED	Channel device not opened for access
MT_RET_NOT_AVAILABLE	This feature is not available on this board
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_API_THREAD_PROTECTED	Another thread is currently accessing this API

Events

None

MTSetECFreezeAdaptation()

Description

MTSetECFreezeAdaptation() allows the user to freeze the values currently used by the echo cancellation feature. Once enabled, the channel will only rely on these values and will no longer adapt to changes on the network. Echo cancellation must first be enabled to use this API. If echo cancellation is reset (using **MTECControl()**), then the adaptive capabilities of echo cancellation is automatically restarted. Use **MTGetECAdaptation()** to check the status of adaptive echo cancellation on a specified channel.

This API is only supported on terminate boards with 09 DSPs (DT3209, DT6409, LD409, LD809).

Prototype

MT_RESULT MTSetECFreezeAdaptation (

```
const CHANNEL      nChannel,
const USHORT      Value
);
```

Parameters

nChannel	channel number
Value	enable = 1 (frozen); disable(adaptive) = 0 (default)

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_CHANNEL_NOT_OPENED	Channel device not opened for access
MT_RET_NOT_AVAILABLE	This feature is not available on this board
MT_RET_NOT_APPLICABLE	echo cancellation is supported by this board, but is currently disabled
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_API_THREAD_PROTECTED	Another thread is currently accessing this API

Events

None

MTGetECFreezeAdaptation()

Description

Use **MTGetECAdaptation()** to check the status of echo cancellation on the specified channel.

This API is only supported on terminate boards with 09 DSPs (DT3209, DT6409, LD409, LD809).

Prototype

MT_RESULT MTSetECFreezeAdaptation (

const CHANNEL

PUSHORT const

);

nChannel,

pValue

Parameters

nChannel	channel number
pValue	enable = 1 (frozen); disable(adaptive) = 0 (default)

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_CHANNEL_NOT_OPENED	Channel device not opened for access
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_API_THREAD_PROTECTED	Another thread is currently accessing this API

Events

None

MTSetECPARAMS()

Description

MTSetECPARAMS() allows the user to define parameters used when echo cancellation is enabled. Echo cancellation must be enabled with **MTSetECControl()** before this API can be used.

This API is only supported on terminate boards with 09 DSPs (DT3209, DT6409, LD409, LD809).

Prototype

```
MT_RESULT MTSetECPARAMS (
    const CHANNEL          nChannel,
    PMT_ECPARAMS const     pParams,
    PULONG const           pLength
);
```

Parameters

nChannel	channel number
pParams	pointer to the data structure PMT_ECPARAMS
pLength	Set by the user application, this is a pointer to the size of the given buffer for the PMT_ECPARAMS structure

NOTE: For backwards compatability with older AudioCodes products, the LENGTH parameter for Set APIs allows the SDK to set up to the specified size. The rest of parameters are not changed. MT_RET_OK is returned.

Structure

The PMT_ECPARAMS structure is defined as follows::

TABLE 38: PMT_ECPARAMS

Type	Name	Description
SHORT	max_comfort_noise (reserved)	Comfort noise range: -40 to -66 dBm inclusive, -66dBm is the default
SHORT	fEnableNLP	Controls the Non Linear Processing switch: 1 for enable, 0 for disable(default) This switch replaces a residual echo with comfort noise at the level defined with the max_comfort_noise parameter

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_CHANNEL_NOT_OPENED	Channel device not opened for access
MT_RET_NOT_AVAILABLE	This feature is not available on this board
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_API_THREAD_PROTECTED	Another thread is currently accessing this API

Events

None

MTGetECPARAMS()

Description

This API uses to PMT_ECPARAMS structure to obtain set values used by the echo cancellation feature. The PMT_ECPARAMS structure is defined where the API **MTSetECPARAMS()** is defined.

This API is only supported on terminate boards with 09 DSPs (DT3209, DT6409, LD409, LD809).

Prototype

MT_RESULT MTGetECPARAMS (

```
const CHANNEL          nChannel,
PMT_ECPARAMS const    pParams,
PULONG const          pLength
);
```

Parameters

nChannel	channel number
pParams	pointer to the data structure PMT_ECPARAMS
pLength	Set by the user application, this is a pointer to the size of the given buffer for the PMT_ECPARAMS structure

NOTE: For backwards compatability with older AudioCodes products, the LENGTH parameter for Set APIs allows the SDK to set up to the specified size. The rest of parameters are not changed. MT_RET_OK is returned.

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_CHANNEL_NOT_OPENED	Channel device not opened for access
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_DATA_TRUNCATED	Data copied but truncated; the buffer size set by the user application is not large enough for this API request. <i>FlagDataTruncated</i> field of the MTSYS_CONFIG or the MTSYS_CONFIGURATION data structure must be enabled (1) to receive this return code, otherwise MT_RET_OK is used, and the truncated data is passed to the user application
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_API_THREAD_PROTECTED	Another thread is currently accessing this API

Events

None

CPM FUNCTIONS

MTGetCpmToneParams()

This API was formerly called *MTGetCpmTone()*. This API will be obsoleted and has been replaced with *MTChGetCPMSignalParams()*. This API should not be used when *MTChSetCPMSignalParms()* was used to create a signal profile.

Description

MTGetCpmToneParams() retrieves the CPM tone parameters used to create a CPM signal profile.

The MT_CPMTONE structure is defined where the API **MTSetCpmToneParams()** is explained.

Prototype

```
MT_RESULT MTGetCpmToneParams (  
    const CHANNEL          nChannel,  
    const USHORT           Index,  
    const PMT_CPMTONE      pCpm,  
    PULONG const           pLength  
);
```

Parameters

nChannel	channel number
Index	CPM tone index 0 -14
pCpm	pointer to a MT_CPMTONE structure
pLength	Set by the user application, this is a pointer to the size of the given buffer for the MT_CPMTONE structure

NOTE: For backwards compatability with older AudioCodes products, the LENGTH parameter for Set APIs allows the SDK to set up to the specified size. The rest of parameters are not changed. MT_RET_OK is returned.

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_CHANNEL_NOT_OPENED	Channel device not opened for access
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_INVALID_PARAM	Invalid argument passed to function

Events

None

CODE EXAMPLE**MTGetCpmToneParams()**

```

if (handleMT(MTGetCpmToneParams(channel, 0, &cpmtone, &size), "MTGetCpmToneParams()
channel %d", channel))

    {
        USHORT index = cpmtone.index;
        USHORT signame = cpmtone.signalname;
        USHORT sigtype = cpmtone.signaltype;
        cout << "Signal index = " << index << endl
        << "Signal name = " << signame << endl
        << "Signal type = " << sigtype << endl <<

endl;

    }

    handleMT(MTSysShutdown(), "MTSysShutdown()");

```

A complete code example is available on the AudioCodes website:

<http://www.audiocodes.com/blades/support.htm>

MTSetCpmToneParams()

This API was formerly called **MTSetCpmTone()**. This API will be obsoleted and has been replaced with **MTChSetCPMSignalParams()**. New development should use this new API.

Description

MTSetCpmToneParams() sets the CPM tone parameters used to create a CPM signal profile.
There are maximum of 15 CPM signal entries and the index ranges from 0 to 14.
Each CPM signal will only be detected if it's parameters are set correctly and it is enabled. To enable a signal profile set the fEnable flag in the CPM_SIGNAL structure to true, or use **MTSetChCPMSignalControl()**.

Prototype

```
MT_RESULT MTSetCpmToneParams (  
    const CHANNEL          nChannel,  
    const PMT_CPMTONE      pCpm,  
    PULONG const           pLength  
);
```

Parameters

nChannel	channel number
pCpm	pointer to a MT_CPMTONE structure
pLength	Set by the user application, this is a pointer to the size of the given buffer for the MT_CPMTONE structure.

NOTE: For backwards compatability with older AudioCodes products, the LENGTH parameter for Set APIs allows the SDK to set up to the specified size. The rest of parameters are not changed. MT_RET_OK is returned.

Structure

The MT_CPMTONE structure is defined as follows::

TABLE 39: MT_CPMTONE

Type	Name	Description
USHORT	index	Signal Tone Index ID 0-14 When a EVT_MON_ is generated for this signal, the Index ID is reported in the XtraInfo field of the MT_EVENT structure.
USHORT	signalname	CPM_DIAL1, etc. The options for this field are defined in the NtiData.h file. Setting this field determines the type of event generated. If CPM_DIAL1 is selected, the event MON_DIAL is generated for this signal.

TABLE 39: MT_CPMTONE

Type	Name	Description
USHORT	signaltype	Tone type (CPM_TONE, SIT_TONE). The options for this field are defined in the NtiData.h file.
CPM_SIGNAL	signal	CPM tone specification: USHORT fEnable; USHORT fDetectSIT; USHORT fDetectLate; USHORT fDetectCycle; USHORT fDetectLost; USHORT nCycleDetect; USHORT Freq[MAX_CPM_FREQ];* CPM_PULSE Pulse[MAX_CPM_PULSE];

*Values for minimum and maximum pulse and gap durations are in msec.

The firmware samples each signal at 20 ms intervals. When defining the cadence pattern it is recommended that all times be specified in increments of 20 ms, and that they provide at least 20 ms of margin from the actual limits of the signal to be detected.

For compatibility with earlier AudioCodes products, values for frequencies are in Hz, even though what is actually stored internally is the filter number. The frequencies specified must conform to the following rules:

Each frequency must fall within the bandwidth of a filter. If the frequency falls within the passbands of two filters, it is assigned to the filter whose band edge is more distant.

If more than one frequency is specified, the frequencies must be given in order, from low to high.

If more than one frequency is specified, each frequency must be assigned to a different filter.

When the contents of an existing profile are read back by the user, the CPM firmware substitutes the nominal frequency of each filter in the profile. These will not necessarily be the frequencies used to specify the profile.

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_CHANNEL_NOT_OPENED	Channel device not opened for access
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_INVALID_PARAM	Invalid argument passed to function

Events

None

CODE EXAMPLE**Code Example MTSetCpmToneParams()**

```
if (handleMT(MTGetCpmToneParams(channel, 0, &cpmtone, &size), "MTGetCpmToneParams()",  
channel %d", channel))  
  
    {  
        USHORT index = cpmtone.index;  
        USHORT signame = cpmtone.signalname;  
        USHORT sigtype = cpmtone.signaltype;  
        cout << "Signal index = " << index << endl  
        << "Signal name = " << signame << endl  
        << "Signal type = " << sigtype << endl <<  
  
endl;  
  
    }  
  
    cpmtone.index = 0;  
    cpmtone.signalname = CPM_BUSY1;  
    cpmtone.signaltype = CPM_TONE;  
    cpmtone.signal.fEnable = 1;  
    cpmtone.signal.fDetectSIT = 0;  
    cpmtone.signal.fDetectLate = 0;  
    cpmtone.signal.fDetectCycle = 0;  
    cpmtone.signal.fDetectLost = 1;  
    cpmtone.signal.nCycleDetect = 1;  
    cpmtone.signal.Freq[0] = 350;  
    cpmtone.signal.Freq[1] = 440;  
    cpmtone.signal.Freq[2] = 0;  
    cpmtone.signal.Pulse[0].onmin = 140;  
    cpmtone.signal.Pulse[0].onmax = 200;  
    cpmtone.signal.Pulse[0].offmin = 140;  
    cpmtone.signal.Pulse[0].offmax = 200;  
    cpmtone.signal.Pulse[1].onmin = 0;  
    cpmtone.signal.Pulse[1].onmax = 0;  
    cpmtone.signal.Pulse[1].offmin = 0;  
    cpmtone.signal.Pulse[1].offmax = 0;  
    cpmtone.signal.Pulse[2].onmin = 0;  
    cpmtone.signal.Pulse[2].onmax = 0;  
    cpmtone.signal.Pulse[2].offmin = 0;  
    cpmtone.signal.Pulse[2].offmax = 0;  
  
    cout << "Setting..." << endl  
    << "signame = " << cpmtone.signalname << endl  
    << "sigtype = " << cpmtone.signaltype << endl  
  
<< endl;
```

A complete code example is available on the AudioCodes website:

<http://www.audiocodes.com/blades/support.htm>

MTResetCpmToneParams()

This API was formerly called **MTResetCpmTone()**. This API will be obsoleted and has been replaced with **MTChResetCPMSignalParams()**. New development should use the new API.

Description

MTResetCpmToneParams() either disables all CPM signal profiles or sets all CPM signal profiles to default settings.

The mode has two possible values:

0 = disable all CPM signal profiles (each signal profile must then be enabled individually)

1 = reset all signal profiles to factory default

To enable a signal, use **MTChCPMSignalControl()** or use **MTGetCPMToneParams()**, modify the fEnable flag to true, then use **MTSetCPMToneParams()** to set this value.

Each board is shipped with default (North American) signal profiles. If a signal profile is modified using the Control Panel, the factory default settings are written over. Signal profiles set by using **MTSetCpmToneParams()** do not modify factory default settings.

When the SmartWORKS DLL is first loaded, the factory default settings are pulled in and used as channel signal profiles.

NOTE: When this API is used, the Call State Machine is not reset.

Prototype

MT_RESULT MTResetCpmToneParams (

```
const CHANNEL      nChannel,
const UCHAR        Mode
);
```

Parameters

nChannel	Channel Number
Mode	CPM tones mode
	0 = disable all signal profiles
	1 = reset all signal profiles to values stored in the registry

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_CHANNEL_NOT_OPENED	Channel device not opened for access
MT_RET_NOT_AVAILABLE	Driver is not loaded

Events

None

MTChResetCPMSignalParams()

This API was formerly called *MTResetCpmToneParams()*.

Description

MTChResetCpmSignalParams() either disables all CPM signal profiles or sets all CPM signal profiles to default settings.

The mode has two possible values:
0 = disable all CPM signal profiles (each signal profile must then be enabled individually)
1 = reset all signal profiles to factory settings.

To enable a signal, use **MTChCPMSignalControl()** or use **MTChGetCPMSignalParams()**, modify the fEnable flag to true, then use **MTChSetCPMSignalParams()** to set this value.

Each board is shipped with default (North American) signal profiles. If a signal profile is modified using the Control Panel, the factory default settings are written over. Signal profiles set by using **MTChSetCpmSignalParams()** do not modify the factory defaults.

When the SmartWORKS DLL is first loaded, these factory settings are used to set the channel signal profiles.

NOTE: When this API is used, the Call State Machine is not reset.

Prototype

MT_RESULT MTChResetCpmSignalParams (

```
const CHANNEL      nChannel,
const UCHAR        Mode
);
```

Parameters

nChannel	Channel Number
Mode	CPM tones mode 0 = disable all signal profiles 1 = reset all signal profiles to values stored in the registry or the NtiDrv.conf file when running Linux

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_CHANNEL_NOT_OPENED	Channel device not opened for access
MT_RET_NOT_AVAILABLE	Driver is not loaded

Events

None

MTChCPMControl()

This API used to be called **MTSetCpmMode()**.

Description

This API enables/disables CPM on a specific channel.

Once enabled, CPM runs in monitoring mode by default and EVT_MON_*signalname* events are returned to the user application.

A monitor enable, controlled through **MTSetMoni()**, controls the CPM monitor condition. A termination enable, controlled through **MTSetTerm()**, controls the CPM termination conditions.

MTGetCpmStatus() retrieves the current channel status.

NOTE: Using **MTSetChannelToDefault()** resets CPM and Voice Detection to enabled.

Prototype

```
MT_RESULT MTChCPMControl (
    const CHANNEL      nChannel,
    const PUSHORT      Mode
);
```

Parameters

nChannel	Channel Number
Mode	set CPM status: 0 = disabled non zero = enabled

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_CHANNEL_NOT_OPENED	Channel device not opened for access
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_INVALID_PARAM	Invalid argument passed to function

Events

None

See Also: [MTSetMoni\(\)](#); [MTSetTerm\(\)](#)

MTChCPMStatus()

This API used to be called *MTGetCpmMode()*.

Description

MTChCPMStatus() retrieves the status of CPM on a specific channel.
A monitor enable, controlled through **MTSetMoni()**, controls the CPM monitor condition. A termination enable, controlled through **MTSetTerm()**, controls the CPM termination conditions.

NOTE: Using *MTSetChannelToDefault()* resets CPM and Voice Detection to enabled.

Prototype

```
MT_RESULT MTChCPMStatus (  
    const CHANNEL          nChannel,  
    BOOLEAN * const        pOn  
);
```

Parameters

nChannel	Channel Number
pOn	pointer to the CPM status: 0 = disabled non zero = enabled

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_CHANNEL_NOT_OPENED	Channel device not opened for access
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_INVALID_PARAM	Invalid argument passed to function

Events

None

MTGetUsrToneParams()

This API was formerly called **MTGetUsrTone()**. This API will be obsoleted and has been replaced with **MTChGetUsrSignalParams()**. Any Usr Tone created with the **MTChSetUsrSignalParams()** API must use this new API to retrieve information.

Description

MTGetUsrToneParams() allows retrieval of a user defined CPM tone. The MT_USRTONE structure is defined where the API **MTSetUsrToneParams()** is explained.

Prototype

MT_RESULT MTGetUsrToneParams (

```
const CHANNEL      nChannel,
const USHORT      Index,
const PMT_USRTONE pUsrTone,
PULONG const      pLength
);
```

Parameters

nChannel	channel number
Index	signal index 15-19
pUsrTone	pointer to MT_USRTONE structure
pLength	Set by the user application, this is a pointer to the size of the given buffer for the MT_USRTONE structure

NOTE: For backwards compatability with older AudioCodes products, the LENGTH parameter for Set APIs allows the SDK to set up to the specified size. The rest of parameters are not changed. MT_RET_OK is returned.

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_CHANNEL_NOT_OPENED	Channel device not opened for access
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_DATA_TRUNCATED	Data copied but truncated; the buffer size set by the user application is not large enough for this API request. <i>FlagDataTruncated</i> field of the MTSYS_CONFIG or the MTSYS_CONFIGURATION data structure must be enabled (1) to receive this return code, otherwise MT_RET_OK is used, and the truncated data is passed to the user application

Events

None

MTChGetUsrSignalParams()

This API was formerly called **MTGetUsrToneParams()**.

Description

MTChGetUsrSignalParams() allows retrieval of a user defined tone. The MT_USR_TONE structure is defined where the API **MTChSetUsrSignalParams()** is explained.

Prototype

MT_RESULT MTChGetUsrSignalParams (

```
const CHANNEL      nChannel,
const USHORT      Index,
const PMT_USR_TONE pUsrTone,
PULONG const      pLength
);
```

Parameters

nChannel	channel number
Index	signal index 15-19
pUsrTone	pointer to MT_USR_TONE structure
pLength	Set by the user application, this is a pointer to the size of the given buffer for the MT_USR_TONE structure

NOTE: For backwards compatability with older AudioCodes products, the LENGTH parameter for Set APIs allows the SDK to set up to the specified size. The rest of parameters are not changed. MT_RET_OK is returned.

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_CHANNEL_NOT_OPENED	Channel device not opened for access
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_DATA_TRUNCATED	Data copied but truncated; the buffer size set by the user application is not large enough for this API request. <i>FlagDataTruncated</i> field of the MTSYS_CONFIG or the MTSYS_CONFIGURATION data structure must be enabled (1) to receive this return code, otherwise MT_RET_OK is used, and the truncated data is passed to the user application

Events

None

MTClearUsrToneParams()

This API was formerly called ***MTClearUsrTone()***. This API will be obsoleted and has been replaced with ***MTChClearUsrSignalParams()***. If ***MTChSetUsrSignalParams()*** was used to create a new usr tone, then this new API should be used to clear it.

Description

MTClearUsrToneParams() clears the user tone table for a specified channel.

Prototype

```
MT_RESULT MTClearUsrToneParams (  
    const CHANNEL          nChannel  
);
```

Parameters

nChannel	channel number

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_CHANNEL_NOT_OPENED	Channel device not opened for access
MT_RET_NOT_AVAILABLE	Driver is not loaded

Events

None

MTChClearUsrSignalParams()

This API was formerly called *MTClearUsrToneParams()*.

Description

MTChClearUsrSignalParams() clears the user tone table for a specified channel.

Prototype

```
MT_RESULT MTChClearUsrSignalParams (  
    const CHANNEL          nChannel  
);
```

Parameters

nChannel channel number

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_CHANNEL_NOT_OPENED	Channel device not opened for access
MT_RET_NOT_AVAILABLE	Driver is not loaded

Events

None

MTSetUsrToneParams()

This API was formerly called **MTSetUsrTone()**. This API will be obsoleted and has been replaced with **MTChSetUsrSignalParams()**. New development should use this new API.

Description

MTSetUsrToneParams() adds the tone parameters used to define a user defined CPM signal.

Prototype

MT_RESULT MTSetUsrToneParams (

```
const CHANNEL          nChannel,
const PMT_USRTONE      pUsrTone,
PULONG const          pLength
);
```

Parameters

nChannel	channel number
pUsrTone	pointer to a MT_USR_TONE structure
pLength	Set by the user application, this is a pointer to the size of the given buffer for the MT_USR_TONE structure

NOTE: For backwards compatability with older AudioCodes products, the LENGTH parameter for Set APIs allows the SDK to set up to the specified size. The rest of parameters are not changed. MT_RET_OK is returned.

Structure

The MT_USRTONE structure is defined as follows:

TABLE 40: MT_USRTONE

Type	Name	Description
MT_CPMTONE	Tone	Tone specifications as defined in the MT_CPMTONE structure. Refer to explanation below.
UCHAR	asCPMTone	Treat this tone as the specified CPM tone: 0 = USR_NONE, Report EVT_UTONE_ON 1 = USR_DIAL, Report EVT_MON_DIAL 2 = USR_RINGBACK Report EVT_MON_RINGBACK 3 = USR_BUSY Report EVT_MON_BUSY 4 = USR_FAX Report EVT_MON_FAX 5 = USR_SIT Report EVT_MON_SIT 6 = USR_LOOPOFF Report EVT_MON_LOOPOFF 7 = USR_DIGIT, Report EVT_DIGIT and the specified digit through DTMF queue
UCHAR	useDigit	The digit placed in the DTMF queue if asCPMTone is set to USR_DIGIT

The MT_CPMTONE structure is defined as follows:

TABLE 41: MT_CPMTONE

Type	Name	Description
USHORT	index	Signal Tone Index ID 15-19 When a EVT_UTONE_ON, or EVT_UTONE_OFF is generated for this sig- nal, the Index ID is reported in the XtraiInfo field of the MT_EVENT structure.
USHORT	signalname	This field is currently not used when creat- ing a Usr tone.
USHORT	signaltype	Tone type (USR_TONE). The options for this field are defined in the NtiData.h file.
CPM_SIGNAL	signal	CPM tone specification: USHORT fEnable; USHORT fDetectSIT; USHORT fDetectLate; USHORT fDetectCycle; USHORT fDetectLost; USHORT nCycleDetect; USHORT Freq[MAX_CPM_FREQ];* CPM_PULSE Pulse[MAX_CPM_PULSE];

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_CHANNEL_NOT_OPENED	Channel device not opened for access
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_INVALID_PARAM	Error in channel, pointer, or tone index (invalid argument passed to function)

Events

None

MTChSetUsrSignalParams()

This API was formerly called *MTSetUsrToneParams()*.

Description

MTChSetUsrSignalParams() adds the tone parameters used to define a user defined CPM signal.

Refer to the application note: *Understanding Signal Profiles* for more information.

Prototype

MT_RESULT MTChSetUsrSignalParams (

```
const CHANNEL          nChannel,
const PMT_USR_TONE     pUsrTone,
PULONG const          pLength
);
```

Parameters

nChannel	channel number
pUsrTone	pointer to a MT_USRTONE structure
pLength	Set by the user application, this is a pointer to the size of the given buffer for the MT_USRTONE structure

NOTE: For backwards compatability with older AudioCodes products, the LENGTH parameter for Set APIs allows the SDK to set up to the specified size. The rest of parameters are not changed. MT_RET_OK is returned.

Structure

The MT_USR_TONE structure is defined as follows:

TABLE 42: MT_USR_TONE

Type	Name	Description
MT_CPMTONE	Tone	Tone specifications as defined in the MT_CPMTONE structure. Refer to explanation below.
UCHAR	asCPMTone	Treat this tone as the specified CPM tone: 0 = USR_NONE, Report EVT_UTONE_ON 1 = USR_DIAL, Report EVT_MON_DIAL 2 = USR_RINGBACK Report EVT_MON_RINGBACK 3 = USR_BUSY Report EVT_MON_BUSY 4 = USR_FAX Report EVT_MON_FAX 5 = USR_SIT Report EVT_MON_SIT 6 = USR_LOOPOFF Report EVT_MON_LOOPOFF 7 = USR_DIGIT, Report EVT_DIGIT and the specified digit through DTMF queue
UCHAR	useDigit	The digit placed in the DTMF queue if asCPMTone is set to USR_DIGIT

The MT_CPT_TONE structure is defined as follows:

TABLE 43: MT_CPT_TONE

Type	Name	Description
USHORT	index	Signal Tone Index ID 15-19 When a EVT_UTONE_ON, or EVT_UTONE_OFF is generated for this sig- nal, the Index ID is reported in the XtrInfo field of the MT_EVENT structure.
USHORT	signalname	This field is currently not used when creat- ing a Usr tone.
USHORT	signaltype	Tone type (USR_TONE). The options for this field are defined in the NtiData.h file.
CPM_SIGNAL	signal	CPM tone specification: USHORT fEnable; USHORT fDetectSIT; USHORT fDetectLate; USHORT fDetectCycle; USHORT fDetectLost; USHORT nCycleDetect; CPM_FIITER Filter[MAX_CPM_FREQ];* CPM_PULSE Pulse[MAX_CPM_PULSE];

Refer to the applicationi note: *Understanding Signal Profiles* for more
information about creating/modifying signal profiles.

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_CHANNEL_NOT_OPENED	Channel device not opened for access
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_INVALID_PARAM	Error in channel, pointer, or tone index (invalid argument passed to function)

Events

None

MTChCPMSignalControl()

Used to be called **MTSetCPMSigSwitch()**.

Description

MTChCPMSignalControl() allows the user to enable or disable the detection of a given CPM signal.

Prototype

MT_RESULT MTChCPMSignalControl (

```
const CHANNEL      nChannel,  
const USHORT       Index,  
const UCHAR        fState  
);
```

Parameters

nChannel	channel number
Index	Signal Index ID 0 -14
fState	tone Mode:: 0=off 1=on (on by default)

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_CHANNEL_NOT_OPENED	Channel device not opened for access
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_INVALID_PARAM	Error in channel, pointer, or tone index

Events

None

MTChCPMSignalStatus()

This API was formerly called *MTGetCPMSigSwitch()*.

Description

MTChCPMSignalStatus() allows the user to retrieve whether detection has been enabled or disabled for a given CPM signal profile.

Prototype

```
MT_RESULT MTChCPMSignalStatus (  
    const CHANNEL          nChannel,  
    const USHORT           Index,  
    const PCHAR            pState  
);
```

Parameters

nChannel	channel number
Index	Signal Index ID 0-14 for CPM
pState	0 for disabled, others for enabled (1 by default)

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_CHANNEL_NOT_OPENED	Channel device not opened for access
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_INVALID_PARAM	Error in channel, pointer, or tone index

Events

None

MTChUsrSignalControl()

Used to be called **MTSetCPMSigSwitch()**.

Description

MTChUsrSignalControl() allows the user to enable or disable the detection of a given Usr defined signal.

Prototype

MT_RESULT MTChUsrSignalControl (

```
const CHANNEL      nChannel,  
const USHORT       Index,  
const UCHAR        fState  
);
```

Parameters

nChannel	channel number
Index	Signal Index ID 15-19
fState	user signal state: 0=off 1=on (on by default)

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_CHANNEL_NOT_OPENED	Channel device not opened for access
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_INVALID_PARAM	Invalid argument passed to function

Events

None

MTChUsrSignalStatus()

This API was formerly called *MTGetCPMSigSwitch()*.

Description

MTChUsrSignalStatus() allows the user to retrieve whether detection has been enabled or disabled for a given Usr defined signal profile.

Prototype

```
MT_RESULT MTChUsrSignalStatus (  
    const CHANNEL      nChannel,  
    const USHORT       Index,  
    const PCHAR        pState  
);
```

Parameters

nChannel	channel number
Index	Signal Index ID 15-19
pState	0 for disabled, others for enabled (1 by default)

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_CHANNEL_NOT_OPENED	Channel device not opened for access
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_INVALID_PARAM	Error in channel, pointer, or signal index

Events

None

MTChSetCPMSignalParams()

This API was formerly called **MTSetCpmToneParams()**

Description

MTChSetCpmSignalParams() sets the CPM tone parameters used to create a CPM signal profile.

There are maximum of 15 CPM tone entries and the index ranges from 0 to 14.

Each CPM signal will only be detected if it's parameters are set correctly and it is enabled. To enable a signal profile, set the fEnable field to true in the CPT_SIGNAL structure or use the **MTChCPMSignalControl()** API.

It is highly recommended that the user read the Application note: Understanding Signal Profiles before creating and modifying signal profiles.

Prototype

MT_RESULT MTChSetCpmSignalParams (

```
const CHANNEL      nChannel,
const PMT_CPT_TONE pCpt,
PULONG const      pLength
);
```

Parameters

nChannel	channel number
pCpt	pointer to a MT_CPT_TONE structure
pLength	Set by the user application, this is a pointer to the size of the given buffer for the MT_CPT_TONE structure.

NOTE: For backwards compatability with older AudioCodes products, the LENGTH parameter for Set APIs allows the SDK to set up to the specified size. The rest of parameters are not changed. MT_RET_OK is returned.

Structure

The MT_CPT_TONE structure is defined as follows::

TABLE 44: MT_CPT_TONE

Type	Name	Description
USHORT	index	0-14: CPM tone. This Signal Index ID is reported in the XtrInfo field of the MT_EVENT structure when the EVT_MON_ event is generated.
USHORT	signalname	CPM_DIAL1, etc. The options for this field are defined in the NtiData.h file. Setting this field determines the type of event generated. If CPM_DIAL1 is selected, the event MON_DIAL is generated for this signal.

TABLE 44: MT_CPT_TONE

Type	Name	Description
USHORT	signaltype	Tone type (CPM_TONE, SIT_TONE). The options for this field are defined in the NtiData.h file.
CPT_SIGNAL	signal	CPM signal specification: USHORT fEnable; USHORT fDetectSIT; USHORT fDetectLate; USHORT fDetectCycle; USHORT fDetectLost; USHORT nCycleDetect; CPM_FILTER Filter[MAX_CPM_FREQ]* CPM_PULSE Pulse[MAX_CPM_PULSE]*;

*Values for minimum and maximum pulse and gap durations are in msec. The firmware samples each signal at 20 ms intervals. When defining the cadence pattern it is recommended that all times be specified in increments of 20 ms, and that they provide at least 20 ms of margin from the actual limits of the signal to be detected.

Values for frequencies are based on a 256 FFT output. A signal's nominal frequency must first be converted to bin numbers using **MTCalculateFilterRange()**. The values must conform to the following rules:

If more than one frequency range is specified (in bins), the frequencies must be given in order, from low to high.

Refer to the application note: *Understanding Signal Profiles* for more information.

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_CHANNEL_NOT_OPENED	Channel device not opened for access
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_INVALID_PARAM	Invalid argument passed to function

Events

None

MTChGetCPMSignalParams()

This API was formerly called *MTGetCpmToneParams()*

Description

MTGetCpmSignalParams() gets the current values for a CPM signal.

There are maximum of 15 CPM signal profiles and the index ranges from 0 to 14.

The MT_CPM_TONE structure is defined where the API **MTChSetCPMSignalParams()** is explained.

Prototype

MT_RESULT MTChGetCpmSignalParams (

```
const CHANNEL      nChannel,
const USHORT       Index,
const PMT_CPT_TONE pCpt,
PULONG const       pLength
);
```

Parameters

nChannel	channel number
Index	the index ID (0-14) for this CPM signal
pCpt	pointer to a MT_CPT_TONE structure
pLength	Set by the user application, this is a pointer to the size of the given buffer for the MT_CPT_TONE structure.

NOTE: For backwards compatability with older AudioCodes products, the LENGTH parameter for Set APIs allows the SDK to set up to the specified size. The rest of parameters are not changed. MT_RET_OK is returned.

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_CHANNEL_NOT_OPENED	Channel device not opened for access
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_INVALID_PARAM	Invalid argument passed to function

Events

None

MTChResetCpmSignalParams()

This API was formerly called *MTResetCpmToneParams()*.

Description

MTChResetCpmSignalParams() either disables all CPM signal profiles or sets all CPM signal profiles to default settings.

The mode has two possible values:
0 = disable all CPM tone detection (each signal profile must then be enabled individually)
1 = reset to the settings which are currently stored in the registry or the NtiDrv.conf file when running Linux.

To enable a signal, use *MTChCPMSignalControl()* or use *MTChGetCPMSignalParams()*, modify the fEnable flag to true, then use *MTChSetCPMSignalParams()* to set this value.

Each board is shipped with default (North American) signal profiles. These parameters are stored in the registry or the NtiDrv.conf file when running Linux of the computer. If a signal profile is modified using the Control Panel, the new settings are pushed to this registry or the NtiDrv.conf file when running Linux, and factory default settings are lost. Signal profiles set by using *MTChSetCpmSignalParams()* do not modify registry or the NtiDrv.conf file settings.

NOTE: When this API is used, the Call State Machine is not reset.

Prototype

MT_RESULT MTChResetCpmSignalParams (

```
const CHANNEL      nChannel,
const UCHAR        Mode
);
```

Parameters

nChannel	Channel Number
Mode	CPM signal mode 0 = all CPM signals disabled 1 = reset to registry settings

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_CHANNEL_NOT_OPENED	Channel device not opened for access
MT_RET_NOT_AVAILABLE	Driver is not loaded

Events

None

MTCalculateFilterRange()

Description

The SmartWORKS API allows user to set the upper and lower frequencies for three unique sinewaves of a signal. When these values are passed to the firmware (**MTChSetCPMSignalParams()** or **MTChSetUsrSignalParams()**) bin numbers based on a 256 FFT output must be used.

The API **MTCalculateFilterRange()** is used to convert a signal's nominal frequency into bins. The bins associated with each nominal frequency values are returned in the CPT_SIGNAL structure.

This calculator is also available on the SmartView interface under the **Settings** menu.

- this function accepts upto three nominal frequencies
- any values not used must be set to '0' (if Freq1 and Freq3 are defined, Freq2 must not be set to '0')
- nominal frequencies must be entered from lower to highest value

A complete discussion is available in the application note: *Understanding Signal Profiles*.

Prototype

MT_RESULT MTCalculateFilterRange (

```
double          dFreq1,
double          dFreq2,
double          dFreq3,
CPT_SIGNAL      pCPTSignal
);
```

Parameters

dFreq1	nominal frequency (HZ)
dFreq2	nominal frequency (HZ)
dFreq3	nominal frequency (HZ)
pCPTSignal	pointer to the CPT_SIGNAL structure

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_PARAMETER	Invalid parameter passed. Verify that both values are within the 350 Hz to 2208 Hz. The lower value must be less than the upper value
MT_RET_NOT_AVAILABLE	Driver is not loaded

Events

None

MTLoadSignalProfile()

Description

The function **MTLoadSignalProfile()** loads one CPT_TONE from the specified file holding signal profiles into the data structure provided.

The profile name is a null terminated string that identifies one of the files holding individual CPM signal profiles. The profile name must not contain a file extension. The signal name is one of the CPM signal names defined in the NtiData.h header file (i.e. CPM_BUSY1).

Once the data structure is loaded, invoke **MTChSetCPMSignalParams()** or **MTChSetUsrSignalParams()** to configure the channel with this signal profile. At the time variables, such as Index ID, should be modified per the needs of the user application.

Refer to the application note: *Understanding Signal Profiles*, for more information about creating/modifying profiles.

Prototype

```
MT_RESULT MTLoadSignalProfile( (
    PCHAR          pProfileName,
    PCHAR          pSignalName,
    PMT_CPT_TONE   pCPTSignal
);
```

Parameters

pProfileName	profile name
pSignalName	the signal name (i.e. CPM_BUST1)
pCPTSignal	a pointer to the MT_CPT_TONE structure

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_FILENAME	If either the profile or signal name cannot be located

Events

None

MTChSetVoiceDetectParams()

Description

The **MTChSetVoiceDetectParams()** sets the Voice detection structure (MT_VOICEDetectParams). This structure controls the detection of human voice or answering machine when the application is running the Call Progress Monitoring (CPM) feature with Voice Detection enabled (**MTChVoiceDetectControl()**).

*If it is necessary to modify a single value, the application should first obtain all the Voice Detect parameters by calling **MTChGetVoiceDetectParams()**, modify the desired parameters and write back all the parameters to the channel.*

Prototype

MT_RESULT MTChSetVoiceDetectParams (

```
const CHANNEL          nChannel,
PMT_VOICEDetectParams const pVoiceDetectParams,
const PULONG           pLength
);
```

Parameters

nChannel	Channel Number
pVoiceDetectParams	pointer to the MT_VOICEDetectParams structure
pLength	Set by the user application, this is a pointer to the size of the given buffer for the MT_VOICEDetectParams structure

NOTE: For backwards compatability with older AudioCodes products, the LENGTH parameter for Set APIs allows the SDK to set up to the specified size. The rest of parameters are not changed. MT_RET_OK is returned.

Structure

The MT_VOICEDetectParams structure is defined as follows:

TABLE 45: MT_VOICEDetectParams

Type	Name	Description	Default	Units
ULONG	threshold_power	the minimum voice-band power required for voice detection	-36	dBm
USHORT	relative_threshold	the minimum proportion of band pass filter power required for detection in any given filter	65	-----
USHORT	threshold_filter_count	the number of band-pass filters where the relative_threshold must be met for voice detection	5	int (1-10?)

TABLE 45: MT_VOICEDetectParams

Type	Name	Description	Default	Units
USHORT	no_voice_min	length of time there is no voice activity to determine the voice has stopped speaking	1000	ms
USHORT	reset	the minimum length silence (no voice detection) required to reset CPM voice detect. If a voice is detected, new events are generated.	15000	ms
USHORT	voice_min	the minimum time voice activity is required before the application generates a voice detected event	150	ms
USHORT	machine_min	the minimum voice activity required to determine that an answer machine has been detected	3000	ms

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_CHANNEL_NOT_OPENED	Channel device not opened for access
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_INVALID_PARAMS	Invalid argument passed to function
MT_RET_DATA_TRUNCATED	Data copied but truncated; the buffer size set by the user application is not large enough for this API request. <i>FlagDataTruncated</i> field of the MTSYS_CONFIG or the MTSYS_CONFIGURATION data structure must be enabled (1) to receive this return code, otherwise MT_RET_OK is used, and the truncated data is passed to the user application

Events

None

MTChGetVoiceDetectParams()

Description

The **MTChGetVoiceDetectParams()** gets the current settings of the Voice detection structure (MT_VOICEDetectPARAMS). This structure controls the detection of human voice or answering machine when the application is running the Call Progress Monitoring (CPM) feature with Voice Detection enabled (**MTChVoiceDetectControl()**).

*If it is necessary to modify a single value, the application should first obtain all the Voice Detect parameters by calling **MTChGetVoiceDetectParams()**, modify the desired parameters and write back all the parameters to the channel.*

Prototype

MT_RESULT MTChGetVoiceDetectParams (

```
const CHANNEL          nChannel,
PMT_VOICEDetectPARAMS const pVoiceDetectParams,
const PULONG           pLength
);
```

Parameters

nChannel	Channel Number
pVoiceDetectParams	pointer to the MT_VOICEDetectPARAMS structure
pLength	Set by the user application, this is a pointer to the size of the given buffer for the MT_VOICEDetectPARAMS structure

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_CHANNEL_NOT_OPENED	Channel device not opened for access
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_INVALID_PARAMS	Invalid argument passed to function
MT_RET_DATA_TRUNCATED	Data copied but truncated; the buffer size set by the user application is not large enough for this API request. <i>FlagDataTruncated</i> field of the MTSYS_CONFIG or the MTSYS_CONFIGURATION data structure must be enabled (1) to receive this return code, otherwise MT_RET_OK is used, and the truncated data is passed to the user application

Events

None

MTChVoiceDetectControl()

Description

This API enables/disables voice detection on a specified channel. By default, when a channel is first opened, Call Progress Monitoring(CPM) and voice detection is enabled.

Prototype

MT_RESULT MTChVoiceDetectControl (

```
const CHANNEL          nChannel,  
Boolean                fOn  
);
```

Parameters

nChannel	Channel Number
fOn	value that sets the state of voice detection: TRUE=enable FALSE=disable

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_CHANNEL_NOT_OPENED	Channel device not opened for access
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_INVALID_PARAMS	Invalid argument passed to function

Events

None

MTChVoiceDetectStatus()

Description

This API retrieves the current status of voice detection on a specified channel. By default, when a channel is first opened, Call Progress Monitoring(CPM) and voice detection is enabled.

Prototype

MT_RESULT MTChVoiceDetectControl (

const CHANNEL

Boolean

);

nChannel,

pOn

Parameters

nChannel	Channel Number
pOn	pointer to the value holding the status of voice detection: TRUE=enable FALSE=disable

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_CHANNEL_NOT_OPENED	Channel device not opened for access
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_INVALID_PARAMS	Invalid argument passed to function

Events

None

TONE FUNCTIONS - ALERT TONES

MTGetAlertToneParams()

This API was formerly called ***MTGetAlertTone()***

Description

Alert tone is played to the decoder and encoder so that alert tone occurs during the recording process, and is part of the recorded data stream.

The alert tone can be enabled through the ALERT_TONE flag inside the MT_IO_CONTROL.StartControl field.

Prototype

MT_RESULT MTGetAlertToneParams(

```
const CHANNEL          nChannel,  
const PMT_ALERTTONE    pAlert,  
PULONG const           pLength  
);
```

Parameters

nChannel	channel number
pAlert	pointer to an MT_ALERTTONE structure
pLength	Set by the user application, this is a pointer to the size of the given buffer for the MT_ALERTTONE structure

Structure

Structure MT_ALERTTONE is defined as the following:

TABLE 46: MT_ALERTTONE

Type	Name	Purpose
float	amplitude	Amplitude in dBm, 3.0 dBm to -60.0 dBm
USHORT	frequency	Frequency in Hz.
USHORT	cycle_duration	Alert tone interval, Repeat duration in unit of 20ms; eg. 750->15sec.
USHORT	tone_duration	Duration in 20 ms unit for this tone.
USHORT	first_cycle_duration	Duration in 20 ms unit for the first cycle only.

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_CHANNEL_NOT_OPENED	Channel device not opened for access
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_INVALID_PARAM	Invalid argument passed to function

MT_RET_DATA_TRUNCATED

Data copied but truncated; the buffer size set by the user application is not large enough for this API request. *FlagDataTruncated* field of the MTSYS_CONFIG or the MTSYS_CONFIGURATION data structure must be enabled (1) to receive this return code, otherwise MT_RET_OK is used, and the truncated data is passed to the user application

EventsNone

MTSetAlertToneParams()

This API was formerly called ***MTSetAlertTone()***

Description

Alert tone is played to the decoder and encoder so that alert tone occurs during the recording process, and is part of the recorded data stream.

The alert tone can be enabled through the ALERT_TONE flag inside the MT_IO_CONTROL.StartControl field.

Cycle - Interval at which tone repeats

Duration - Time length of tone and/or silence

Please note, each board implements this feature differently. The SmartWORKS DT along with the SmartWORKS LD puts the tone out onto the line. The SmartWORKS NGX, and DP won't play the tone out onto the line, however, it will be recorded with the conversation.

Prototype

MT_RESULT MTSetAlertToneParams (

```
const CHANNEL      nChannel,
const PMT_ALERTTONE pAlert,
PULONG const      pLength
);
```

NOTE: For backwards compatability with older AudioCodes products, the LENGTH parameter for Set APIs allows the SDK to set up to the specified size. The rest of parameters are not changed. MT_RET_OK is returned.

Parameters

nChannel	channel number
pAlert	pointer to an MT_ALERTTONE structure
pLength	Set by the user application, this is a pointer to the size of the given buffer for the MT_ALERTTONE structure

Structure

Structure MT_ALERTTONE is defined as the following:

TABLE 47: MT_ALERTTONE

Type	Name	Purpose
float	amplitude	Amplitude in dBm, 3.0 dBm to -60.0 dBm
USHORT	frequency	Frequency in Hz
USHORT	cycle_duration	Alert tone interval, Repeat duration in unit of 20ms; eg. 750->15sec
USHORT	tone_duration	Duration in 20 ms unit for this tone

TABLE 47: MT_ALERTTONE

Type	Name	Purpose
USHORT	first_cycle_duration	Duration in 20 ms unit for the first cycle only

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_CHANNEL_NOT_OPENED	Channel device not opened for access
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_INVALID_PARAM	Invalid argument passed to function

Events

None

MTStartAlertTone()

Description

Alert tone is played to the decoder and encoder so that alert tone occurs during the recording process, and is part of the recorded data stream.

The alert tone can be enabled through the ALERT_TONE flag inside the MT_IO_CONTROL.StartControl field.

Scenario

The streaming buffer is allocated and maintained inside the DLL. The user application does not need to provide a buffer through MT_IO_CONTROL.

Prototype

MT_RESULT MTStartAlertTone(

const CHANNEL nChannel
);

Parameters

nChannel channel number

Structure

Structure MT_ALERTTONE is defined as the following:

TABLE 48: MT_ALERTTONE

Type	Name	Purpose
float	amplitude	Amplitude in dBm, 3.0 dBm to -60.0 dBm
USHORT	frequency	Frequency in Hz
USHORT	cycle_duration	Alert tone interval, Repeat duration in unit of 20ms; eg. 750->15sec
USHORT	tone_duration	Duration in 20 ms unit for this tone
USHORT	first_cycle_duration	Duration in 20 ms unit for the first cycle only

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_CHANNEL_NOT_OPENED	Channel device not opened for access
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_SERVICE_ALREADY_STARTED	Requested service already started

Events

None

See Also: [MTStopStreaming\(\)](#), [MTStreamBufIn\(\)](#), [MTStreamBufOut\(\)](#)

MTStopAlertTone()

Description

Alert tone is played to the decoder and encoder so that alert tone occurs during the recording process, and is part of the recorded data stream.

The alert tone can be enabled through the ALERT_TONE flag inside the MT_IO_CONTROL.StartControl field.

Prototype

MT_RESULT MTStopAlertTone(

```
const CHANNEL          nChannel
);
```

Parameters

nChannel channel number

Structure

Structure MT_ALERTTONE is defined as the following:

TABLE 49: MT_ALERTTONE

Type	Name	Purpose
float	amplitude	Amplitude in dBm, 3.0 dBm to -60.0 dBm
USHORT	frequency	Frequency in Hz
USHORT	cycle_duration	Alert tone interval, Repeat duration in unit of 20ms; eg. 750->15sec
USHORT	tone_duration	Duration in 20 ms unit for this tone
USHORT	first_cycle_duration	Duration in 20 ms unit for the first cycle only

The alert tone occurs at the end of a cycle.

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_CHANNEL_NOT_OPENED	Channel device not opened for access
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_INVALID_PARAM	Invalid argument passed to function

Events

None

TONE FUNCTIONS - PLAY TONES

MTPlayTone(), MTPlayToneAsync(), MTPlayToneEx()

This API was formerly called *MTPlayUsrTone()*

Description

Each of these APIs starts a DLL background function to play the specified tone(s). A return of MT_RET_IO_PENDING indicates the successful queuing of the background task inside the channel function queue. At the completion of the function, an event will be added to the channel event queue.

The OVERLAPPED structure provides to the DLL an application created event. The user application waits for an event to signal the completion of the background function.

MTCALLBACK and LPARAM provide the DLL the address of the API completion routine and the user expected parameter for invoking the completion routine.

The tone structure is defined using the data structure MT_PLAYTONE, whose address is recorded into the Buffer field of the structure MT_IO_CONTROL. MT_IO_CONTROL.Length field contains the byte size of the MT_PLAYTONE array. To play one tone, set MT_IO_CONTROL.Length field to the size of MT_PLAYTONE; to play twenty tones, set MT_IO_CONTROL.Length to twenty times of the size of MT_PLAYTONE structure.

Flag REPEAT_TONE can be set in MT_IO_CONTROL.StartControl field to specify a repeat playing of the tones. The supported maximum repeat tones is 10. Tone repetition can be terminated either through a termination condition such as DTMF detected, or application termination through MTStopChannel().

MAX_REPEAT_TONE_COUNT - The maximum number of tones to play in MTPlayTone() repeat mode set through MT_IO_CONTROL.StartControl field with bit REPEAT_TONE set.

To generate a dial tone until a DTMF is detected, set the tone duration to 3 seconds, set the silence duration to 0, set the REPEAT_TONE flag, and set the MT_IO_CONTROL.MaxDigits to 1.

Scenario

Use this API function to play dial tone, SIT tone, ring-back tone, etc.

Prototype

MT_RESULT MTPlayTone (

```
const CHANNEL          nChannel,
const PMT_IO_CONTROL   ploCtl
);
```

MT_RESULT MTPlayToneEx(

```
const CHANNEL          nChannel,
```

```
PMT_IO_CONTROL const    ploCtl,  
const LOVERLAPPED      pOverlapped  
);
```

MT_RESULT MTPlayToneAsync(

```
const CHANNEL          nChannel,  
PMT_IO_CONTROL const   ploCtl,  
const MTCOMPLETE       pCompletionRoutine,  
const LPARAM           lParam  
);
```

Parameters

nChannel	channel number
ploCtl	pointer to a The MT_IO_CONTROL Data Structure structure (defined in Appendix A)
pOverlapped	pointer to OVERLAPPED structure
pCompletionRoutine	pointer to completion routine
lParam	parameter to echo back to completion routine

Structure

Structure MT_PLAYTONE is defined as the following.

TABLE 50: MT_PLAYTONE

Type	Name	Purpose
float	ampl1	Amplitude 1 in dBm, 3.0 dBm to -60.0 dBm
USHORT	freq1	Frequency 1 in Hz (300 Hz to 3400 Hz, 0 is also valid)
float	ampl2	Amplitude 2 in dBm (+3.0 dbm to -60.0 dbm)
USHORT	freq2	Frequency 2 in Hz ((300 Hz to 3400 Hz, 0 is also valid)
USHORT	durtone	Duration in 125 us of generated tone
USHORT	dursilence	Duration in 125 us of post tone silence

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_CHANNEL_NOT_OPENED	Channel device not opened for access
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_IO_PENDING	Background I/O operation is in progress or queued

MT_RET_NO_MEM

An operation could not be completed due to insufficient memory or an error occurred while attempting to allocate memory

Events

EVT_TONEP_DONE
EVT_MAXDIGITS
EVT_TERMDIGIT
EVT_TERMSILENCE

TERMINATION/MONITOR FUNCTIONS

MTGetLastTerm()

Description

Gets the previous termination event code.

Prototypes

```
MT_RESULT  MTGetLastTerm (  
    const CHANNEL    nChannel,  
    PULONG const     pEventCode  
);
```

Parameters

nChannel	channel number
pEventCode	pointer to a buffer for event code

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_CHANNEL_NOT_OPENED	Channel device not opened for access
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_INVALID_PARAM	Invalid argument passed to function

Events

None

MTSetTerm()

Description

Termination enable bits control the termination conditions on a per channel basis. When a termination condition is met, all background functions running on the channel are stopped and events are generated.

One EVT_CPM_STOP event is generated for each task that was stopped. The Subreason and XtraInfo fields indicate the line condition and the FuncCode field indicates the type of media task that was stopped.

If **MTSetChannelToDefault()** is invoked, these settings are put back to default. The enabled default TERM_ bits are represented in the table below and mapped to settings that have been maintained for backwards compatibility.

TERM_ Bits	v3.2 or earlier
TERM_CONNECT (answer machine, human)	TERM_CONNECT (answer machine, human, fax)
TERM_MON_BUSY1	TERM_BUSY (normal)
TERM_MON_BUSY2	TERM_BUSY (trunk busy)
TERM_MON_SIT1, TERM_MON_SIT2, TERM_MON_SIT3, TERM_MON_SIT4, TERM_MON_SIT5	TERM_INTERCEPT
TERM_NO_ANSWER	TERM_NO_ANSWER
TERM_LOOP_DROP (loop voltage/ current on hook)	TERM_LOOP_DROP (loop voltage/ current on hook)
TERM_LVOLTAGE_NOTOFFHOOK (SmartWORKS LD only)	TERM_LVOLTAGE_NOTOFFHOOK (SmartWORKS LD only)
TERM_MON_FAX1, TERM_MON_FAX2	TERM_CONNECT (answer machine, human, fax)
TERM_MAX_SILENCE	TERM_MAX_SILENCE
TERM_MAX_ACTIVITY	TERM_MAX_ACTIVITY

NOTE: Termination conditions can also be set for a specific task using the *TermEnable* field of the MT_IO_CONTROL structure.

The following bits can be put together with “OR” to enable termination. The following table lists each termination conditions, plus the corresponding Subreason and XtrInfo field generated when the EVT_CPM_STOP event is reported:

MTSetTerm() Control	Subreason	XtrInfo
TERM_LOOP_DROP	CPM_MON_LOOP_DROP	
TERM_MON_SIL	CPM_MON_SILENCE	
TERM_MON_ACT	CPM_MON_ACTIVITY	
TERM_MON_DIAL*+	CPM_MON_DIAL	The Index of the signal
TERM_MON_BUSY*+	CPM_MON_BUSY	The Index of the signal (normal busy or trunk busy)
TERM_CONNECT+	(Using MTCallString()) CPM_CONNECT ----- (Using other background functions) CPM_MON_HUMAN CPM_MON_MACHINE	CPM_MON_HUMAN, CPM_MON_MACHINE -----
TERM_NO_ANSWER+ (use only with MTCallString())	CPM_NO_ANSWER	
TERM_INTERCEPT*+	CPM_MON_INTERCEPT	The index of the signal
TERM_NOR_SILENCE	CPM_MON_MAX_SILENCE CPM_MAX_SILENCE	
TERM_NOR_ACTIVITY	CPM_MON_MAX_ACTIVITY CPM_MAX_ACTIVITY	
TERM_LVOLTAGE_ ABOVEORBELOW	CPM_MON_LVOLTAGE_DROP	
TERM_LVOLTAGE_ NOTOFFHOOK	(same as TERM_LVOLTAGE_ ABOVE_OR_BELOW	
TERM_MON_UTONE+	CPM_MON_UTONE	The Index of the signal
TERM_MON_CALLWAITING+	CPM_MON_CALLWAITING	
TERM_MON_RECEIVEOFF+	CPM_MON_RECEIVOFF	
TERM_MON_DIAL1+	CPM_MON_DIAL1	
TERM_MON_DIAL2+	CPM_MON_DIAL2	
TERM_MON_DIAL3+	CPM_MON_DIAL3	
TERM_MON_BUSY1+	CPM_MON_BUSY1	
TERM_MON_BUSY2+	CPM_MON_BUSY2	
TERM_MON_FAX1+	CPM_MON_FAX1	
TERM_MON_FAX2+	CPM_MON_FAX2	
TERM_MON_SIT1+	CPM_MON_SIT1	
TERM_MON_SIT2+	CPM_MON_SIT2	
TERM_MON_SIT3+	CPM_MON_SIT3	
TERM_MON_SIT4+	CPM_MON_SIT4	
TERM_MON_SIT5+	CPM_MON_SIT5	

*This option has been maintained for backwards compatibility. New development should not use these options.

+ Call Progress Monitoring must be enabled to detect these line conditions:
MTChCPMControl() (or **MTSetCPMMode** if using v. 3.2 or earlier).

Prototype

MT_RESULT MTSetTerm (

const CHANNELnChannel,

const ULONGTermEnable

);

Parameters

nChannelchannel number

TermEnablemonitor event monitoring enable bit

Return Codes

MT_RET_OKNo error

MT_RET_INVALID_CHANNELChannel number is out of range

MT_RET_CHANNEL_NOT_OPENEDChannel device not opened for access

MT_RET_NOT_AVAILABLEDriver is not loaded

MT_RET_INVALID_PARAMInvalid argument passed to function

Events

None

MTGetTerm()

Description

MTGetTerm() retrieves the current termination enable bits. A termination enable bit controls termination conditions linked to the termination of background functions. NOTE: Termination enable bits can also be set through the TermEnable field in the IO_CONTROL structure for a specific task..

The following bits can be put together with "OR" to enable monitor events:

Prototype

MT_RESULT MTGetTerm (

```
const CHANNEL      nChannel,  
const PULONG       pTermEnable  
);
```

Parameters

nChannel	channel number
pTermEnable	pointer to termination event enable bits

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_CHANNEL_NOT_OPENED	Channel device not opened for access
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_INVALID_PARAM	Invalid argument passed to function

Events

None

MTSetMoni()

Description

Monitor event enable bits control which monitoring events are reported to the user application. **MTSetMoni()** is similar to **MTSetEventFilters()** but defines CPM specific monitor events. **MTGetMoni()** retrieves the current CPM monitoring conditions.

If **MTSetChannelToDefault()** is invoked, these settings are returned to the default. By default, MONI_BUSY1, MONI_RINGBACK1, MONI_DIAL1, MONI_DIAL2, and MONI_DIAL3 (MONI_BUSY, MONI_RINGBACK, and MONI_DIAL for v 3.2 or earlier) are enabled.

This bits control the CPM monitoring on a per channel basis. Call Progress Monitoring must be enabled when these bits are enabled.

NOTE: Monitoring options can be set for a specific task using the *MoniEnable* field of the MT_IO_CONTROL structure.

NOTE: MONI_REVERSAL is not a CPM signal. Call Progress Monitoring does not have to be enabled to detect this line condition.

The following bits can be put together with "OR" to enable monitor events:

Signal Name	MTSetMoni() Control	Corresponding Event
BUSY1	MONI_BUSY1	EVT_MON_BUSY1
BUSY1	MONI_BUSY*	EVT_MON_BUSY1 (EVT_MON_BUSY)
BUSY2	MONI_BUSY2	EVT_MON_BUSY2
BUSY2	MONI_TBUSY*	EVT_MON_BUSY2 EVT_MON_TBUSY)
DIAL1-3	MONI_DIAL*	EVT_MON_DIAL1-3
CALLWAITING	MONI_CALLWAITING	EVT_MON_CALLWAITING
RECEIVEOFF	MONI_RECEIVEOFF	EVT_MON_RECEIVEOFF
RINGBACK1	MONI_RINGBACK1	EVT_MON_RINGBACK1
RINGBACK1	MONI_RINGBACK*	EVT_MON_RINGBACK1 (EVT_MON_RINGBACK)
RINGBACK2	MONI_RINGBACK2	EVT_MON_RINGBACK2
RINGBACK2	MONI_DRINGBACK*	EVT_MON_RINGBACK2 EVT_MON_DRINGBACK)
SIT 1-5	MONI_SIT*	EVT_MON_SIT1-5
FAX1 & FAX2	MONI_FAX*	EVT_MON_FAX
FAX1	MONI_FAX1	EVT_MON_FAX1
FAX2	MONI_FAX2	EVT_MON_FAX2
	MONI_UTONE	EVT_UTONE_ON+
	MONI_SIGNAL_CYCLE	EVT_MON_SIGNAL_CYCLE
SIT1	MONI_SIT1	EVT_MON_SIT1
SIT2	MONI_SIT2	EVT_MON_SIT2

SIT3	MONI_SIT3	EVT_MON_SIT3
SIT4	MONI_SIT4	EVT_MON_SIT4
SIT5	MONI_SIT5	EVT_MON_SIT5
DIAL1	MONI_DIAL1	EVT_MON_DIAL1
DIAL2	MONI_DIAL2	EVT_MON_DIAL2
DIAL3	MONI_DIAL3	EVT_MON_DIAL3
	MONI_VOICE	EVT_MON_VOICE EVT_MON_HUMAN EVT_MON_MACHINE EVT_MON_NOVOICE
	MONI_REVERSAL	EVT_MON_REVERSAL

*These options have been maintained for backwards compatibility. New development should not use these options.

Prototype

MT_RESULT MTSetMoni (

```
const CHANNEL      nChannel,
const ULONG        MoniEnable
);
```

Parameters

nChannel	channel number
MoniEnable	monitor event monitoring enable bit

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_CHANNEL_NOT_OPENED	Channel device not opened for access
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_INVALID_PARAM	Invalid argument passed to function

Events

None

MTGetMoni()

Description

Monitor event enable bits control CPM monitoring conditions. By default, this is set to all zeros. With **MTSetMoni()** the application can enable several additional CPM monitor events. **MTSetMoni()** is similar to **MTSetEventFilters()** but defines CPM monitor events. **MTGetMoni()** retrieves the current CPM monitoring conditions.

NOTE: Bits can also be set in conjunction with a specific background function using the *MoniEnable* field of the *IO_CONTROL* structure.

Prototype

```
MT_RESULT MTGetMoni (  
    const CHANNEL          nChannel,  
    const PULONG           pMoniEnable  
);
```

Parameters

nChannel	Channel number
pMoniEnable	pointer to value of event monitoring enable bit

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_CHANNEL_NOT_OPENED	Channel device not opened for access
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_INVALID_PARAM	Invalid argument passed to function

Events

None

CT BUS FUNCTIONS

CT BUS / MUX CONTROL

The MUX system type is either MUX_MVIP, or MUX_H100, or MUX_NONE. This MUX type is added as a registry (or the NtiDrv.conf file when running Linux) configurable parameter for the SmartWORKS DLL under the name *CTBusType*. This field can be set via the **MTBoardSetConfig()** function or via the Control Panel.

The SmartWORKS hardware can be either a resource card or a network card when equipped with a T1/E1 card. The MUX interface on SmartWORKS is designed to switch board Network Interfaces (NI), DSPs (DR) and Channels.

MTArmMasterAlarm()

Description

MTArmMasterAlarm() arms the report of master clock synchronization related errors through EVT_MASTER_PLL_ALARM.

Prototype

```
MT_RESULT MTArmMasterAlarm (  
    const BOARD          nBoard  
);
```

Parameters

nBoard	board number

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_BOARD	Board number is out of range
MT_RET_BOARD_NOT_OPENED	Board is not opened for access
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_API_THREAD_PROTECTED	Another thread is currently accessing this API

Events

EVT_MASTER_PLL_ALARM

MTDisarmMasterAlarm()

Description

MTDisarmMasterAlarm() clears the arming for reporting of master clock synchronization related errors.

Prototype

MT_RESULT MTDisarmMasterAlarm (

```
    const BOARD          nBoard  
    );
```

Parameters

nBoard	board number

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_BOARD	Board number is out of range
MT_RET_BOARD_NOT_OPENED	Board is not opened for access
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_API_THREAD_PROTECTED	Another thread is currently accessing this API

Events

None

MTGetMasterAlarmStatus()

Description

MTGetMasterAlarmStatus() retrieves the arming status for reporting of the master clock synchronization error. The BOOLEAN parameter will contain TRUE if alarm armed; FALSE if disarmed.

Prototype

```
MT_RESULT MTGetMasterAlarmStatus (  
    const BOARD          nBoard,  
    BOOLEAN * const      pEnabled  
);
```

Parameters

nBoard	board number
pEnabled	pointer to information accessed

Return Codes

MT_RET_OK	No error
MT_RET_BOARD_NOT_OPENED	Board is not opened for access
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_INVALID_BOARD	Board number is out of range
MT_RET_API_THREAD_PROTECTED	Another thread is currently accessing this API

Events

None

MTArmCTAlarm()

Description

MTArmCTAlarm() arms the alarm for monitoring MVIP/H.100 bus related errors. When armed, detected errors will be reported once and the alarm will be disarmed. MVIP bus related errors are reported as event EVT_MVIP_ALARM only when its alarm is armed. H.100 bus related errors are reported as events EVT_H100_MASTER_A_ALARM and EVT_H100_MASTER_B_ALARM only when their respective alarms are armed.

Prototype

MT_RESULT MTArmCTAlarm(

```
const BOARD      nBoard,  
const int        BusType,  
const int        nMaster  
);
```

Parameters

nBoard	board index number
BusType	MUX_MVIP or MUX_H100
nMaster	MODE_MASTER_A or MODE_MASTER_B for MUX_H100; MODE_MASTER for MUX_MVIP

Return Code

MT_RET_OK	No error
MT_RET_BOARD_NOT_OPENED	Board is not opened for access
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_INVALID_BOARD	Board number is out of range
MT_RET_INVALID_BUSTYPE	Failure to specify MUX_MVIP or MUX_H100
MT_RET_API_THREAD_PROTECTED	Another thread is currently accessing this API

Events

None

MTDisarmCTAlarm()

Description

MTDisarmCTAlarm() disables reporting of MVIP/H.100 bus related errors.

Prototype

```
MT_RESULT MTDisarmCTAlarm(  
    const BOARD      nBoard,  
    const int         BusType,  
    const int         nMaster  
);
```

Parameters

nBoard	board index number
BusType	MUX_MVIP or MUX_H100
nMaster	MODE_MASTER_A or MODE_MASTER_B for MUX_H100; MODE_MASTER for MUX_MVIP

Return Codes

MT_RET_OK	No error
MT_RET_BOARD_NOT_OPENED	Board is not opened for access
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_INVALID_BOARD	Board number is out of range
MT_RET_INVALID_BUSTYPE	Failure to specify MUX_MVIP or MUX_H100
MT_RET_API_THREAD_PROTECTED	Another thread is currently accessing this API

Events

None

Note - MTDisarmCTAlarm() is provided as a macro that redirects to this API

MTGetCTAlarmStatus()

This API was formerly called **MTGetMVIPAlarmStatus()**

Description

MTGetCTAlarmStatus() retrieves the arming status of the MVIP/H.100 alarm. The BOOLEAN parameter will return TRUE if the alarm is armed, FALSE if disarmed.

Prototype

```
MT_RESULT MTGetCTAlarmStatus (
    const BOARD          nBoard,
    const int             BusType,
    const int             nMaster,
    BOOLEAN * const       pArmed
);
```

Parameters

nBoard	board index number
BusType	MUX_MVIP or MUX_H100
nMaster	MODE_MASTER, MODE_MASTER_A or MODE_MASTER_B
pArmed	pointer to retrieve the alarm's arming status

Return Codes

MT_RET_OK	No error
MT_RET_BOARD_NOT_OPENED	Board is not opened for access
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_INVALID_BOARD	Board number is out of range
MT_RET_INVALID_BUSTYPE	Failure to specify MUX_MVIP or MUX_H100
MT_RET_API_THREAD_PROTECTED	Another thread is currently accessing this API

Events

None

Note - MTGetCTAlarmStatus() is provided as a macro that redirects to this API

MTSetCTRoute()

This API was formerly called **MTSetRoute()**

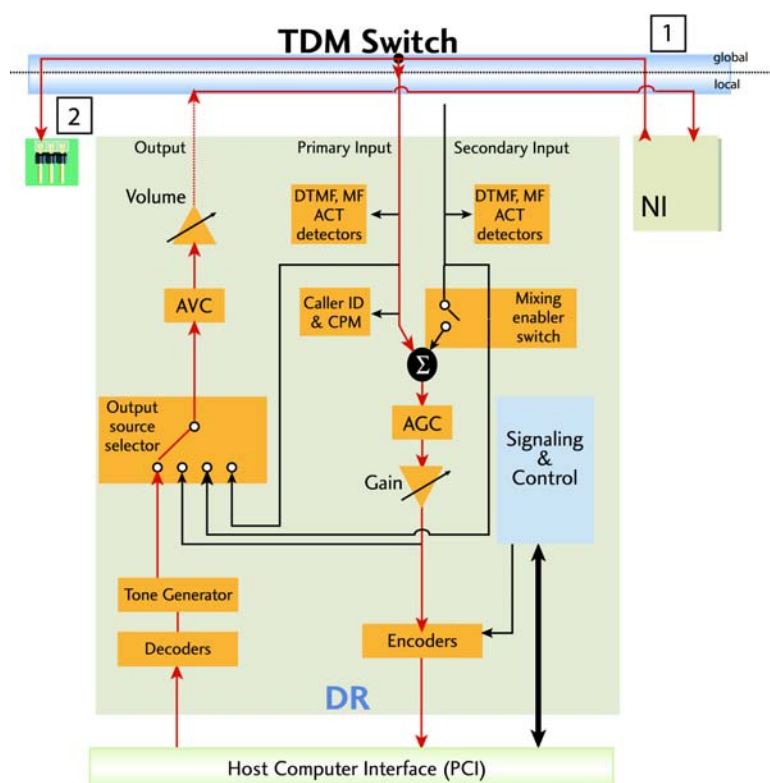
Description

MTSetCTRoute() sets a link between a channel and the CT bus on a specified CT stream and CT bus timeslot. Link directions can be DIR_TALK, DIR_LISTEN or DIR_SUPER. DIR_TALK and DIR_LISTEN are used with the primary input and output. DIR_SUPER is used for listening on the secondary input.

Scenario

This API is most commonly used when data passes from the CT bus to a channel (NI).

Using MTSetCTRoute for live monitoring while recording



STEP 1: MTSetCTRoute - Make a link between channel and bus for Live Monitoring

nChannel = GCI index
Direction = DIR_TALK
Stream = CT Bus Stream Index
Slot = CT Bus Timeslot Index

STEP 2: MTAJListen - Set Audio Jack to Receive from specified CT Bus timeslot number for desired input.

iAudioJack = audio jack index
iStream = Index of TDM stream
iSlot = Index of TDM slot on a stream

If connection to Audio Jack needs to be broken, use **MTResetAJListen**

NOTES:

In the above scenario the channel still has an output connection to the Network Interface (NI). As a result, tone generation, is still available when using a board that supports this feature. When opening a connection to the TDM bus, it is important to understand that the DSP is bypassed. The audio passed along this route cannot be altered with volume and gain control. This is the recommended output method to use with the SmartWORKS AT, DT and the SmartWORKS PT boards.

Prototype**MT_RESUTL MTSetCTRoute (Parameters**

```
const CHANNEL      nChannel,  
const UCHAR        Direction,  
const int           Stream,  
const int           Slot  
);
```

nChannel	channel number
Direction	DIR_TALK, DIR_LISTEN or DIR_SUPER(listen secondary input only)
Stream	CT bus stream index: 0-15 for MVIP, 0-31 for H100
Slot	CT bus slot index within the specified stream 0-31 for MVIP, 0-31 for H100 with a stream speed of 2048 KHz 0-63 for H100 with stream speed of 4096 KHz 0-127 for H100 with stream speed of 8192 KHz

Return Codes

MT_RET_OK	No error
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_CHANNEL_NOT_OPENED	Channel device not opened for access

Events

None

MTGetCTRoute()

Description

Retrieves a specified channel's MVIP/H.100 link and it's specified link direction (TALK, LISTEN, SUPER). If the channel is not connected to the MVIP/H.100 bus, pStream will be returned as -1.

Prototype

MT_RESULT MTGetCTRoute (

const CHANNEL nChannel,
const UCHAR Direction,
int * const pStream,
int * const pSlot
);

Parameters

nChannel	channel number
Direction	DIR_TALK, DIR_LISTEN or DIR_SUPER
pStream	buffer for the returned stream index -1 = channel is not connected to the bus
pSlot	buffer for the returned slot index

Return Codes

MT_RET_OK	No error
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_CHANNEL_NOT_OPENED	Channel device not opened for access
MT_RET_INVALID_PARAM	Invalid argument passed to function

Events

None

MTResetCTRoute()

This API was formerly called **MTResetRoute()**

Description

MTResetCTRoute() resets the MVIP/H.100 link. A link's direction can be DIR_TALK, DIR_LISTEN, or DIR_SUPER. **MTResetCTRoute()** requires users to pass exact parameters (stream and timeslot) both of which are set by **MTSetCTRoute()**. If users don't want to store the stream and timeslot information, that information can be retrieved by **MTGetCTRoute()** first, after which **MTResetCTRoute()** can be called.

Prototype

MT_RESULT MTResetCTRoute (

```
const CHANNEL      nChannel,
const UCHAR        Direction,
const int           Stream,
const int           Slot
);
```

Parameters

nChannel	channel number
Direction	DIR_TALK, DIR_LISTEN or DIR_SUPER
Stream	CT bus stream index: 0-15 for MVIP, 0-31 for H100
Slot	CT bus slot index within the specified stream 0-31 for MVIP, 0-31 for H100 with a stream speed of 2048 KHz 0-63 for H100 with stream speed of 4096 KHz 0-127 for H100 with stream speed of 8192 KHz

Return Codes

MT_RET_OK	No error
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_CHANNEL_NOT_OPENED	Channel device not opened for access
MT_RET_INVALID_PARAM	Invalid argument passed to function

Events

None

MTSetInputs()

Description

MTSetInputs() sets the CT bus connection of the primary and secondary input of the specified channel.

When the CT bus type is MVIP, the first eight (8) MVIP streams (0 - 7) are on the forward direction which uses 256 time slots. The next eight (8) streams (8 - 15) are on the reverse direction which also use 256 time slots. The MVIP time slot index is from 0 to 255.

If the CT bus type is H.100, the number of timeslots available per streams (0 - 31) depends on which H.100 stream speed is selected in the SmartControl panel.
At 2048 KHz, 32 timeslots per stream are available.
At 4096 KHz, 64 timeslots per stream are available.
At 8192 KHz, 128 timeslots per stream are available.

Scenario

Use Supervise APIs to monitor remote conversations on the same CT bus.

Prototype

MT_RESULT MTSetInputs(

```
const CHANNEL      nChannel,
const int           iStream1,
const int           iSlot,
const int           iStream2,
const int           iSlot2
);
```

Parameters

nChannel	channel number
iStream1	channel primary input connection stream, Index of TDM stream: 0 - 15 for MVIP
iSlot1	channel primary input connection timeslot: 0 to 31 for MVIP number varies for H.100 (see above) -1 for not connected
iStream2	channel secondary input connection stream, Index of TDM stream: 0 - 15 for MVIP
iSlot2	channel secondary input connection timeslot: 0 to 31 for MVIP number varies for H.100 (see above) -1 for not connected

Return Codes

MT_RET_OK	No error
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_CHANNEL_NOT_OPENED	Channel device not opened for access
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_BUSY	Channel is busy

Events

None

MTGetInputs()

Description

MTGetInputs() retrieves the CT bus connection of the primary and secondary input of the specified channel.

When the CT bus type is MVIP, the first eight (8) MVIP streams (0 - 7) are on the forward direction which uses 256 time slots. The next eight (8) streams (8 - 15) are on the reverse direction which also use 256 time slots. The MVIP time slot index is from 0 to 255.

If the CT bus type is H.100, the number of timeslots available per streams 0 - 15 depends on which H.100 stream speed is selected in the SmartControl panel.

At 2048 KHz, 32 timeslots per stream are available.

At 4096 KHz, 64 timeslots per stream are available.

At 8192 KHz, 128 timeslots per stream are available.

Scenario

Use Supervise APIs to monitor remote conversations on the same CT bus.

Prototype

```
MT_RESULT MTGetInputs (  
    const CHANNEL          nchannel,  
    int * const             pStream,  
    int * const             pSlot1,  
    int * const             pStream2,  
    int * const             pSlot2  
);
```

Parameters

nchannel	channel number
pStream1	channel primary input connection stream, Pointer for TDM stream index: 0 - 15, -1 for not connected
pSlot1	channel primary input connection timeslot: 0 to 31 for MVIP number varies for H.100 (see above) when stream is -1, a board counter is returned for the slot value
pStream2	channel secondary input connection stream, Pointer for TDM stream index: 0 - 15, -1 for not connected
pSlot2	channel secondary input connection timeslot: 0 to 31 for MVIP number varies for H.100 (see above) when stream is -1, a board counter is returned for the slot value

Return Codes

MT_RET_OK	No error
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_CHANNEL_NOT_OPENED	Channel device not opened for access
MT_RET_INVALID_PARAM	Invalid argument passed to function

Events

None

MTResetInputs()

Description

Calling this API will disconnect the primary and secondary input connections from the CT bus (MVIP or H.100). If a channel is not connected to the CT bus, the return value for all directions and timeslots will be -1. If a channel is connected, proper values according to the settings (made with **MTSetInputs()**) will be returned.

Scenario

Use Supervise APIs to monitor remote conversations on the same CT bus.

Prototype

```
MT_RESULT MTResetInputs(  
    const CHANNEL          nChannel  
);
```

Parameters

nChannel channel number

Return Codes

MT_RET_OK	No error
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_CHANNEL_NOT_OPENED	Channel device not opened for access
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_MUX_OFF	Multiplexer system was not started

Events

None

MTSetOutput()

Description

MTSetOutput() connects the channel's DSP output to the specified MVIP/H.100 time slot for monitoring. **MTResetOutput()** clears the connection.

Note: Only one channel can transmit (talk) on a time slot, however, multiple channels can receive (listen) on a time slot (See **MTSetInputs()**).

When used, the channel source must be defined by using **MTSetOutputSource()**. As the incoming data passes through the DSPs, the following can still be applied to channel data: DTMF/MF, Caller ID, Activity detection, Automatic Gain Control and Gain. The data is passed into the CT Bus and also processed on the board for recording. Refer to the section that explains **MTSetOutputSource()** for a drawing of the logical diagram.

Prototypes

MT_RESULT MTSetOutput(

```
const CHANNEL      nChannel,  
const int          iStream,  
const int          iSlot  
):
```

Parameters

nChannel	channel number
iStream	stream number (0-15 for MVIP, 0 - 31 for H.100)
iSlot	time slot number (0-31 for MVIP, 0 - 127 for H.100 - assuming 8192 KHz)

Return Codes

MT_RET_OK	No error
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_CHANNEL_NOT_OPENED	Channel device not opened for access
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_BUSY	When the specified timeslot is already taken

Events

None

See Also: [MTSetOutputSource\(\)](#)

MTGetOutput()

Description

Retrieves the MVIP/H.100 bus connectivity of a specified channel's DSP output. If a DSP output is not connected to the MVIP/H.100 bus, pStream will be returned with the value of -1.

Prototype

```
MT_RESULT MTGetOutput(  
    const CHANNEL          nChannel,  
    int * const             pStream,  
    int * const             pSlot  
);
```

Parameters

nChannel	channel number
pStream	buffer for the returned stream index -1 = not connected
pSlot	buffer for the returned slot index

Return Codes

MT_RET_OK	No error
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_CHANNEL_NOT_OPENED	Channel device not opened for access
MT_RET_INVALID_PARAM	Invalid argument passed to function

Events

None

MTResetOutput()

Description

MTResetOutput() clears a channel's connection to the MVIP/H.100 timeslot set by **MTSetOutput()**.

Prototypes

MT_RESULT MTResetOutput(

```
const CHANNEL      nChannel  
);
```

Parameters

nChannel	channel number

Return Codes

MT_RET_OK	No error
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_INVALID_CHANNEL	Channel number is out of range
MT_RET_CHANNEL_NOT_OPENED	Channel device not opened for access
MT_RET_INVALID_PARAM	Invalid argument passed to function

Events

None

MTGetFramerOutput()

Description

Used to get framer data from MVIP or H.100 Bus. More information is provided on the next page that explains **MTSetFramerOutput()**.

Prototype

```
MT_RESULT MTGetFramerOutput (  
    const BOARD          nBoard,  
    const USHORT         nFramer,  
    const USHORT         nSlot,  
    int * const          pStream,  
    int * const          pSlot  
);
```

Parameters

nBoard	Board index
nFramer	Index of framer on the specified board: 0-3 for the DP, and 0-1 for the DT board, 0-23 NGX (24 channel card)
nSlot	Index of framer slot: 1-24 for T1, 1-30 for E1
pStream	Index of TDM stream: 0-15 for MVIP, 0-31 for H100
pSlot	Index of TDM slot on a stream: 0-31 for MVIP

Return Codes

MT_RET_NOT_AVAILABLE	Driver is not loaded

Events

None

MTSetFramerOutput()

Description

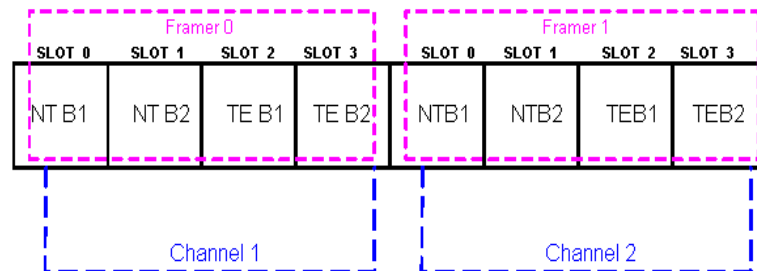
The API gives the user the ability to route the data from the output of the framer to the CT Bus while still passing the same input data to the DSPs by default. The data routed to the CT Bus directly from the framers is not processed by the DSPs, therefore no gain control is applied.

When using the SmartWORKS DP card the following rules apply: There are two framers per trunk. Each framer contains a total of 24 (T1) or 31(E1) Timeslots.

Before developing applications that work in 4-wire scenarios, it is important to understand how framing and timeslot numbering was implemented on the SmartWORKS NGX product. The following diagram and table are used to illustrate framing and timeslot numbering. A standard 2-wire implementation is compared to a 4-wire implementation. A mapping table is also provided:

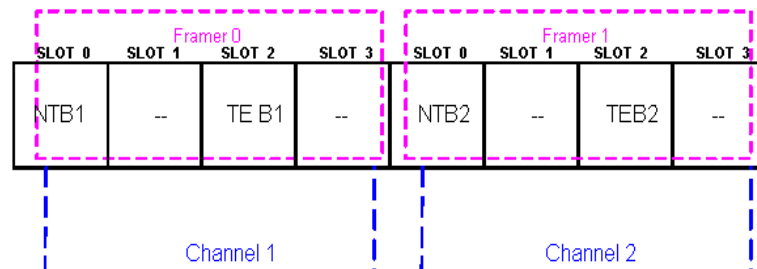
NGX

2-Wire Implementation



NOTE: In both diagrams, Framer 0 represents incoming traffic, and Framer 1 represents outgoing traffic.

4-Wire Implementation



The following table maps framer and timeslot numbering between a 2-wire implementation versus a 4-wire.

FRAMER AND TIMESLOT NUMBERING (FRAMERS 0-3)

Framer	Timeslot	2 Wire	Line	4 Wire	Line
0	0	NT B1	1	NT B1	1
0	1	NT B2	1	--	--
0	2	TE B1	1	TE B1	1
0	3	TE B2	1	--	--
1	0	NT B1	2	NT B2	1
1	1	NT B2	2	--	--
1	2	TE B1	2	TE B2	1
1	3	TE B2	2	--	--

•
•
•

Framer	Timeslot	2 Wire	Line	4 Wire	Line
23	0	NT B1	24	NT B2	12
23	1	NT B2	24	--	--
23	2	TE B1	24	TE B2	12
23	3	TE B2	24	--	--

Prototype

MT_RESULT MTSetFramerOutput (

const BOARD

const USHORT

const USHORT

const int

const int

);

nBoard,

nFramer,

nSlot,

iStream,

iSlot

Parameters

nBoard

nFramer

Board index

Index of framer on the specified board:
0-3 for the DP, 0-1 DT, and 0-23 for the
NGX (24 channel card)

nSlot	Index of framer slot: 1-24 for T1, 0-31 for E1
iStream	Index of TDM stream: 0-15 for MVIP, 0-31 for H100
iSlot	Index of TDM slot on a stream: 0-31 for MVIP

Return Codes

MT_RET_NOT_AVAILABLE	Driver is not loaded

Events

None

MTResetFramerOuput()

Description

Resets framer output. More information is provided in the section that explains [MTSetFramerOutput\(\)](#).

Prototype

```
MT_RESULT MTResetFramerOutput (  
    const BOARD          nBoard,  
    const USHORT         nFramer,  
    const USHORT         nSlot  
);
```

Parameters

nBoard	Board index
nFramer	Index of framer on the specified board
nSlot	Index of framer slot: 1-24 for T1, 0-31 for E1

Return Codes

MT_RET_NOT_AVAILABLE	Driver is not loaded

Events

None

LOOP VOLTAGE / LOOP CURRENT / RING DETECTION STATUS FUNCTIONS

MTGetLVDetectState()

Description

This API is only supported on the PT board.

MTGetLVDetectState() retrieves the detection state of loop voltage. If a value of 0 is returned, this indicates that there is no voltage - the channel is currently OFF_HOOK or has been disconnected.

Prototype

```
MT_RESULT MTGetLVDetectState (  
    const CHANNEL          nChannel,  
    const PCHAR             pState  
);
```

Parameters

nChannel	channel number
pState	pointer to buffer for the detect state, 0 indicates no voltage

Return codes

MT_RET_OK	no error
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_INVALID_PARAM	invalid board index or framer index
MT_RET_INVALID_CHANNEL	Channel index error
MT_RET_CHANNEL_NOT_OPENED	Channel not owned by the caller

Events

None

MTGetLCDetectState()

Description

This API is only supported on AT and LD boards.

This API gets loop current detect state per channel. If the value of 0 is returned this indicates that there is no loop current - the line is either disconnected or ON_HOOK.

Prototype

MT_RESULT MTGetLCDetectState (

```
const CHANNEL      nChannel,  
const PCHAR        pState  
):
```

Parameters

nChannel	channel number
pState	pointer to buffer for the detect state, 0 for not detecting none

Return codes

MT_RET_OK	no error
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_INVALID_PARAM	Invalid board index or framer index
MT_RET_CHANNEL_NOT_OPENED	Channel not owned by the caller

Events

None

MTGetRingDetectState()

Description

MTGetRingDetectState() retrieves the current detection state of ring. A returned value of 0 indicates that the channel is currently not detecting incoming ring. A value other than 0 indicates a ring is currently being detected.

Prototypes

```
MT_RESULT MTGetRingDetectState (  
    const CHANNEL          nChannel,  
    const PCHAR            pState  
);
```

Parameters

nChannel	channel index
pState	pointer to buffer for the detect state, 0 for none detected

Return codes

MT_RET_OK	no error
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_INVALID_PARAM	Invalid board index or framer index
MT_RET_CHANNEL_NOT_OPENED	Channel not owned by the caller
MT_RET_INVALID_CHANNEL	Channel index error

Events

None

MTSetLVParams()

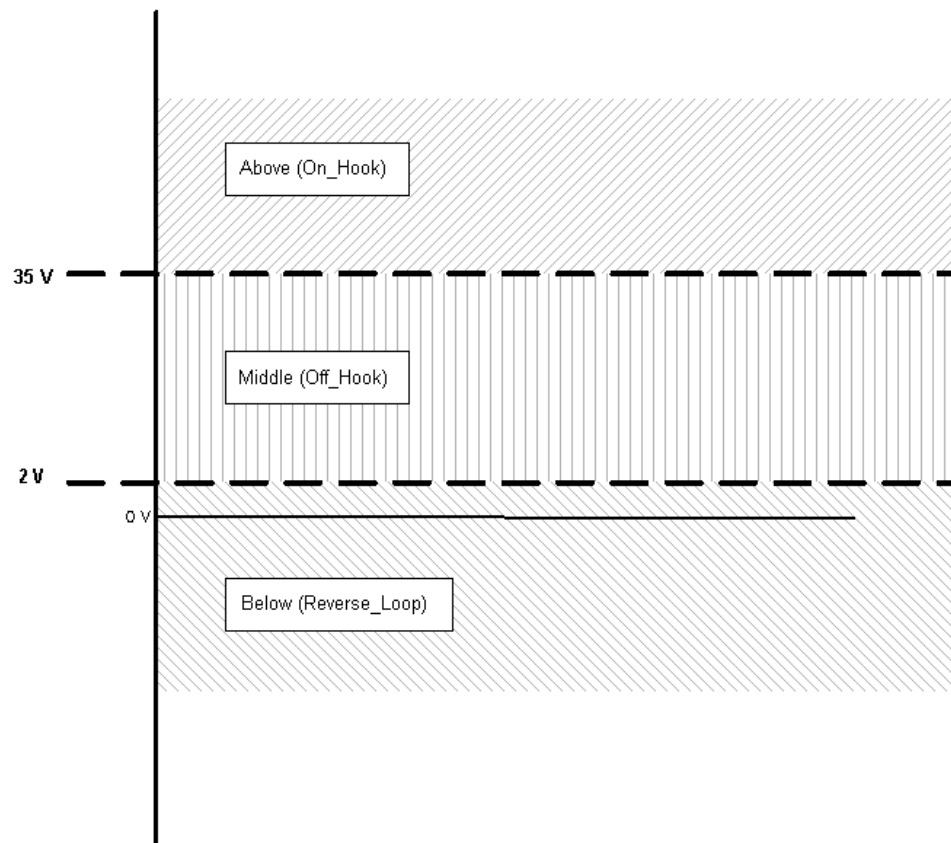
Description

The API is only supported on the SmartWORKS LD boards. LDA provides the user capability to set the thresholds of voltage high and voltage low. This allows custom definition of the three loop voltage states of ABOVE, BELOW, and MIDDLE (e.g. ONHOOK, REVERSE, and OFFHOOK states are the common terms for ABOVE, BELOW, and MIDDLE states).

With LD's capability of detecting the three states of loop voltage change, a wink, and current voltage reading. The Default value for Threshold_high was selected to maintain compatibility with SmartWORKS PT 1609.

The field loop_deglitch in MT_PSTN structure is for loop current de-bouncing.

A complete explanation of loop voltage detection is provided in the SmartWORKS Developer's Guide. The following is a visual representation of voltage states (ABOVE, BELOW, and MIDDLE).



Prototype

```
MT_RESULT MTSetLVParams(  
    const CHANNEL      nChannel,  
    const SHORT        ThresholdLow,  
    const SHORT        ThresholdHigh,  
    const USHORT       msDeglitchTime  
);
```

Parameters

nChannel	channel number
ThresholdLow	LV threshold low: -60V to 60V, default = -16V
ThresholdHigh	LV threshold high: 60V to -60V, default = +16V
msDeglitchTime	De-bouncing time in unit of 10ms, valid range is 10 ms to 2550 ms, default 50ms

Return codes

MT_RET_OK	no error
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_INVALID_PARAM	Invalid board index or framer index
MT_RET_CHANNEL_NOT_OPENED	Channel not owned by the caller

Events

None

MTGetLVParams()

Description

The API is only supported on the SmartWORKS LD boards.

LDA provides user the configuration capability to set the thresholds of voltage high and voltage low. This allows custom definition of the three loop voltage states of ABOVE, BELOW, and MIDDLE (e.g. ONHOOK, REVERSE, and OFFHOOK states are the common terms for ABOVE, BELOW, and MIDDLE states).

With LDA's capability of detecting the three states of loop voltage change, LDA can detect the presence of a wink, a state PT channel cannot detect. LDA can also provide current voltage reading, a capability PT channel does not have.

The field loop_deglitch in MT_PSTN structure is for loop current de-bouncing

A complete explanation of loop voltage detection is provided at the beginning of this section: [Loop Voltage / Loop Current / Ring Detection Status Functions](#).

Prototype

```
MT_RESULT MTGetLVParams (
    const CHANNEL      nChannel,
    const PSHORT       pThreshold_low,
    const PSHORT       pThreshold_high,
    const PUSHORT      pmsDeglitchTime
);
```

Parameters

nChannel	channel number
pThreshold_low	LV threshold low: -60V to 60V
pThreshold_high	LV threshold high: 60V to -60V
pmsDeglitchTime	10-2550 ms, in increments of 10 ms

Return codes

MT_RET_OK	no error
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_INVALID_PARAM	Invalid board index or framer index
MT_RET_CHANNEL_NOT_OPENED	Channel not owned by the caller

Events

None

MTGetLVDetectValue()

Description

The API is only supported on the SmartWORKS LD boards. It is used for the purpose of measuring the voltage on any particular channel.

A complete explanation of loop voltage detection is provided at the beginning of this section: [Loop Voltage / Loop Current / Ring Detection Status Functions](#).

Prototype

```
MT_RESULT MTGetLVDetectValue (  
    const CHANNEL          nChannel,  
    const PSHORT           pValue  
);
```

Parameters

nChannel	channel number
pValue	Current LV reading in increments of 2V

Return codes

MT_RET_OK	no error
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_INVALID_PARAM	Invalid board index or framer index
MT_RET_CHANNEL_NOT_OPENED	Channel not owned by the caller

Events

None

MTSetReverseLoopPolarity()

Description

The API is only supported on the SmartWORKS LD boards.

This API (***MTSetReverseLoopPolarity()***) is used to set a parameter if the loop polarity on a line is reversed. First, use ***MTGetLVDetect()*** to obtain the current ONHOOK voltage reading. If the voltage is negative, then the line polarity is reversed. Use this API to reverse the line polarity through a software change rather than a wiring change.

Prototype

MT_RESULT MTSetReverseLoopPolarity (

```
const CHANNEL      nChannel,  
const BOOLEAN      fReversed  
);
```

Parameters

nChannel	channel number
fReversed	FALSE for normal, TRUE for reversed The default is normal when a channel is opened.

Return Codes

MT_RET_OK	no error
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_INVALID_PARAM	Invalid board index or framer index
MT_RET_CHANNEL_NOT_OPENED	Channel not owned by the caller

Events

None

MTGetReverseLoopPolarity()

Description

The API is only supported on the SmartWORKS LD boards.

Line status for LDA card will be represented through status bit LINE_ONHOOK, LINE_POLARITY(0 for normal; 1 for reversed), LINE_NO_LOOP, and LINE_NO_LVOLTAGE_MIDDLE

A complete explanation of loop voltage detection is provided at the beginning of this section: [Loop Voltage / Loop Current / Ring Detection Status Functions](#).

Prototype

MT_RESULT MTGetReverseLoopPolarity (

const CHANNEL

BOOLEAN * const

);

nChannel,

pReversed

Parameters

nChannel	channel number
pReversed	Pointer of buffer for reversed polarity reading

Return Codes

MT_RET_OK	no error
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_INVALID_PARAM	Invalid board index or framer index
MT_RET_CHANNEL_NOT_OPENED	Channel not owned by the caller

Events

None

SUMMATION FUNCTIONS

MTGetSummationInfo ()

This API was previously called **MTGetSummationResource()**, and **MTGetSummationInputCount()**.

Description

This API is used to return the total number of inputs the specified summation resource supports.

It is suggested that the user application use this API to query for the maximum number of inputs supported by a summation resource before launching the summation resource to work.

Prototype

MT_RESULT MTGetSummationInfo (

```
const int                iSummation,  
PMTSUMMATION_INFO const pInfo,  
PULONG const            pLength  
);
```

Parameters

iSummation	Index of summation resource: 0 for the first one
pInfo	Pointer to resource information structure, MTSUMMATION_INFO.
pLength	Set by the user application, this is a pointer to the size of the given buffer for the MTSUMMATION_INFO structure.

Structure

The following table explains each field of the MTSUMMATION_INFO structure:

TABLE 51: MTSUMMATION_INFO

Type	Name	Function
int	tResource	Summation resource sub-type, if any
int	iResource	Summation resource system index
int	iChGCI	GCI index of the channel which controls the summation resource
int	iBoard	System index of the board the summation resides
int	iOnBoard	Summation index within the board it resides
int	nInputs	Number of inputs for the summation

Return codes

MT_RET_OK	no error
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_INVALID_PARAM	Invalid board index or framer index

Events

None

MTResetSummation()

Description

Clears the all time slot connections of inputs and outputs on the specified summation function. **MTResetSummation()** disconnects the previously connected summation inputs and outputs.

For summation resources, when the input is added, the gain will be set to the specified value. When the input is removed, the gain is reset to the default, -50dB (mute).

Prototype

MT_RESULT MTResetSummation(

const int iSummation
);

Parameters

iSummation	Index of summation function, 0 for the first

Return Code

MT_RET_OK	no error
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_INVALID_PARAM	Invalid board index or framer index
MT_RET_NOT_STARTED	The specified input is not connected

Events

None

See Also: [MTRemoveSummationInput\(\)](#)

MTStartSummation()

Description

MTStartSummation() connects the specified summation output to the specified stream and time slot. When the CT bus is set to MVIP, the stream index is from 0 to 15 for the 16 available streams, and the time slot index is from 0 to 31 for the available time slots on each stream.

If the CT bus type is H.100, the stream index is from 0-31 for the 32 available streams. The number of timeslots available per stream depends on which H.100 stream speed is selected in the SmartControl panel.

At 2048 KHz, 32 timeslots per stream are available.

At 4096 KHz, 64 timeslots per stream are available.

At 8192 KHz, 128 timeslots per stream are available.

Prototypes

MT_RESULT MTStartSummation (

```
const int          iSummation,  
const int          iStream,  
const int          iSlot  
);
```

Parameters

iSummation	summation function index: 0 for the first one
iStream	summation output stream index, 0 - 15 for MVIP, 0-31 for H.100
iSlot	summation output time slot index, 0 - 31 for MVIP, for H.100 the value is determined by user configured stream speed (see description above).

Return Code

MT_RET_OK	no error
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_INVALID_PARAM	Invalid board index or framer index
MT_RET_BUSY	On output time slot taken

Events

None

MTStopSummation()

Description

MTStopSummmation() disconnects the previously connected summation output. No other settings are changed.

Prototype

```
MT_RESULT MTStopSummation (  
    const int                iSummation  
);
```

Parameters

iSummation	summation function index: 0 for the first one

Return codes

MT_RET_OK	no error
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_INVALID_PARAM	Invalid board index or framer index
MT_RET_NOT_STARTED	On summation not started

Events

None

MTAddSummationInput()

Description

MTAddSummationInput() connects an input timeslot to an input unit on the specified summation function. Should there be a timeslot previously connected to that input unit, the previous connection will be replaced with the new time slot. The rest of the summation inputs will not be altered.

Valid gain values range from -50 dB to +24dB.

For summation resources, when an input is added, the gain will be set to the specified value. When the input is removed, the gain is reset to the default, -50dB(mute).

When the CT bus is set to MVIP, the stream index is from 0 to 15 for the 16 available streams, and the time slot index is from 0 to 31 for the available time slots on each stream.

If the CT bus type is H.100, the stream index is from 0-31 for the 32 available streams. The number of timeslots available per stream depends on which H.100 stream speed is selected in the SmartControl panel.

At 2048 KHz, 32 timeslots per stream are available.

At 4096 KHz, 64 timeslots per stream are available.

At 8192 KHz, 128 timeslots per stream are available.

Prototype

MT_RESULT MTAddSummationInput (

```
const int          iSummation,
const int          iInput,
const int          iStream,
const int          iSlot,
const SHORT        Gain
);
```

Parameters

iSummation	Index of summation function, 0 for the first
iInput	Index of summation input: 0 - 15 for 16 summation inputs.
iStream	Index of TDM output stream: 0 - 15 for MVIP, 0-31 for H.100
iSlot	Index of TDM output slot on a stream: 0 - 31 for MVIP, 0 - 31 for MVIP, for H.100 the value is determined by user configured stream speed (see description above).
Gain	Initial gain for the specified input: valid values: =24dB to -50dB.

Return codes

MT_RET_OK	no error
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_INVALID_PARAM	Invalid board index or framer index
MT_RET_BUSY	On time slot connection error

Events

None

MTRemoveSummationInput()

Description

MTRemoveSummationInput() disconnects the previously connected summation input.

Prototypes

MT_RESULT MTRemoveSummationInput (

```
const int          iSummation,  
const int          iInput  
);
```

Parameters

iSummation	Summation function index: 0 for the first one
iInput	Summation input index: 0 to 15 for 16 summation inputs

Return Codes

MT_RET_OK	no error
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_INVALID_PARAM	Invalid board index or framer index
MT_RET_NOT_STARTED	The specified input is not connected

Events

None

MTGetSummationInputs()

Description

MTGetSummationInputs() retrieves the time slot connections for a specified number of inputs on the specified summation function. The operating state of the specified summation has no bearing on this function. Inputs for a summation can always be connected without the summation function being turned on.

The pInput parameter is a pointer to an integer specifying the total number of inputs the caller intends to retrieve. This function always works from the 1st input. For example, if this parameter sets the total number to 5, input settings are retrieved for the inputs numbered 0-4.

Parameters pStream, pSlot, and pGain all work as an array. This array must be large enough to accept the total number of input settings specified with the pInput parameter. When returned, this integer contains the number of inputs retrieved into the provided buffer. Fox example, if a summation unit has a maximum of 16 inputs and the caller specifies the retrieval of 32 inputs, the API will send the updated information into the provided buffer and update the input count integer from 32 to the actual number of 16.

Prototype

MT_RESULT MTGetSummationInputs (

const int	iSummation,
int * const	pInputs,
int * const	pStream,
int * const	pSlot,
SHORT * const	pGain
);	

Parameters

iSummation	Summation function index: 0 for the first one
pInputs	Pointer to the total number of input settings to retrieve.
pStream	Pointer as an array to the streams for the 16 inputs
pSlot	Pointer as an array to time slots for the 16 summation inputs
pGain	Pointer as an array to gains for the 16 summation inputs

Return Codes

MT_RET_OK	no error
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_INVALID_PARAM	Invalid board index or framer index

MT_RET_CHANNEL_NOT_OPENED

Occurs when there's no ownership of the specified summation function

EventsNone

MTGetSummationOutput()

Description

MTGetSummationOutput() retrieves the time slot connection for the output on the specified summation function.

When the specified summation function is turned off, the returned output time slot information will indicate that the output is not connected. This is done by setting both the stream and time slot values to -1.

Prototypes

MT_RESULT MTGetSummationInputs (

const int

int * const

int * const

PSHORT const

);

iSummation,

pStream,

pSlot,

pVolume

Parameters

iSummation	Summation function index: 0 for the first one
pStream	Pointer for stream for the summation output
pSlot	Pointer for time slot for the summation output
pVolume	Pointer for the current volume for the summation output

Return Codes

MT_RET_OK	no error
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_INVALID_PARAM	Invalid board index or framer index
MT_RET_CHANNEL_NOT_OPENED	Occurs when there's no ownership of the specified summation function
MT_RET_NOT_STARTED	The specified output is not connected

Events

None

MTSetSummationGain()

Description

MTSetSummationGain() sets the gain for summation input on the specified summation function. The operating state of the specified summation has no bearing on this function. Gain for a summation input can be changed whether or not the summation function is turned on.

Valid gain values range from -50 dB to +24dB.

For summation resources, when an input is added, the gain will be set to the specified value. When the input is removed, the gain is reset to the default, -50dB (mute).

Prototype

MT_RESULT MTSetSummationGain (

```
const int          iSummation,  
const int          ilInput,  
const SHORT        Gain  
);
```

Parameters

iSummation	Summation function index: 0 for the first one
ilInput	Index of summation input: 0 - 15 for 16 summation inputs
Gain	The gain setting, which ranges from -50dB to +24dB,

Return codes

MT_RET_OK	no error
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_INVALID_PARAM	Invalid board index or framer index
MT_RET_CHANNEL_NOT_OPENED	On failure to gain ownership of the specified summation function

Events

None

MTGetSummationGain()

Description

MTGetSummationGain() gets the gain for the summation input on the specified summation function. The operating state of the specified summation has no bearing on this function. Gain for a summation input can be read whether or not the summation function is turned on.

Valid gain values range from -50 dB to +24dB.

For summation resources, when an input is added, the gain will be set to the specified value. When the input is removed, the gain is reset to the default, -50dB(mute).

Prototype

```
MT_RESULT MTGetSummationGain (  
    const int          iSummation,  
    const int          iInput,  
    const SHORT        pGain  
);
```

Parameters

iSummation	Summation function index: 0 for the first one
iInput	Index of summation input: 0 - 15 for 16 inputs.
pGain	Pointer to the gain setting

Return codes

MT_RET_OK	no error
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_INVALID_PARAM	Invalid board index or framer index
MT_RET_CHANNEL_NOT_OPENED	On failure to gain ownership of the specified summation function

Events

None

MTSetSummationVolume()

Description

MTSetSummationVolume() sets the volume for summation output on the specified summation function. The operating state of the specified summation has no bearing on this function. Volume for a summation output can be changed whether or not the summation function is turned on.

Prototype

MT_RESULT MTSetSummationVolume (

```
const int          iSummation,  
const SHORT        Volume  
);
```

Parameters

iSummation	Summation function index: 0 for the first one
Volume	Output volume for the specified summation output. Valid values: 24 db to -50dB.

Return codes

MT_RET_OK	no error
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_INVALID_PARAM	Invalid board index or framer index
MT_RET_CHANNEL_NOT_OPENED	On failure to gain ownership of the specified summation function

Events

None

MTGetSummationVolume()

Description

MTGetSummationVolume() gets the volume for summation output on the specified summation function. The operating state of the specified summation has no bearing on this function. Volume for a summation output can be read whether or not the summation function is turned on.

Prototype

```
MT_RESULT MTGetSummationVolume (  
    const int          iSummation,  
    PSHORT const      pVolume  
);
```

Parameters

iSummation	Summation function index: 0 for the first one
pVolume	The volume setting, Valid values: 24 dB to -50dB

Return Codes

MT_RET_OK	no error
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_INVALID_PARAM	Invalid board index or framer index
MT_RET_CHANNEL_NOT_OPENED	On failure to gain ownership of the specified summation function

Events

None

MTSetSummationAVC()

Description

MTSetSummationAVC() sets the Automatic Volume Control (AVC) for summation output on the specified summation function. The operating state of the specified summation has no bearing on this function. The AVC setting for a summation output can be changed whether or not the summation function is turned on. Configuring the AVC settings does not enable AVC functionality. Use **MTSummationAVCControl()** to enable or disable AVC on summation output.

Prototype

MT_RESULT MTSetSummationAVC (

```
const int          iSummation,  
const SHORT        TMA,  
const SHORT        AVCR,  
const SHORT        DVCR,  
const SHORT        MA  
);
```

Parameters

iSummation	Summation function index: 0 for the first one
TMA	Target maximum amplitude (TMA), 1 to -45 dBm, def=-6 dBm
AVCR	Attack Volume Change Rate, in unit of -0.00212 dB/ms, def=400 value must be <=0
DVCR	Decay Volume Change Rate, in unit of 0.00212 dB/ms, def=4 value must be <=0
MA	Max. Amplification, 6 dB to 48 dB in 6dB steps, def=30 dB

Return codes

MT_RET_OK	no error
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_INVALID_PARAM	Invalid board index or framer index
MT_RET_CHANNEL_NOT_OPENED	On failure to gain ownership of the specified summation function

Events

None

MTGetSummationAVC()

Description

MTGetSummationAVC() retrieves the AVC settings for the output on the specified summation function. The operating state of the specified summation has no bearing on this function. AVC settings for a summation output can be read whether or not the summation function is turned on.

Prototypes

MT_RESULT MTGetSummationAVC (

const int

PSHORT const

PSHORT const

PSHORT const

PSHORT const

);

iSummation,

pTMA,

pAVCR,

pDVCR,

pMA

Parameters

iSummation	Summation function index: 0 for the first one
pTMA	Pointer for target max. amplitude (TMA)
pAVCR	Pointer for attack volume change rate (AVCR)
pDVCR	Pointer for decay volume change rate (DVCR)
pMA	Pointer for max amplification (MA)

Return Codes

MT_RET_OK	no error
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_INVALID_PARAM	Invalid board index or framer index
MT_RET_CHANNEL_NOT_OPENED	On failure to gain ownership of the specified summation function

Events

None

MTSummationAVCControl()

Description

MTSummationAVCControl() controls the AVC for the output on the specified summation function. The operating state of the specified summation has no bearing on this function. AVC for a summation output can be configured whether or not the summation function is turned on.

Prototype

MT_RESULT MTSummationAVCControl (

```
const int          iSummation,  
const BOOLEAN      fOn  
);
```

Parameter

iSummation	Summation function index: 0 for the first one
fOn	TRUE for enable; FALSE for disable the AVC

Return Codes

MT_RET_OK	no error
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_INVALID_PARAM	Invalid board index or framer index
MT_RET_CHANNEL_NOT_OPENED	On failure to gain ownership of the specified summation function

Events

None

MTSummationAVCStatus()

Description

MTSummationAVCStatus() reads the AVC status for the output on the specified summation function. The operating state of the specified summation has no bearing on this function. AVC enabled/disabled status for a summation output can be read whether or not the summation function is turned on.

Prototype

MT_RESULT MTSummationAVCStatus (

const int

BOOLEAN * const

);

iSummation,

pOn

Parameters

iSummation	Summation function index: 0 for the first one
pOn	Pointer for the status

Return Codes

MT_RET_OK	no error
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_INVALID_PARAM	Invalid board index or framer index
MT_RET_CHANNEL_NOT_OPENED	On failure to gain ownership of the specified summation function

Events

None

TRUNK SPECIFIC APIs

MTOffHook()

Description

MTOffHook() takes a specified channel off hook provided that the channel is in the on hook state. MT_RET_BUSY will be returned if the channel is already in the off hook state. When a channel successfully goes off hook, EVT_OFFHOOK will be issued [if enabled through **MTSetEventFilters()**].

This API is only available on ANALOG_VFXO_LS_E channels on the SmartWORKS LD.

If loop current starts flowing by going off-hook, EVT_OFFHOOK will be followed by EVT_LOOP_ON, if enabled. If EVT_OFFHOOK is disabled with the MTSetEvents() function, the application must poll the line status field of MTCHAN_STATUS structure to determine when this function terminates.

Prototype

```
MT_RESULT MTOffHook (  
    CHANNEL          nChannel  
);
```

Parameter

nChannel	channel number

Return Codes

MT_RET_OK	no error
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_INVALID_CHANNEL	Channel number is out of range.
MT_RET_CHANNEL_NOT_OPENED	Channel device not opened for access
MT_RET_BUSY	Channel is already off hook

Events

EVT_OFFHOOK
EVT_LOOP_ON
EVT_LOOP_REVERSE

MTOnHook()

Description

MTOnHook() changes a specified channel that is off hook to the on hook state. MT_RET_BUSY will be returned if the specified channel is already on hook. When a channel successfully goes on hook, EVT_ONHOOK will be issued (if enabled through **MTSetEventFilters()**).

This API is only available on *ANALOG_VFXO_LS_E* type channels on the SmartWORKS LD card.

Prototype

MT_RESULT MTOnHook (

CHANNEL nChannel
);

Parameters

nChannel channel number

Return Codes

MT_RET_OK	no error
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_INVALID_CHANNEL	Channel number is out of range.
MT_RET_CHANNEL_NOT_OPENED	Channel device not opened for access
MT_RET_BUSY	Channel is already on hook

Events

EVT_ONHOOK
EVT_LOOP_DROP

MTGetHookState()

Description

Returns the on/off hook state for an identified channel.

Prototype

```
MT_RESULT MTGetHookState (
CHANNEL          nChannel,
const PCHAR      pState
);
```

Parameters

nChannel	channel number
pState	0 for onhook

Return Codes

MT_RET_OK	no error
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_INVALID_CHANNEL	Channel number is out of range.
MT_RET_CHANNEL_NOT_OPENED	Channel device not opened for access

Events

None

MTWink()

TBD, at this time this API is not supported.

Description

MTWink() perform a wink, the sequence of going off-hook, waiting for a moment, and going on-hook again. The function terminates with an EVT_WINKDONE event. During this function, the status fields of the MTCHAN_STATUS structure are set to MODE_WINK.

This function can only be called when the channel is on-hook.

The timing of this function is controlled by two parameters that can be set with the **MTSetPSTNParams()** function. The wink-delay specifies the delay after issuing the function call before actually going off-hook. The wink-time specifies the time that the channel stays off-hook. Both parameters are specified in 1 ms units and have a default value of 150 ms.

Prototype

```
MT_RESULT MTWink (  
    CHANNEL          nChannel  
);
```

Parameters

nChannel	channel number

Return Codes

MT_RET_OK	no error
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_INVALID_CHANNEL	Channel number is out of range.
MT_RET_CHANNEL_NOT_OPENED	Channel device not opened for access
MT_RET_BUSY	Channel is busy

Events

EVT_WINKDONE

See Also: [MTSetPSTNParams\(\)](#)

WAVE HEADER UTILITY

MakeWaveGSMHeader()

Description

MakeWaveGSMHeader() takes the specified header data size and generates a RIFF header for the GSM format. Refer to [NtiData.h](#) for the definition of GSM_WAVE_HEADER, a standard Microsoft header format.

Note:

Only Microsoft GSM is supported.

Scenario

In the situation where wave header is not to be generated by APIs (such as streaming and device IO functions), the user application can use **MakeWaveGSMHeader()** to write the header when desired.

Prototype

```
void MakeWaveGSMHeader (
    PGSM_WAVE_HEADER    pWaveHeader,
    int                  BytesWritten
);
```

Parameters

pWaveHeader	pointer to buffer to take the header
BytesWritten	size of the header data (size of GSM_WAVE_HEADER)

Return Codes

MT_RET_NOT_AVAILABLE	Driver is not loaded

Events

None

CODE EXAMPLE

MakeWaveGSMHeader()

```
MakeWaveGSMHeader(&header, size of (GSM_WAV_HEADER)); //GSM_WAVE_HEADER = 60
```

A complete code example is available on the AudioCodes website:

<http://www.audiocodes.com/blades/support.htm>

MakeWavePCMHeader()

Description

MakeWavePCMHeader() takes the specified data size and generates a RIFF header for the PCM format. Refer to [NtiData.h](#) for the definition of PPCM_WAVE_HEADER, a standard Microsoft header format.

Scenario

In the situation where wave header is not to be generated by APIs (such as streaming and device IO functions), the user application can use **MakeWavePCMHeader()** to write the header when desired.

Prototype

```
void MakeWavePCMHeader (  
    PPCM_WAVE_HEADER    pWaveHeader,  
    int                  BytesWritten,  
    USHORT               Format  
);
```

Parameters

pWaveHeader	pointer to buffer to take the header
BytesWritten	size of the data
Format	format of 8 or 16 bit PCM

Return Codes

MT_RET_NOT_AVAILABLE	Driver is not loaded

Events

None

MTReturnEventBuffer()

Description

MTReturnEventBuffer() returns the event buffer to the API after the user application is finished processing the event. **NOTE:** This API is only required when call control events (EVT_CC_) are reported.

The `ptrBuffer` field of the `MT_EVENT` structure is used for two purposes:

1. The SmartWORKS DLL has terminated a media function based on a termination event. The `ptrBuffer` field indicates the pointer to the data buffer on which the event happened.
2. An event that contains extra information was reported. These events include call control events (`EVT_CC_`). In this scenario, `ptrBuffer` field points to the buffer allocated by the SmartWORKS DLL. When the event information is retrieved, the user application must invoke ***MTReturnEventBuffer()*** to return the allocated buffer.

If the user application relies on the callback method: the *ptrXtraBuffer* field is used to point to the buffer holding the extra data and the *XtraDataLength* field identifies the data length. When this new method of event handling is used, ***MTReturnEventBuffer()*** should not be used.

When using the polling method: The SmartWORKS DLL no longer allocates a buffer for the extra information. The user application must now allocate and manage this buffer. In the MT_EVENT structure set the *pXtraBuffer* field to identify the buffer created for extra information, and set the buffer length with the *XtraBufferLength* field. The *XtraDataLength* field is set by the SmartWORKS DLL and shows the actual length of data passed into the buffer. **NOTE:** If the actual data length exceeds the allocated buffer length, the data is truncated. Bit 2 of the *EventFlag* field in the MT_EVENT structure is set to 0x000000004 to notify the user application that the data has been truncated. When this new method of event handling is used, ***MTReturnEventBuffer()*** should not be used.

Prototype

```
void MTReturnEventBuffer(
    PVOID ptrBuffer
);
```

Parameters

ptrBuffer pointer to the buffer to return

Return Codes

MT RET NOT AVAILABLE	Driver is not loaded

Events

None

MTStartMUXOnChannel()

Description

MTStartMUXOnChannel() is called to prepare MUX operation on a specified channel before MUX connections can be made. All configurations will be set to default based on the configuration of the TDMCardType and TDMBusType. **MTStartMUX()** needs to be called before **MTStartMUXOnChannel()**.

Refer to **MTStartMUX()** for default MUX mapping on a specified channel.

Prototype

MT_RESULT MTStartMUXOnChannel (

const CHANNEL nChannel
);

Parameters

nChannel channel number

Return Codes

MT_RET_OK	no error
MT_RET_INVALID_CHANNEL	Channel number is out of range.
MT_RET_CHANNEL_NOT_OPENED	Channel device not opened for access
MT_RET_NO_MUX	Multiplexer is not available
MT_TIMEOUT	Wait for event timed out

Events

None

SMARTWORKS PCM SPECIFIC APIs

The following section contains APIs that are specific for use with the SmartWORKS PCM boards.

MTBoardSetIdleCoding()

Description

Sets the coding format for the idle packet. When the trunk is reporting loss of synchronization (EVT_LOSYNC_ALARM), the framer replaces the data stream normally passed to the TDM with an idle packet signal. By default, μ -Law idle packet is used. This API is only used with SmartWORKS PCM cards.

Prototype

MTBoardSetIdleCoding (

```
const BOARD      iBrd,  
const int        iTrunk,  
const int        iCodingFormat  
);
```

Parameters

iBrd	Board number
iTrunk	0 for the 1st trunk, 1 for the 2nd trunk
iCodingFormat	ULAWPXD or ALAWPXD

Return Codes

MT_RET_OK	No error
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_TIMEOUT	Wait for event timed out
MT_RET_INVALID_PARAM	Invalid argument passed to function

Events

None

MTBoardGetIdleCoding()

Description

Gets the coding format that is currently used for the idle packet sent out when the trunk is reporting loss of synchronization (EVT_LOSYNC_ALARM). By default, μ -Law idle packet is used.

Prototype

```
MTBoardGetIdleCoding (  
    const BOARD          iBrd,  
    const int            iTrunk,  
    int * const          iCodingFormat  
);
```

Parameters

iBrd	Board number
iTrunk	0 for the 1st trunk, 1 for the 2nd trunk
pCodingFormat	pointer to the coding format: ULAW-PXD or ALAWPX

Return Codes

MT_RET_OK	No error
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_TIMEOUT	Wait for event timed out
MT_RET_INVALID_PARAM	Invalid argument passed to function

Events

None

MTBoardSetTerminationImpedance()

Description

Sets the termination impedance for the identified trunk. By default, all trunks are set to Hi-Impedance. This API is used with SmartWORKS PCM cards only.

Prototype

MTBoardSetTerminationImpedance (

```
const BOARD      iBrd,  
const int        iTrunk,  
const int        iImpedance  
);
```

Parameters

iBrd	Board number
iTrunk	0 for the 1st trunk, 1 for the 2nd trunk
iImpedance	0 for Hi-Impedance(default), 1 for Low-Impedance

Return Codes

MT_RET_OK	No error
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_TIMEOUT	Wait for event timed out
MT_RET_INVALID_PARAM	Invalid argument passed to function

Events

None

MTBoardGetTerminationImpedance()

Description

Gets the termination impedance for the identified trunk.

Prototype

MTBoardGetTerminationImpedance (

```
const BOARD      iBrd,  
const int        iTrunk,  
int * const      plmpedance  
);
```

Parameters

iBrd	Board number
iTrunk	0 for the 1st trunk, 1 for the 2nd trunk
plmpedance	pointer to the current termination impedance: 0 for Hi-Impedance, 1 for Low-Impedance

Return Codes

MT_RET_OK	No error
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_TIMEOUT	Wait for event timed out
MT_RET_INVALID_PARAM	Invalid argument passed to function

Events

None

MTBoardPCMSetSignalPolarity()

This API replaces *MTBoardPCMSetSignalConfig()* (v.3.6.1). New application development should use this function.

Description

Sets the PCM Signal Configuration for the specified trunk. When this function is invoked the signal is reversed. This allows the application to adjust the pin interpretation if the paired wiring was reversed during installation. The default setting is normal for all signals.

Prototype

MTBoardPCMSetSignalPolarity (

```
const BOARD          iBrd,
const int            iTrunk,
MTBOARD_PCM_SIGNAL_POLARITY const pSignal,
PULONG const        pLength
);
```

Parameters

iBrd	Board number
iTrunk	0 for the 1st trunk, 1 for the 2nd trunk
pSignal	pointer to the PMTBOARD_PCM_SIGNAL_POLARITY structure
pLength	number of bytes for structure Signal

Structure

TABLE 52: MTBOARD_PCM_SIGNAL_POLARITY STRUCTURE

Type	Name	Values
BOOL	TxSignal	false for Normal (default), true for invert
BOOL	RxSignal	false for Normal (default), true for invert
BOOL	FrameSyncSignal	false for Normal (default), true for invert

Return Codes

MT_RET_OK	No error
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_TIMEOUT	Wait for event timed out
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_DATA_TRUNCATED	Data copied but truncated; the buffer size set by the user application is not large enough for this API request. <i>FlagDataTruncated</i> field of the MTSYS_CONFIG or the MTSYS_CONFIGURATION data structure must be enabled (1) to receive this return code, otherwise MT_RET_OK is used, and the truncated data is passed to the user application
MT_RET_NOT_ENOUGH_DATA	The amount of collected data does not satisfy request with timeout set to 0

Events

None

MTBoardPCMGetSignalPolarity()

This API replaces *MTBoardPCMGetSignalConfig()* (v3.6.1). New application development should use this function.

Description

Gets the PCM Signal Configuration for the specified trunk.

Prototype

MTBoardPCMGetSignalPolarity (

```
const BOARD          iBrd,  
const int            iTrunk,  
MTBOARD_PCM_SIGNAL_POLARITY const pSignal,  
PULONG const        pLength  
);
```

Parameters

iBrd	Board number
iTrunk	0 for the 1st trunk, 1 for the 2nd trunk
pSignal	pointer to the Signal structure
pLength	number of bytes for the structure pointed to by pSignal

Return Codes

MT_RET_OK	No error
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_TIMEOUT	Wait for event timed out
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_DATA_TRUNCATED	Data copied but truncated; the buffer size set by the user application is not large enough for this API request. <i>FlagDataTruncated</i> field of the MTSYS_CONFIG or the MTSYS_CONFIGURATION data structure must be enabled (1) to receive this return code, otherwise MT_RET_OK is used, and the truncated data is passed to the user application
MT_RET_NOT_ENOUGH_DATA	The amount of collected data does not satisfy request with timeout set to 0

Events

None

MTBoardPCMTrunkFrameSyncControl()

Description

By default, TDM frame reference is received on trunk 0 (framer 0). The SmartWORKS PCM cards can also transmit frame sync. This API can be used to switch the trunk reference or enable/disable frame sync transmission. By default, frame sync transmission is disabled on both trunks.

Prototype

MTBoardPCMTrunkFrameSyncControl (

```
const BOARD      iBrd,  
const int        iTrunk,  
const BOOL       fTxFrameSyncEnable  
);
```

Parameters

iBrd	Board number
iTrunk	0 for the 1st trunk, 1 for the 2nd trunk
fTxFrameSyncEnable	false for Disable (default), true for Enable

Return Codes

MT_RET_OK	No error
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_TIMEOUT	Wait for event timed out
MT_RET_INVALID_PARAM	Invalid argument passed to function

Events

None

MTBoardPCMTrunkFrameSyncStatus()

Description

Gets the current frame sync status on the specified trunk.

Prototype

MTBoardPCMTrunkFrameSyncStatus (

```
const BOARD      iBrd,  
const int        iTrunk,  
BOOL * const     pTxFrameSyncEnable  
);
```

Parameters

iBrd	Board number
iTrunk	0 for the 1st trunk, 1 for the 2nd trunk
pTxFrameSyncEnable	pointer to the frame sync status: false for Disable (default), true for Enable

Return Codes

MT_RET_OK	No error
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_TIMEOUT	Wait for event timed out
MT_RET_INVALID_PARAM	Invalid argument passed to function

Events

None

MTBoardPCMTrunkSlotTxControl()

Description

Enables/disables transmission on any specified timeslot. By default, transmission is disabled on all SmartWORKS PCM timeslots. The first channel on the board is connected to the first timeslot on the first trunk. On the PCM6409 the 33rd channel is connected to the first timeslot on the second trunk.

Prototype

MTBoardPCMTrunkSlotTxControl(

```
const BOARD          iBrd,  
const int            iTrunk,  
const DWORD32        fEnableBitMap  
);
```

Parameters

iBrd	Board number
iTrunk	0 for the 1st trunk, 1 for the 2nd trunk
fEnableBitMap	bit value 0 for disable, 1 for enable bit 0 for first slot on the specified trunk and etc.

Return Codes

MT_RET_OK	No error
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_TIMEOUT	Wait for event timed out
MT_RET_INVALID_PARAM	Invalid argument passed to function

Events

None

MTBoardPCMTrunkSlotTxStatus()

Description

Gets the current transmission status on the specified timeslot. The first channel is connected to the first timeslot on the first trunk. On the PCM6409 the 33rd channel is connected to the first timeslot on the second trunk.

Prototype

MTBoardPCMTrunkSlotTxStatus(

```
const BOARD      iBrd,  
cosnt int        iTrunk,  
PDWORD32         pEnableBitMap  
);
```

Parameters

iBrd	Board number
iTrunk	0 for the 1st trunk, 1 for the 2nd trunk
pEnableBitMap	pointer to a 4-byte (32 bits) slot table

Return Codes

MT_RET_OK	No error
MT_RET_NOT_AVAILABLE	Driver is not loaded
MT_RET_TIMEOUT	Wait for event timed out
MT_RET_INVALID_PARAM	Invalid argument passed to function

Events

None

IPX (VoIP) FUNCTIONS

The following section details the APIs that have been added to the SmartWORKS DLL to support the IPX board.

A logical discussion of all IPX (VoIP) functions is available in the *IPX Integration Guide*.

CONFIGURATION APIs

MTIpEnableSignalingProtocol()

Description

MTIpEnableSignalingProtocol() allows an application to enable (start) a specified protocol stack running on the IPX. More than one stack may be enabled at a single time.

NOTE: To change parameters associated with this function, the protocol must first be disabled and then enabled.

Prototype

int MTIpEnableSignalingProtocol (

const BOARDnBoard,

const USHORTnProtocolID,

PMT_IP_PROTOCOL_PARAMS pParams,

PULONG const pLength

);

Parameters

nBoard	Board number
nProtocolID	protocol ID for the signaling protocol: MT_IP_CISCO = 1 MT_IP_AVAYA_H323 = 2 MT_IP_NORTEL_UNISTIM = 4 MT_IP_ERICSSON_H323 = 5
pParams	Protocol dependent parameter structure.*
pLength	Pointer to the size (in bytes) of the MT_IP_PROTOCOL_PARAMS structure

* PBX specific configuration instructions are available in the *IPX Integration Guide* per each PBX manufacturer.

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_BOARD	Board number is out of range
MT_RET_BOARD_NOT_OPENED	Board is not opened for access
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_NOT_AVAILABLE	Function not available on this board

Events

None

MTIpDisableSignalingProtocol()

Description

MTIpDisableSignalingProtocol() allows an application to disable (stop) a specified protocol stack running on the IPX. More than one stack may be enabled at a single time.

This function is designed for synchronous operation only.

NOTE: To change parameters associated with this function, the protocol must first be disabled and then enabled.

Prototype

int MTIpDisableSignalingProtocol (

```
const BOARD          nBoard,  
const USHORT         nProtocolID,  
);
```

Parameters

nBoard	Board number
nProtocolID	protocol ID for the signaling protocol: MT_IP_CISCO = 1 MT_IP_AVAYA_H323 = 2 MT_IP_NORTEL_UNISTIM = 4 MT_IP_ERICSSON_H323 = 5

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_BOARD	Board number is out of range
MT_RET_BOARD_NOT_OPENED	Board is not opened for access
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_NOT_AVAILABLE	Function not available on this board

Events

None

MTIpSignalingProtocolStatus()

Description

MTIpSignalingProtocolStatus() returns the current runtime status (enabled/disabled) and the current configuration settings for the specified protocol

This function is designed for synchronous operation only.

Prototype

```
int MTIpSignalingProtocolStatus (  
    const BOARD                nBoard,  
    const USHORT               nProtocolID,  
    BOOLEAN * const            pOn,  
    PMT_IP_PROTOCOL_PARAMS const pParams,  
    PULONG const               pLength  
);
```

Parameters

nBoard	Board number
nProtocolID	protocol ID for the signaling protocol: MT_IP_CISCO = 1 MT_IP_AVAYA_H323 = 2 MT_IP_NORTEL_UNISTIM = 4 MT_IP_ERICSSON_H323 = 5
pOn	True = enable, FALSE = disabled
pParams	Protocol dependent parameter structure
pLength	Pointer to size (in bytes) of MT_IP_PROTOCOL_PARAMS structure

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_BOARD	Board number is out of range
MT_RET_BOARD_NOT_OPENED	Board is not opened for access
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_NOT_AVAILABLE	Function not available on this board

Events

None

MTIpDChannelEventControl()

Description

MTIpDChannelEventControl() allows an application to enable (start) or disable (stop) D-Channel event reporting for an IP signaling protocol on the specified board. Before using this API, the signaling stack for this protocol must first be enabled.

This function is designed for synchronous operation only.

Prototype

int MTIpDChannelEventControl (

```
const BOARD          nBoard,  
const USHORT         nProtocolID,  
const BOOLEAN        fOn  
);
```

Parameters

nBoard	Board number
nProtocolID	protocol ID for the signaling protocol: MT_IP_CISCO = 1 MT_IP_AVAYA_H323 = 2 MT_IP_NORTEL_UNISTIM = 4 MT_IP_ERICSSON_H323 = 5
fOn	Desired operation: TRUE = enabled, FALSE = disabled (default)

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_BOARD	Board number is out of range
MT_RET_BOARD_NOT_OPENED	Board is not opened for access
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_NOT_AVAILABLE	Function not available on this board

Events

None

MTIpDChannelEventStatus()

Description

MTIpDChannelEventStatus() retrieves the current operational status of D-Channel event reporting for the specified signaling protocol. Upon successful completion, pStatus will be set to TRUE(1) when enabled. If D-channel event reporting is disabled, then FALSE(0) is reported.

This function is designed for synchronous operation only.

Prototype

```
int MTIpDChannelEventStatus (  
    const BOARD          nBoard,  
    const USHORT         nProtocolID,  
    BOOLEAN * const      pOn  
);
```

Parameters

nBoard	Board number
nProtocolID	protocol ID for the signaling protocol: MT_IP_CISCO = 1 MT_IP_AVAYA_H323 = 2 MT_IP_NORTEL_UNISTIM = 4 MT_IP_ERICSSON_H323 = 5
pOn	Variable that returns the current operational status; TRUE = enabled, FALSE = disabled

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_BOARD	Board number is out of range
MT_RET_BOARD_NOT_OPENED	Board is not opened for access
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_NOT_AVAILABLE	Function not available on this board

Events

None

MTIpDChannelEventFilteringControl()

Description

MTIpDChannelEventFilteringControl() enables or disables DChannel event filtering when tapping with the SmartWORKS IPX. This operation is consistent with the NGX board. When enabled, DChannel event filtering removes duplicate events which would otherwise be reported to the user application.

This function is designed for synchronous operation only.

Prototype

int MTIpDChannelEventFilteringControl (

```
const BOARD          nBoard,
const USHORT         nProtocolID,
const BOOLEAN        fOn
);
```

Parameters

nBoard	Board number
nProtocolID	protocol ID for the signaling protocol: MT_IP_CISCO = 1 MT_IP_AVAYA_H323 = 2 MT_IP_NORTEL_UNISTIM = 4 MT_IP_ERICSSON_H323 = 5
fOn	Desired operation: TRUE = enabled FALSE = disabled (default)

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_BOARD	Board number is out of range
MT_RET_BOARD_NOT_OPENED	Board is not opened for access
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_NOT_AVAILABLE	Function not available on this board
MT_RET_TIMEOUT	
MT_RET_INTERNAL_ERROR	Internal DLL error

Events

None

MTIpDChannelEventFilteringStatus()

Description

MTIpDChannelEventFilteringStatus() returns the status of the DChannel event filtering feature when tapping with the SmartWORKS IPX. This operation is consistent with the NGX board. When enabled, DChannel event filtering removes duplicate events which would otherwise be reported to the user application. This function is designed for synchronous operation only.

Prototype

```
int MTIpDChannelEventFilteringStatus(  
    const BOARD          nBoard,  
    const USHORT         nProtocolID,  
    BOOLEAN * const      pOn  
);
```

Parameters

nBoard	Board number
nProtocolID	protocol ID for the signaling protocol: MT_IP_CISCO = 1 MT_IP_AVAYA_H323 = 2 MT_IP_NORTEL_UNISTIM = 4 MT_IP_ERICSSON_H323 = 5
pOn	Pointer to the status returned to the user application: TRUE = enabled FALSE = disabled (default)

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_BOARD	Board number is out of range
MT_RET_BOARD_NOT_OPENED	Board is not opened for access
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_NOT_AVAILABLE	Function not available on this board
MT_RET_TIMEOUT	
MT_RET_INTERNAL_ERROR	Internal DLL error

Events

None

MTIpCCEventControl()

Description

MTIpCCEventControl() allows an application to enable (start) or disable (stop) call control event reporting for an IP signaling protocol on the specified board. Prior to calling this function, the signaling stack for the specified protocol must first be enabled.

This function is designed for synchronous operation only.

Prototype

int MTIpCCEventControl (

```
const BOARD          nBoard,  
const USHORT         nProtocolID,  
const BOOLEAN        fOn  
);
```

Parameters

nBoard	Board number
nProtocolID	protocol ID for the signaling protocol: MT_IP_CISCO = 1 MT_IP_AVAYA_H323 = 2 MT_IP_NORTEL_UNISTIM = 4 MT_IP_ERICSSON_H323 = 5
fOn	Desired setting of operation; TRUE = enabled, FALSE = disabled (default)

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_BOARD	Board number is out of range
MT_RET_BOARD_NOT_OPENED	Board is not opened for access
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_NOT_AVAILABLE	Function not available on this board

Events

None

MTIpCCEventStatus()

Description

MTIpCCEventStatus() retrieves the current operational status of call control event reporting for the specified signaling protocol. Upon successful completion, pStatus will be set to ENABLE if call control event reporting is on or DISABLE if call control event recording is off.

This function is designed for synchronous operation only.

Prototype

```
int MTIpCCEventStatus (  
    const BOARD          nBoard,  
    const USHORT         nProtocolID,  
    BOOLEAN * const      pOn  
);
```

Parameters

nBoard	Board number
nProtocolID	protocol ID for the signaling protocol: MT_IP_CISCO = 1 MT_IP_AVAYA_H323 = 2 MT_IP_NORTEL_UNISTIM = 4 MT_IP_ERICSSON_H323 = 5
pOn	Variable that returns the current operational setting; TRUE = enabled, FALSE = disabled

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_BOARD	Board number is out of range
MT_RET_BOARD_NOT_OPENED	Board is not opened for access
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_NOT_AVAILABLE	Function not available on this board

Events

None

MTIpLoadLicense()

Description

MTIpLoadLicense() allows an application to load a media forwarding license via their application.

NOTE: The IPX-C board default, without any license file loaded, is 30 sessions.

Prototype

int MTIpLoadLicense(

```
const BOARD          nBoard,  
const FILE           *pFileName  
);
```

Parameters

nBoard	Board number
*pFileName	pointer to the file name

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_PARAM	Invalid parameter
MT_RET_NOT_AVAILABLE	Function not available on this board
MT_RET_LICENSE_FILE_CORRUPT	License file is corrupt or invalid
MT_RET_INVALID_FILE_NAME	Invalid filename (i.e. License File does not have valid extension, incorrect filename, or has exceeded maximum file size limit-256b)
MT_RET_LICENSE_INVALID_SERIAL_NUMBER	Serial number associated with this license file does not match the serial number of this board
MT_RET_INVALID_PRODUCT_ID	Invalid product ID
MT_RET_INTERNAL_ERROR	Internal DLL error

Events

None

MTIpGetLicenseInfo()

Description

Retrieve license file information from the board.

Prototype

int MTIpGetLicenseInfo(

const BOARDnBoard,

const PMT_IP_LICENSE_INFOpLicenseInfo

);

Parameters

nBoardBoard number

pLicenseInfopointer to the PMT_IP_LICENSE_INFO data structure

Definitions

LIC_STRING_LEN20

LIC_LONG_STRING_LEN256

LICENSE_KEY_SIZE2200

Structure

Type	Name	Description
UCHAR	LicBoardSerialNumber[LIC_STRING_LEN]	Serial number of the board
ULONG	LicBoardMaxSessions	Maximum sessions supported by board
UCHAR	LicBoardProductID[LIC_STRING_LEN]	Product ID of the board (PCB assembly code of the board.

Return Codes

MT_RET_OKNo error

MT_RET_INVALID_PARAMInvalid parameter

MT_RET_NOT_AVAILABLEFunction not available on this board

MT_RET_INTERNAL_ERRORInternal DLL error

Events

None

STATION MANAGER APIs

MTIpGetStationCount()

Description

Returns the total number of station endpoints presently detected by the board.

A station device is typically a phone set, however other station devices may be detected on the network. For example computers running soft phone applications and other portable devices running compatible software applications.

This function is designed for synchronous operation only.

Prototype

```
int MTIpGetStationCount (  
    const BOARD          nBoard,  
    PULONG const         pCount  
);
```

Parameters

nBoard	Board number
pCount	Pointer to a variable that returns the total phone count

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_BOARD	Board number is out of range
MT_RET_BOARD_NOT_OPENED	Board is not opened for access
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_NOT_AVAILABLE	Function not available on this board

Events

None

MTIpGetStationList()

Description

Returns the count of the active stations with their Protocol IDs and Station IDs.

When using this function the *pCount* parameter needs to be set with number of array elements in the *pParams* array before calling the function. Upon completion, the DLL sets this number to the number of stations for which valid data was copied. If

MTIpGetStationList() is invoked with a count of 120, but there are only 20 stations known to the IPX, the function returns MT_RET_OK, *pCount* will be set to 20, and the first 20 elements in the *pParams* array will contain valid data.

To obtain specific information about each station on the network, users must invoke **MTIpGetStationParams()**.

This function is designed for synchronous operation only.

Prototype

int MTIpGetStationList (

```
const BOARD          nBoard,
const PULONG         pCount,
const PMT_IP_STATION pParams,
PULONG const        pLength
);
```

Parameters

nBoard	Board number
pCount	The number of array elements contained in pParams
pParams	Pointer to an array of PMT_IP_STATION structures
pLength	Pointer to the size of a PMT_IP_STATION structure

Structure

TABLE 53: PMT_IP_STATION

Type	Name	Description
ULONG	StationID	The unique Station ID assigned to this endpoint by the IPX
USHORT	ProtocolID	Protocol ID for the signaling protocol: MT_IP_AVAYA = 2 MT_IP_CISCO_SKINNY = 1

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_BOARD	Board number is out of range
MT_RET_BOARD_NOT_OPENED	Board is not opened for access

MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_NOT_AVAILABLE	Function not available on this board
MT_RET_DATA_TRUNCATED	Data copied but truncated; the buffer size set by the user application is not large enough for this API request. <i>FlagDataTruncated</i> field of the MTSYS_CONFIG or the MTSYS_CONFIGURATION data structure must be enabled (1) to receive this return code, otherwise MT_RET_OK is used, and the truncated data is passed to the user application

Events

None

MTGetStationMacAddress()

Description

Returns the MAC address of the station.

This function is designed for synchronous operation only.

Developer's NOTE: This function does **NOT** return an array of characters that can be printed out like a regular string. This function returns an array where each character corresponds to a particular value of the MAC.

~ Refer to the example code below.

NOTE: The IPX returns only the MAC address visible on the network relative to the tapped position. Many networks use switches or routers which strip the endpoint's MAC address and sublant another one. The user *must* verify that the station's endpoint is visible at the tap site.

Prototype

int MTIpGetStationMacAddress (

```
const BOARD          nBoard,  
const USHORT         nStation,  
PCHAR const          pMacAddress,  
PULONG const         pLength  
);
```

Parameters

nBoard	Board number
nStation	Station ID
pMacAddress	pointer to the MAC address returned to user
pLength	length of the pointer of the Mac address

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_BOARD	Board number is out of range
MT_RET_BOARD_NOT_OPENED	Board is not opened for access
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_NOT_AVAILABLE	Function not available on this board

MT_RET_DATA_TRUNCATED	Data copied but truncated; the buffer size set by the user application is not large enough for this API request. <i>FlagDataTruncated</i> field of the MTSYS_CONFIG or the MTSYS_CONFIGURATION data structure must be enabled (1) to receive this return code, otherwise MT_RET_OK is used, and the truncated data is passed to the user application
MT_RET_NO_MEM	the SDK was not able to allocate enough memory for the requested buffer size
MT_TIMEOUT	the board did not respond to the request in time

Events

None

Example Code

```

if(pEvent->EventCode==174) //EVT_STATION_ADDED
{
    stationID = pEvent->XtrInfo & 0xFFFF;
    printf("station %d added.\n", stationID);
    //print out mac address
    CHAR MAC[6];
    ULONG size_MAC = sizeof(MAC);
    MT_RESULT MacResult =
MTGetStationMacAddress(boardNum,stationID,&MAC[0],&size_MA
C);

    if(MacResult == MT_RET_OK)
    {
        char str[18];
        ULONG size_STR = sizeof(str);
        memset(str,0,size_STR);
        printf("Mac = ");

        //convert the MAC address into a string to be printed out
        //each character of MAC must be treated individually -
        if MAC[0] = 0x00,
        //we need the resulting string to be str[0] = 0, str[1] = 0.
        //include the ':' dividers between the mac sections - str[2] = ':'.
        //repeat above for each character in MAC.

        char temp;
        int j = 0;
        for(int i=0;i<size_MAC;i++)

```

```

        {
            if(j>=size_STR)
                break;
            //use itoa to convert the number into a character, base 16.
            itoa((MAC[i] & 0x00F0),&temp,16);
            str[j] = temp;
            itoa((MAC[i] & 0x000F),&temp,16);
            str[j+1] = temp;
            //do not add ':' at the end of the MAC
            if(i<(size_MAC-1))
                str[j+2] = ':';
            j = j+3;
        }
        printf("%s\n",str);
    }
}

```

After getting a station added event, here is the output:

```

station 0 added.
Mac   = 00:a0:de:37:86:3c

```

MTGetStationMacAddressEx()

Description

Returns the MAC address of the station.

This function is designed for synchronous operation only.

This function returns the MAC address presented in both Rx and Tx traffic so that the user application may select the correct value depending on their tap location.

Developer's NOTE: This function does **NOT** return an array of characters that can be printed out like a regular string. This function returns an array where each character corresponds to a particular value of the MAC.

~ Refer to the example code as presented in the section above where ***MTGetStationMacAddress()*** is documented.

NOTE: The IPX returns only the MAC address visible on the network relative to the tapped position. Many networks use switches or routers which strip the endpoint's MAC address and sublant another one. The user *must* verify that the station's endpoint is visible at the tap site.

Prototype

```
int MTIpGetStationMacAddressEx (  
    const BOARD          nBoard,  
    const USHORT         nStation,  
    const PMT_ST_MAC_ADDRESS nMacAddress,  
    ULONG const          pLength  
);
```

Parameters

nBoard	Board number
nStation	Station ID
nMacAddress	pointer to the PMT_ST_MAC_ADDRESS data structure
pLength	length of the pointer of the Mac address

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_BOARD	Board number is out of range
MT_RET_BOARD_NOT_OPENED	Board is not opened for access
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_NOT_AVAILABLE	Function not available on this board

MT_RET_DATA_TRUNCATED	Data copied but truncated; the buffer size set by the user application is not large enough for this API request. <i>FlagDataTruncated</i> field of the MTSYS_CONFIG or the MTSYS_CONFIGURATION data structure must be enabled (1) to receive this return code, otherwise MT_RET_OK is used, and the truncated data is passed to the user application
MT_RET_NO_MEM	the SDK was not able to allocate enough memory for the requested buffer size
MT_TIMEOUT	the board did not respond to the request in time

MTIpGetStationParams()

Description

Retrieves the current information for the specified station.

This function is designed for synchronous operation only.

Prototype

```
int MTIpGetStationParams(  
    const BOARD          nBoard,  
    const USHORT         nStation,  
    const PMT_IP_STATION_PARAMS pParams,  
    PULONG const         pLength  
);
```

Parameters

nBoard	Board number
nStation	The Station ID of the VoIP endpoint
pParams	Pointer to a protocol specific station parameter structure.
pLength	Length of the PMT_IP_STATION_PARAMS structure

Cisco Parameters:

Protocol Type - the type of protocol (MT_TCP or MT_UDP) used to transmit signaling data

Port - the port used by the Cisco Call Manager for listening to signaling requests from the VoIP endpoints

Avaya Parameters:

H225CS - the protocol type (MT_TCP or MT_UDP) used to transmit signaling data, along with the port on the IP PBX used for listening to signaling requests

H225RAS - the H225 Registration Admission and Status IP protocol type and port number. If RAS is not enabled on the network, these fields must be set to 'NULL'. **NOTE:** RAS is not implemented in the Beta 2 release.

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_BOARD	Board number is out of range
MT_RET_BOARD_NOT_OPENED	Board is not opened for access
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_NOT_AVAILABLE	Function not available on this board

MT_RET_DATA_TRUNCATED

Data copied but truncated; the buffer size set by the user application is not large enough for this API request. *FlagDataTruncated* field of the MTSYS_CONFIG or the MTSYS_CONFIGURATION data structure must be enabled (1) to receive this return code, otherwise MT_RET_OK is used, and the truncated data is passed to the user application

EventsNone

MTGetExtension()

Description

The **MTGetExtension()** retrieves the extension number of the specified line appearance on the specified station.

The function will always ensure that the string placed into *pExtension* is NULL terminated. In addition, the function will modify the *pLength* argument before returning to the calling application. The meaning of the value of *pLength* is:

- If the function returns MT_RET_OK, *pLength* will be set to the size of the string copied into the buffer pointed to by *pExtension* or ZERO. A value of ZERO indicates that the extension number is not known to the NetTAP board at this time. (Note, *pExtension* will be set to the NULL string in this case).

pLength includes the NULL terminator; for example for the string "1234" *pLength* would be equal to 5.

NOTE: This function has only been implemented on Avaya and Cisco networks. This API is synchronous only.

Prototype

int MTGetExtension (

const BOARD	nBoard,
const USHORT	nStation,
const ULONG	nLine,
PCHAR const	pExtension,
PULONG const	pLength
);	

Parameters

nBoard	Board number
nStationID	Station ID
nLineID	Target Line ID
pExtension	Pointer to buffer to receive the extension number
pLength	Pointer to buffer length

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_BOARD	Board number is out of range
MT_RET_BOARD_NOT_OPENED	Board is not opened for access
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_NOT_AVAILABLE	Function not available on this board

MT_RET_DATA_TRUNCATED	Data copied but truncated; the buffer size set by the user application is not large enough for this API request. <i>FlagDataTruncated</i> field of the MTSYS_CONFIG or the MTSYS_CONFIGURATION data structure must be enabled (1) to receive this return code, otherwise MT_RET_OK is used, and the truncated data is passed to the user application
MT_RET_NO_MEM	the SDK was not able to allocate enough memory for the requested buffer size
MT_TIMEOUT	the board did not respond to the request in time

Events

None

MTIpGetStationCallStatistics()

Description

Retrieves call statistics for the specified Ip station. This function reports statistics from the time of board startup. Users can invoke the function *MTIpClearStationCallStatistics()* to reset the counters.

Prototype

int MTIpGetStationCallStatistics (

const BOARDnBoard,

const USHORTnStation,

const PMT_IP_STATION_CALL_STATS pStats,

PULONG const pLength

);

Parameters

nBoard	Board number
nStation	Station ID is the unique number assigned to the station by the IPX station manager
pStats	Pointer to the PMT_IP_STATION_CALL_STATS structure
pLength	Length of the station stats structure

Structure

Type	Name	Description
USHORT	StationID	Station ID. Station ID is the unique number assigned to the station by the IPX station manager
ULONG	InboundCallCount	Total number of inbound calls to this phone (this count includes all calls, even calls that were rejected)
ULONG	OutboundCallCount	Total number of outbound calls initiated by this station. This count includes calls that may not have been connected.
ULONG	Average Call Duration	The average duration of all calls in 10 ms units. This data applies only to calls in the connected state.
ULONG	PeakCallDuration	The duration of the longest call associated with this station. In units of 10 ms.This data only applies to calls in the connected state.
ULONG	CallsConnected	The total number of connected calls.
ULONG	CallsAbandoned	The total number of calls that were abandoned
ULONG	CallsRejected	The total number of calls rejected by this station

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_BOARD	Board number is out of range
MT_RET_BOARD_NOT_OPENED	Board is not opened for access
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_NOT_AVAILABLE	Function not available on this board
MT_RET_DATA_TRUNCATED	Data copied but truncated; the buffer size set by the user application is not large enough for this API request. <i>FlagDataTruncated</i> field of the MTSYS_CONFIG or the MTSYS_CONFIGURATION data structure must be enabled (1) to receive this return code, otherwise MT_RET_OK is used, and the truncated data is passed to the user application

Events

None

MTIpClearStationCallStatistics()

Description

Allows an application to reset, or clear, the statistics collected for the specified station.

This function is designed for synchronous operation only.

Prototype

```
int MTIpClearStationCallStatistics (  
    const BOARD          nBoard,  
    const USHORT          nStation  
);
```

Parameters

nBoard	Board number
nStation	Station ID is the unique number assigned to the station by the IPX phone manager

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_BOARD	Board number is out of range
MT_RET_BOARD_NOT_OPENED	Board is not opened for access
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_NOT_AVAILABLE	Function not available on this board

Events

None

NETWORK STATISTICS

MTIpGetPassiveNetworkTransportStatistics()

Description

Retrieves the current statistical data maintained by the board, relative transport layer statistics .

This function reports statistics from the time of board startup. Users can invoke the function

MTIpClearPassiveNetworkTransportStatistics() to reset the counters.

This function is designed for synchronous operation only.

Prototype

int MTIpGetPassiveNetworkTransportStatistics (

```
const BOARD          nBoard,  
const PMT_IP_TRANSPORT_STATS  pStats,  
PULONG const        pLength  
);
```

Parameters

nBoard	Board number
pStats	Pointer to the PMT_IP_TRANSPORT_STATS structure
pLength	Length of the network statistics structure

Data Structure

NOTE: The following TCP statistics relate to VoIP packets that match the protocol enabled by the user on the IPX. All other packets are ignored by the IPX and not included in these statistics.

Type	Name	Description
ULONG	IPChecksumErrors	Total number of IP packets received with checksum error.
ULONG	TCPChecksumErrors	Total number of recieved packets with TCP checksum error.
ULONG	TCPOutofOrderPackets	Total number of packets recieved that arrived out of order
ULONG	TCPDroppedSegments	Total number of packets dropped
ULONG	TCPTotalPacketsReceived	Total number of packets received
ULONG	TCPTotalPacketsProcessed	Total number of TCP packets processed by the IPX after non-VoIP packets are filtered by the IPX.
ULONG	UDPChecksumErrors	Total number of packets received that were dropped by the IPX due to checksum error
ULONG	UDPTotalPacketsRecieved	Total number of UDP packets received

Type	Name	Description
ULONG	UDPTotalPacketsProcessed	Total number of UDP packets processed by the IPX after non-VoIP packets are filtered by the IPX.

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_BOARD	Board number is out of range
MT_RET_BOARD_NOT_OPENED	Board is not opened for access
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_NOT_AVAILABLE	Function not available on this board
MT_RET_DATA_TRUNCATED	Data copied but truncated; the buffer size set by the user application is not large enough for this API request. <i>FlagDataTruncated</i> field of the MTSYS_CONFIG or the MTSYS_CONFIGURATION data structure must be enabled (1) to receive this return code, otherwise MT_RET_OK is used, and the truncated data is passed to the user application

Events

None

MTIpClearPassiveNetworkTransportStatistics()

Description

Allows an application to reset, or clear, the statistics collected for the specified board.

This function is designed for synchronous operation only.

Prototype

```
int MTIpClearPassiveNetworkTransportStatistics (  
    const BOARD          nBoard,  
);
```

Parameters

nBoard	Board number
nStation	Station ID is the unique number assigned to the station by the IPX phone manager

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_BOARD	Board number is out of range
MT_RET_BOARD_NOT_OPENED	Board is not opened for access
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_NOT_AVAILABLE	Function not available on this board

Events

None

TARGET RECORDING APIs

MTStationMappingControl()

Description

Used to enable the target recording feature of the IPX-C/HPX. This allows the user application to identify, or target, endpoints for recording.

Prototype

int MTStationMappingControl (

```
const BOARD          nBoard,  
const BOOLEAN        fOn  
);
```

Parameters

nBoard	Board number
fOn	TRUE = enabled, FALSE = disabled

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_BOARD	Board number is out of range
MT_RET_BOARD_NOT_OPENED	Board is not opened for access
MT_RET_NOT_AVAILABLE	Function not available on this board
MT_RET_BUSY	An IP protocol is already enabled on the board
MT_RET_BOARD_TIMEOUT	Request has timed out without effecting change

Events

None

MTStationMappingStatus()

Description

Returns the current status of the target recording, or mapping feature.

Prototype

int MTStationMappingStatus (

```
const BOARD          nBoard,  
BOOLEAN * const      pOn  
);
```

Parameters

nBoard	Board number
pOn	pointer to the status, TRUE = enabled, FALSE = disabled

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_BOARD	Board number is out of range
MT_RET_BOARD_NOT_OPENED	Board is not opened for access
MT_RET_NOT_AVAILABLE	Function not available on this board
MT_RET_BUSY	An IP protocol is already enabled on the board
MT_RET_BOARD_TIMEOUT	Request has timed out without effecting change

Events

None

MTAddStationMapping()

Description

Used to add a new target to the station mapping list. Only stations that have been “targeted” by the user application are reported as EVT_STATION_ADDED by the IPX-C/HPX when the Station Mapping feature is enabled.

Prototype

int MTAddStationMapping(

const BOARDnBoard,

PMT_STATION_MAPPING_INFOpStationMap,

PULONG constpLength

);

Parameters

nBoardBoard number

pStationMappointer to
PMT_STATION_MAPPING_INFO

pLengthLength of data structure

Structure

Type	Name	Description
USHORT	nStation	Any unique value that is used by the application to identify and label the targeted phone
StationNetworkIdType	NetworkIdType	STATION_NETWORK_ID_TYPE_IPV4_ADDRESS - or - STATION_NETWORK_ID_TYPE_MAC_ADDRESS *see below

* union

{
 ULONG IpV4Address; (if Network Type = IP Address, supply IP address in hex format)
 unsigned char MacAddress[MT_IP_PHY_PORT_ADDRESS_LENGTH]; (if Network type =
MAC address, then supply MAC Address here in string format)
}

Return Codes

MT_RET_OKNo error

MT_RET_BOARD_NOT_OPENEDBoard is not opened for access

MT_RET_NOT_AVAILABLEStation mapping feature is currently not enabled or feature not supported on this board

MT_RET_INVALID_BOARDBoard number is out of range

MT_RET_INVALID_PARAMETERStation ID not a valid number

MT_RET_BOARD_TIMEOUTRequested board failed to return before timing out

MT_RET_QUOTA_EXCEEDEDStation is not added to the mapping list as the quota of the license file has been met

Events

None

MTRemoveStationMapping()

Description

Remove a target phone from the station mapping list.

Prototype

int MTRemoveStationMapping (

```
const BOARD          nBoard,  
USHORT              nStation  
);
```

Parameters

nBoard	Board number
nStation	Station ID

Return Codes

MT_RET_OK	No error
MT_RET_BOARD_NOT_OPENED	Board is not opened for access
MT_RET_INVALID_BOARD	Board number is out of range
MT_RET_NOT_AVAILABLE	Station Mapping feature is currently not enabled, or feature not supported on this board
MT_RET_BOARD_TIMEOUT	Request has timed out without effecting change
MT_RET_INVALID_PARAMETER	Station ID is not recognized as a mapped station

Events

None

MTGetStationMappingInfo()

Description

Returns the current status of the target recording, or mapping feature.

Prototype

int MTGetStationMappingInfo (

```
const BOARD          nBoard,  
PMT_STATION_MAPPING_INFO pStationMap,  
PULONG const         pLength  
);
```

Parameters

nBoard	Board number
pStationMap	Pointer to a structure containing Station ID and value information
pLength	Pointer to size (bytes) of data structure

Return Codes

MT_RET_OK	No error
MT_RET_BOARD_NOT_OPENED	Board is not opened for access
MT_RET_INVALID_BOARD	Board number is out of range
MT_RET_NOT_AVAILABLE	Station Mapping feature is currently not enabled, or feature not available on this board
MT_RET_BOARD_TIMEOUT	Request has timed out without effecting change
MT_RET_INVALID_PARAMETER	Station ID is not recognized as a mapped Station

Events

None

MEDIA SESSION APIs

MTIpCreateMediaSession()

Description

This function may be used if the application is receiving call state information from an external source. The application uses this API to create a media session on the IPX board. The IPX returns a Session ID to the application which then relies on this ID number to control media routing.

This function is designed for synchronous operation only.

NOTE: When creating a media session with RTP timeout enabled, the user application should confirm that the RTP is actually present on the line or the IPX/HPX will terminate the session and delete the SessionID.

Developer's Note

When the user application invokes this API to create a media session on the IPX board, a corresponding **MTIpSetSessionMediaDest()** must also be invoked. None of the general routing rules are applied to this session, and the user must manually set a session based routing rule so that the RTP packets can be forwarded from the IPX board to a recording device.

Prototype

```
int MTIpCreateMediaSession (
    const BOARD          nBoard,
    const PMT_IP_SESSION_PARAMS pParams,
    PULONG const         pLength
    PULONG const         pSessionID
);
```

Parameters

nBoard	Board number
pSessionParams	Pointer to the PMT_IP_SESSION_PARAMS
pLength	Size of the session parameters structure
pSessionID	pointer to the buffer that holds the Session ID returned to the application from the IPX board

Structure

NOTE: This data structure has been changed as of the 3.9 release.

Type	Name	Description
ULONG	SessionID	The Session ID associated with this RTP media connection
USHORT	PrimaryUDPPort	The UDP port used by the phone(on the tapped network) to receive RTP packets *
USHORT	SecondaryUDPPort	The UDP port used by the VoIP endpoint (not on the tapped network to receive RTP packets*

ULONG	PrimaryIPAddress	The IP address of the phone on the tapped network*
ULONG	SecondaryIPAddress	The IP address of the VoIP endpoint that is not on the tapped network*
union {		
ULONG	Codec	Codec of the RTP packets. This field is used for backward compability, being obsoleted
ULONG	PrimaryCodec	Codec used by the primary station when transmitting RTP.
} (end union)		
ULONG	SecondaryCodec	Codec used by the secondary station when transmitting RTP.
ULONG	CallRef	This field is only populated when using the IPX for SIP trunk recording. This value is a call reference number maintained by the IPX to identify the call on the network.

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_BOARD	Board number is out of range
MT_RET_BOARD_NOT_OPENED	Board is not opened for access
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_NOT_AVAILABLE	Function not available on this board
MT_RET_DATA_TRUNCATED	Data copied but truncated; the buffer size set by the user application is not large enough for this API request. <i>FlagDataTruncated</i> field of the MTSYS_CONFIG or the MTSYS_CONFIGURATION data structure must be enabled (1) to receive this return code, otherwise MT_RET_OK is used, and the truncated data is passed to the user application

Events

None

MTIpDeleteMediaSession()

Description

Removes a media session created by the application via the ***MTIpCreateMediaSession()*** function.

This function is designed for synchronous operation only.

Prototype

int MTIpDeleteMediaSession (

```
const BOARD          nBoard,  
const ULONG          nSessionID  
);
```

Parameters

nBoard	Board number
nSessionID	The ID of the session that is to be deleted

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_BOARD	Board number is out of range
MT_RET_BOARD_NOT_OPENED	Board is not opened for access
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_NOT_AVAILABLE	Function not available on this board

Events

None

MTIpGetMediaSessionInfo()

Description

Retrieves the parameters associated with the specified Session ID.

Prototype

int MTIpGetMediaSessionInfo (

const BOARD

const ULONG

const PMT_IP_SESSION_PARAMS

PULONG const

);

nBoard,

nSessionID,

pParams,

pLength

Parameters

nBoard	Board number
nSessionID	Session ID
pParams	Pointer to the PMT_IP_SESSION_PARAMS
pLength	Size of the session parameters structure

Structure

NOTE: This data structure has been changed as of the 3.9 release.

Type	Name	Description
ULONG	SessionID	The Session ID associated with this RTP media connection
USHORT	PrimaryUDPPort	The UDP port used by the phone(on the tapped network) to receive RTP packets *
USHORT	SecondaryUDPPort	The UDP port used by the VoIP endpoint (not on the tapped network to receive RTP packets*
ULONG	PrimaryIPAddress	The IP address of the phone on the tapped network*
ULONG	SecondaryIPAddress	The IP address of the VoIP endpoint that is not on the tapped network*
union {		
ULONG	Codec	Codec of the RTP packets. This field is used for backward compability, being obsoleted
ULONG	PrimaryCodec	Codec used by the primary station when transmitting RTP.
} (end union)		
ULONG	SecondaryCodec	Codec used by the secondary station when transmitting RTP.

ULONG	CallRef	This field is only populated when using the IPX for SIP trunk recording. This value is a call reference number maintained by the IPX to identify the call on the network.

*When a media session is established between two phones on the tapped network, then the Primary IP Address and UDP port are associated to the phone generating the events.

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_BOARD	Board number is out of range
MT_RET_BOARD_NOT_OPENED	Board is not opened for access
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_NOT_AVAILABLE	Function not available on this board
MT_RET_DATA_TRUNCATED	Data copied but truncated; the buffer size set by the user application is not large enough for this API request. <i>FlagDataTruncated</i> field of the MTSYS_CONFIG or the MTSYS_CONFIGURATION data structure must be enabled (1) to receive this return code, otherwise MT_RET_OK is used, and the truncated data is passed to the user application

Events

None

MEDIA FORWARDING APIs

MTIpSetMediaRoutingParams()

Description

The **MTIpSetMediaRoutingParams()** allows an application to create rules to automatically assign media processing resources to new media sessions.

Media forwarding by the IPX is limited by a license key. By default, the IPX is capable of forwarding a maximum of 8 concurrent media sessions. A license key may be purchased, in increments of 60, to support additional media forwarding capabilities. The IPX maintains a count of the total number of media sessions it is currently forwarding. When media routing rules are used, should the number of forwarded sessions matches the maximum allowed by the license key then EVT_QUOTA_EXCEEDED is reported.

Should an error occur where the destination device is unreachable, then the EVT_ETH_DEST_UNREACHABLE event is reported indicating the reason for failure.

Developer's Note

A theoretical explanation of this API is provided in the *IPX Integration Guide* in the chapter that explains application development. User are encouraged to read this section prior to using this API to gain an understanding of the following:

- port numbering (managing port resources)
- managing multiple rules

NOTE: If a media packet matches two rules, then the rule with the lowest index number takes the highest priority.

Prototype

int MTIpSetMediaRoutingParams (

```
const BOARD          nBoard,
const ULONG          nRuleNo
PMT_IP_ROUTING_PARAMS pParams,
const
PULONG const         pLength
);
```

Parameters

nBoard	Board number
nRuleNo	The index number associated with this rule. (1... MT_IP_MAX_MEDIA_ROUTING_RULES) NOTE: The rule with the lowest index number takes the highest priority.
pParams	Pointer to the PMT_IP_ROUTING_PARAMS structures
pLength	Size of the session parameters structure

Structure (MT_IP_ROUTING_PARAMS)

Type	Name	Description
IP_TARGET_OPTS	fTarget	Flag indicating the destination target type. Currently only: NETWORK is supported with this API.
MT_IP_MEDIA_RULE_TYPE	RuleType	The type of the routing rule: ROUTE_BY_NONE = 0, ROUTE_BY_SESSION = 1, ROUTE_BY_STATION = 2, ROUTE_BY_IP = 3
union { MT_IP_ROUTE_BY_SESSION_PARAMS MT_IP_ROUTE_BY_STATION_PARAMS MT_IP_ROUTE_BY_IP_PARAMS }		A union of structures. Each data structure corresponds to a specific rule type.

MT_IP_ROUTE_BY_SESSION_PARAMS

Type	Name	Description
ULONG	MinSession	starting session number
ULONG	MaxSession	ending session number
ULONG	DestPrimaryStationIPAddr	The IP address of the recording device that receives media packets that the primary station is transmitting.
USHORT	DestPrimaryIncOffset	value required for calculating port number
USHORT	DestPrimaryStationBaseID	starting port number used to calculate the receiving port
ULONG	DestSecondaryStationIPAddr	The IP address of the recording device that receives media packets that the secondary station is transmitting.
USHORT	DestSecondaryIncOffset	value required for calculating port number
USHORT	DestSecondaryStationBaseID	starting port number used to calculate the receiving port

MT_IP_ROUTE_BY_STATION_PARAMS

Type	Name	Description
ULONG	MinStation	starting station number
ULONG	MaxStation	ending station number
ULONG	DestPrimaryStationIPAddr	The IP address of the recording device that receives media packets that the primary station is transmitting.
USHORT	DestPrimaryIncOffset	value required for calculating port number
USHORT	DestPrimaryStationBaseID	starting port number used to calculate the receiving port
ULONG	DestSecondaryStationIPAddr	The IP address of the recording device that receives media packets that the secondary station is transmitting.
USHORT	DestSecondaryIncOffset	value required for calculating port number
USHORT	DestSecondaryStationBaseID	starting port number used to calculate the receiving port

MT_IP_ROUTE_BY_IP_ADDRESS

Type	Name	Description
ULONG	MinPrimaryStationIPAddress	starting primary station IP address
ULONG	MaxPrimaryStationIPAddress	ending primary station IP address
ULONG	DestPrimaryStationIPAddr	The IP address of the recording device that receives media packets that the primary station is transmitting.
USHORT	DestPrimaryIncOffset	value required for calculating port number
USHORT	DestPrimaryStationBaseID	starting port number used to calculate the receiving port
ULONG	DestSecondaryStationIPAddr	The IP address of the recording device that receives media packets that the secondary station is transmitting.
USHORT	DestSecondaryIncOffset	value required for calculating port number
USHORT	DestSecondaryStationBaseID	starting port number used to calculate the receiving port

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_BOARD	Board number is out of range
MT_RET_BOARD_NOT_OPENED	Board is not opened for access
MT_RET_INVALID_PARAM	Invalid argument passed to function (this is returned if the value of nRuleNo exceeds the limit defined MT_IP_MAX_MEDIA_ROUTING_RULES.
MT_RET_NOT_AVAILABLE	Function not available on this board
MT_RET_DATA_TRUNCATED	Data copied but truncated; the buffer size set by the user application is not large enough for this API request. <i>Flag-DataTruncated</i> field of the MTSYS_CONFIG or the MTSYS_CONFIGURATION data structure must be enabled (1) to receive this return code, otherwise MT_RET_OK is used, and the truncated data is passed to the user application

Events

EVT_ETH_DEST_UNREACHABLE

MTIpGetMediaRoutingParams()

Description

The **MTIpGetMediaRoutingParams()** allows an application to retrieve media routing rules set for a specified rule index number.

Prototype

int MTIpGetMediaRoutingParams (

```
const BOARD          nBoard,  
const ULONG          nRuleNo,  
PMT_IP_ROUTING_PARAMS const pParams,  
PULONG const         pLength  
);
```

Parameters

nBoard	Board number
nRuleNo	The rule index number
pParams	Pointer to the PMT_IP_ROUTING_PARAMS structure
pLength	Size of the session parameters structure

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_BOARD	Board number is out of range
MT_RET_BOARD_NOT_OPENED	Board is not opened for access
MT_RET_INVALID_PARAM	Invalid argument passed to function (this is returned if the value of nCount exceeds the limit defined MT_IP_ADDR_MAX.
MT_RET_NOT_AVAILABLE	Function not available on this board
MT_RET_DATA_TRUNCATED	Data copied but truncated; the buffer size set by the user application is not large enough for this API request. <i>FlagDataTruncated</i> field of the MTSYS_CONFIG or the MTSYS_CONFIGURATION data structure must be enabled (1) to receive this return code, otherwise MT_RET_OK is used, and the truncated data is passed to the user application

Events

None

MTIpSetSessionMediaDest()

Description

The **MTIpSetSessionMediaDest()** allows an application to set rules for forwarding media packets associated with a specified Session ID. Once the Session ID is no longer in use, then this rule no longer applies.

Media forwarding by the IPX is limited by a license key. By default, the IPX is capable of forwarding a maximum of 8 concurrent media sessions. A license key may be purchased, in increments of 60, to support additional media forwarding capabilities. The IPX maintains a count of the total number of media sessions it is currently forwarding. Should this function be invoked when the number of forwarded sessions matches the maximum allowed by the license key then this function returns MT_RET_QUOTA_EXCEEDED.

Should an error occur where the destination device is unreachable, then the EVT_ETH_DEST_UNREACHABLE event is reported indicating the reason for failure.

Stopping Media Forwarding

When the event EVT_MEDIA_SESSION_STOPPED is reported, the IPX automatically stops forwarding media, and returns the license used for this forwarding rule. In the scenario where the IPX does not report EVT_MEDIA_SESSION_STOPPED, the user application can invoke this function setting all MT_IP_DEST parameters to '0'. This stops the media forwarding process and returns the license to the user application.

Developer's Note

1. This rule overrides all general rules defined on the board while the specified session is active
2. If the user application creates a Session using **MTIpCreateMediaSession()**, then this API must be invoked to set a destination. General media forwarding rules do not apply to packets associated with a Session ID created by the user.
3. Incoming upstream and downstream packets are not summed by the IPX. As a result, packets for a single session are forwarded to two unique ports on the destination device.

Prototype

```
int MTIpSetSessionMediaDest (  
    const BOARD          nBoard,  
    const ULONG          nSessionID,  
    PMT_IP_SESSION_DEST const pParams,  
    PULONG const         pLength  
);
```

Parameters

nBoard	Board number
nSessionID	Session ID

fTarget	Flag indicating the target type. May be one of the following: NETWORK HOST (future release)
pParams	Pointer to the PMT_IP_SESSION_DEST structure
pLength	Size of the pParams structure

Structure

Type	Name	Description
IP_TARGET_OPTS	fTarget	Flag indicating the destination target type. May be one of the following: NETWORK HOST (future releases)
ULONG	DestPrimaryStationIPAddr	The IP address of the recording device that receives media packets that the primary station is transmitting.
USHORT	DestPrimaryStationBaseID	When fTarget = NETWORK - the port on the recording device where packets associated with the primary station are sent
ULONG	DestSecondaryStationIPAddr	The IP address of the recording device that receives media packets that the secondary station is transmitting.
USHORT	DestSecondaryStationBaseID	When fTarget = NETWORK - the port on the recording device where packets associated with the secondary station are sent

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_BOARD	Board number is out of range
MT_RET_BOARD_NOT_OPENED	Board is not opened for access
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_NOT_AVAILABLE	Function not available on this board
MT_RET_DATA_TRUNCATED	Data copied but truncated; the buffer size set by the user application is not large enough for this API request. <i>FlagDataTruncated</i> field of the MTSYS_CONFIG or the MTSYS_CONFIGURATION data structure must be enabled (1) to receive this return code, otherwise MT_RET_OK is used, and the truncated data is passed to the user application
MT_RET_QUOTA_EXCEEDED	The maximum number of sessions allowed by the license key has been met.

Events

EVT_ETH_DEST_UNREACHABLE

MTIpGetSessionMediaDest()

Description

The **MTIpGetSessionMediaDest()** retrieves the rule defined for forwarding media packets associated with a specified Session ID.

Prototype

int MTIpGetSessionMediaDest (

```
const BOARD          nBoard,  
const ULONG          nSessionID,  
PMT_IP_SESSION_DEST const pParams,  
PULONG const         pLength  
);
```

Parameters

nBoard	Board number
nSessionID	Session ID
pParams	Pointer to the PMT_IP_SESSION_DEST structures
pLength	Pointer to a variable set to the size of a MT_IP_SESSION_DEST structure

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_BOARD	Board number is out of range
MT_RET_BOARD_NOT_OPENED	Board is not opened for access
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_NOT_AVAILABLE	Function not available on this board
MT_RET_DATA_TRUNCATED	Data copied but truncated; the buffer size set by the user application is not large enough for this API request. <i>FlagDataTruncated</i> field of the MTSYS_CONFIG or the MTSYS_CONFIGURATION data structure must be enabled (1) to receive this return code, otherwise MT_RET_OK is used, and the truncated data is passed to the user application

Events

None

PORT STATUS AND STATISTICS APIs

MTIpGetNetworkPortStatus()

Description

Retrieves the current statistical data maintained by the board, relative transport layer statistics.

This function is designed for synchronous operation only.

Developer's Note

The SmartWORKS SDK also provides asynchronous events should a link go down and a port is no longer connected to the network. Refer to the *Events* chapter where EVT_PHY_LINK_DOWN, and EVT_PHY_LINK_UP are defined.

Prototype

int MTIpGetNetworkPortStatus (

```
const BOARD          nBoard,  
const USHORT         nNetworkInterface,  
const PMT_IP_PORT_STATUS pStats,  
PULONG              pLength  
);
```

Parameters

nBoard	Board number
nNetworkInterface	ID of the port interface (0-2)
pStats	Pointer to the PMT_IP_PORT_STATUS structure
pLength	Length of the network statistics structure

Data Structure

Type	Name	Description
UCHAR	PhysicalLinkStatus	0=down, 1=up
UCHAR	PhysicalIPAddress	MAC Address of the port: [MT_IP_PHY_PORT_ADDRESS_LENGTH]
ULONG	IPAddress	IP Address of this port
ULONG	SubNetMask	The Subnet mask assigned to this port

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_BOARD	Board number is out of range
MT_RET_BOARD_NOT_OPENED	Board is not opened for access
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_NOT_AVAILABLE	Function not available on this board
MT_RET_DATA_TRUNCATED	Data copied but truncated; the buffer size set by the user application is not large enough for this API request. <i>FlagDataTruncated</i> field of the MTSYS_CONFIG or the MTSYS_CONFIGURATION data structure must be enabled (1) to receive this return code, otherwise MT_RET_OK is used, and the truncated data is passed to the user application

Events

None

MTIpGetNetworkPortStatistics()

Description

Retrieves the current statistical data maintained by the board, for the specified protocol.

This function is designed for synchronous operation only.

NOTE: This API replaces the *MTIpGetNetworkInterfaceStatistics()* function which as never officially released.

Prototype

int MTIpGetNetworkPortStatistics (

```
const BOARD          nBoard,
const USHORT         nNetworkInterface,
const PMT_IP_PORT_STATS pStats,
PULONG const        pLength
);
```

Parameters

nBoard	Board number
nNetworkInterface	ID of the port interface (0-2)
pStats	Pointer to the PMT_IP_PORT_STATS structure
pLength	Length of the network statistics structure

Data Structure

Type	Name	Description
ULONG	PacketsTransmitted	Total number of packets transmitted (active port only)
ULONG	BytesTransmitted	Total number of bytes transmitted (active port only)
ULONG	PacketsReceived	Total number of packets received
ULONG	BytesReceived	Total bytes received
ULONG	TxFifoErrorCount	Number of transmit Fifo Errors detected(active port only)Detected when packet dropped due to full transmit queue
ULONG	RxDroppedPacketCount	Number of dropped packets detected; detected when packets are dropped because the the receiving queue is full
ULONG	RxMalformedErrorCount	Number of malformed packets detected
ULONG	MulticastErrorCount	Number of multicast errors detected
ULONG	CollisionErrorCount	Number of collisions errors detected

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_BOARD	Board number is out of range
MT_RET_BOARD_NOT_OPENED	Board is not opened for access
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_NOT_AVAILABLE	Function not available on this board
MT_RET_INTERNAL_ERROR	Internal DLL error

Events

None

MTIpClearNetworkPortStatistics()

Description

Reset or clear the statistics collected for the specified network port. This function is designed for synchronous operation only.

NOTE: This API replaces the ***MTIpClearNetworkInterfaceStatistics()*** function which as never officially released.

Prototype

int MTIpClearNetworkPortStatistics(

```
const BOARD          nBoard,  
const USHORT         nNetworkInterface,  
);
```

Parameters

nBoard	Board number
nNetworkInterface	ID of the port interface (0-2)

Return Codes

MT_RET_OK	No error
MT_RET_INVALID_BOARD	Board number is out of range
MT_RET_BOARD_NOT_OPENED	Board is not opened for access
MT_RET_INVALID_PARAM	Invalid argument passed to function
MT_RET_NOT_AVAILABLE	Function not available on this board
MT_RET_INTERNAL_ERROR	Internal DLL error

Events

None

Chapter 3

Event Code Library

Event Codes

Event codes can be found in the EventCode field of the [MT_EVENT](#) structure. They are of the [EVT_ . . .](#) type and are defined in NtiEvent.h. In the following, the type has the X/Y format where X is an event and Y is an event type.

NOTE: All event codes are returned to the user application as a hex value. To retrieve a text description use the API ***MTGetEventCodeDescription()***.

A block has been reserved for user defined event codes. Events can be created using the hex value of 0x8000 through 0x8fff. Application developers can define event codes and put them in the appropriate event queues using ***MTPutChannelEvent()*** and ***MTPutBoardEvent()***.

All event codes are defined below:

Board Type

DT - SmartWORKS DT

DP - SmartWORKS DP

NGX - SmartWORKS NGX

LD - SmartWORKS LD

PCM - SmartWORKS PCM

IPX - SmartWORKS IPX

HPX - SmartWORK HPX

Event Type

B - Board event

C - Channel event

S - System event

R - Regular, non-terminating event

T - Terminating event

M - Monitor event

MC - Monitor cadence event

1 - This event could indicate a possible interrupt or address conflict, or true technical failure.

2 - End of File (EVT_EOF or EVT_MAXBYTES) is also reported during [MTPlayIndex\(\)](#) when all segments are played.

3 - For information purposes only and will occur in an idle state.

4 - These events may be enabled or disabled with the function [MTSetMoni\(\)](#).

When a function terminates with the [EVT_CPM_STOP](#) code, the SubReason parameter of the [MT_EVENT](#) block indicates the precise reason.

SMARTWORKS EVENT CODES

Board	Hex	Dec	Event	Type	Description
All	0x01	1	EVT_MAXDIGITS	C/T	Maximum DTMF digits received
All	0x02	2	EVT_TERMDIGIT	C/T	Terminating DTMF digit received
All	0x03	3	EVT_STOP	C/T	User stopped
All	0x04	4	EVT_SYS_ERROR	C/T	System response error
All	0x05	5	EVT_MAXBYTES	C/T	Max bytes reached on play or record
All	0x06	6	EVT_ERROR	C/T (1)	Hardware failure
All	0x07	7	EVT_MAXTIME	C/T	Max time reached
AT, LD, NGX	0x08	8	EVT_OFFHOOK	C	Offhook complete
All	0x09	9	EVT_DIALED	C/T	Dialing complete
All	0x0A	10	EVT_MAX_SILENCE	C/R/ M	Maximum silence received
All	0x0B	11	EVT_EOF	C/T (2)	EOF reached on playback
AT, LD	0x0C	12	EVT_LOOP_STOP	C/T	Terminate by drop in loop current
All	0x0D	13	EVT_DISK_FULL	C/T	Disk full
AT, LD, NGX	0x0E	14	EVT_ONHOOK	C	Onhook complete
All	0x0F	15	EVT_DISK_WRITE_ERROR	C/T	File write error
All	0x10	16	EVT_BOARD_PANIC_ERROR	/B/R	Board Panic error
All	0x11	17	EVT_TERMSILENCE	C/T	Termination silence reached on play or record
AT, LD, DT	0x12	18	EVT_CPM_STOP	C/T	Call analysis termination
AT, LD	0x13	19	EVT_LOOP_REVERSE	C/R/ M/4	Loop current battery reversal
AT, LD	0x14	20	EVT_LOOP_DROP	C/R (3)	Loop current dropped
AT, PT, LD	0x15	21	EVT_RINGS	C/R	Ring received
All	0x16	22	EVT_MON_ACTIVITY	C/R	Silence off, activity on
All	0x17	23	EVT_MON_SILENCE	C/R	Silence on, activity off
All	0x18	24	EVT_IDLE	C	Channel back in IDLE state
DP, DT, NGX	0x19	25	EVT_FRAME_HEADER_ERROR	C/T	Frame header error, function terminated

SMARTWORKS EVENT CODES (CONTINUED)

Board	Hex	Dec	Event	Type	Description
AT, LD	0x1A	26	EVT_LOOP_ON	C/R	Loop current on event
AT, LD	0x1C	28	EVT_LVOLTAGE_REVERSAL	C/T	Indicates a voltage change of state
All	0x1D	29	EVT_DIGITENABLE	C/T	Terminated by enabled DTMF digit
All	0x1E	30	EVT_INTERDIGIT	C/T	Interdigit delay exceeded
All	0x1F	31	EVT_MAX_ACTIVITY	C/R/ M	Terminated by a max non-silence
LD	0x20	32	EVT_LVOLTAGE_MIDDLE	C/R/ M/4	Loop voltage between ThresholdHigh and ThresholdLow
LD	0x21	33	EVT_LVOLTAGE_ABOVEORBELOW	C/R/ M/4	Loop voltage not in MIDDLE state
LD	0x22	34	EVT_TERMLVOLTAGE_ABOVEORBELOW	C/R/ M/4	Termination on loop voltage state not in MIDDLE
PT	0x20	32	EVT_LVOLTAGE_OFFHOOK (EVT_LVOLTAGE_MIDDLE)	C/R/ M/4	Loop voltage detected
PT	0x21	33	EVT_LVOLTAGE_NOTOFFHOOK (EVT_LVOLTAGE_ABOVEORBELOW)	C/R/ M/4	Loop voltage dropped
PT	0x22	34	EVT_TERMLVOLTAGE_NOTOFFHOOK (EVT_TERMLVOLTAGE_ABOVEORBELOW)	C/T	Termination on loop voltage changed to notoffhook
LD	0x23	35	EVT_LVOLTAGE_ABOVE	C/T	Loop voltage above ThresholdHigh
LD	0x24	36	EVT_LVOLTAGE_BELOW	C/T	Loop voltage below ThresholdLow
All	0x25	37	EVT_SETPAR_DONE	N/A	Parameter updated
None	0x26	38	EVT_WINKDONE	C/R/ T	Wink protocol complete
All	0x27	39	EVT_DIGIT_DROPPED	C/R	Received DTMF digit dropped due to queue full the queue allows a maximum of 256 digits
All	0x28	40	EVT_DIGIT	C/R	Received DTMF digit queued, the queue allows a maximum of 256 digits
AT, LD	0x2B	43	EVT_LCURRENT_REVERSAL	C/R	Loop current reversal
All	0x2F	47	EVT_TONEPERR	C/R	Tone generation template error
All	0x30	48	EVT_TONEP_DONE	C/T	Tone generation complete
All	0x31	49	EVT_MFTONE_DROPPED	C/R	Received MF(R2) tone dropped due to queue full
All	0x32	50	EVT_MFTONE	C/R	Received MF(R2) tone queued

SMARTWORKS EVENT CODES (CONTINUED)

Board	Hex	Dec	Event	Type	Description
PT, LD	0x33	51	EVT_LVOLTAGE_RECORD	C/R	Recording started on detection of loop voltage offhook
All	0x35	53	EVT_VOICE_RECORD	C/R	Recording started on detection of voice
AT, LD	0x36	54	EVT_LOOPON_RECORD	C/R	Recording started on detection of loop current
All	0x37	55	EVT_STREAMIN_WATERMARK	C/R	Data ENCODE streaming watermark reached
All	0x38	56	EVT_STREAMOUT_WATERMARK	C/R	Data DECODE streaming watermark reached
All	0x39	57	EVT_STREAMIN_START	C/R	Data ENCODE streaming started
All	0x3A	58	EVT_STREAMOUT_START	C/R	Data DECODE streaming started
All	0x3B	59	EVT_STREAM_MAXSIL	C/R	Max silence during data streaming
All	0x3C	60	EVT_STREAMIN_STOP	C/T	Data ENCODE streaming stopped
None	0x3D	61	EVT_STREAMIN_COMP	C/R	Input streaming complete
None	0x3E	62	EVT_STREAMOUT_COMP	C/R	Output streaming complete
All	0x3F	63	EVT_STREAMIN_DROPPED	C/R	Input streaming data dropped
All	0x40	64	EVT_STREAMOUT_STOP	C/T	Data DECODE streaming stopped
All	0x41	65	EVT_STREAMOUT_EMPTY	C/R	Output stream buffer empty
All	0x42	66	EVT_STREAMOUT_PAUSED	C/R	Output streaming paused
All	0x43	67	EVT_STREAMOUT_RESUMED	C/R	Output streaming resumed
All	0x45	69	EVT_INDEX_PLAYED	C/R/ 2	Indexed byte length played
All	0x4D	77	EVT_DEVICE_WRITE_ERROR	C/T	User device write error
All	0x4E	78	EVT_DEVICE_READ_ERROR	C/T	User device read error
All	0x4F	79	EVT_DEVICE_SEEK_ERROR	C/T	User device seek error
All	0x50	80	EVT_DSPRESET	C/R	DSP status event
All	0x51	81	EVT_NOT_AVAILABLE	/BC/ R	Functionality not available or busy
All	0x52	82	EVT_NOT_SUPPORTED	/BC/ R	Functionality not supported
All	0x53	83	EVT_NO_EFFECT	/BC/ R	Functionality not necessary (already done)
PCM	0x54	84	EVT_LOSYNC_ALARM	B	Loss of Sync. alarm

SMARTWORKS EVENT CODES (CONTINUED)

Board	Hex	Dec	Event	Type	Description
DP, DT, NGX	0x55	85	EVT_LOS_ALARM	/B/R	Loss Of Signal alarm
DP, DT	0x56	86	EVT_LOF_ALARM	/B/R	Loss Of Framers alarm
DP, DT	0x57	87	EVT_AIS_ALARM	/B/R	Alarm Indication alarm
DP, DT	0x58	88	EVT_YELLOW_ALARM	/B/R	Yellow Alarm
DP, DT	0x59	89	EVT_INCOMING_SIGCHANGE	C/R	Change of Signaling Bits (ABCD) on incoming line
DT	0x5A	90	EVT_OUTGOING_SIGCHANGE	C/R	Change of Signaling Bits (ABCD) on outgoing line
All, except LD 409	0x5B	91	EVT_MVIP_ALARM	/B/R	MVIP error detected
All	0x5C	92	EVT_MASTER_PLL_ALARM	/B/R	Master PLL error detected
DP, DT	0x5D	93	EVT_LOSMF_ALARM	/B/R	E1: Loss Of Signaling Multiframe Alignment alarm
DP, DT	0x5E	94	EVT_LOCRCMF_ALARM	/B/R	E1: Loss Of CRC Multiframe Alignment alarm
DP, DT	0x5F	95	EVT_TS16RAI_ALARM	/B/R	E1: Timeslot 16 Remote Alarm Indication alarm
All	0x66	102	EVT_H100_MASTER_A_ALARM	B	H100 Master A clock error detected
All	0x67	103	EVT_H100_MASTER_B_ALARM	B	H100 Master B clock error detected
DP, NGX	0x6E	110	EVT_CALL_ALERTING EVT_CC_CALL_ALERTING	C	ISDN Call Alerting
DP, NGX	0x6F	111	EVT_CALL_CONNECTED EVT_CC_CALL_CONNECTED	C	ISDN Call Connected
DP, NGX	0x70	112	EVT_CALL_RELEASED EVT_CC_CALL_RELEASED	C	ISDN Call Released
DP, NGX	0x71	113	EVT_CALL_SUSPENDED EVT_CC_CALL_SUSPENDED	C	ISDN Call Suspended
DP, NGX	0x72	114	EVT_CALL_RESUMED EVT_CC_CALL_RESUMED	C	ISDN Call Resumed
DP, NGX	0x73	115	EVT_CALL_HELD EVT_CC_CALL_HELD	C	ISDN Call Held
DP, NGX	0x74	116	EVT_CALL_RETRIEVED EVT_CC_CALL_RETRIEVED	C	ISDN Call retrieved from Hold
DP, NGX	0x75	117	EVT_CALL_ABANDONED EVT_CC_CALL_ABANDONED	C	ISDN Call abandoned by the caller
DP, NGX	0x76	118	EVT_CALL_REJECTED EVT_CC_CALL_REJECTED	C	ISDN Call rejected by destination

SMARTWORKS EVENT CODES (CONTINUED)

Board	Hex	Dec	Event	Type	Description
DP	0X77	119	EVT_CALL_USER_INFO EVT_USER_INFO EVT_CC_USERINFO_IND	C	ISDN user info
DT	0x78	120	EVT_CC_CONNECT_IND	B	Incoming call Indication
DT	0x79	121	EVT_CC_INFO_IND	B	ISDN called party digits
DT	0x7A	122	EVT_CC_PROGRESS_IND	B	ISDN call progress Indication
DT	0x7B	123	EVT_CC_DISC_IND	B	Disconnect Indication
DT	0x7C	124	EVT_CC_DISC_CONF	B	Disconnect Confirmation
DT	0x7D	125	EVT_CC_CONNECT_CONF	B	Connect Confirmation
DT	0x7E	126	EVT_CC_ALERT_IND	B	ISDN Alert Indication
DT	0X7F	127	EVT_CC_FACILITY_IND	B	ISDN Facility Indication
DT	0X80	128	EVT_CC_FACILITY_CONF	B	ISDN Facility Confirmation
DT	0X6D	109	EVT_SUSPEND_CONF	B	ISDN Suspend Confirmation
DT	0X6C	108	EVT_RESUME_CONF	B	ISDN Resume Confirmation
All	0x81	129	EVT_MON_BUSY EVT_MON_BUSY1	C/R (4)	Monitor busy cadence cycle
All	0x82	130	EVT_MON_BUSY_TRUNK EVT_MON_BUSY2	C/R (4)	Monitor trunk busy cycle
All	0x88	136	EVT_MON_DIAL EVT_MON_DIAL1	C/R/ MC (4)	Monitor Dial1 signal
All	0x89	137	EVT_MON_DIAL2	C/R/ MC (4)	Monitor Dial2 signal
All	0x8A		EVT_MON_DIAL3	C/R/ MC (4)	Monitor Dial3 signal
All	0x8E	142	EVT_MON_RINGBACK EVT_MON_RINGBACK1	C/R/ MC (4)	Monitor Ring back
All	0x8F	143	EVT_MON_DRINGBACK EVT_MON_RINGBACK2	C/R/ MC (4)	Monitor Double Ring back
All	0x90	144	EVT_MON_SIT EVT_MON_SIT1	C/R/ MC (4)	Monitor Intercept signal SIT1
All	0x84	132	EVT_MON_SIT2	C/R/ MC (4)	Monitor Intercept signal SIT2

SMARTWORKS EVENT CODES (CONTINUED)

Board	Hex	Dec	Event	Type	Description
All	0x85	133	EVT_MON_SIT3	C/R/ MC (4)	Monitor Intercept signal SIT3
All	0x86	134	EVT_MON_SIT4	C/R/ MC (4)	Monitor Intercept signal SIT4
All	0x87	135	EVT_MON_SIT5	C/R/ MC (4)	Monitor Intercept signal SIT5
All	0x8C	140	EVT_MON_CALLWAITING	C/R/ MC (4)	Monitor Calwaiting signal
All	0x8D	141	EVT_MON_RECEIVEOFF	C/R/ MC (4)	Monitor Receive off signal
All	0x91	145	EVT_MON_FAX EVT_MON_FAX1	C/R/ MC (4)	Monitor Fax1 signal
All	0xAB	171	EVT_MON_FAX2	C/R/ MC (4)	Monitor Fax2 signal
All	0x92	146	EVT_MON_REVERSAL	C/R/ MC (4)	Monitor polarity reversal
All	0x93	147	EVT_MON_VOICE	C/R/ MC (4)	Monitor voice (human or machine)
All	0x94	148	EVT_MON_NOVOICE	C/R/ MC (4)	Monitor loss of voice
All	0x95	149	EVT_MON_MACHINE	C/R/ MC (4)	Monitor answer machine
All	0x96	150	EVT_MON_HUMAN	C/R/ MC (4)	Monitor Human voice
All	0x97	151	EVT_CALLID_DROPPED	C/R/ MC	CallerID detected and dropped due to queue full
All	0x98	152	EVT_CALLID_STOP	C/R/ MC	CallerID end detected
None	0x9F	159	EVT_RECOFFHOOK	C/R	Recording offhook
All	0xA0	160	EVT_REVERSAL	C/R	Loop current reversal

SMARTWORKS EVENT CODES (CONTINUED)

Board	Hex	Dec	Event	Type	Description
All	0xAA	170	EVT_MON_SIGNAL_CYCLE	C/R/ MC (4)	used to count one complete cycle of CPM or Usr signal
All	0xC8	200	EVT_SYS_SYNCTIME_OLD	S	resync time
All	0xC9	201	EVT_SYS_SYNCTIME_NEW	S	resync time
All	0xCA	202	EVT_SYS_BOARD_ADDED	S	new board added to system
All	0xCB	203	EVT_SYS_BOARD_REMOVED	S	-not implemented yet-

D-CHANNEL EVENT CODES AND PBX SUPPORT (NGX ONLY)

A listing of the D-channel events returned to the user per PBX is now available on online support. Refer to the Files and Documents section and look for the NGX PBX Support Matrix.

MT_EVENT

The **EVENT** structure is used to retrieve event information generated by the board. Users have two options for retrieving event information: polling method or callback method. When polling, data is retrieved from the event queue with **MTGetChannelEvent()** or **MTWaitForChannelEvent()**. Another method is to rely on the call back method to populate the **MT_EVENT** structure. Both of these methods are discussed in detail in the section that explains Event Control.

The user application can insert an event into the event queue through **MTPutChannelEvent()** or **MTPutBoardEvent()** with user application specific value added to the UserEvent and UserData fields of this structure. The SmartWORKS API leaves these values unmodified. This structure is declared in NtiEvent.h.

TABLE 54: MT_EVENT

Type	Name	Function
LARGE_INTEGER(WIN32) timeval (Linux)	TimeStamp	Time stamp when the event occurred. In FILETIME format (WIN32) or in gettimeofday() format (Linux)
ULONG	UserStatus	User defined value
ULONG	EventCode	The event code
ULONG	SubReason	Extended event information
ULONG	XtraInfo	Extra information such as termination enable, etc.
ULONG	FuncCode	The function that generated this event. <i>See below for a definition of each FuncCode.</i>
ULONG	Board	Board the event occurred on
ULONG	Channel	Channel the event occurred on
PVOID	PtrBuffer	Pointer to event associated buffer, if any

TABLE 54: MT_EVENT (CONTINUED)

Type	Name	Function
ULONG	DataLength	Size of exercised data of the associated buffer, if any
PVOID	ptrXtraBuffer*	pointer to buffer containing extra information
ULONG	XtraBufferLength*	Length of ptrXBuffer
ULONG	XtraDataLength*	Length of data put into the prtXtraBuffer by the DLL.
ULONG	EventFlag	Flag for XtraBuffer information: Bit 0x00000001 1 - Appl. created this event 0 - DLL created this event Bit 0x00000002: 1 - Appl. allocated the buffer 0 - DLL allocated the buffer Bit 0x00000004 1 - data has been truncated 0 - data has not been truncated

* Used only with release 2.8 or later and the CompilewithSDK flag has been set to a value greater than 0x02086600.

NOTE: All event codes are returned as a hex value. To retrieve a text description use the API `MTGetEventCodeDescription()`.

TIMESTAMP

The time stamp of when the event occurred. It is presented in FILETIME format (WIN32) or gettimeofday() format (Linux). During system startup, the board is synchronized with the system time. The board uses its TDM clock to generate the TimeStamp. The TDM clock may be synchronized with a local oscillator (50 ppm stability), the CT bus, or the T1/E1 clock. The TimeStamp and system time may differ depending on the board's clock source stability and system clock stability.

If an event is inserted by using ***MTPutChannelEvent()*** or ***MTPutBoardEvent()*** then the user application is responsible for initialization of the time stamp.

USERSTATUS

User defines and maintains this field. The API does not use, change or maintain this field. All events generated by the API have this field set to 0. A user-defined event can be inserted with this field set for a special purpose defined by the user application.

EVENTCODE

The event code is returned in `EventCode` and is defined as `EVT_...` defines in `NtiEvent.h`.

SUBREASON

This field supplies additional information for a number of events.

For the `EVT_CPM_STOP` event, this field contains the extended termination code of the type `CPM_MONI_ . . .`

For digit termination events, this field contains the digit that terminated the background function.

For `EVT_CALLID_STOP` event, this field contains the length of the caller ID packet.

For line transition events such as `EVT_LOOP_ON`, `EVT_MON_ACTIVITY`, `EVT_MON_SILENCE`, `EVT_OFFHOOK`, `EVT_ONHOOK`, and `EVT_REVERSAL`, this field contains timing information about the previous state in 1 ms units, with a maximum of 65535 ms.

XTRAINFO

For the `EVT_MFTONE` event, this field contains the source of either primary or secondary stream of the R2 tone.

For the `EVT_CALLID_STOP` event, this field contains the first 4 bytes of the caller ID packet.

FUNCCode

FuncCode is a code assigned to each I/O API. When the `EVENT` structure is used to retrieve events, the name of the FuncCode generating the event is captured in the FuncCode field. The FuncCode was designed to enhance the developer's ability to monitor encoding and decoding actions.

Each FuncCode is defined in the `NtiEvent.h` file.

Since the FuncCode is used to only monitor **encoding** or **decoding** actions, the following is defined:

```
#define ENCODE_BASE    0x1000    // Record function
#define DECODE_BASE    0x2000    // Playback function
```

Next, the following **methods** used to encode/decode data are defined:

```
#define BUFFER_BASE    0x0100    // Buffer I/O
#define FILE_BASE      0x0200    // File I/O
#define DEVICE_BASE    0x0400    // Device I/O
#define STREAM_BASE    0x0800    // Streaming I/O
```

As a result, the following eight(8) FuncCode values can be returned. Each FuncCode value is a combination an **encode/decode** value with a **method** value.

For example:

If data has been **encoded** (0x1000) to a **buffer** (0x0100) the structure returns the hexadecimal value of (0x1100).

The following table illustrates the hexadecimal value of each FuncCode, as well as a value that can be pulled into the developer's GUI. Please note, the FuncCode for Playback of file index is an exception.

Hexadecimal	FuncCode	Description
0x1100	F_RECB	Record to Buffer
0x1200	F_RECF	Record to file
0x1400	F_RECD	Record to device
0x1800	F_RECS	Record to stream buffer
0x2100	F_PLAYB	Playback from buffer
0x2200	F_PLAYF	Playback from file
0x2201	F_PLAYFI	Indexed Playback from file
0x2400	F_PLAYD	Playback from device
0x2800	F_PLAYS	Playback from stream buffer
0x1801	F_RECS_START	Start record to stream buffer
0x2801	F_PLAYS_START	Start playback from stream buffer
0x1001	F_GETDTMFS	Get DTMF digits
0x2001	F_PLAYDTMFS	Play DTMF digits (using MTDialString())
0x2002	F_PLAYTONES	Play tones
0x0000	F_IDLE	Idle
0x2004	F_CallDTMFS	MTCallString()

BOARD

The board where the event occurred.

CHANNEL

The channel where event occurred.

PTRBUFFER

Pointer to data buffer on which the event happened (media control). If this field is populated because of an event that contains extra information (such as a call control event EVT_CC_) then this field contains the pointer to a buffer containing the extra information. **MTReturnEventBuffer()** must be invoked to return the buffer. This is further explained where **MTReturnEventBuffer()** is discussed in the Functional Reference Library.

DATALength

Length of exercised data in buffer when the event happened (22).

PTRXTRABUFFER

As of release 2.8, events with extra information such as call control events (EVT_CC) are handled differently. **NOTE:** To use this method, use **MTSetSystemConfig()** to set the *CompiledWithSDK* field to a value greater than 0x02086600.

When using the callback method to retrieve events, the *ptrXtraBuffer* field provides the pointer to the buffer containing the extra information and the *XtraBufferLength* indicates the data length. When using the polling method: the *ptrXtraBuffer* field must be set by the user application, along with the *XtraBufferLength* field.

MTReturnEventBuffer() should not be used. This is further explained where **MTReturnEventBuffer()** is discussed in the Functional Reference Library.

XTRABUFFERLENGTH

Length of the *ptrXBuffer*. This is further explained where **MTReturnEventBuffer()** is discussed in the Functional Reference Library.

XTRADATALENGTH

Length of the actual data in the *XtraBuffer*. This value does not exceed the value set by *XtraBufferLength*. When the polling method is used, Bit 2 in the *EventFlag* field is used to alert the user application when the actual data length exceeds the allocated *XtraBuffer*.

EVENTFLAG

This field is informational only. All bits are managed by the SmartWORKS DLL.

This field contains two bit values:

Bit 0x000000001

- 1- application created the event (this event was put in queue by the user application using **MTPutChannelEvent()** or **MTPutBoardEvent()**.)
- 0 - the DLL created the event

Bit 0x000000002

- 1- application allocated buffer for extra event information (**MTReturnEventBuffer()** must be used)
- 0 - the SmartWORKS DLL allocated the buffer for extra event information

Bit 0x000000004

- 1 - data has been truncated
 - 0 - data has not been truncated
-

Event Codes Presented Alphabetically

A block has been reserved for user defined event codes. Events can be created using the hex value of 0x8000 through 0x8fff. Application developers can define event codes and put them in the appropriate event queues using ***MTPutChannelEvent()*** and ***MTPutBoardEvent()***.

EVT_ABANDONED

(DChannel event, reported by the NGX and IPX only)

Description: For the EVT_ABANDONED with Definity, the event indicates the far-end has disconnected from a connected call state. This is different from the definition used when monitoring call states on a BRI network: the EVT_CC_CALL_ABANDONED occurs when the calling party disconnects from the call attempt in the alerting state. The Definity **only** provides this message when the far-end hangs up first.

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_ABANDONED	An octet containing the number of the framer reporting the event	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_AIS_ALARM

Description: Alarm Indication Signal (AIS) alarm (T1 and E1 framers) has been detected. This indicates that the remote site is not receiving data. To receive this event, the framer alarm must be enabled using the **MTArmFramerAlarm()** API.

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_AIS_ALARM	An octet containing the number of the framer reporting the event	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_ANSWER_BUTTON_PRESSED

(DChannel event, reported by the NGX and IPX only)

Description: Indicates that the answer button has been pressed

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME for-mat	0-n/a	EVT_ANSWER_BUTTON_PRESSED	0-n/a	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_ANSWER_LIGHT_FASTFLASHING
EVT_ANSWER_LIGHT_VERY_FASTFLASHING
EVT_ANSWER_LIGHT_FLASHING
EVT_ANSWER_LIGHT_OFF
EVT_ANSWER_LIGHT_ON
EVT_ANSWER_LIGHT_QUICKFLASH
EVT_ANSWER_LIGHT_WINK

(DChannel event, reported by the NGX and IPX only)

Purpose:

Indicates that the answer light on the phone is on, off, or flashing.

Description:

Observation of various phone models enabled the AudioCodes lab to categorize phone lights as flashing, fast flashing, very fast flashing or quick flashing. The cadence varies depending on PBX models, and this may also be configured. Our definition of flashing lights stems from the signal passed from the PBX to the phone, and is based on factory default configurations for each model. Please refer to your PBX documentation for more information.

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME for-mat	0-n/a	EVT_ANSWER_LIGHT_	*See Below	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

Sub-Reason(this field is populated only when light is flashing):

NOTE: These may vary depending on phone model and PBX.

Bits 0-7 = Light Number
 Bits 8 = Green
 Bit 9 = Red
 Bit 10 = Amber
 Bits 11-31 = Reserved

EVT_ALERTING_AUTO_ANSWER

(DChannel event, reported by the NGX and IPX only)

Check for call alerting with auto answer enabled

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File- TIME for- mat	0-n/a	EVT_ALERTING_AUTO_ ANSWER	0-n/a	0-n/a	0-n/a	BoardID	Global- Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_AUDIO_CHANGE

(DChannel event, reported by the NGX and IPX only)

Description: Indicates a change in the audio setting on the phone

NOTE: The Nortel Meridian does send the EVT_AUDIO_CHANGE, however results may vary per network environment. On the 2216 and 3905 phones the PBX turns the audio on and doesn't shut it off between calls. This event cannot be used alone to determine recording.

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File- TIME for- mat	0-n/a	EVT_AUDIO_CHANGE	see below*	0-n/a	0-n/a	BoardID	Global- Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

* Bit 0 = Handset transmit (Mic)
 Bit 1 = Handset receive (Spkr)
 Bit 2 = Speaker phone (handsfree) transmit (Mic)
 Bit 3 = Speaker phone (handsfree) receive (Spkr)
 Bits 4-31 = Reserved

~more information is available on the next page~

This field is a 31 bit field. Bits 4-31 are reserved. Transmit and receive are always in respect to the phone's position. Transmit - phone to PBX, and Receive - PBX to phone. The following table lists all possible options. Not all states are observed with each PBX.

TABLE 55: EVT_AUDIO_CHANGE BIT OPTIONS

Device State	SPKR RECV	SPKR TRANS	HDSET RECV	HDSET TRANS (LSB)	HEX VALUE
All devices are off	0	0	0	0	0x0000
	0	0	0	1	0x0001
	0	0	1	0	0x0002
Handset active	0	0	1	1	0x0003
	0	1	0	0	0x0004
	0	1	0	1	0x0005
	0	1	1	0	0x0006
	0	1	1	1	0x0007
	1	0	0	0	0x0008
	1	0	0	1	0x0009
Handset & SPR receive only	1	0	1	0	0x000A
	1	0	1	1	0x000B
	1	1	0	0	0x000C
	1	1	0	1	0x000D
	1	1	1	0	0x000E
All devices are active	1	1	1	1	0x000F

EVT_AUDIO_OFF

(DChannel event, reported by the NGX and IPX only)

Description: Although this event exists in the header file, it is not currently supported by the SmarkWORKS API. Refer to EVT_AUDIO_CHANGE.

EVT_AUDIO_ON

(DChannel event, reported by the NGX and IPX only)

Description: Although this event exists in the header file, it is not currently supported by the SmarkWORKS API. Refer to EVT_AUDIO_CHANGE.

EVT_AUTO_ANSWER

(DChannel event, reported by the NGX and IPX only)

Description: Indicates that auto answer has activated

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_AUTO_ANSWER	0-n/a	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_AUTO_ANSWER_CANCELED

(DChannel event, reported by the NGX and IPX only)

Description: Indicates that auto answer has been cancelled

Structure

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_AUTO_ANSWER_CANCELED	0-n/a	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_AUX_MEDIA_SESSION_STARTED

(IPX only)

Description: In the event that the IPX is visible to both endpoints associated with a single session, the IPX now reports a media session started event for both endpoints. EVT_MEDIA_SESSION_STARTED is reported for the first endpoint that establishes the media session, while EVT_AUX_MEDIA_SESSION_STARTED is reported (with the same Session ID) for the other endpoint).

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_AUX_MEDIA_SESSION_STARTED	0-n/a	Protocol ID/ Station ID	0-n/a	BoardID	0-n/a	pointer to data structure	Length of data structure
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_AUX_MEDIA_SESSION_STOPPED

(IPX only)

Description: In the event that the IPX is visible to both endpoints associated with a single session, the IPX now reports a media session started event for both endpoints. EVT_MEDIA_SESSION_STARTED is reported for the first endpoint that establishes the media session, while EVT_AUX_MEDIA_SESSION_STARTED is reported (with the same Session ID) for the other endpoint). When the stopped event is reported, this indicates that the media session is no longer active.

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_AUX_MEDIA_SESSION_STOPPED	0-n/a	Protocol ID/ Station ID	0-n/a	BoardID	0-n/a	pointer to data structure	Length of data structure
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_BACKSPACE_BUTTON_PRESSED

(DChannel event, reported by the NGX and IPX only)

Indicates that the backspace button was pressed.

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME for-mat	0-n/a	EVT_BACKSPACE_BUTTON_PRESSED	0-n/a	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_BOARD_PANIC_ERROR

Description: Indicates a board panic error

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_BOARD_PANIC_ERROR	0-n/a	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_BOARD_WATCHDOG_ALARM

Description: Indicates a board error. Currently, this event is only reported on the IPX/HPX products.

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_BOARD_WATCHDOG_ALARM	See Below	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

Subreason:

One subreason currently exists: WATCHDOG_ALARM_PROTOCOL_ENGINE_RESTARTED without user application command.

Other messages will be added per need.

EVT_CALLBACK_BUTTON_PRESSED

(DChannel event, reported by the NGX and IPX only)

Indicates that the callback button was pressed.

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_CALLBACK_BUTTON_PRESSED	0-n/a	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_CALLID_CSUMERR

Description: Indicates that CallerID was detected, but was not reported (EVT_CALLID_STOP) to the user application due to bad checksum error condition.

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_CALLID_CSUMERR	0-n/a	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_CALLID_DROPPED

Description: Indicates that CallerID was detected and has been dropped due to a full queue

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File- TIME for- mat	0-n/a	EVT_CALLID_DROPPED	See Below*	See Below*	0-n/a	BoardID	Global- Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

SubReason - Lenght of the full packet including the CRC

XtraInfo - the first 4 bytes of the packet

EVT_CALLID_STOP

Description: Indicates that CallerID (in a well formed 1200bps FSK data packet) has been detected and received

Structure:

TimeStamp	UserStatus	EventCode	SubReason	Xtralnfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_CALLID_DROPPED	See Below*	See Below*	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

SubReason - Length of the full packet including the CRC

Xtralnfo - the first 4 bytes of the packet

EVT_CALLWAITING_BUTTON_PRESSED

(DChannel event, reported by the NGX and IPX only)

Indicates that the callwaiting button was pressed.

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME for-mat	0-n/a	EVT_CALLWAITING_BUTTON_PRESSED	0-n/a	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_CALLWAITING_LIGHT_FASTFLASHING
EVT_CALLWAITING_LIGHT_FLASHING
EVT_CALLWAITING_LIGHT_OFF
EVT_CALLWAITING_LIGHT_ON
EVT_CALLWAITING_LIGHT_QUICKFLASH
EVT_CALLWAITING_LIGHT_VERY_FASTFLASHING

(DChannel event, reported by the NGX and IPX only)

Purpose:

Indicates that the callwaiting light on the phone is on, off or flashing.

Description:

Observation of various phone models enabled the AudioCodes lab to categorize phone lights as flashing, fast flashing, very fast flashing or quick flashing. The cadence varies depending on PBX models, and this may also be configured. Our definition of flashing lights stems from the signal passed from the PBX to the phone, and is based on factory default configurations for each model. Please refer to your PBX documentation for more information.

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File- TIME for- mat	0-n/a	EVT_ANSWER_LIGHT_	*See Below	0-n/a	0-n/a	BoardID	Global- Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

Sub-Reason(this field is populated only when light is flashing):

NOTE: These may vary depending on phone model and PBX.

Bits 0-7 = Light Number

Bits 8 = Green

Bit 9 = Red

Bit 10 = Amber

Bits 11-31 = Reserved

EVT_CANCEL_BUTTON_PRESSED

(DChannel event, reported by the NGX and IPX only)

Indicates that the cancel button was pressed.

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File- TIME for- mat	0-n/a	EVT_CANCEL_ BUTTON_PRESSED	0-n/a	0-n/a	0-n/a	BoardID	Global- Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_CANCEL_BUTTON_RELEASED

(DChannel event, reported by the NGX and IPX only)

Indicates that the cancel button was released.

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File- TIME for- mat	0-n/a	EVT_CANCEL_ BUTTON_RELEASED	0-n/a	0-n/a	0-n/a	BoardID	Global- Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_CC_CALL_ALERTING

EVT_CALL_ALERTING

Description: Indicates a phone is ringing on a monitored line.

SmartWORKS DP and SmartWORKS NGX (BRI only) cards - ISDN call alerting. This event is reported as a channel event.

SmartWORKS IPX - reported only when call control is supported on a particular PBX integration (refer to the NGX/IPX PBX Matrix on the online support system) and call control event reporting is enabled via the application. This event is reported as a board event.

Note: When the polling method is used, events prefixed with EVT_CC_ typically require user applications to call `MTReturnEventBuffer()` passing in `MT_EVENT.ptrBuffer` to free the allocated memory. See the Functional Reference Library where `MTReturnEventBuffer()` is explained for more information.

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtrInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_CC_CALL_ALERTING	0-n/a	See Below*	0-n/a	BoardID	Global-Channel ID (GCI)	See Below*	See Below*
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

* XtrInfo:

SmartWORKS DP and NGX: This field contains four (4) bytes of data.

channel = Event.XtrInfo & 0x0000FFFF

The two (2) most significant bytes represent trunk #.

The two (2) least significant represent timeslot #

SmartWORKS IPX: This field contains eight(8) bytes of data.

protocol ID = (Event.XtrInfo & 0xFFFF0000) >> 16

station ID = Event.XtrInfo & 0x0000FFFF

ptrBuffer:

ptrBuffer is a pointer to the MT_CALL_INFO data structure.

DataLength:

DataLength is the length in bytes of the data structure pointed to by ptrBuffer. The MT_CALL_INFO structure is populated with more information and is displayed on the next page. This structure, along with other information, is available in the Call Processing section of the SmartWORKS Developer's Guide.

Type	Name	Purpose
ULONG	CallRef	The unique value assigned to this call by the ISDN network.
ULONG	CallSource	Indicates whether this is an incoming or outgoing call. Possible values: MT_CC_INCOMING_CALL, MT_CC_OUTGOING_CALL
ULONG	CallState	All call states are defined in the DataCC.h file.
ULONG	CallTrunk	On the AudioCodes board, the trunk number where this call is connected.
ULONG	CallDuration	The total call duration
ULONG	Layer1Coding	All layer protocol values are defined in the header file DataCC.h.
ULONG	Cause	The cause of the call state. All values are defined in the DataCC.h file.
MT_CC_CHANNEL_ID	ChannelId	Information about the channel used on the AudioCodes board.
MT_CC_PARTY_NUMBER	CallerNumber	The phone number where this call originated.
MT_CC_PARTY_SUBADDR	CalledSubAddr	When available, the originating number's extension.
MT_CC_PARTY_NUMBER	CalledNumber	The number that was dialed.
MT_CC_PARTY_SUBADDR	CalledSubAddr	When available, the extension number of the called phone.
MT_CC_PARTY_NUMBER	ConnectedNumber	When call forwarding is in use, this is the number where the call is actually connected.
MT_CC_PARTY_SUBADDR	ConnectedSubAddr	When call forwarding is in use, the extension of the phone where the call is actually connected.
MT_CC_PARTY_NUMBER	RedirectingNumber	If the call was re-directed to another phone, this is the number of the phone that initiated the re-direction.

Type	Name	Purpose
MT_CC_CALL_IDENTITY	CallIdentity	When available, any name associated with this phone.

EVT_CC_ALERT_IND

Description: ISDN Alert Indication, indicates a phone is ringing on the monitored line. Generated by the SmartWORKS DT card. All events are boards events except when using the RBS protocol in which case it is reported as a channel event.

Note: When the polling method is used, events prefixed with EVT_CC_ typically require user applications to call `MTReturnEventBuffer()` passing in `MT_EVENT.ptrBuffer` to free the allocated memory. See the Functional Reference Library where `MTReturnEventBuffer()` is explained for more information.

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtrInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME for-mat	0-n/a	EVT_CC_ALERT_IND	0-n/a	See Below*	0-n/a	BoardID	Global-Channel ID (GCI)	See Below*	See Below*
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

***XtrInfo:**

This field contains four (4) bytes of data:

channel = Event.XtrInfo & 0x0000FFFF

The two (2) most significant bytes represent trunk #.

The two (2) least significant represent channel #

ptrBuffer:

ptrBuffer is a pointer to a null terminated string that contains the screen data.

DataLength:

DataLength is the length in bytes of the string pointed to by ptrBuffer.

The MT_CC_ALERT_IND structure is populated with more data.

typedef struct

```
{
    int      ConnectionId ;
    int      CallRef ;
} MT_CC_ALERT_IND, * PMT_CC_ALERT_IND;
```

EVT_CC_CALL_ABANDONED

EVT_CALL_ABANDONED

Description: Indicates that a call was abandoned.

SmartWORKS DP and SmartWORKS NGX (BRI only) cards - ISDN call alerting. This event is reported as a channel event.

SmartWORKS IPX - reported only when call control is supported on a particular PBX integration (refer to the NGX/IPX PBX Matrix on the online support system) and call control event reporting is enabled via the application. This event is reported as a board event.

Note: When the polling method is used, events prefixed with EVT_CC_ typically require user applications to call `MTReturnEventBuffer()` passing in `MT_EVENT.ptrBuffer` to free the allocated memory. See the Functional Reference Library where `MTReturnEventBuffer()` is explained for more information.

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtrInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_CC_CALL_ABANDONED	0-n/a	See Below*	0-n/a	BoardID	Global-Channel ID (GCI)	See Below*	See Beolw*
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

*XtrInfo:

SmartWORKS DP and NGX: This field contains four (4) bytes of data.

channel = Event.XtrInfo & 0x0000FFFF

The two (2) most significant bytes represent trunk #.

The two (2) least significant represent timeslot #

SmartWORKS IPX: This field contains eight(8) bytes of data.

protocol ID = (Event.XtrInfo & 0xFFFF0000) >> 16

station ID = Event.XtrInfo & 0x0000FFFF

ptrBuffer:

ptrBuffer is a pointer to the MT_CALL_INFO data structure.

DataLength:

DataLength is the length in bytes of the string pointed to by ptrBuffer.

The MT_CALL_INFO structure is populated with more information. This structure is defined where the event [EVT_CALL_ALERTING](#) is explained.

EVT_CC_CALL_CONNECTED EVT_CALL_CONNECTED

Description: Indicates that a call is connected.

SmartWORKS DP and SmartWORKS NGX (BRI only) cards - This event is reported as a channel event.

SmartWORKS IPX - reported only when call control is supported on a particular PBX integration (refer to the NGX/IPX PBX Matrix on the online support system) and call control event reporting is enabled via the application. This event is reported as a board event.

Note: When the polling method is used, events prefixed with EVT_CC_ typically require user applications to call MTReturnEventBuffer() passing in MT_EVENT.ptrBuffer to free the allocated memory. See the Functional Reference Library where MTReturnEventBuffer() is explained for more information.

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_CC_CALL_CONNECTED	0-n/a	See Below*	0-n/a	BoardID	Global-Channel ID (GCI)	See Below*	See Below*
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

* XtraInfo:

SmartWORKS DP and NGX: This field contains four (4) bytes of data.

channel = Event.XtraInfo & 0x0000FFFF

The two (2) most significant bytes represent trunk #.

The two (2) least significant represent timeslot #

SmartWORKS IPX: This field contains eight(8) bytes of data.

protocol ID = (Event.XtraInfo & 0xFFFF0000) >> 16

station ID = Event.XtraInfo & 0x0000FFFF

ptrBuffer:

ptrBuffer is a pointer to the MT_CALL_INFO data structure.

DataLength:

DataLength is the length in bytes of the string pointed to by ptrBuffer.

The MT_CALL_INFO structure is populated with more information. This structure is defined where the event [EVT_CALL_ALERTING](#) is explained.

EVT_CC_CALL_HELD EVT_CALL_HELD

Indicates that a call has been held.

SmartWORKS DP and SmartWORKS NGX (BRI only) cards - ISDN call alerting. This event is reported as a channel event.

SmartWORKS IPX - reported only when call control is supported on a particular PBX integration (refer to the NGX/IPX PBX Matrix on the online support system) and call control event reporting is enabled via the application. This event is reported as a board event.

Note: When the polling method is used, events prefixed with EVT_CC_ typically require user applications to call MTReturnEventBuffer() passing in MT_EVENT.ptrBuffer to free the allocated memory. See the Functional Reference Library where MTReturnEventBuffer() is explained for more information.

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File- TIME for- mat	0-n/a	EVT_CC_CALL_HELD	0-n/a	See Below*	0-n/a	BoardID	Global- Channel ID (GCI)	See Below*	See Below*
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

*XtraInfo:

SmartWORKS DP and NGX: This field contains four (4) bytes of data.

channel = Event.XtraInfo & 0x0000FFFF

The two (2) most significant bytes represent trunk #.

The two (2) least significant represent timeslot #

SmartWORKS IPX: This field contains eight(8) bytes of data.

protocol ID = (Event.XtraInfo & 0xFFFF0000) >> 16

station ID = Event.XtraInfo & 0x0000FFFF

ptrBuffer:

ptrBuffer is a pointer to the MT_CALL_INFO data structure.

DataLength:

DataLength is the length in bytes of the string pointed to by ptrBuffer.

This structure is defined where the event [EVT_CALL_ALERTING](#) is explained.

EVT_CC_CALL_PROCEEDING EVT_CALL_PROCEEDING

Indicates that an ISDN call is pending. Generated by the SmartWORKS DP and SmartWORKS NGX card (BRI ISDN only) This event is reported as a channel event.

Note: When the polling method is used, events prefixed with **EVT_CC_** typically require user applications to call **MTReturnEventBuffer()** passing in **MT_EVENT.ptrBuffer** to free the allocated memory. See the **Functional Reference Library** where **MTReturnEventBuffer()** is explained for more information.

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File- TIME for- mat	0-n/a	EVT_CC_CALL_ PROCEEDING	0-n/a	0-n/a	0-n/a	BoardID	Global- Channel ID (GCI)	See Below*	See Below*
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

*ptrBuffer:

ptrBuffer is a pointer to a null terminated string that contains the screen data.

DataLength:

DataLength is the length in bytes of the string pointed to by ptrBuffer.

The MT_CALL_INFO structure is populated with more information. This structure is defined where the event [EVT_CALL_ALERTING](#) is explained.

EVT_CC_CALL_REJECTED

EVT_CALL_REJECTED

Indicates that a call has been rejected.

SmartWORKS DP and SmartWORKS NGX (BRI only) cards - ISDN call alerting. This event is reported as a channel event.

SmartWORKS IPX - reported only when call control is supported on a particular PBX integration (refer to the NGX/IPX PBX Matrix on the online support system) and call control event reporting is enabled via the application. This event is reported as a board event.

Note: When the polling method is used, events prefixed with EVT_CC_ typically require user applications to call `MTReturnEventBuffer()` passing in `MT_EVENT.ptrBuffer` to free the allocated memory. See the Functional Reference Library where `MTReturnEventBuffer()` is explained for more information.

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtrInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_CC_CALL_REJECTED	0-n/a	See Below*	0-n/a	BoardID	Global-Channel ID (GCI)	See Below*	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

*XtrInfo:

SmartWORKS DP and NGX: This field contains four (4) bytes of data.

channel = Event.XtrInfo & 0x0000FFFF

The two (2) most significant bytes represent trunk #.

The two (2) least significant represent timeslot #

SmartWORKS IPX: This field contains eight(8) bytes of data.

protocol ID = (Event.XtrInfo & 0xFFFF0000) >> 16

station ID = Event.XtrInfo & 0x0000FFFF

ptrBuffer is a pointer to the MT_CALL_INFO data structure.

This structure is defined where the event [EVT_CALL_ALERTING](#) is explained.

EVT_CC_CALL_RELEASED EVT_CALL_RELEASED

Indicates that a call has been released.

SmartWORKS DP and SmartWORKS NGX (BRI only) cards - ISDN call alerting. This event is reported as a channel event.

SmartWORKS IPX - reported only when call control is supported on a particular PBX integration (refer to the NGX/IPX PBX Matrix on the online support system) and call control event reporting is enabled via the application. This event is reported as a board event.

Note: When the polling method is used, events prefixed with EVT_CC_ typically require user applications to call MTReturnEventBuffer() passing in MT_EVENT.ptrBuffer to free the allocated memory. See the Functional Reference Library where MTReturnEventBuffer() is explained for more information.

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtrInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File- TIME for- mat	0-n/a	EVT_CC_CALL_RELEASED	0-n/a	See Below*	0-n/a	BoardID	Global- Channel ID (GCI)	See Below*	See Below*
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

*XtrInfo:

SmartWORKS DP and NGX: This field contains four (4) bytes of data.

channel = Event.XtrInfo & 0x0000FFFF

The two (2) most significant bytes represent trunk #.

The two (2) least significant represent timeslot #

SmartWORKS IPX: This field contains eight(8) bytes of data.

protocol ID = (Event.XtrInfo & 0xFFFF0000) >> 16

station ID = Event.XtrInfo & 0x0000FFFF

ptrBuffer:

ptrBuffer is a pointer to the MT_CALL_INFO data structure.

DataLength:

DataLength is the length in bytes of the string pointed to by ptrBuffer.

The MT_CALL_INFO structure is populated with more information. This structure is defined where the event [EVT_CALL_ALERTING](#) is explained.

EVT_CC_CALL_RESUMED EVT_CALL_RESUMED

Indicates that a call has resumed.

SmartWORKS DP and SmartWORKS NGX (BRI only) cards - ISDN call alerting. This event is reported as a channel event.

SmartWORKS IPX - reported only when call control is supported on a particular PBX integration (refer to the NGX/IPX PBX Matrix on the online support system) and call control event reporting is enabled via the application. This event is reported as a board event.

Note: When the polling method is used, events prefixed with EVT_CC_ typically require user applications to call MTReturnEventBuffer() passing in MT_EVENT.ptrBuffer to free the allocated memory. See the Functional Reference Library where MTReturnEventBuffer() is explained for more information.

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtralInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File- TIME for- mat	0-n/a	EVT_CC_CALL_RESUMED	0-n/a	See Below*	0-n/a	BoardID	Global- Channel ID (GCI)	See Below*	See Below*
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

*XtralInfo:

SmartWORKS DP and NGX: This field contains four (4) bytes of data.

channel = Event.XtralInfo & 0x0000FFFF

The two (2) most significant bytes represent trunk #.

The two (2) least significant represent timeslot #

SmartWORKS IPX: This field contains eight(8) bytes of data.

protocol ID = (Event.XtralInfo & 0xFFFF0000) >> 16

station ID = Event.XtralInfo & 0x0000FFFF

ptrBuffer is a pointer to the MT_CALL_INFO data structure.

The MT_CALL_INFO structure is populated with more information. This structure is defined where the event [EVT_CALL_ALERTING](#) is explained.

EVT_CC_CALL_RETRIEVED EVT_CALL_RETRIEVED

Indicates that a call has been retrieved.

SmartWORKS DP and SmartWORKS NGX (BRI only) cards - ISDN call alerting. This event is reported as a channel event.

SmartWORKS IPX - reported only when call control is supported on a particular PBX integration (refer to the NGX/IPX PBX Matrix on the online support system) and call control event reporting is enabled via the application. This event is reported as a board event.

Note: When the polling method is used, events prefixed with EVT_CC_ typically require user applications to call MTReturnEventBuffer() passing in MT_EVENT.ptrBuffer to free the allocated memory. See the Functional Reference Library where MTReturnEventBuffer() is explained for more information.

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtralInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File- TIME for- mat	0-n/a	EVT_CC_CALL_RETRIEVED	0-n/a	See Below*	0-n/a	BoardID	Global- Channel ID (GCI)	See Below*	See Below*
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

*XtralInfo:

SmartWORKS DP and NGX: This field contains four (4) bytes of data.

channel = Event.XtralInfo & 0x0000FFFF

The two (2) most significant bytes represent trunk #.

The two (2) least significant represent timeslot #

SmartWORKS IPX: This field contains eight(8) bytes of data.

protocol ID = (Event.XtralInfo & 0xFFFF0000) >> 16

station ID = Event.XtralInfo & 0x0000FFFF

ptrBuffer:

ptrBuffer is a pointer to the MT_CALL_INFO data structure.

DataLength:

DataLength is the length in bytes of the string pointed to by ptrBuffer.

The MT_CALL_INFO structure is populated with more information. This structure is defined where the event [EVT_CALL_ALERTING](#) is explained.

EVT_CC_CALL_SUSPENDED EVT_CALL_SUSPENDED

Indicates that a call has been suspended.

SmartWORKS DP and SmartWORKS NGX (BRI only) cards - ISDN call alerting. This event is reported as a channel event.

SmartWORKS IPX - reported only when call control is supported on a particular PBX integration (refer to the NGX/IPX PBX Matrix on the online support system) and call control event reporting is enabled via the application. This event is reported as a board event.

Note: When the polling method is used, events prefixed with EVT_CC_ typically require user applications to call MTReturnEventBuffer() passing in MT_EVENT.ptrBuffer to free the allocated memory. See the Functional Reference Library where MTReturnEventBuffer() is explained for more information.

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtralInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File- TIME for- mat	0-n/a	EVT_CC_CALL_SUSPENDED	0-n/a	See Below*	0-n/a	BoardID	Global- Channel ID (GCI)	See Below*	See Below*
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

*XtralInfo:

SmartWORKS DP and NGX: This field contains four (4) bytes of data.

channel = Event.XtralInfo & 0x0000FFFF

The two (2) most significant bytes represent trunk #.

The two (2) least significant represent timeslot #

SmartWORKS IPX: This field contains eight(8) bytes of data.

protocol ID = (Event.XtralInfo & 0xFFFF0000) >> 16

station ID = Event.XtralInfo & 0x0000FFFF

ptrBuffer:

ptrBuffer is a pointer to a null terminated string that contains the screen data.

DataLength:

DataLength is the length in bytes of the string pointed to by ptrBuffer.

The MT_CALL_INFO structure is populated with more information. This structure is defined where the event [EVT_CALL_ALERTING](#) is explained.

EVT_CC_CONNECT_CONF

Indicates connect confirmation. Generated by the SmartWORKS DT card. All events are boards events except when using the RBS protocol in which case it is reported as a channel event.

Note: When the polling method is used, events prefixed with **EVT_CC_** typically require user applications to call **MTReturnEventBuffer()** passing in **MT_EVENT.ptrBuffer** to free the allocated memory. See the **Functional Reference Library** where **MTReturnEventBuffer()** is explained for more information.

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME for-mat	0-n/a	EVT_CC_CONNECT_CONF	0-n/a	See Below*	0-n/a	BoardID	Global-Channel ID (GCI)	See Below*	See Below*
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

***XtraInfo:**

This field contains four (4) bytes of data:

channel = Event.XtraInfo & 0x0000FFFF

The two (2) most significant bytes represent trunk #.

The two (2) least significant represent channel #

ptrBuffer:

ptrBuffer is a pointer to a null terminated string that contains the screen data.

DataLength:

DataLength is the length in bytes of the string pointed to by ptrBuffer.

The MT_CC_CONNECT_CONF structure is populated with this information:

typedef struct

```
{  
    int      ConnectionId;  
    int      CallRef;
```

```
int      Flags ;  
MT_CC_CHANNEL_ID  ChannelId;  
} MT_CC_CONNECT_CONF, * PMT_CC_CONNECT_CONF;
```

EVT_CC_CONNECT_IND

Indicates an incoming call. Generated by the SmartWORKS DT card. All events are boards events except when using the RBS protocol in which case it is reported as a channel event.

Note: When the polling method is used, events prefixed with **EVT_CC_** typically require user applications to call **MTReturnEventBuffer()** passing in **MT_EVENT.ptrBuffer** to free the allocated memory. See the **Functional Reference Library** where **MTReturnEventBuffer()** is explained for more information.

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_CC_CONNECT_IND	0-n/a	See Below*	0-n/a	BoardID	Global-Channel ID (GCI)	See Below*	See Below*
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

*** XtraInfo:**

This field contains four (4) bytes of data:

channel = Event.XtraInfo & 0x0000FFFF

The two (2) most significant bytes represent trunk #.

The two (2) least significant represent channel #

ptrBuffer:

ptrBuffer is a pointer to a null terminated string that contains the screen data.

DataLength:

DataLength is the length in bytes of the string pointed to by ptrBuffer.

The MT_CC_CONNECT_IND structure is populated with more information:

typedef struct

```
{
    int      CallRef ;
```

```
int      SendingComplete ; NOTE: Used with RBS only.  
int      Flags ;  
MT_CC_CHANNEL_ID  ChannelId;  
MT_CC_BEARER_CAPS  BearerCaps;  
MT_CC_PARTY_NUMBER  CalledPartyAddr;  
MT_CC_PARTY_SUBADDR CalledPartySubAddr;  
MT_CC_PARTY_NUMBER  CallingPartyAddr;  
MT_CC_PARTY_SUBADDR CallingPartySubAddr;  
MT_CC_U2U      UserInfo ;  
MT_CC_FACILITY   Facility ;  
} MT_CC_CONNECT_IND, * PMT_CC_CONNECT_IND;
```

EVT_CC_DISC_CONF

Indicates disconnect confirmation. Generated by the SmartWORKS DT card. All events are boards events except when using the RBS protocol in which case it is reported as a channel event.

Note: When the polling method is used, events prefixed with **EVT_CC_** typically require user applications to call **MTReturnEventBuffer()** passing in **MT_EVENT.ptrBuffer** to free the allocated memory. See the **Functional Reference Library** where **MTReturnEventBuffer()** is explained for more information.

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_CC_DISC_CONF	0-n/a	See Below*	0-n/a	BoardID	Global-Channel ID (GCI)	See Below*	See Below*
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

*** XtraInfo:**

This field contains four (4) bytes of data:

channel = Event.XtraInfo & 0x0000FFFF

The two (2) most significant bytes represent trunk #.

The two (2) least significant represent channel #

ptrBuffer:

ptrBuffer is a pointer to a null terminated string that contains the screen data.

DataLength:

DataLength is the length in bytes of the string pointed to by ptrBuffer.

The MT_CC_DISC_CONF structure is populated with this information:

typedef struct

```
{  
    int      ConnectionId ;
```



```
int      CallRef;  
int      Cause ;  
} MT_CC_DISC_CONF, *PMT_CC_DISC_CONF;
```

EVT_CC_DISC_IND

Indicates a disconnect. Generated by the SmartWORKS DT card. All events are boards events except when using the RBS protocol in which case it is reported as a channel event.

Note: When the polling method is used, events prefixed with **EVT_CC_** typically require user applications to call **MTReturnEventBuffer()** passing in **MT_EVENT.ptrBuffer** to free the allocated memory. See the **Functional Reference Library** where **MTReturnEventBuffer()** is explained for more information.

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME for-mat	0-n/a	EVT_CC_DISC_IND	0-n/a	See Below*	0-n/a	BoardID	Global-Channel ID (GCI)	See Below*	See Below*
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

*** XtraInfo:**

This field contains four (4) bytes of data:

channel = Event.XtraInfo & 0x0000FFFF

The two (2) most significant bytes represent trunk #.

The two (2) least significant represent channel #

ptrBuffer:

ptrBuffer is a pointer to a null terminated string that contains the screen data.

DataLength:

DataLength is the length in bytes of the string pointed to by ptrBuffer.

The MT_CC_DISC_IND structure is populated with this information:

typedef struct

```
{  
    int      ConnectionId;  
    int      CallRef;
```

```
int      Cause ;  
MT_CC_U2U      UserInfo ;  
MT_CC_FACILITY      Facility ;  
} MT_CC_DISC_IND, * PMT_CC_DISC_IND;
```

EVT_CC_FACILITY_CONF

A facility confirmation. Generated by the SmartWORKS DT card. All events are boards events except when using the RBS protocol in which case it is reported as a channel event.

Note: When the polling method is used, events prefixed with **EVT_CC_** typically require user applications to call **MTReturnEventBuffer()** passing in **MT_EVENT.ptrBuffer** to free the allocated memory. See the **Functional Reference Library** where **MTReturnEventBuffer()** is explained for more information.

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_CC_FACILITY_CONF	0-n/a	See Below*	0-n/a	BoardID	Global-Channel ID (GCI)	See Below*	See Below*
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

*** XtraInfo:**

This field contains four (4) bytes of data:

channel = Event.XtraInfo & 0x0000FFFF

The two (2) most significant bytes represent trunk #.

The two (2) least significant represent channel #

ptrBuffer:

ptrBuffer is a pointer to a null terminated string that contains the screen data.

DataLength:

DataLength is the length in bytes of the string pointed to by ptrBuffer.

The MT_CC_FACILITY_CONF is populated with this information:

typedef struct

```
{  
    int      ConnectionId;  
};
```

```
int      CallRef ;
int      AccessId ;
int      Action ;
int      Code ;
int      NetCause ;
int      Flags ;
MT_CC_CHANNEL_ID  ChannelId;
MT_CC_PARTY_NUMBER CalledAddr;
MT_CC_PARTY_NUMBER CallingAddr;
MT_CC_VOICE_SUPSRV Voice ;
MT_CC_FACILITY    Facility ;
} MT_CC_FACILITY_CONF, * PMT_CC_FACILITY_CONF;
```

EVT_CC_FACILITY_IND

A facility indication. Generated by the SmartWORKS DT card. All events are boards events except when using the RBS protocol in which case it is reported as a channel event.

Note: When the polling method is used, events prefixed with **EVT_CC_** typically require user applications to call **MTReturnEventBuffer()** passing in **MT_EVENT.ptrBuffer** to free the allocated memory. See the **Functional Reference Library** where **MTReturnEventBuffer()** is explained for more information.

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME for-mat	0-n/a	EVT_CC_FACILITY_IND	0-n/a	See Below*	0-n/a	BoardID	Global-Channel ID (GCI)	See Below*	See Below*
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

*** XtraInfo:**

This field contains four (4) bytes of data:

channel = Event.XtraInfo & 0x0000FFFF

The two (2) most significant bytes represent trunk #.

The two (2) least significant represent channel #

ptrBuffer:

ptrBuffer is a pointer to a null terminated string that contains the screen data.

DataLength:

DataLength is the length in bytes of the string pointed to by ptrBuffer.

The MT_CC_FACILITY_IND structure is populated with this information:

typedef struct

```
{  
    int      ConnectionId;  
    int      CallRef;
```

```
int      AccessId ;
int      Action ;
int      Code ;
int      Flags ;
MT_CC_CHANNEL_ID  ChannelId;
MT_CC_PARTY_NUMBER CalledAddr;
MT_CC_PARTY_NUMBER CallingAddr;
MT_CC_VOICE_SUPSRV Voice ;
MT_CC_FACILITY    Facility ;

} MT_CC_FACILITY_IND, * PMT_CC_FACILITY_IND;
```

EVT_CC_INFO_IND

Indicates ISDN called party digits. Generated by the SmartWORKS DT card. All events are boards events except when using the RBS protocol in which case it is reported as a channel event.

Note: When the polling method is used, events prefixed with **EVT_CC_** typically require user applications to call **MTReturnEventBuffer()** passing in **MT_EVENT.ptrBuffer** to free the allocated memory. See the **Functional Reference Library** where **MTReturnEventBuffer()** is explained for more information.

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME for-mat	0-n/a	EVT_CC_INFO_IND	0-n/a	See Below*	0-n/a	BoardID	Global-Channel ID (GCI)	See Below*	See Below*
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

*** XtraInfo:**

This field contains four (4) bytes of data:

channel = Event.XtraInfo & 0x0000FFFF

The two (2) most significant bytes represent trunk #.

The two (2) least significant represent channel #

ptrBuffer:

ptrBuffer is a pointer to a null terminated string that contains the screen data.

DataLength:

DataLength is the length in bytes of the string pointed to by ptrBuffer.

The MT_CC_INFO_IND structure is populated with this information:

typedef struct

```
{  
    int      ConnectionId;  
    int      CallRef;
```



```
int      Flags ;  
MT_CC_PARTY_NUMBER CalledPartyAddr;  
} MT_CC_INFO_IND, * PMT_CC_INFO_IND;
```

EVT_CC_PROGRESS_IND

Indicates ISDN call progress. Generated by the SmartWORKS DT card. All events are boards events except when using the RBS protocol in which case it is reported as a channel event.

Note: When the polling method is used, events prefixed with **EVT_CC_** typically require user applications to call **MTReturnEventBuffer()** passing in **MT_EVENT.ptrBuffer** to free the allocated memory. See the **Functional Reference Library** where **MTReturnEventBuffer()** is explained for more information.

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME for-mat	0-n/a	EVT_CC_PROGRESS_IND	0-n/a	See Below*	0-n/a	BoardID	Global-Channel ID (GCI)	See Below*	See Below*
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

*** XtraInfo:**

This field contains four (4) bytes of data:

channel = Event.XtraInfo & 0x0000FFFF

The two (2) most significant bytes represent trunk #.

The two (2) least significant represent channel #

ptrBuffer:

ptrBuffer is a pointer to a null terminated string that contains the screen data.

DataLength:

DataLength is the length in bytes of the string pointed to by ptrBuffer.

The MT_CC_PROGRESS_IND structure is populated with this information:

typedef struct

```
{  
    int      ConnectionId ;  
    int      CallRef ;  
    int      Flags ;
```

```
    MT_CC_CHANNEL_ID  ChannelId;  
} MT_CC_PROGRESS_IND, * PMT_CC_PROGRESS_IND;
```

EVT_CC_RESUME_CONF

A ISDN call was resumed. Generated by the SmartWORKS DT card. All events are boards events except when using the RBS protocol in which case it is reported as a channel event.

Note: When the polling method is used, events prefixed with **EVT_CC_** typically require user applications to call **MTReturnEventBuffer()** passing in **MT_EVENT.ptrBuffer** to free the allocated memory. See the **Functional Reference Library** where **MTReturnEventBuffer()** is explained for more information.

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtralInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME for-mat	0-n/a	EVT_CC_RESUME_CONF	0-n/a	See Below*	0-n/a	BoardID	Global-Channel ID (GCI)	See Below*	See Below*
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

*** XtralInfo:**

This field contains four (4) bytes of data:

channel = Event.XtralInfo & 0x0000FFFF

The two (2) most significant bytes represent trunk #.

The two (2) least significant represent channel #

ptrBuffer:

ptrBuffer is a pointer to a null terminated string that contains the screen data.

DataLength:

DataLength is the length in bytes of the string pointed to by ptrBuffer.

The MT_CC_RESUME_CONF structure is populated with this information:

typedef struct

```
{  
    int      ConnectionId ;
```

```
int      CallRef;  
int      RetCode ;  
int      NetCause ;  
MT_CC_CHANNEL_ID  ChannelId;  
MT_CC_CALL_IDENTITY CallIdentity ;  
  
} MT_CC_RESUME_CONF, *PMT_CC_RESUME_CONF;
```

EVT_CC_SUSPEND_CONF

A ISDN call was suspended. Generated by the SmartWORKS DT card. All events are boards events except when using the RBS protocol in which case it is reported as a channel event.

Note: When the polling method is used, events prefixed with EVT_CC_ typically require user applications to call `MTReturnEventBuffer()` passing in `MT_EVENT.ptrBuffer` to free the allocated memory. See the Functional Reference Library where `MTReturnEventBuffer()` is explained for more information.

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME for-mat	0-n/a	EVT_CC_SUSPEND_CONF	0-n/a	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	See Below*	See Below*
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

*** ptrBuffer:**

ptrBuffer is a pointer to a null terminated string that contains the screen data.

DataLength:

DataLength is the length in bytes of the string pointed to by ptrBuffer.

The MT_CC_SUSPEND_CONF structure is populated with this information:

typedef struct

```
{
    int      ConnectionId ;
    int      CallRef ;
    int      RetCode ;
    int      NetCause ;
    MT_CC_CALL_IDENTITY CallIdentity ;

} MT_CC_SUSPEND_CONF, *PMT_CC_SUSPEND_CONF;
```

EVT_CC_USERINFO_IND

Indicates user information. Generated by the SmartWORKS DT card. All events are boards events except when using the RBS protocol in which case it is reported as a channel event.

Note: When the polling method is used, events prefixed with **EVT_CC_** typically require user applications to call **MTReturnEventBuffer()** passing in **MT_EVENT.ptrBuffer** to free the allocated memory. See the **Functional Reference Library** where **MTReturnEventBuffer()** is explained for more information.

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtralInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_CC_USER_INFO_IND	0-n/a	See Below*	0-n/a	BoardID	Global-Channel ID (GCI)	See Below*	See Below*
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

*** XtralInfo:**

This field contains four (4) bytes of data:

channel = Event.XtralInfo & 0x0000FFFF

The two (2) most significant bytes represent trunk #.

The two (2) least significant represent channel #

ptrBuffer:

ptrBuffer is a pointer to a null terminated string that contains the screen data.

DataLength:

DataLength is the length in bytes of the string pointed to by ptrBuffer.

The MT_CC_USERINFO_IND structure is populated with the following information:

```
{
    ULONG      CallRef ;
    ULONG      Source ;
    MT_CC_CHANNEL_ID  ChannelId ;
    ULONG      Type ;
```

```
ULONG      MoreData ;  
MT_CC_U2U   UserInfo ;  
} MT_CC_USERINFO_IND, * PMT_CC_USERINFO_IND;
```


EVT_CC_USER_INFO EVT_USER_INFO

ISmartWORKS DP and SmartWORKS NGX (BRI only) cards - ISDN call alerting. This event is reported as a channel event and is passed to the user application with a data structure containing extra call information. When tapping a PRI network that supports codeset 5, this data structure is also used to support one information element of the codeset 5 - display information. Refer to the explanation below where this data structure has been explained.

Note: When the polling method is used, events prefixed with EVT_CC_ typically require user applications to call MTReturnEventBuffer() passing in MT_EVENT.ptrBuffer to free the allocated memory. See the Functional Reference Library where MTReturnEventBuffer() is explained for more information.

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtrInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME for-mat	0-n/a	EVT_CC_USER_INFO	0-n/a	See Below*	0-n/a	BoardID	Global-Channel ID (GCI)	See Below*	See Below*
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

* XtrInfo:

This field contains four (4) bytes of data:

channel = Event.XtrInfo & 0x0000FFFF

The two (2) most significant bytes represent trunk #.

The two (2) least significant represent channel #

ptrBuffer:

ptrBuffer is a pointer to a null terminated string that contains the screen data.

DataLength:

DataLength is the length in bytes of the string pointed to by ptrBuffer.

The MT_CC_USERINFO_IND structure is populated with the following information:

```
{
    ULONG      CallRef;
    ULONG      Source;
```

```
MT_CC_CHANNEL_ID  ChannelId ;
ULONG            Type ;
ULONG            MoreData ;
MT_CC_U2U        UserInfo ;
} MT_CC_USERINFO_IND, * PMT_CC_USERINFO_IND;
```

When tapping a PRI network that supports **NI 1 codeset 5**, the following applies (ISDN_BRI_NI1 firmware is required):

NI1 Codeset 5 display IE:

The “Type” field is set to ‘d’ for display.

The “UserInfo” field is populated with the raw display data found with information element.

The “CallRef”, “ChannelId”, and “Source” fields will be populated the same as with the MT_CC_CALL_INFO structure.

EVT_CC_USERINFO_REQ

Indicates ISDN user information has been requested. Generated by the SmartWORKS DT card. All events are boards events except when using the RBS protocol in which case it is reported as a channel event.

Note: When the polling method is used, events prefixed with **EVT_CC_** typically require user applications to call **MTReturnEventBuffer()** passing in **MT_EVENT.ptrBuffer** to free the allocated memory. See the **Functional Reference Library** where **MTReturnEventBuffer()** is explained for more information.

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtrInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME for-mat	0-n/a	EVT_CC_USERINFO_REQ	0-n/a	See Below*	0-n/a	BoardID	Global-Channel ID (GCI)	See Below*	See Below*
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

*** XtrInfo:**

This field contains four (4) bytes of data:

channel = Event.XtrInfo & 0x0000FFFF

The two (2) most significant bytes represent trunk #.

The two (2) least significant represent channel #

ptrBuffer:

ptrBuffer is a pointer to a null terminated string that contains the screen data.

DataLength:

DataLength is the length in bytes of the string pointed to by ptrBuffer.

The MT_CC_USERINFO_REQ structure is populated with this information:

typedef struct

```
{  
    int      ConnectionId ;  
    int      CallRef ;
```

```
MT_CC_USERINFO    UserInfo;
```

```
} MT_CC_USERINFO_REQ, * PMT_CC_USERINFO_REQ;
```

EVT_CFWD

(DChannel event, reported by the NGX and IPX only)

Indicates call forward activation

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME for-mat	0-n/a	EVT_CFWD	n/a	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	See Below*	See Below*
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

ptrBuffer - pointer to the buffer containing a call string of the phone number the call was forwarded to.

DataLength - the length of the data contained in the buffer

EVT_CFWD_BUTTON_PRESSED

(DChannel event, reported by the NGX and IPX only)

Indicates that the call forwarding button was pressed.

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME for-mat	0-n/a	EVT_CFWD_BUTTON_PRESSED	See Below*	See Below*	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

*On Cisco IP PBXs, there are three call forward options. The type of call forward option is reported in the subreason field of the MT_EVENT structure - Forward All (0x00000000), Forward Busy (0x00000001) and Forward No Answer (0x00000002).

EVT_CFWD_CANCELED

(DChannel event, reported by the NGX and IPX only)

Indicates call forward cancellation

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME for-mat	0-n/a	EVT_CFWD_CANCELED	0-n/a	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_CFWD_LIGHT_FASTFLASHING
EVT_CFWD_LIGHT_FLASHING
EVT_CFWD_LIGHT_OFF
EVT_CFWD_LIGHT_ON
EVT_CFWD_LIGHT_QUICKFLASH
EVT_CFWD_LIGHT_VERY_FASTFLASHING
EVT_CFWD_LIGHT_WINK

(DChannel event, reported by the NGX and IPX only)

Purpose:

Indicates that the call forwarding light on the phone is fast on, off, or flashing.

Description:

Observation of various phone models enabled the AudioCodes lab to categorize phone lights as flashing, fast flashing, very fast flashing, quick flashing or winking. The cadence varies depending on PBX models, and this may also be configured. Our definition of flashing lights stems from the signal passed from the PBX to the phone, and is based on factory default configurations for each model. Please refer to your PBX documentation for more information.

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File- TIME for- mat	0-n/a	EVT_CFWD_LIGHT_	*See Below	*See Below	0-n/a	BoardID	Global- Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

Sub-Reason(this field is only populated when light is flashing or winking):

NOTE: These may vary depending on phone model and PBX.

Bits 0-7 = Light Number
 Bits 8 = Green
 Bit 9 = Red
 Bit 10 = Amber
 Bits 11-31 = Reserved

XtraInfo: On Cisco IP PBXs, there are three call forward options. The type of call forward option is reported in the XtraInfo field of the MT_EVENT structure - Forward All (0x00000000), Forward Busy (0x00000001) and Forward No Answer (0x00000002).

EVT_CISCO_SCCP_CALL_INFO

(This event is only reported on the IPX when tapping a Cisco network)

Provides call information for a single call that is currently active.

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtrInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File- TIME for- mat	0-n/a	EVT_CISCO_SCCP_ CALL_INFO	0-n/a	0-n/a	0-n/a	BoardID	Global- Channel ID (GCI)	See Below*	See Below*
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

*The MT_CISCO_SCCP_CALL_INFO data structure is passed over with this event:

• .

Type	Name	Description
UCHAR	CallingPartyName	When a name can be associated with this number (callerID when possible, or configured via PBX)
UCHAR	CallingParty	The number of the caller.
UCHAR	CalledPartyName	When a name can be associated with this number (callerID when possible, or configured via PBX)
UCHAR	CalledParty	The number that was originally dialed.
ULONG	LineInstance	Which phone line the call is active on. This is phone model dependant.
ULONG	CallId	A unique number assigned by the IPX to this call. This number is unique per each Station and is not unique to the complete system. Same as call ref number provided with the call control events reported by the IPX.

Type	Name	Description
ULONG	CallType	#define MT_CISCO_SCCP_CALL_TYPE_INBOUND 1 MT_CISCO_SCCP_CALL_TYPE_OUTBOUND 2 MT_CISCO_SCCP_CALL_TYPE_FORWARD 3
UCHAR	OrigCalledParty	The number that was originally dialed. For example, this field is relevant when the original number is configured for call forwarding.
UCHAR	OrigCalledPartyName	relevant when call has been redirected or forwarded
UCHAR	LastRedirectingPartyName	When the call has been re-directed, this is the name associated with the re-directing phone number (when available)
UCHAR	LastRedirectingParty	When the call has been re-directed, this is the phone number which re-directed the call.
ULONG	OrigCalledPartyRedirect Reason	The reason the original called number was re-directed on the far side.
ULONG	LastRedirectReason	The reason the last call was re-directed.
UCHAR	CallingPartyVoiceMailbox	When directed to Voicemail, the mailbox number used by the calling party.
UCHAR	CalledPartyVoiceMailbox	When directed to Voicemail, the mailbox number used by the called party.
UCHAR	OriginalCalledPartyVoice-Mailbox	When the call has been forwarded, the original VoiceMailbox used by the called party.
UCHAR	LastRedirectVoiceMailbox	When the call has been re-directed, the original VoiceMailbox used by the called party.

EVT_CLEAR_BUTTON_PRESSED

(DChannel event, reported by the NGX and IPX only)

Indicates that the clear button was pressed.

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File- TIME for- mat	0-n/a	EVT_CLEAR_ BUTTON_PRESSED	0-n/a	0-n/a	0-n/a	BoardID	Global- Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_CONF_BUTTON_PRESSED

(DChannel event, reported by the NGX and IPX only)

Indicates that the Conference button was pressed

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File- TIME for- mat	0-n/a	EVT_CONF_BUTTON_ PRESSED	0-n/a	0-n/a	0-n/a	BoardID	Global- Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_CONF_BUTTON_RELEASED

(DChannel event, reported by the NGX and IPX only)

Indicates that the Conference button was released

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME for-mat	0-n/a	EVT_CONF_BUTTON_RELEASED	0-n/a	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_CONFERENCE_LIGHT_FASTFLASHING
EVT_CONFERENCE_LIGHT_FLASHING
EVT_CONFERENCE_LIGHT_OFF
EVT_CONFERENCE_LIGHT_ON
EVT_CONFERENCE_LIGHT_QUICKFLASH
EVT_CONFERENCE_LIGHT_VERY_FASTFLASHING
EVT_CONFERENCE_LIGHT_WINK

(DChannel event, reported by the NGX and IPX only)

Purpose:

Indicates that the conference light on the phone is fast on, off, or flashing.

Description:

Observation of various phone models enabled the AudioCodes lab to categorize phone lights as flashing, fast flashing, very fast flashing, quick flashing or winking. The cadence varies depending on PBX models, and this may also be configured. Our definition of flashing lights stems from the signal passed from the PBX to the phone, and is based on factory default configurations for each model. Please refer to your PBX documentation for more information.

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File- TIME for- mat	0-n/a	EVT_CONFERENCE_LIGHT_	*See Below	0-n/a	0-n/a	BoardID	Global- Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

Sub-Reason(this field is only populated when light is flashing or winking):

NOTE: These may vary depending on phone model and PBX.

Bits 0-7 = Light Number

Bits 8 = Green

Bit 9 = Red

Bit 10 = Amber

Bits 11-31 = Reserved

EVT_CONNECTED

Indicates connection

Structure: (DChannel event, reported by the NGX and IPX only)

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME for-mat	0-n/a	EVT_CONNECTED	0-n/a	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_COPY_BUTTON_PRESSED

Indicates that the copy button was pressed

Structure: (DChannel event, reported by the NGX and IPX only)

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME for-mat	0-n/a	EVT_COPY_BUTTON_PRESSED	0-n/a	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_COPY_BUTTON_RELEASED

Indicates that the copy button was released.

Structure: (DChannel event, reported by the NGX and IPX only)

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME for-mat	0-n/a	EVT_COPY_BUTTON_RELEASED	0-n/a	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_CPM_STOP

Indicates a termination condition has been met. If a termination condition has been enabled (using **MTSetTerm()** or *TermEnable* field of the IO_CONTROL structure) it results in an EVT_CPM_STOP. The associated background function(s) are terminated and the channel either enters into an idle state or executes the next background function if one is queued. The termination condition is identified in the Subreason and XtraInfo fields. The media function that has been stopped is identified in the FuncCode field. One CPM_STOP event is generated for each background function that has been stopped. **NOTE: MTCallString()** is a background function. When a termination condition has been met, **MTCallString()** stops and a CPM_STOP event is generated.

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_CPM_STOP	See Below*	See Below*	FuncCode	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

*

Subreason and Xtra Info

This information is also populated in the CpmTerm field of the CALL_RESULT structure. The current background function is terminated and the channel either enters into an idle state or executes the next background function if one is queued. The following table lists all valid termination reasons plus the information presented in the XtraInfo field.:

Subreason (EVT_CPM_STOP)	Xtra Info
CPM_MON_LOOP_DROP	
CPM_MON_SILENCE	
CPM_MON_ACTIVITY	
CPM_MON_DIAL	The Index of the signal
CPM_MON_BUSY*	The Index of the signal (normal or trunk busy)
(using MTCallString()) CPM_CONNECT	CPM_CONNECT_MACHINE CPM_CONNECT_HUMAN
----- (using other background tasks) CPM_MON_HUMAN CPM_MON_MACHINE	-----
CPM_NO_ANSWER	

CPM_MON_INTERCEPT*	the Index of the SIT signal
CPM_MON_MAX_SILENCE CPM_MAX_SILENCE	
CPM_MON_MAX_ACTIVITY CPM_MAX_ACTIVITY	
CPM_MON_LVOLTAGE_DROP	
(same as TERM_LVOLTAGE_ ABOVE_OR_BELOW	
CPM_MON_UTONE	Signal Index ID
CPM_MON_CALLWAITING	
CPM_MON_RECEIVOFF	
CPM_MON_DIAL1	
CPM_MON_DIAL2	
CPM_MON_DIAL3	
CPM_MON_BUSY1	
CPM_MON_BUSY2	
CPM_MON_FAX1	
CPM_MON_FAX2	
CPM_MON_SIT1	
CPM_MON_SIT2	
CPM_MON_SIT3	
CPM_MON_SIT4	
CPM_MON_SIT5	

*This option has been maintained for backwards compatibility.

EVT_CTRL_BUTTON_PRESSED

Indicates that the control button was pressed

Structure: (DChannel event, reported by the NGX and IPX only)

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_CTRL_BUTTON_PRESSED	0-n/a	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_CTRL_BUTTON_RELEASED

Indicates that the control button was released.

Structure: (DChannel event, reported by the NGX and IPX only)

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_CTRL_BUTTON_RELEASED	0-n/a	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_DESTINATION_BUSY

(DChannel event, reported by the NGX and IPX only)

Indicates that the destination called is busy

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_DESTINATION_BUSY	0-n/a	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_DESTINATION_BUTTON_PRESSED

Indicates that the destination button was pressed.

Structure: (DChannel event, reported by the NGX and IPX only)

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME for-mat	0-n/a	EVT_DESTINATION_BUTTON_PRESSED	0-n/a	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_DEVICE_READ_ERROR

Indicates a user device read error

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME for-mat	0-n/a	EVT_DEVICE_READ_ERROR	0-n/a	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_DEVICE_SEEK_ERROR

Indicates a user device seek error

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME for-mat	0-n/a	EVT_DEVICE_SEEK_ERROR	0-n/a	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_DEVICE_WRITE_ERROR

Indicates a user device writer error

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_DEVICE_WRITE_ERROR	0-n/a	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_DIALED

Indicates that dialing is complete

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_DIALED	0-n/a	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_DIGIT

Indicates the received DTMF digit queued. A maximum of 256 digits is allowed in the queue.

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_DIGIT	See Below*	0= outgoing 1= incoming	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

*Byte 0 = ASCII value of the digit pressed

Byte 1-31 = Reserved

EVT_DIGIT_DROPPED

Indicates that a received DTMF digit was dropped due to a full queue. A maximum of 256 digits is allowed in the queue.

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_DIGIT_DROPPED	*See Below	0= outgoing 1= incoming	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

*Bit 0 = ASCII value of the digit pressed

Bits 1-31 = Reserved

EVT_DIGIT_PRESSED

(DChannel event, reported by the NGX and IPX only)

Indicates that a digit button was pressed

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_DIGIT_PRESSED	See Below*	0-n/a	Reserved	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

*Bit 0 = ASCII value of the digit pressed
Bits 1-31 = Reserved

EVT_DIGIT_RELEASED

(DChannel event, reported by the NGX and IPX only)

Indicates that a digit button was released

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_DIGIT_RELEASED	See Below*	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

*Bit 0 = ASCII value of the digit pressed
Bits 1-31 = Reserved

EVT_DIGITENABLE

Indicates termination of a background function by an enabled DTMF digit

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_DIGITENABLE	Termination Digit	0-n/a	FuncCode	BoardID	Global-Channel ID (GCI)	See Below*	See Below*
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

ptrBuffer - pointer to either a file or buffer (depending on which API was called, playing a file or from a buffer)

DataLength - length (bytes) of exercised data at termination

EVT_DIRECTORY_BUTTON_PRESSED

(DChannel event, reported by the NGX and IPX only)

Description: Indicates that the directory button has been pressed

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_DIRECTORY_BUTTON_PRESSED	0-n/a	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_DIRECTORY_BUTTON_RELEASED

(DChannel event, reported by the NGX and IPX only)

Description: Indicates that the directory button has been released.

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME for-mat	0-n/a	EVT_DIRECTORY_BUTTON_RELEASED	0-n/a	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_DIRECTORY_LIGHT_FASTFLASHING
EVT_DIRECTORY_LIGHT_VERY_FASTFLASHING
EVT_DIRECTORY_LIGHT_FLASHING
EVT_DIRECTORY_LIGHT_OFF
EVT_DIRECTORY_LIGHT_ON
EVT_DIRECTORY_LIGHT_QUICKFLASH

(DChannel event, reported by the NGX and IPX only)

Purpose:

Indicates that the directory light on the phone is on, off, or flashing.

Description:

Observation of various phone models enabled the AudioCodes lab to categorize phone lights as flashing, fast flashing, very fast flashing or quick flashing. The cadence varies depending on PBX models, and this may also be configured. Our definition of flashing lights stems from the signal passed from the PBX to the phone, and is based on factory default configurations for each model. Please refer to your PBX documentation for more information.

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File- TIME for- mat	0-n/a	EVT_DIRECTORY_LIGHT_	*See Below	0-n/a	0-n/a	BoardID	Global- Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

Sub-Reason(this field is populated only when light is flashing):

NOTE: These may vary depending on phone model and PBX.

Bits 0-7 = Light Number

Bits 8 = Green

Bit 9 = Red

Bit 10 = Amber

Bits 11-31 = Reserved

EVT_DISCONNECTED

(DChannel event, reported by the NGX and IPX only)

Indicates a disconnect

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME for-mat	0-n/a	EVT_DISCONNECTED	0-n/a	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_DISK_FULL

Indicates that the disk is full

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME for-mat	0-n/a	EVT_DISK_FULL	0-n/a	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_DISK_WRITE_ERROR

Indicates that a file write error occurred

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME for-mat	0-n/a	EVT_DISK_WRITE_ERROR	0-n/a	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_DISPLAY_CLEAR

(DChannel event, reported by the NGX and IPX only)

Indicates that the phone display has been cleared

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_DISPLAY_CLEAR	0-n/a	0-n/a*	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

*Several of the Bosch Integral phones support a multi-windowed display. The X-tra info field specifies which window the display event has occurred in.

EVT_DISPLAY_CLOCK

(DChannel event, reported by the NGX and IPX only)

Indicates that the PBX has told the phone to display the clock (usually date and time)

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_DISPLAY_CLOCK	0-n/a	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	*See below	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

*Pointer to a tm structure indicating the pbx time and date. User applications must call **MTReturnEventBuffer()** if ptrBuffer is valid. This clears the buffer. ptrBuffer may not be available on all pbxs.

EVT_DISPLAY_MESSAGE

(DChannel event, reported by the NGX and IPX only)

Purpose:

Indicates that the PBX has commanded the phone to display a message or phrase. In most cases the phone is programmed with the messages, and the PBX passes over a message ID. Only three lines of LCD text is passed over to the event files. Any messages larger than that will be truncated. When other data is displayed on the LCD, ie phone number, agent id, the EVT_MESSAGE_CHANGE event is reported.

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_DISPLAY_MESSAGE	MSG_ID/Window ID*	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

*This 32-bit field displays the Message ID in the lower 16 bits. On PBXs which have LCDs with multiple "windows", the window ID is passed over in the upper 16 bits.

EVT_DISPLAY_TIMER

(DChannel event, reported by the NGX and IPX only)

Indicates that the PBX has told the phone to display the timer (displaying the call duration).

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_DISPLAY_TIMER	0-n/a	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_DIVERT_BUTTON_PRESSED

(DChannel event, reported by the NGX and IPX only)

Indicates that the divert button was pressed.

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File- TIME for- mat	0-n/a	EVT_DIVERT_ BUTTON_PRESSED	0-n/a	0-n/a	0-n/a	BoardID	Global- Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_DIVERT_LIGHT_FASTFLASHING
EVT_DIVERT_LIGHT_VERY_FASTFLASHING
EVT_DIVERT_LIGHT_FLASHING
EVT_DIVERT_LIGHT_OFF
EVT_DIVERT_LIGHT_ON
EVT_DIVERT_LIGHT_QUICKFLASH

(DChannel event, reported by the NGX and IPX only)

Purpose:

Indicates that the divert light on the phone is on, off, or flashing.

Description:

Observation of various phone models enabled the AudioCodes lab to categorize phone lights as flashing, fast flashing, very fast flashing or quick flashing. The cadence varies depending on PBX models, and this may also be configured. Our definition of flashing lights stems from the signal passed from the PBX to the phone, and is based on factory default configurations for each model. Please refer to your PBX documentation for more information.

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME for-mat	0-n/a	EVT_DIVERT_LIGHT_	*See Below	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

Sub-Reason(this field is populated only when light is flashing):

NOTE: These may vary depending on phone model and PBX.

Bits 0-7 = Light Number

Bits 8 = Green

Bit 9 = Red

Bit 10 = Amber

Bits 11-31 = Reserved

EVT_DONGLE_ADDED

(reported by the HPX only)

Indicates that a dongle license lock has been installed to the host system.

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File- TIME for- mat	0-n/a	EVT_DONGLE_ADDED	The dongle serial #	0-n/a	0-n/a	BoardID	0- n/a	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_DONGLE_REMOVED

(reported by the HPX only)

Indicates that a dongle license lock has been removed from the USB port of the host system.

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File- TIME for- mat	0-n/a	EVT_DONGLE_REMOVED	The dongle serial #	0-n/a	0-n/a	BoardID	0- n/a	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_DND_BUTTON_PRESSED

(DChannel event, reported by the NGX and IPX only)

Indicates that the do not disturb button was pressed.

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File- TIME for- mat	0-n/a	EVT_DND_ BUTTON_PRESSED	0-n/a	0-n/a	0-n/a	BoardID	Global- Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_DND_LIGHT_FASTFLASHING
EVT_DND_LIGHT_VERY_FASTFLASHING
EVT_DND_LIGHT_FLASHING
EVT_DND_LIGHT_OFF
EVT_DND_LIGHT_ON
EVT_DND_LIGHT_QUICKFLASH
EVT_DND_LIGHT_MEDIUM_WINK
EVT_DND_LIGHT_WINK

(DChannel event, reported by the NGX and IPX only)

Purpose:

Indicates that the do not disturb light on the phone is on, off, or flashing.

Description:

Observation of various phone models enabled the AudioCodes lab to categorize phone lights as flashing, fast flashing, very fast flashing, quick flashing or winking. The cadence varies depending on PBX models, and this may also be configured. Our definition of flashing lights stems from the signal passed from the PBX to the phone, and is based on factory default configurations for each model. Please refer to your PBX documentation for more information.

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File- TIME for- mat	0-n/a	EVT_DND_LIGHT_	*See Below	0-n/a	0-n/a	BoardID	Global- Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

Sub-Reason(this field is populated only when light is flashing or winking):

NOTE: These may vary depending on phone model and PBX.

Bits 0-7 = Light Number

Bits 8 = Green

Bit 9 = Red

Bit 10 = Amber

Bits 11-31 = Reserved

EVT_DOWNLOAD_STATUS

Not supported. Although this event exists in the header file, it is not currently supported by the SmartWORKS API.

EVT_DSPRESET

Indicates that the DSPs just reset

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_DSPRESET	0-n/a	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_ENTER_BUTTON_PRESSED

(DChannel event, reported by the NGX and IPX only)

Indicates that the enter button has been pressed

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_ENTER_BUTTON_PRESSED	0-n/a	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_EOF

Indicates that the end of file was reached on playback.

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_EOF	0-n/a	0-n/a	FuncCode	BoardID	Global-Channel ID (GCI)	file handle	length of exercised data
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_ERROR

Indicates a hardware failure

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_ERROR	0-n/a	Internal Error Code	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_ETH_DEST_UNREACHABLE

(Used by IPX only). Indicates that the RTP media destination is unreachable when an ARP command that was forwarded to an RTP destination device has failed. This event is generated per a single stream (half-duplex of the conversation).

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_ETH_DEST_UNREACHABLE	See below	Station ID	0-n/a	BoardID	Global-Channel ID (GCI)	file handle	length of exercised data
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

Subreason:

ptrBuffer:

The MT_IP_DEST_UNREACHABLE_INFO data structure is passed over to user:

- ULONG SessionId; - Session ID associated with this error
- ULONG IpAddr; IP Address of the unreachable destination device
- USHORT Port; - port number on the destination device that this stream was supposed to be forwarded to
- MT_IP_FWD_DEST FwdDest; - Primary = 1, Secondary = 2

EVT_EXIT_BUTTON_PRESSED

(DChannel event, reported by the NGX and IPX only)

Indicates that the Exit button has been pressed

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME for-mat	0-n/a	EVT_EXIT_BUTTON_PRESSED	0-n/a	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_EXIT_BUTTON_RELEASED

Indicates that the Exit button has been released

Structure: (DChannel event, reported by the NGX and IPX only)

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_EXIT_BUTTON_RELEASED	0-n/a	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_EXPAND_BUTTON_PRESSED

Indicates that the expand button has been pressed

Structure: (DChannel event, reported by the NGX and IPX only)

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_EXPAND_BUTTON_PRESSED	0-n/a	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_EXPAND_BUTTON_RELEASED

Indicates that the expand button has been released

Structure: (DChannel event, reported by the NGX and IPX only)

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_EXPAND_BUTTON_RELEASED	0-n/a	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_FAXRECVEOP
EVT_FAXSEND_DONE
EVT_FAXRECV_DONE
EVT_FAX_ERROR
EVT_FAX_PHASEB_COMP
EVT_FAX_PHASED_COMP

Although these are documented in the header file, these events are not supported by the API.

EVT_FEATURE_BUTTON_PRESSED

(DChannel event, reported by the NGX and IPX only)
Indicates that a feature button has been pressed
Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME for-mat	0-n/a	EVT_FEATURE_BUTTON_PRESSED	0-n/a	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_FEATURE_LIGHT_FASTFLASHING
EVT_FEATURE_LIGHT_FLASHING
EVT_FEATURE_LIGHT_ON
EVT_FEATURE_LIGHT_OFF
EVT_FEATURE_LIGHT_QUICKFLASHING
EVT_FEATURE_LIGHT_VERY_FASTFLASHING

(DChannel event, reported by the NGX and IPX only)

Purpose:

Indicates that a feature light on the phone is on, off or flashing.

Description:

Observation of various phone models enabled the AudioCodes lab to categorize phone lights as flashing, fast flashing, very fast flashing or quick flashing. The cadence varies depending on PBX models, and this may also be configured. Our definition of flashing lights stems from the signal passed from the PBX to the phone, and is based on factory default configurations for each model. Please refer to your PBX documentation for more information.

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME for-mat	0-n/a	EVT_FEATURE_LIGHT_	*See Below	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

Sub-Reason(this field is only populated when the light is flashing):

NOTE: These may vary depending on phone model and PBX.

Bits 0-7 = Light Number

Bits 8 = Green

Bit 9 = Red

Bit 10 = Amber

Bits 11-31 = Reserved

Alcatel phones do not have lights, instead pictures are presented on the phone display. The following bits are set:

Bits 0-7 = "Light" Number

Bits 8 = square, located on the right side of the display area

Bit 9 = music symbol
 Bit 10 = square, located in the center of the display area
 Bit 11 = handset symbol
 Bit 12 = square, located on the left side of the display area
 Bit 13 = alert symbol
 Bits 14=31 Reserved

EVT_FINISHED_PLAY

Not supported. Although this event exists in the header file, it is not currently supported by the SmartWORKS API.

EVT_FEATURE_HEADER_ERROR

Indicates a frame header error, terminating the function

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME for-mat	0-n/a	EVT_FEATURE_HEADER_ERROR	0-n/a	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_FLASH_BUTTON_PRESSED

(DChannel event, reported by the NGX and IPX only)

Description: Indicates that the flash button has been pressed

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME for-mat	0-n/a	EVT_FLASH_BUTTON_PRESSED	0-n/a	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_FLASH_BUTTON_RELEASED

(DChannel event, reported by the NGX and IPX only)

Description: Indicates that the flash button has been released

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File- TIME for- mat	0-n/a	EVT_FLASH_BUTTON_ RELEASED	0-n/a	0-n/a	0-n/a	BoardID	Global- Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_FLASH_LIGHT_FASTFLASHING
EVT_FLASH_LIGHT_VERY_FASTFLASHING
EVT_FLASH_LIGHT_FLASHING
EVT_FLASH_LIGHT_OFF
EVT_FLASH_LIGHT_ON
EVT_FLASH_LIGHT_QUICKFLASH
EVT_FLASH_LIGHT_MEDIUM_WINK

(DChannel event, reported by the NGX and IPX only)

Purpose:

Indicates that the flash light on the phone is on, off, or flashing.

Description:

Observation of various phone models enabled the AudioCodes lab to categorize phone lights as flashing, fast flashing, very fast flashing, quick flashing or winking. The cadence varies depending on PBX models, and this may also be configured. Our definition of flashing lights stems from the signal passed from the PBX to the phone, and is based on factory default configurations for each model. Please refer to your PBX documentation for more information.

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File- TIME for- mat	0-n/a	EVT_FLASH_LIGHT_	*See Below	0-n/a	0-n/a	BoardID	Global- Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

Sub-Reason(this field is populated only when light is flashing or winking):

NOTE: These may vary depending on phone model and PBX.

Bits 0-7 = Light Number
 Bits 8 = Green
 Bit 9 = Red
 Bit 10 = Amber
 Bits 11-31 = Reserved

EVT_FORWARD_BUTTON_PRESSED

Description: Indicates that the forward button has been pressed

Structure: (DChannel event, reported by the NGX and IPX only)

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_FORWARD_BUTTON_PRESSED	0-n/a	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_FRAME_DATA_ERROR

Indicates that a background function (playback) has terminated due to a frame data error

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_FEATURE_BUTTON_PRESSED	0-n/a	0-n/a	FuncCode	BoardID	Global-Channel ID (GCI)	file handle	length of exercised data
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_FUNCTION_BUTTON_RELEASED

Indicates that a function button has been released

Structure: (DChannel event, reported by the NGX and IPX only)

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_FUNCTION_BUTTON_PRESSED	*see below	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

* Bit 0 = Button number 0-255
Bits 1-31 = Reserved

EVT_FUNCTION_BUTTON_PRESSED

(DChannel event, reported by the NGX and IPX only)

Description: Indicates that the function button has been pressed

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME for-mat	0-n/a	EVT_FUNCTION_BUTTON_PRESSED	0-n/a	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_FUNCTION_LIGHT_FASTFLASHING
EVT_FUNCTION_LIGHT_FLASHING
EVT_FUNCTION_LIGHT_OFF
EVT_FUNCTION_LIGHT_ON
EVT_FUNCTION_LIGHT_QUICKFLASH
EVT_FUNCTION_LIGHT_VERY_FASTFLASHING
EVT_FUNCTION_LIGHT_WINK
EVT_FUNCTION_LIGHT_SLOW_WINK
EVT_FUNCTION_LIGHT_MEDIUM_WINK

(DChannel event, reported by the NGX and IPX only)

Purpose:

Indicates that the function light on the phone is on, off or flashing.

Description:

Observation of various phone models enabled the AudioCodes lab to categorize phone lights as flashing, fast flashing, very fast flashing or quick flashing. The cadence varies depending on PBX models, and this may also be configured. Our definition of flashing lights stems from the signal passed from the PBX to the phone, and is based on factory default configurations for each model. Please refer to your PBX documentation for more information. The "WINK" events were added with SmartWORKS release 2.10.0, and are supported on the Siemens / Rolm IBM 9751. There is a difference in on/off patterns when comparing winks to flashes. The subreason fields will be the same as all other light events.

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File- TIME for- mat	0-n/a	EVT_FUNCTION_LIGHT_	*See Below	0-n/a	0-n/a	BoardID	Global- Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

Sub-Reason(this field is populated only when light is flashing):

NOTE: These may vary depending on phone model and PBX.

Bits 0-7 = Light Number

Bits 8 = Green

Bit 9 = Red

Bit 10 = Amber

Bits 11-31 = Reserved (~ more information on nextpage~)

Alcatel phones do not have lights, instead pictures are presented on the phone display. The following bits are set:

Bits 0-7 = "Light" Number

Bits 8 = square, located on the right side of the display area

Bit 9 = music symbol

Bit 10 = square, located in the center of the display area

Bit 11 = handset symbol

Bit 12 = square, located on the left side of the display area

Bit 13 = alert symbol

Bits 14=31 Reserved

EVT_GROUP_BUTTON_PRESSED

(DChannel event, reported by the NGX and IPX only)

Indicates that the group button was pressed.

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME for-mat	0-n/a	EVT_GROUP_BUTTON_PRESSED	0-n/a	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_GROUP_LIGHT_FASTFLASHING
EVT_GROUP_LIGHT_VERY_FASTFLASHING
EVT_GROUP_LIGHT_FLASHING
EVT_GROUP_LIGHT_OFF
EVT_GROUP_LIGHT_ON
EVT_GROUP_LIGHT_QUICKFLASH

(DChannel event, reported by the NGX and IPX only)

Purpose:

Indicates that the group light on the phone is on, off, or flashing.

Description:

Observation of various phone models enabled the AudioCodes lab to categorize phone lights as flashing, fast flashing, very fast flashing or quick flashing. The cadence varies depending on PBX models, and this may also be configured. Our definition of flashing lights stems from the signal passed from the PBX to the phone, and is based on factory default configurations for each model. Please refer to your PBX documentation for more information.

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File- TIME for- mat	0-n/a	EVT_GROUP_LIGHT_	*See Below	0-n/a	0-n/a	BoardID	Global- Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

Sub-Reason(this field is populated only when light is flashing):

NOTE: These may vary depending on phone model and PBX.

Bits 0-7 = Light Number

Bits 8 = Green

Bit 9 = Red

Bit 10 = Amber

Bits 11-31 = Reserved

EVT_H100_MASTER_A_ALARM

Indicates H100 Master A clock error detection event. This event will be generated when it has been armed by an Arm H.100 Clock A Error Event command, and CT812 reports an error condition on line CT_C8_A on the H.100 bus.

This event is only applicable when the switching handler is in H.100 mode.

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File- TIME for- mat	0-n/a	EVT_H100_MASTER_A_ ALARM	0-n/a	0-n/a	0-n/a	BoardID	Global- Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_H100_MASTER_B_ALARM

Indicates H100 Master B clock error detection. This event will be generated when it has been armed by an Arm H.100 Clock B Error Event command, and CT812 reports an error condition on line CT_C8_B on the H.100 bus.

This event is only applicable when the switching handler is in H.100 mode.

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File- TIME for- mat	0-n/a	EVT_H100_MASTER_B_ ALARM	0-n/a	0-n/a	0-n/a	BoardID	Global- Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_HANDSFREE_BUTTON_PRESSED

(DChannel event, reported by the NGX and IPX only)

Description: Indicates that the handsfree button has been pressed

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File- TIME for- mat	0-n/a	EVT_HANDSFREE_BUTTON _PRESSED	0-n/a	0-n/a	0-n/a	BoardID	Global- Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_HANDSFREE_BUTTON_RELEASED

(DChannel event, reported by the NGX and IPX only)

Description: Indicates that the handsfree button has been released.

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File- TIME for- mat	0-n/a	EVT_HANDSFREE_BUTTON _RELEASED	0-n/a	0-n/a	0-n/a	BoardID	Global- Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_HANDSFREE_LIGHT_FASTFLASHING
EVT_HANDSFREE_LIGHT_VERY_FASTFLASHING
EVT_HANDSFREE_LIGHT_FLASHING
EVT_HANDSFREE_LIGHT_OFF
EVT_HANDSFREE_LIGHT_ON
EVT_HANDSFREE_LIGHT_QUICKFLASH

(DChannel event, reported by the NGX and IPX only)

Purpose:

Indicates that the handsfree light on the phone is on, off, or flashing.

Description:

Observation of various phone models enabled the AudioCodes lab to categorize phone lights as flashing, fast flashing, very fast flashing or quick flashing. The cadence varies depending on PBX models, and this may also be configured. Our definition of flashing lights stems from the signal passed from the PBX to the phone, and is based on factory default configurations for each model. Please refer to your PBX documentation for more information.

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME for-mat	0-n/a	EVT_HANDSFREE_LIGHT_	*See Below	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

Sub-Reason(this field is populated only when light is flashing):

NOTE: These may vary depending on phone model and PBX.

Bits 0-7 = Light Number

Bits 8 = Green

Bit 9 = Red

Bit 10 = Amber

Bits 11-31 = Reserved

EVT_HEADSET_BUTTON_PRESSED

(DChannel event, reported by the NGX and IPX only)

Description: Indicates that the headset button has been pressed

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_HEADSET_BUTTON_PRESSED	0-n/a	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_HEADSET_BUTTON_RELEASED

(DChannel event, reported by the NGX and IPX only)

Description: Indicates that the headset button has been released.

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_HEADSET_BUTTON_RELEASED	0-n/a	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_HEADSET_LIGHT_OFF EVT_HEADSET_LIGHT_ON

(DChannel event, reported by the NGX and IPX only)

Purpose:

Indicates that the headset light on the phone is on, off, or flashing.

Description:

Observation of various phone models enabled the AudioCodes lab to categorize phone lights as flashing, fast flashing, very fast flashing or quick flashing. The cadence varies depending on PBX models, and this may also be configured. Our definition of flashing lights stems from the signal passed from the PBX to the phone, and is based on factory default configurations for each model. Please refer to your PBX documentation for more information.

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME for-mat	0-n/a	EVT_HEADSET_LIGHT_	*See Below	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

Sub-Reason(this field is populated only when light is flashing):

NOTE: These may vary depending on phone model and PBX.

Bits 0-7 = Light Number

Bits 8 = Green

Bit 9 = Red

Bit 10 = Amber

Bits 11-31 = Reserved

EVT_HELP_BUTTON_PRESSED

(DChannel event, reported by the NGX and IPX only)

Indicates that the help button has been pressed

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME for-mat	0-n/a	EVT_HELP_BUTTON_PRESSED	0-n/a	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_HELP_LIGHT_FASTFLASHING
EVT_HELP_LIGHT_FLASHING
EVT_HELP_LIGHT_OFF
EVT_HELP_LIGHT_ON
EVT_HELP_LIGHT_QUICKFLASH
EVT_HELP_LIGHT_VERY_FASTFLASHING

(DChannel event, reported by the NGX and IPX only)

Purpose:

Indicates that the help light on the phone is on, off or flashing.

Description:

Observation of various phone models enabled the AudioCodes lab to categorize phone lights as flashing, fast flashing, very fast flashing or quick flashing. The cadence varies depending on PBX models, and this may also be configured. Our definition of flashing lights stems from the signal passed from the PBX to the phone, and is based on factory default configurations for each model. Please refer to your PBX documentation for more information.

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME for-mat	0-n/a	EVT_HELP_LIGHT_	*See Below	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

Sub-Reason(this field is populated only when light is flashing):

NOTE: These may vary depending on phone model and PBX.

Bits 0-7 = Light Number

Bits 8 = Green

Bit 9 = Red

Bit 10 = Amber

Bits 11-31 = Reserved

EVT_HOLD_BUTTON_PRESSED

(DChannel event, reported by the NGX and IPX only)

Indicates that the hold button has been pressed

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File- TIME for- mat	0-n/a	EVT_HOLD_BUTTON_ PRESSED	0-n/a	0-n/a	0-n/a	BoardID	Global- Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_HOLD_BUTTON_RELEASED

(DChannel event, reported by the NGX and IPX only)

Indicates that the hold button has been released

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File- TIME for- mat	0-n/a	EVT_HOLD_BUTTON_ RELEASED	0-n/a	0-n/a	0-n/a	BoardID	Global- Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_HOLD_LIGHT_FASTFLASHING
EVT_HOLD_LIGHT_FLASHING
EVT_HOLD_LIGHT_OFF
EVT_HOLD_LIGHT_ON
EVT_HOLD_LIGHT_QUICKFLASH
EVT_HOLD_LIGHT_VERY_FASTFLASHING
EVT_HOLD_LIGHT_WINK

(DChannel event, reported by the NGX and IPX only)

Purpose:

Indicates that the hold light on the phone is on, off or flashing.

Description:

Observation of various phone models enabled the AudioCodes lab to categorize phone lights as flashing, fast flashing, very fast flashing or quick flashing. The cadence varies depending on PBX models, and this may also be configured. Our definition of flashing lights stems from the signal passed from the PBX to the phone, and is based on factory default configurations for each model. Please refer to your PBX documentation for more information.

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File- TIME for- mat	0-n/a	EVT_HOLD_LIGHT_	*See Below	0-n/a	0-n/a	BoardID	Global- Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

Sub-Reason(this field is populated only when light is flashing):

NOTE: These may vary depending on phone model and PBX.

Bits 0-7 = Light Number

Bits 8 = Green

Bit 9 = Red

Bit 10 = Amber

Bits 11-31 = Reserved

EVT_IDLE

Indicates that the channel is in the IDLE state after the channel has flushed all operations

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME for-mat	0-n/a	EVT_IDLE	0-n/a	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_INCOMING_SIGCHANGE

Indicates a Change of Signaling Bits (ABCD) on incoming line

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME for-mat	0-n/a	EVT_INCOMING_SIGCHANGE	See Below*	See Below*	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	See Below*
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

***.SubReason:**

an octect the new values of the signaling bits A, B, C, and D, in bits 3, 2, 1, and 0, respectively. Bits 7, 6, 5, and 4 contain the previous values of the signaling bits.

XtraInfo:

the number of the framer reporting the event (least significant bit)

the timeslot on which the change was detected(most significant bit)

DataLength:

the duration of the previous state in 10 ms units

EVT_INDEX_PLAYED

Indicates the indexed byte length played

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_INDEX_PLAYED	0-n/a	0-n/a	FuncCode	BoardID	Global-Channel ID (GCI)	file handle	length of exercised data
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_INTERCOM_BUTTON_PRESSED

(DChannel event, reported by the NGX and IPX only)

Indicates that the intercom button has been pressed

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_INTERCOM_BUTTON_PRESSED	0-n/a	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_INTERCOM_LIGHT_FASTFLASHING
EVT_INTERCOM_LIGHT_FLASHING
EVT_INTERCOM_LIGHT_ON
EVT_INTERCOM_LIGHT_OFF
EVT_INTERCOM_LIGHT_QUICKFLASH
EVT_INTERCOM_LIGHT_VERY_FASTFLASHING

(Indicates that the intercom light on the phone is on, off, or flashing.

Structure: DChannel event, reported by the NGX and IPX only)

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME for-mat	0-n/a	EVT_INTERCOM_LIGHT_	*See below	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

* **Subreason: (This field is only populated when the light is flashing)**

Bits 0-7 = Light number

Bit 8 = Green light

Bit 9 = Red light

Bits 10-31 = Reserved

EVT_INTERDIGIT

Indicates termination of a background media function because the maximum Interdigit delay has been exceeded

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME for-mat	0-n/a	EVT_INTERDIGIT	0-n/a	0-n/a	FuncCode	BoardID	Global-Channel ID (GCI)	See Below*	See Below*
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

ptrBuffer - pointer to either a file or buffer (depending on which API was called, playing a file or from a buffer)

DataLength - length (bytes) of exercised data at termination

EVT_ISDN_SUPPL_SERVICES

Indicates that more information is available relative to ISDN events.

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_ISDN_SUPPL_SERVICES	0-n/a	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

The following data structure is passed over with this event:

```
typedef struct
```

```
{  
    ULONG CallRef;  
    MT_CC_CHANNEL_ID ChannelId;  
    ULONG NetworkId; /* the fields following NetworkId are network-specific */  
    ULONG NoldInfoCause;  
} MT_ISDN_SUPPL_SERVICES, *PMT_ISDN_SUPPL_SERVICES;
```

```
// Network Identification Values
```

```
#define NETWORK_ID_UNKNOWN 0x0
```

```
#define NETWORK_ID_EURO_ISDN 0x1
```

```
#define NETWORK_ID_NAT_ISDN1 0x2
```

```
#define NETWORK_ID_JAPAN_INS 0x3
```

```
// Japan INS - No Id Information Cause
```

```
#define INS_NO_ID_INFO_CAUSE_NOT_AVAILABLE 0x0
```

```
#define INS_NO_ID_INFO_CAUSE_REJECTED_BY_USER 0x1
```

```
#define INS_NO_ID_INFO_CAUSE_OTHER_SERVICES 0x2
```

```
#define INS_NO_ID_INFO_CAUSE_COIN_LINE 0x3
```

EVT_LIGHT_FASTFLASHING
EVT_LIGHT_FLASHING
EVT_LIGHT_ON
EVT_LIGHT_OFF
EVT_LIGHT_QUICKFLASH
EVT_LIGHT_VERY_FASTFLASHING

(DChannel event, reported by the NGX and IPX only)
 Indicates that the light on the phone is on, off, or flashing.

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File- TIME for- mat	0-n/a	EVT_LIGHT_	*See below	0-n/a	0-n/a	BoardID	Global- Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

*** Subreason: (This field is only populated when the light is flashing)**

Bits 0-7 = Light number
 Bit 8 = Green light
 Bit 9 = Red light
 Bits 10-31 = Reserved

EVT_LINE_BUTTON_PRESSED

(DChannel event, reported by the NGX and IPX only)
 Indicates that the line button has been pressed

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File- TIME for- mat	0-n/a	EVT_LINE_BUTTON_ PRESSED	0-n/a	0-n/a	0-n/a	BoardID	Global- Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_LINE_LIGHT_FASTFLASHING
EVT_LINE_LIGHT_FLASHING
EVT_LINE_LIGHT_ON
EVT_LINE_LIGHT_OFF
EVT_LINE_LIGHT_QUICKFLASH
EVT_LINE_LIGHT_VERY_FASTFLASHING
EVT_LINE_LIGHT_WINK

(DChannel event, reported by the NGX and IPX only)

Purpose:

Indicates that the line light on the phone is on, off or flashing.

Description:

Observation of various phone models enabled the AudioCodes lab to categorize phone lights as flashing, fast flashing, very fast flashing or quick flashing. The cadence varies depending on PBX models, and this may also be configured. Our definition of flashing lights stems from the signal passed from the PBX to the phone, and is based on factory default configurations for each model. Please refer to your PBX documentation for more information.

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File- TIME for- mat	0-n/a	EVT_LINE_LIGHT_	*see below	0-n/a	0-n/a	BoardID	Global- Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

These may vary depending on phone model and PBX. The subreason field is only populated when the light is flashing.

*Bits 8 = Green

Bit 9 = Red

Bit 10 = Amber

Bits 11-31 = Reserved

EVT_LINE_STATE_CHANGED

(DChannel event, reported by the NGX only)

Some phones may have multi line keys. The user may make these lines "active", "on hold" or "inactive". i.e. line 1 "on hold", line 2 "active", and line 3 "inactive". Currently, this event can handle up to 4 lines. When this event is reported, the status of all 4 lines is reported.

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME for-mat	0-n/a	EVT_LINE_STATE_CHANGED	See Below*	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

***Subreason** - The sub reason uses low 8 bits. The high 24 bits are reserved. The low 8-bit of sub reason was divided to 4 groups. Bit 1~0 tells the status of line 1; Bit 3~2 tells the status of line 2; Bit 5~4 tells the status of line 3; and Bit 7~6 tells the status of line 4. "00" means "inactive"; "11" means "active"; "01" means "on hold"; and "10" is reserved.

EVT_LOCRCMF_ALARM

Indicates a LOSS ofCRC Multiframe Alignment alarm (E1 only). For these events to generate, framer alarms must be enabled using the **MTArmFramerAlarm()** API.

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_LOCRCMF_ALARM	0-n/a	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_LOF_ALARM

Indicates a Loss of Framer alarm on T1 or E1 networks. For these events to generate, framer alarms must be enabled using the **MTArmFramerAlarm()** API.

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_LOF_ALARM	An octet containing the number of the framer reporting the event	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_LOOP_DROP

Indicates a Loop current dropped, meaning a phone went back on hook

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_LOOP_DROP	See Below*	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	See Below*
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

SubReason:

Current & Previous state

DataLen:

Duration of previous state in ms.

EVT_LOOP_ON

Indicates a Loop current on event, meaning a phone is off hook

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_LOOP_ON	See Below*	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	See Below*
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

SubReason:

Current & Previous state

DataLen:

Duration of previous state in ms.

EVT_LOOP_REVERSE

Indicates a Loop current battery reversal (current flow is ring to tip)

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_LOOP_REVERSE	See Below*	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	See Below*
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

SubReason:

Current & Previous state

DataLen:

Duration of previous state in ms

EVT_LOOP_STOP

Indicates termination of background media function due to a drop in loop current (phone went back on hook)

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_LOOP_STOP	0-n/a	0-n/a	FuncCode	BoardID	Global-Channel ID (GCI)	Reserved	Reserved
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_LOOPON_RECORD

Indicates recording started on detection of loop current

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_LOOPON_RECORD	0-n/a	0-n/a	FuncCode	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_LOSYNC_ALARM

Indicates a Loss of synchronization alarm on the SmartWORKS PCM framer. This event is only generated if framer alarms are enabled using the **MTArmFramerAlarm()** API.

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_LOSYNC_ALARM	An octet containing the number of the framer reporting the event	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_LOS_ALARM

Indicates a Loss of Signal alarm on the framer. This event is only generated if framer alarms are enabled using the *MTArmFramerAlarm()* API.

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtrInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_LOS_ALARM	An octet containing the number of the framer reporting the event	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_LOSMF_ALARM

Indicates loss of signaling multiframe alignment alarm (E1 only). This event is only generated if framer alarms are enabled using the **MTArmFramerAlarm()** API.

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_LOSMF_ALARM	An octet containing the number of the framer reporting the event	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_LVOLTAGE_ABOVE

Indicates a loop voltage has been detected that falls above the set threshold (ThresholdHigh parameters in the MTSetLVParams() API). This event is only generated by the SmartWORKS LD board.

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_LVOLTAGE_ABOVE	See Below*	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	See Below*
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

***SubReason:**

Current & Previous voltage state

DataLen:

Duration of previous state in 10 ms increments

EVT_LVOLTAGE_BELOW

Indicates a loop voltage detected that falls above the below the set threshold (LowThreshold parameter used by the MTSetLVParams() API). This event is only generated by the SmartWORKS LD board.

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_LVOLTAGE_BELOW	See Below*	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	See Below*
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

***SubReason:**

Current & Previous voltage state

DataLen:

Duration of previous state in 10 ms increments

EVT_LVOLTAGE_MIDDLE

Indicates a loop voltage detected that falls between the set thresholds (LowThreshold and HighThreshold parameters used by the MTSetLVParams() API). This event is only generated by the SmartWORKS LD board. This indicates that the phone is on hook.

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_LVOLTAGE_MIDDLE	See Below*	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	See Below*
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

***SubReason:**

Current & Previous voltage state

DataLen:

Duration of previous state in 10 ms increments

EVT_LVOLTAGE_NOTOFFHOOK

EVT_LVOLTAGE_ABOVEORBELOW

Indicates a loop voltage state that is not off hook (it is either above or below the threshold high or threshold low range).

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File- TIME for- mat	0-n/a	EVT_LVOLTAGE_ NOTOFFHOOK	0-n/a	0-n/a	0-n/a	BoardID	Global- Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

***SubReason:**

Current & Previous voltage state

DataLen:

Duration of previous state in 10 ms increments

EVT_LVOLTAGE_OFFHOOK

EVT_LVOLTAGE_MIDDLE

Indicates a loop voltage detected

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_LVOLTAGE_OFFHOOK	0-n/a	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_LVOLTAGE_RECORD

Indicates recording started on detection of offhook loop voltage. On the SmartWORKS LD board this occurs when the event EVT_LVOLTAGE_ABOVE is generated.

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_LVOLTAGE_RECORD	0-n/a	0-n/a	FuncCode	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_MAPPED_STATION_DETECTED

Reported by the IPX and HPX when operating in “target mode”. Indicates that a station on the station mapping list is visible to the IPX or HPX products.

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_MAPPED_STATION_DETECTED	n/a	see below*	n/a	BoardID	n/a	pointer to data structure	length of data
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

*The station ID and protocol ID.

NOTE: To obtain more information for this VoIP endpoint, use the *MTIpGetStationParams()* function.

EVT_MAPPED_STATION_TIMEOUT

Reported by the IPX and HPX when operating in “target mode”. Indicates that a station on the station mapping list is no longer visible to the HPX/IPX products.

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File- TIME for- mat	0-n/a	EVT_MAPPED_STATION_ TIMEOUT	n/a	see below*	n/a	BoardID	n/a	pointer to data structure	length of data
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

*The station ID and protocol ID.

NOTE: To obtain more information for this VoIP endpoint, use the *MTIpGetStationParams()* function.

EVT_MASTER_PLL_ALARM

Indicates master PLL error detected. This event will be generated when it has been armed by an Arm Master PLL Error Event command, and CT812 reports a Master PLL error condition.

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_MASTER_PLL_ALARM	0-n/a	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_MAX_ACTIVITY

Issued by a channel device when activity detection is enabled and the activity timer has reached its specified maximum value. Upon issuing this event, the activity timer is reset to zero (0).

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_MAX_ACTIVITY	See Below*	0= outgoing 1= incoming	FuncCode	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

*Timing Information - displays the amount of time (in ms) the line remained in the of the previous state. Information is presented in 1ms units, with a maximum of 65535ms.

EVT_MAX_SILENCE

Issued by a channel device when activity detection is enabled and the silence timer has reached its specified maximum. Upon issuing this event, the silence timer is reset to zero (0).

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_MAX_SILENCE	See Below*	0= outgoing 1= incoming	FuncCode	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

*Timing Information - displays the amount of time (in ms) the line remained in the of the previous state. Information is presented in 1ms units, with a maximum of 65535ms.

EVT_MAXBYTES

Indicates maximum bytes reached on play or record

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_MAXBYTES	0-n/a	0-n/a	FuncCode	BoardID	Global-Channel ID (GCI)	pointer to file handel	length of exercised data
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_MAXDIGITS

Indicates termination of background function when a maximum DTMF digits has been received

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_MAXDIGITS	# of digits	0-n/a	FuncCode	BoardID	Global-Channel ID (GCI)	See Below*	# of digits or length of exercised data
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

ptrBuffer - pointer to either a file or buffer (depending on which API was called, playing a file or from a buffer)

EVT_MAXTIME

Indicates termination of background function when the maximum time (ms) has been reached.

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_MAXTIME	See Below*	0-n/a	FuncCode	BoardID	Global-Channel ID (GCI)	pointer to data structure	length of data
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_MEDIA_SESSION_STARTED

Reported by the IPX only. This event indicates that a media session has been established on the network.

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME for-mat	0-n/a	EVT_MEDIA_SESSION_STARTED	n/a	Protocol ID/ Station ID	n/a	BoardID	n/a	pointer to data structure	length of exercised data
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

When this event is reported the MT_IP_SESSION_PARAMS data structure is populated.

EVT_MEDIA_SESSION_STOPPED

Reported by the IPX only. This event indicates that a media session has been torn down on the network.

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME for-mat	0-n/a	EVT_MEDIA_SESSION_STOPPED	n/a	Protocol ID/ Station ID	n/a	BoardID	n/a	pointer to data structure	length of exercised data
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

When this event is reported the MT_IP_SESSION_PARAMS data structure is populated.

EVT_MEDIA_SESSION_STOPPED

Reported by the IPX only. This event indicates that a media session has been torn down on the network.

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File- TIME for- mat	0-n/a	EVT_MEDIA_SESSION_ STOPPED	n/a	Protocol ID/ Station ID	n/a	BoardID	n/a	pointer to data structure	length of exercised data
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

When this event is reported the MT_IP_SESSION_PARAMS data structure is populated.

EVT_MEDIA_SESSION_QOS_REPORT

This event will not be reported on the IPX unless this feature is enabled using the SmartControl panel. After EVT_MEDIA_SESSION_STOPPED is reported, the IPX generates this event to provide useful quality of service (QOS) information for this particular session.

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_MEDIA_SESSION_QOS_REPORT	0-n/a	Protocol ID/ Station ID	0-n/a	BoardID	Global-Channel ID (GCI)	pointer to data structure	length of exercised data
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

Data Structure:

```
typedef struct
```

```
{
```

```
    ULONG SessionID; // Session ID
```

```
    USHORT PrimaryStationRxJitter_Average; /**-----provided by network RTCP reports, average delay between packets**
```

```
    USHORT PrimaryStationRxJitter_Peak; ----- calculated by the IPX based on data from RTCP reports, largest delay for this session**
```

```
    USHORT SecondaryStationRxJitter_Average;**
```

```
    USHORT SecondaryStationRxJitter_Peak;**
```

```
    ULONG PrimaryStationRtpTotalPktCnt; ----- total packets transmitted by this endpoint; calculations derived by number received by IPX
```

```
    ULONG SecondaryStationRtpTotalPktCnt;
```

```
} MT_IP_SESSION_QOS_REPORT;
```

** - if RTCP is not enabled or not supported on the tapped network; these values will be '0'.

EVT_MENU_BUTTON_RELEASED

Indicates that a menu button has been released.

Structure: (DChannel event, reported by the NGX and IPX only)

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_MENU_BUTTON_RELEASED	0-n/a	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_MESSAGE_BUTTON_PRESSED

Indicates that a message button has been pressed

Structure: (DChannel event, reported by the NGX and IPX only)

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File- TIME for- mat	0-n/a	EVT_MESSAGE_BUTTON_ PRESSED	0-n/a	0-n/a	0-n/a	BoardID	Global- Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_MESSAGE_BUTTON_RELEASED

(DChannel event, reported by the NGX and IPX only)

Indicates that a message button has been released.

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File- TIME for- mat	0-n/a	EVT_MESSAGE_BUTTON_ RELEASED	0-n/a	0-n/a	0-n/a	BoardID	Global- Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_MESSAGE_CHANGE

(DChannel event, reported by the NGX and IPX only)

Indicates a message change on the LCD of the phone. On many networks information such as agent ID and phone number are displayed on the LCD. This information is passed to the user application in ASCII format. This string is null terminated.

Other phrases displayed on the phone LCD generate EVT_DISPLAY_MESSAGE.

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File- TIME for- mat	0-n/a	EVT_MESSAGE_CHANGE	32 bit field*	0-n/a	0-n/a	BoardID	Global- Channel ID (GCI)	*see below	*see below
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

* **Subreason** field is a 32 bit field. Some phone LCDs are separated into sections or “windows”. The upper 16 bits are used to identify the window ID when available.

ptrBuffer is a pointer to a null terminated string that contains the screen data.

DataLength is the length in bytes of the string (including the null termination) pointed to by ptrBuffer.

EVT_MESSAGE_LIGHT_FASTFLASHING
EVT_MESSAGE_LIGHT_FLASHING
EVT_MESSAGE_LIGHT_ON
EVT_MESSAGE_LIGHT_OFF
EVT_MESSAGE_LIGHT_QUICKFLASH
EVT_MESSAGE_LIGHT_VERY_FASTFLASHING
EVT_MESSAGE_LIGHT_MEDIUM_WINK

(DChannel event, reported by the NGX and IPX only)

Purpose:

Indicates that the message light on the phone is on, off or flashing.

Description:

Observation of various phone models enabled the AudioCodes lab to categorize phone lights as flashing, fast flashing, very fast flashing, quick flashing or winking. The cadence varies depending on PBX models, and this may also be configured. Our definition of flashing lights stems from the signal passed from the PBX to the phone, and is based on factory default configurations for each model. Please refer to your PBX documentation for more information.

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File- TIME for- mat	0-n/a	EVT_MESSAGE_LIGHT_	*See Below	0-n/a	0-n/a	BoardID	Global- Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

Sub-Reason(This field is only populated when the light is flashing or winking):

NOTE: These may vary depending on phone model and PBX.

Bits 0-7 = Light Number

Bits 8 = Green

Bit 9 = Red

Bit 10 = Amber

Bits 11-31 = Reserved

EVT_MFTONE

Indicates that received MF(R2) tone has been queued

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_MFTONE	See Below*	0= outgoing 1= incoming	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

*an octet containing an unsigned number indicating the MF tone received. This value will be 0x01 through 0x0F for forward tones 1 through 15, and 0x11 through 0x1F for backward tones 1 through 15, respectively. Thus, bits 0 through 3 contain the tone number, and bit 4 is reset for forward tones, and set for backward tones.

EVT_MFTONE_DROPPED

Indicates that received MF(R2) tone has been dropped due to a full queue

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_MFTONE_DROPPED	See Below*	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

*an octet containing an unsigned number indicating the MF tone received. This value will be 0x01 through 0x0F for forward tones 1 through 15, and 0x11 through 0x1F for backward tones 1 through 15, respectively. Thus, bits 0 through 3 contain the tone number, and bit 4 is reset for forward tones, and set for backward tones.

EVT_MIC_BUTTON_PRESSED

(DChannel event, reported by the NGX and IPX only)

Indicates that a microphone button has been pressed

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_MIC_BUTTON_PRESSED	0-n/a	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_MIC_BUTTON_RELEASED

(DChannel event, reported by the NGX and IPX only)

Indicates that a microphone button has been released

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_MIC_BUTTON_RELEASED	0-n/a	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_MIC_LIGHT_FASTFLASHING
EVT_MIC_LIGHT_FLASHING
EVT_MIC_LIGHT_ON
EVT_MIC_LIGHT_OFF
EVT_MIC_LIGHT_QUICKFLASH
EVT_MIC_LIGHT_VERY_FASTFLASHING

(DChannel event, reported by the NGX and IPX only)

Purpose:

Indicates that a microphone light on the phone is on, off, or flashing.

Description:

Observation of various phone models enabled the AudioCodes lab to categorize phone lights as flashing, fast flashing, very fast flashing or quick flashing. The cadence varies depending on PBX models, and this may also be configured. Our definition of flashing lights stems from the signal passed from the PBX to the phone, and is based on factory default configurations for each model. Please refer to your PBX documentation for more information.

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File- TIME for- mat	0-n/a	EVT_MIC_LIGHT_	*See Below	0-n/a	0-n/a	BoardID	Global- Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

Sub-Reason(This field is only populated when the light is flashing):

NOTE: These may vary depending on phone model and PBX.

Bits 0-7 = Light Number

Bits 8 = Green

Bit 9 = Red

Bit 10 = Amber

Bits 11-31 = Reserved

EVT_MODE_BUTTON_PRESSED

(DChannel event, reported by the NGX and IPX only)

Indicates that a mode button has been pressed

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File- TIME for- mat	0-n/a	EVT_MODE_BUTTON_ PRESSED	0-n/a	0-n/a	0-n/a	BoardID	Global- Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_MON_ACTIVITY

Indicates silence off and activity on

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File- TIME for- mat	0-n/a	EVT_MON_ACTIVITY	See Below*	0= outgoing 1= incoming	0-n/a	BoardID	Global- Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

*Timing Information - displays the amount of time (in ms) the line remained in the of the previous state. Information is presented in 1ms units, with a maximum of 65535ms.

EVT_MON_BUSY

Indicates CPM detection of busy cadence cycle. The signal Index ID is passed in the XtrInfo field.

NOTE: This event was obsoleted with v3.3. It has been mapped to EVT_MON_BUSY1 for backwards compatibility.

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtrInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME for-mat	0-n/a	EVT_MON_BUSY1	See Below*	See Below*	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

*

Subreason -

The value of 0x0000 indicates a signal has been detected.

The value of 0xFFFFFFFF indicates loss of signal (MT_CPM_SIGNAL.fDetectLost must be enabled in the Signal Profile).

XtrInfo - the index number of the signal in the CPM signal table

More information is available in the application notes, "Call Progress Monitoring" and "Managing CPM Events" located on the AudioCodes website:

<http://www.audiocodes.com/blades/support.htm>

EVT_MON_BUSY1

Indicates CPM detection of BUSY1 signal. The signal Index ID is passed in the XtrInfo field.

NOTE: EVT_MON_BUSY was obsoleted with v3.3. It has been mapped to EVT_MON_BUSY1 for backwards compatibility.

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtrInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME for-mat	0-n/a	EVT_MON_BUSY1	See Below*	See Below*	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

*

Subreason -

The value of 0x0000 indicates a signal has been detected.

The value of 0xFFFFFFFF indicates loss of signal (MT_CPM_SIGNAL.fDetectLost must be enabled in the Signal Profile).

XtrInfo - the index number of the signal in the CPM signal table

More information is available in the application notes, "Call Progress Monitoring" and "Managing CPM Events" located on the AudioCodes website:

<http://www.audiocodes.com/blades/support.htm>

EVT_MON_BUSY2

Indicates CPM detection of BUSY2 signal. The signal Index ID is passed in the XtraInfo field. By default, this signal profile contains values for the North American tunk (fast) busy.

NOTE: EVT_MON_TBUSY was obsoleted with v3.3. It has been mapped to EVT_MON_BUSY2 for backwards compatibility.

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME for-mat	0-n/a	EVT_MON_BUSY2	See Below*	See Below*	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

*

Subreason -

The value of 0x0000 indicates a signal has been detected.

The value of 0xFFFFFFFF indicates loss of signal (MT_CPM_SIGNAL.fDetectLost must be enabled in the Signal Profile).

XtraInfo - the index number of the signal in the CPM signal table

More information is available in the application notes, "Call Progress Monitoring" and "Managing CPM Events" located on the AudioCodes website:

<http://www.audiocodes.com/blades/support.htm>

EVT_MON_CALLWAITING

Indicates CPM detection of CALLWAITING signal. The signal Index ID is passed in the XtrInfo field.

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtrInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_MON_CALLWAITING	See Below*	See Below*	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

*Subreason -

The value of 0x0000 indicates a signal has been detected.

The value of 0xFFFFFFFF indicates loss of signal (MT_CPM_SIGNAL.fDetectLost must be enabled in the Signal Profile).

XtrInfo - the index number of the signal in the CPM signal table

More information is available in the application notes, “Call Progress Monitoring” and “Managing CPM Events” located on the AudioCodes website:

<http://www.audiocodes.com/blades/support.htm>

EVT_MON_CPM_CYCLE

Used to display the total cycle count of a detected CPM signal. One EVT_MON_CPM_CYCLE is generated for each complete cycle of a signal. The fDetectCycle field must be enabled in the signal profile. This option should only be enabled for signals with a repeating cadence pattern.

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtrInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_MON_BUSY	See Below*	n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

*Total cycle count, incremented until the cycle is no longer detected.

EVT_MON_DIAL

EVT_MON_DIAL1, EVT_MON_DIAL2, EVT_MON_DIAL3

Indicates CPM detection of a dial tone on the line. The signal Index ID is passed in the XtraInfo field.

NOTE: As of release v3.3 this event has been obsoleted. When MONI_DIAL is enabled, then events EVT_MON_DIAL1, EVT_MON_DIAL2, and EVT_MON_DIAL3 are generated respectively.

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_MON_DIAL	See Below*	See Below*	FuncCode	BoardID	Global-Channel ID (GCI)	pointer to data structure	length of exercised data
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

*Subreason -

The value of 0x0000 indicates a signal has been detected.

The value of 0xFFFFFFFF indicates loss of signal (MT_CPM_SIGNAL.fDetectLost must be enabled).

XtraInfo - the index number of the signal in the CPM signal table

More information is available in the application notes, "Call Progress Monitoring" and "Managing CPM Events" located on the AudioCodes website:

<http://www.audiocodes.com/blades/support.htm>

EVT_MON_DIAL1

Indicates CPM detection of a DIAL1 signal on the line. The signal Index ID is passed in the XtraInfo field.

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_MON_DIAL1	See Below*	See Below*	FuncCode	BoardID	Global-Channel ID (GCI)	pointer to data structure	length of data
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

*Subreason -

The value of 0x0000 indicates a signal has been detected.

The value of 0xFFFFFFFF indicates loss of signal (MT_CPM_SIGNAL.fDetectLost must be enabled).

XtraInfo - the index number of the signal in the CPM signal table

More information is available in the application notes, "*Call Progress Monitoring*" and "*Managing CPM Events*" located on the AudioCodes website:

<http://www.audiocodes.com/blades/support.htm>

EVT_MON_DIAL2

Indicates CPM detection of a DIAL2 signal on the line. The signal Index ID is passed in the XtrInfo field.

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtrInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_MON_DIAL2	See Below*	See Below*	FuncCode	BoardID	Global-Channel ID (GCI)	pointer to data structure	length of data
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

***Subreason -**

The value of 0x0000 indicates a signal has been detected.

The value of 0xFFFFFFFF indicates loss of signal (MT_CPM_SIGNAL.fDetectLost must be enabled).

XtrInfo - the index number of the signal in the CPM signal table

More information is available in the application notes, "Call Progress Monitoring" and "Managing CPM Events" located on the AudioCodes website:

<http://www.audiocodes.com/blades/support.htm>

EVT_MON_DIAL3

Indicates CPM detection of a DIAL3 signal on the line. The signal Index ID is passed in the XtraInfo field.

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_MON_DIAL3	See Below*	See Below*	FuncCode	BoardID	Global-Channel ID (GCI)	pointer to data structure	length of data
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

*Subreason -

The value of 0x0000 indicates a signal has been detected.

The value of 0xFFFFFFFF indicates loss of signal (MT_CPM_SIGNAL.fDetectLost must be enabled).

XtraInfo - the index number of the signal in the CPM signal table

More information is available in the application notes, *"Call Progress Monitoring"* and *"Managing CPM Events"* located on the AudioCodes website:

<http://www.audiocodes.com/blades/support.htm>

EVT_MON_DRINGBACK

Indicates CPM detection of a RINGBACK2 signal. The signal Index ID is passed in the XtraInfo field.

NOTE: This event has been obsoleted as of v3.3 release. It has been mapped to EVT_MON_RINGBACK2 for backwards compatibility.

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME for-mat	0-n/a	EVT_MON_RINGBACK2	See Below*	See Below*	FuncCode	BoardID	Global-Channel ID (GCI)	pointer to data structure	length of data
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

*

Subreason -

The value of 0x0000 indicates a signal has been detected.

The value of 0xFFFFFFFF indicates loss of signal (MT_CPM_SIGNAL.fDetectLost must be enabled).

XtraInfo - the index number of the signal in the CPM signal table

More information is available in the application notes, “Call Progress Monitoring” and “Managing CPM Events” located on the AudioCodes website:

<http://www.audiocodes.com/blades/support.htm>

EVT_MON_FAX EVT_MON_FAX1

Indicates CPM detection of a FAX1 signal. The signal Index ID is passed in the XtraInfo field.

NOTE: The event EVT_MON_FAX has been obsoleted as of v3.3 release. It has been mapped to EVT_MON_FAX1 for backwards compatibility.

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File- TIME for- mat	0-n/a	EVT_MON_FAX1	0-n/a	See Below*	FuncCode	BoardID	Global- Channel ID (GCI)	pointer to data structure	length of data
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

*

Subreason -

The value of 0x0000 indicates a signal has been detected.

The value of 0xFFFFFFFF indicates loss of signal (MT_CPM_SIGNAL.fDetectLost must be enabled).

XtraInfo - the index number of the signal in the CPM signal table

More information is available in the application notes, “Call Progress Monitoring” and “Managing CPM Events” located on the AudioCodes website:

<http://www.audiocodes.com/blades/support.htm>

EVT_MON_FAX1

Indicates CPM detection of a FAX1 signal. The signal Index ID is passed in the XtrInfo field.

NOTE: The event EVT_MON_FAX has been obsoleted as of v3.3 release. It has been mapped to EVT_MON_FAX1 for backwards compatibility.

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtrInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME for-mat	0-n/a	EVT_MON_FAX1	0-n/a	See Below*	FuncCode	BoardID	Global-Channel ID (GCI)	pointer to data structure	length of data
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

*

Subreason -

The value of 0x0000 indicates a signal has been detected.

The value of 0xFFFFFFFF indicates loss of signal (MT_CPM_SIGNAL.fDetectLost must be enabled).

XtrInfo - the index number of the signal in the CPM signal table

More information is available in the application notes, "Call Progress Monitoring" and "Managing CPM Events" located on the AudioCodes website:

<http://www.audiocodes.com/blades/support.htm>

EVT_MON_FAX2

Indicates CPM detection of a FAX2 signal. The signal Index ID is passed in the XtrInfo field.

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtrInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_MON_FAX2	0-n/a	See Below*	FuncCode	BoardID	Global-Channel ID (GCI)	pointer to data structure	length of data
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

*

Subreason -

The value of 0x0000 indicates a signal has been detected.

The value of 0xFFFFFFFF indicates loss of signal (MT_CPM_SIGNAL.fDetectLost must be enabled).

XtrInfo - the index number of the signal in the CPM signal table

More information is available in the application notes, “Call Progress Monitoring” and “Managing CPM Events” located on the AudioCodes website:

<http://www.audiocodes.com/blades/support.htm>

EVT_MON_HUMAN

Indicates that the CPM detection of voice activity, after complete analysis is a live person (human).

More information is available in the application notes, “Call Progress Monitoring” and “Managing CPM Events” located on the AudioCodes website:

<http://www.audiocodes.com/blades/support.htm>

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtrInfo	FuncCode	Board	Channel	ptrBuffer	DaaLength
In File-TIME format	0-n/a	EVT_MON_HUMAN	0-n/a	0-n/a	N/A	BoardID	Global-Channel ID (GCI)	pointer to data structure	length of data
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_MON_MACHINE

Indicates that the CPM detection of voice activity, after complete analysis is an answering machine.

More information is available in the application notes, *"Call Progress Monitoring"* and *"Managing CPM Events"* located on the AudioCodes website:

<http://www.audiocodes.com/blades/support.htm>

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DaaLength
In File- TIME for- mat	0-n/a	EVT_MON_MACHINE	0-n/a	0-n/a	N/A	BoardID	Global- Channel ID (GCI)	pointer to data structure	length of data
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_MON_NOVOICE

This event is only generated after the CPM feature detects voice on the line. If silence is detected (no voice activity) then this event is generated.

More information is available in the application notes, *"Call Progress Monitoring"* and *"Managing CPM Events"* located on the AudioCodes website:

<http://www.audiocodes.com/blades/support.htm>

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DaaLength
In File- TIME for- mat	0-n/a	EVT_MON_NOVOICE	0-n/a	0-n/a	N/A	BoardID	Global- Channel ID (GCI)	pointer to data structure	length of data
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_MON_RECEIVEOFF

Indicates CPM detection of a RECEIVEOFF signal. The signal Index ID is passed in the XtraInfo field.

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_MON_RECEIVEOFF	0-n/a	See Below*	FuncCode	BoardID	Global-Channel ID (GCI)	pointer to data structure	length of data
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

*

Subreason -

The value of 0x0000 indicates a signal has been detected.

The value of 0xFFFFFFFF indicates loss of signal (MT_CPM_SIGNAL.fDetectLost must be enabled).

XtraInfo - the index number of the signal in the CPM signal table

More information is available in the application notes, “Call Progress Monitoring” and “Managing CPM Events” located on the AudioCodes website:

<http://www.audiocodes.com/blades/support.htm>

EVT_MON_REVERSAL

Indicates the detection of a loop current polarity reversal.

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_MON_REVERSAL	0-n/a	0-n/a	FuncCode	BoardID	Global-Channel ID (GCI)	pointer to data structure	length of data
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_MON_RINGBACK

EVT_MON_RINGBACK1

Indicates CPM detection of a RINGBACK1 signal. The signal Index ID is passed in the XtraInfo field.

NOTE: This event was obsoleted as of v3.3 release. It has been mapped to event EVT_MON_RINGBACK1 for backwards compatibility.

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_MON_RINGBACK1	See Below*	See Below*	FuncCode	BoardID	Global-Channel ID (GCI)	pointer to data structure	length of data
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

*

Subreason -

The value of 0x0000 indicates a signal has been detected.

The value of 0xFFFFFFFF indicates loss of signal (MT_CPM_SIGNAL.fDetectLost must be enabled).

XtraInfo - the index number of the signal in the CPM signal table

More information is available in the application notes, *"Call Progress Monitoring"* and *"Managing CPM Events"* located on the AudioCodes website:

<http://www.audiocodes.com/blades/support.htm>

EVT_MON_RINGBACK1

Indicates CPM detection of a RINGBACK1 signal. The signal Index ID is passed in the XtrInfo field.

NOTE: The event EVT_MON_RINGBACK was obsoleted as of v3.3 release. It has been mapped to event EVT_MON_RINGBACK1 for backwards compatibility.

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtrInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME for-mat	0-n/a	EVT_MON_RINGBACK1	See Below*	See Below*	FuncCode	BoardID	Global-Channel ID (GCI)	pointer to data structure	length of data
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

*

Subreason -

The value of 0x0000 indicates a signal has been detected.

The value of 0xFFFFFFFF indicates loss of signal (MT_CPM_SIGNAL.fDetectLost must be enabled).

XtrInfo - the index number of the signal in the CPM signal table

More information is available in the application notes, "Call Progress Monitoring" and "Managing CPM Events" located on the AudioCodes website:

<http://www.audiocodes.com/blades/support.htm>

EVT_MON_RINGBACK2

Indicates CPM detection of a RINGBACK2 signal. The signal Index ID is passed in the XtrInfo field. By default, this signal profile contains values that define the North American double ringback.

NOTE: The event EVT_MON_DRINGBACK was obsoleted as of v3.3 release. It has been mapped to event EVT_MON_RINGBACK2 for backwards compatibility.

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtrInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME for-mat	0-n/a	EVT_MON_RINGBACK2	See Below*	See Below*	FuncCode	BoardID	Global-Channel ID (GCI)	pointer to data structure	length of data
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

*

Subreason -

The value of 0x0000 indicates a signal has been detected.

The value of 0xFFFFFFFF indicates loss of signal (MT_CPM_SIGNAL.fDetectLost must be enabled).

XtrInfo - the index number of the signal in the CPM signal table

More information is available in the application notes, "Call Progress Monitoring" and "Managing CPM Events" located on the AudioCodes website:

<http://www.audiocodes.com/blades/support.htm>

EVT_MON_SIGNAL_CYCLE

Used as a counting mechanism when CPM or Usr signals that have a distinct ON/OFF cadence pattern have been detected. The MONI_SIGNAL_CYCLE bit must be enabled as well as the *fDetectCycle* flag in the signal's profile. This event is generated once for each complete cycle detected by the firmware. The SubReason field is used as a counter.

The signal Index ID is passed in the XtraInfo field.

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME for-mat	0-n/a	EVT_MON_SIGNAL_CYCLE	See Below*	See Below*	N/A	BoardID	Global-Channel ID (GCI)	pointer to data structure	length of data
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

***Subreason -**

The total cycle count.

XtraInfo - the index number of the signal in the CPM signal table

More information is available in the application notes, *"Call Progress Monitoring"* and *"Managing CPM Events"* located on the AudioCodes website:

<http://www.audiocodes.com/blades/support.htm>

EVT_MON_SILENCE

Indicates the detection of silence on the line.

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME for-mat	0-n/a	EVT_MON_SILENCE	See Below*	0-n/a	FuncCode	BoardID	Global-Channel ID (GCI)	pointer to data structure	length of data
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

***Timing Information** - displays the amount of time (in ms) the line remained in the of the previous state. Information is presented in 1ms units, with a maximum of 65535ms.

EVT_MON_SIT

EVT_MON_SIT1, EVT_MON_SIT2, EVT_MON_SIT3, EVT_MON_SIT4, EVT_MON_SIT5

Indicates the CPM detection of a (SIT) signal. The signal Index ID is passed in the XtraInfo field. When MONI_SIT is enabled, then EVT_MON_SIT(1-5) is generated per respective signal.

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME for-mat	0-n/a	EVT_MON_SIT(1-5)	See Below*	See Below*	FuncCode	BoardID	Global-Channel ID (GCI)	pointer to data structure	length of data
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

*

Subreason -

The value of 0x0000 indicates a signal has been detected.

The value of 0xFFFFFFFF indicates loss of signal (MT_CPM_SIGNAL.fDetectLost must be enabled).

XtraInfo - the index number of the signal in the CPM signal table

More information is available in the application notes, "Call Progress Monitoring" and "Managing CPM Events" located on the AudioCodes website:

<http://www.audiocodes.com/blades/support.htm>

EVT_MON_SIT1

Indicates the CPM detection of a SIT1 signal. The signal Index ID is passed in the XtraInfo field.

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_MON_SIT1	See Below*	See Below*	FuncCode	BoardID	Global-Channel ID (GCI)	pointer to data structure	length of data
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

*

Subreason -

The value of 0x0000 indicates a signal has been detected.

The value of 0xFFFFFFFF indicates loss of signal (MT_CPM_SIGNAL.fDetectLost must be enabled).

XtraInfo - the index number of the signal in the CPM signal table

More information is available in the application notes, “*Call Progress Monitoring*” and “*Managing CPM Events*” located on the AudioCodes website:

<http://www.audiocodes.com/blades/support.htm>

EVT_MON_SIT2

Indicates the CPM detection of a SIT2 signal. The signal Index ID is passed in the XtraInfo field.

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_MON_SIT2	See Below*	See Below*	FuncCode	BoardID	Global-Channel ID (GCI)	pointer to data structure	length of data
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

*

Subreason -

The value of 0x0000 indicates a signal has been detected.

The value of 0xFFFFFFFF indicates loss of signal (MT_CPM_SIGNAL.fDetectLost must be enabled).

XtraInfo - the index number of the signal in the CPM signal table

More information is available in the application notes, *"Call Progress Monitoring"* and *"Managing CPM Events"* located on the AudioCodes website:

<http://www.audiocodes.com/blades/support.htm>

EVT_MON_SIT3

Indicates the CPM detection of a SIT3 signal. The signal Index ID is passed in the XtrInfo field.

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtrInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_MON_SIT3	See Below*	See Below*	FuncCode	BoardID	Global-Channel ID (GCI)	pointer to data structure	length of data
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

*

Subreason -

The value of 0x0000 indicates a signal has been detected.

The value of 0xFFFFFFFF indicates loss of signal (MT_CPM_SIGNAL.fDetectLost must be enabled).

XtrInfo - the index number of the signal in the CPM signal table

More information is available in the application notes, “Call Progress Monitoring” and “Managing CPM Events” located on the AudioCodes website:

<http://www.audiocodes.com/blades/support.htm>

EVT_MON_SIT4

Indicates the CPM detection of a SIT4 signal. The signal Index ID is passed in the XtraInfo field.

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_MON_SIT4	See Below*	See Below*	FuncCode	BoardID	Global-Channel ID (GCI)	pointer to data structure	length of data
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

*

Subreason -

The value of 0x0000 indicates a signal has been detected.

The value of 0xFFFFFFFF indicates loss of signal (MT_CPM_SIGNAL.fDetectLost must be enabled).

XtraInfo - the index number of the signal in the CPM signal table

More information is available in the application notes, "Call Progress Monitoring" and "Managing CPM Events" located on the AudioCodes website:

<http://www.audiocodes.com/blades/support.htm>

EVT_MON_SIT5

Indicates the CPM detection of a SIT5 signal. The signal Index ID is passed in the XtraInfo field.

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_MON_SIT5	See Below*	See Below*	FuncCode	BoardID	Global-Channel ID (GCI)	pointer to data structure	length of data
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

*

Subreason -

The value of 0x0000 indicates a signal has been detected.

The value of 0xFFFFFFFF indicates loss of signal (MT_CPM_SIGNAL.fDetectLost must be enabled).

XtraInfo - the index number of the signal in the CPM signal table

More information is available in the application notes, “*Call Progress Monitoring*” and “*Managing CPM Events*” located on the AudioCodes website:

<http://www.audiocodes.com/blades/support.htm>

EVT_MON_TBUSY

Indicates CPM detection of BUSY2 signal. The signal Index ID is passed in the XtrInfo field. By default, the signal profiles contain values defining a North American trunk (fast) busy signal.

NOTE: This event has been obsoleted with the v3.3 release. It has been mapped to EVT_MON_BUSY2 for backwards compatibility.

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtrInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME for-mat	0-n/a	EVT_MON_BUSY2	See Below*	See Below*	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

*

Subreason -

The value of 0x0000 indicates a signal has been detected.

The value of 0xFFFFFFFF indicates loss of signal (MT_CPM_SIGNAL.fDetectLost must be enabled).

XtrInfo - the index number of the signal in the CPM signal table

More information is available in the application notes, *"Call Progress Monitoring"* and *"Managing CPM Events"* located on the AudioCodes website:

<http://www.audiocodes.com/blades/support.htm>

EVT_MON_UTONE

Indicates CPM detection of user defined signal (Usr Tone). The signal Index ID is passed in the XtrInfo field. This event is only generated if the Usr Tone has not been defined with an alias.

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtrInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_MON_UTONE	See Below*	See Below*	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

*Subreason -

The value of 0x0000 indicates a signal has been detected.

The value of 0xFFFFFFFF indicates loss of signal (MT_CPM_SIGNAL.fDetectLost must be enabled).

XtrInfo - the index number of the signal in the CPM signal table

More information is available in the application notes, “Call Progress Monitoring” and “Managing CPM Events” located on the AudioCodes website:

<http://www.audiocodes.com/blades/support.htm>

EVT_MON_VOICE

Indicates the CPM detection of voice activity on the line. The line will be analyzed further to determine whether the voice activity is either a live person (human) or answer machine (machine).

More information is available in the application notes, “Call Progress Monitoring” and “Managing CPM Events” located on the AudioCodes website:

<http://www.audiocodes.com/blades/support.htm>

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtrInfo	FuncCode	Board	Channel	ptrBuffer	DaaLength
In File-TIME format	0-n/a	EVT_MON_VOICE	0-n/a	0-n/a	FuncCode	BoardID	Global-Channel ID (GCI)	pointer to data structure	length of data
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_MUTE_BUTTON_PRESSED

(DChannel event, reported by the NGX and IPX only)

Indicates that the mute button has been pressed on the client phone.

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File- TIME for- mat	0-n/a	EVT_MUTE_BUTTON_ PRESSED	0-n/a	0-n/a	0-n/a	BoardID	Global- Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_MUTE_BUTTON_RELEASED

(DChannel event, reported by the NGX and IPX only)

Indicates that the mute button has been released.

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File- TIME for- mat	0-n/a	EVT_MUTE_BUTTON_ RELEASED	0-n/a	0-n/a	0-n/a	BoardID	Global- Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_MUTE_LIGHT_FASTFLASHING
EVT_MUTE_LIGHT_VERY_FASTFLASHING
EVT_MUTE_LIGHT_FLASHING
EVT_MUTE_LIGHT_OFF
EVT_MUTE_LIGHT_ON
EVT_MUTE_LIGHT_QUICKFLASH
EVT_MUTE_LIGHT_WINK

(DChannel event, reported by the NGX and IPX only)

Purpose:

Indicates that the mute light on the phone is on, off, or flashing.

Description:

Observation of various phone models enabled the AudioCodes lab to categorize phone lights as flashing, fast flashing, very fast flashing, quick flashing or winking. The cadence varies depending on PBX models, and this may also be configured. Our definition of flashing lights stems from the signal passed from the PBX to the phone, and is based on factory default configurations for each model. Please refer to your PBX documentation for more information.

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File- TIME for- mat	0-n/a	EVT_MUTE_LIGHT_	*See Below	0-n/a	0-n/a	BoardID	Global- Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

Sub-Reason(this field is populated only when light is flashing or winking):

NOTE: These may vary depending on phone model and PBX.

Bits 0-7 = Light Number

Bits 8 = Green

Bit 9 = Red

Bit 10 = Amber

Bits 11-31 = Reserved

EVT_MVIP_ALARM

Indicates an MVIP alarm detected. This event will be generated when it has been armed by an Arm MVIP Error Event command, and CT812 reports an MVIP error condition.

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME for-mat	0-n/a	EVT_MVIP_ALARM	0-n/a	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_NAVIGATION_BUTTON_PRESSED

(DChannel event, reported by the NGX and IPX only)

Indicates that the navigation button has been pressed on the client phone.

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME for-mat	0-n/a	EVT_NAVIGATION_BUTTON_PRESSED	0-n/a	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_NAVIGATION_BUTTON_RELEASED

(DChannel event, reported by the NGX and IPX only)

Indicates that the navigation button has been released.

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME for-mat	0-n/a	EVT_NAVIGATION_BUTTON_RELEASED	0-n/a	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_NEWCALL_BUTTON_PRESSED

(DChannel event, reported by the NGX and IPX only)

Indicates that the new call button was pressed. Normally, this button is used if an agent attempted to call someone and the line is busy or they dialed the wrong extension. Rather than hang up the hookswitch, they can press the "new call" button. It has very similar behavior as the hookswitch.

Also, if a dialed is busy number, and another call is coming into the line at the same time, the agent can press the new call button to retrieve the incoming call.

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File- TIME for- mat	0-n/a	EVT_NEWCALL_ BUTTON_PRESSED	0-n/a	0-n/a	0-n/a	BoardID	Global- Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_NEXT_BUTTON_PRESSED

(DChannel event, reported by the NGX and IPX only)

Indicates that the Next button has been pressed

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File- TIME for- mat	0-n/a	EVT_NEXT_BUTTON_ PRESSED	0-n/a	0-n/a	0-n/a	BoardID	Global- Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_NEXT_BUTTON_RELEASED

Indicates that the Next button has been released.

Structure: (DChannel event, reported by the NGX and IPX only)

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_NEXT_BUTTON_RELEASED	0-n/a	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_NOT_AVAILABLE

Indicates functionality not available or busy

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_NOT_AVAILABLE	0-n/a	Internal Error Code	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_NOT_SUPPORTED

Indicates functionality not supported

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_NOT_SUPPORTED	0-n/a	Internal Error Code	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_OFFHOOK

Indicates Offhook complete

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_OFFHOOK	0-n/a	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_ONHOOK

Indicates Onhook complete

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_ONHOOK	0-n/a	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_OFF_HOOK

(DChannel event, reported by the NGX and IPX only)

Indicates that the user of the tapped phone (call agent) has removed the handset, or enabled the speaker or handsfree capabilities on the phone to initiate or answer a call.

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_OFF_HOOK	0-n/a	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_ON_HOOK

Indicates that the user of the tapped phone (call agent) has replaced the handset, or disabled the speaker or handsfree capabilities on the phone to terminate a call.

Structure: (DChannel event, reported by the NGX and IPX only)

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_ON_HOOK	0-n/a	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_OUTGOING_BUTTON_PRESSED

Indicates that the outgoing button has been pressed

Structure: (DChannel event, reported by the NGX and IPX only)

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_OUTGOING_BUTTON_PRESSED	0-n/a	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_OUTGOING_SIGCHANGE

Indicates a change of Signaling Bits (ABCD) on outgoing line

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtrInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_OUTGOING_SIGCHANGE	See Below*	See Below*	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	See Below*
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

***.SubReason:**

an octet the new values of the signaling bits A, B, C, and D, in bits 3, 2, 1, and 0, respectively. Bits 7, 6, 5 and 4 contain the previous values of the signaling bits.

XtrInfo:

the number of the framer reporting the event (least significant bit)

the timeslot on which the change was detected(most significant bit)

DataLength:

the duration of the previous state in 10 ms units

EVT_PAGE_BUTTON_PRESSED

(DChannel event, reported by the NGX and IPX only)

Indicates that the page button was pressed.

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtrInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_PAGE_BUTTON_PRESSED	0-n/a	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_PAGE_BUTTON_RELEASED

(DChannel event, reported by the NGX and IPX only)

Indicates that the page button was released.

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME for-mat	0-n/a	EVT_PAGE_BUTTON_RELEASED	0-n/a	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_PAGE_LIGHT_FASTFLASHING
EVT_PAGE_LIGHT_FLASHING
EVT_PAGE_LIGHT_OFF
EVT_PAGE_LIGHT_ON
EVT_PAGE_LIGHT_QUICKFLASH
EVT_PAGE_LIGHT_VERY_FASTFLASHING

(DChannel event, reported by the NGX and IPX only)

Purpose:

Indicates that the page light on the phone is on, off or flashing.

Description:

Observation of various phone models enabled the AudioCodes lab to categorize phone lights as flashing, fast flashing, very fast flashing or quick flashing. The cadence varies depending on PBX models, and this may also be configured. Our definition of flashing lights stems from the signal passed from the PBX to the phone, and is based on factory default configurations for each model. Please refer to your PBX documentation for more information.

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File- TIME for- mat	0-n/a	EVT_PAGE_LIGHT_	*See Below	0-n/a	0-n/a	BoardID	Global- Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

Sub-Reason(this field is populated only when light is flashing):

NOTE: These may vary depending on phone model and PBX.

Bits 0-7 = Light Number

Bits 8 = Green

Bit 9 = Red

Bit 10 = Amber

Bits 11-31 = Reserved

EVT_PARK_BUTTON_PRESSED

(DChannel event, reported by the NGX and IPX only)

Indicates that the park button was pressed. A 'park' is similar to a call on hold, except that once the call is 'parked' other extensions can access the held call.

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME for-mat	0-n/a	EVT_PARK_BUTTON_PRESSED	0-n/a	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_PARK_LIGHT_FASTFLASHING
EVT_PARK_LIGHT_FLASHING
EVT_PARK_LIGHT_OFF
EVT_PARK_LIGHT_ON
EVT_PARK_LIGHT_QUICKFLASH
EVT_PARK_LIGHT_VERY_FASTFLASHING
EVT_PARK_LIGHT_WINK

(DChannel event, reported by the NGX and IPX only)

Purpose:

Indicates that the park light on the phone is on, off or flashing.

Description:

Observation of various phone models enabled the AudioCodes lab to categorize phone lights as flashing, fast flashing, very fast flashing or quick flashing. The cadence varies depending on PBX models, and this may also be configured. Our definition of flashing lights stems from the signal passed from the PBX to the phone, and is based on factory default configurations for each model. Please refer to your PBX documentation for more information.

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File- TIME for- mat	0-n/a	EVT_PARK_LIGHT_	*See Below	0-n/a	0-n/a	BoardID	Global- Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

Sub-Reason(this field is populated only when light is flashing):

NOTE: These may vary depending on phone model and PBX.

Bits 0-7 = Light Number

Bits 8 = Green

Bit 9 = Red

Bit 10 = Amber

Bits 11-31 = Reserved

EVT_PAUSED_BUTTON_PRESSED

(DChannel event, reported by the NGX and IPX only)

Indicates that the paused button has been pressed

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME for-mat	0-n/a	EVT_PAUSED_BUTTON_PRESSED	0-n/a	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_PBX_DCH

Not Supported. Although this event exists in the header file, it is not currently supported by the SmartWORKS API.

EVT_PHN_DCH

Not Supported. Although this event exists in the header file, it is not currently supported by the SmartWORKS API.

EVT_PAUSED_BUTTON_PRESSED

(DChannel event, reported by the NGX and IPX only)

Indicates that the paused button has been pressed

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME for-mat	0-n/a	EVT_PAUSED_BUTTON_PRESSED	0-n/a	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_PHY_LINK_DOWN

(Reported by IPX only)

Indicates that a network link is down.

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME for-mat	0-n/a	EVT_PHY_LINK_DOWN	See Below	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

SubReason: An integer that represents the IPX port number associated with this event (0-2)

EVT_PHY_LINK_UP

(Reported by IPX only)

Indicates that a network link has re-gained connection and is up.

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File- TIME for- mat	0-n/a	EVT_PHY_LINK_UP	See Below	0-n/a	0-n/a	BoardID	Global- Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

SubReason: An integer that represents the IPX port number associated with this event

EVT_PREVIOUS_BUTTON_RELEASED

(DChannel event, reported by the NGX and IPX only)

Indicates that the previous button has been released

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME for-mat	0-n/a	EVT_PREVIOUS_BUTTON_RELEASED	0-n/a	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_PROGRAM_BUTTON_PRESSED

(DChannel event, reported by the NGX and IPX only)

Indicates that the program button was pressed

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME for-mat	0-n/a	EVT_PROGRAM_BUTTON_PRESSED	0-n/a	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_PROGRAM_LIGHT_FASTFLASHING
EVT_PROGRAM_LIGHT_FLASHING
EVT_PROGRAM_LIGHT_OFF
EVT_PROGRAM_LIGHT_ON
EVT_PROGRAM_LIGHT_QUICKFLASH
EVT_PROGRAM_LIGHT_VERY_FASTFLASHING
EVT_PROGRAM_LIGHT_WINK

(DChannel event, reported by the NGX and IPX only)

Purpose:

Indicates that the program light on the phone is on, off or flashing.

Description:

Observation of various phone models enabled the AudioCodes lab to categorize phone lights as flashing, fast flashing, very fast flashing, quick flashing or winking. The cadence varies depending on PBX models, and this may also be configured. Our definition of flashing lights stems from the signal passed from the PBX to the phone, and is based on factory default configurations for each model. Please refer to your PBX documentation for more information.

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File- TIME for- mat	0-n/a	EVT_PROGRAM_LIGHT_	*See Below	0-n/a	0-n/a	BoardID	Global- Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

Sub-Reason(this field is populated only when light is flashing or winking):

NOTE: These may vary depending on phone model and PBX.

Bits 0-7 = Light Number
 Bits 8 = Green
 Bit 9 = Red
 Bit 10 = Amber
 Bits 11-31 = Reserved

EVT_QUOTA_EXCEEDED

Indicates that a board limit has been met.

When reported by the IPX board, this event indicates that the allowed number of media sessions that can be forwarded has been met.

When reported by the HPX board, this event indicates that the allowed number of media sessions that can be forwarded has been met or the allowed number of station endpoints that can be monitored has been met.

These maximum values are determined by the license key.

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_QUOTA_EXCEEDED	See Below*	0-n/a	0-n/a	BoardID	n/a	Pointer to data structure*	length of data
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

*Subreason

The resource type that has been exceeded.

For the IPX board, SubReason only can be 0x01 that means the allowed number of media sessions that can be forwarded has been met.

For the HPX board, SubReason can have the following two values:

0x01 that means the allowed number of media sessions that can be forwarded has been met.

0x02 that means the allowed number of station endpoints that can be monitored has been met.

ptrBuffer

When reported by the IPX, this field is a pointer to the MT_IP_SESSION_PARAMS data structure.

EVT_READY_BUTTON_PRESSED

(DChannel event, reported by the NGX and IPX only)

Indicates that the ready button was pressed

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_READY_BUTTON_PRESSED	0-n/a	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_READY_BUTTON_RELEASED

(DChannel event, reported by the NGX and IPX only)

Indicates that the ready button was released

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME for-mat	0-n/a	EVT_READY_BUTTON_RELEASED	0-n/a	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_READY_LIGHT_FASTFLASHING
EVT_READY_LIGHT_FLASHING
EVT_READY_LIGHT_OFF
EVT_READY_LIGHT_ON
EVT_READY_LIGHT_QUICKFLASH
EVT_READY_LIGHT_VERY_FASTFLASHING

(DChannel event, reported by the NGX and IPX only)

Purpose:

Indicates that the ready light on the phone is on, off or flashing.

Description:

Observation of various phone models enabled the AudioCodes lab to categorize phone lights as flashing, fast flashing, very fast flashing or quick flashing. The cadence varies depending on PBX models, and this may also be configured. Our definition of flashing lights stems from the signal passed from the PBX to the phone, and is based on factory default configurations for each model. Please refer to your PBX documentation for more information.

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME for-mat	0-n/a	EVT_READY_LIGHT_	*See Below	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

Sub-Reason(this field is populated only when light is flashing):

NOTE: These may vary depending on phone model and PBX.

Bits 0-7 = Light Number

Bits 8 = Green

Bit 9 = Red

Bit 10 = Amber

Bits 11-31 = Reserved

EVT_RECALL_BUTTON_PRESSED

Indicates that the Recall button has been pressed

Structure: (DChannel event, reported by the NGX and IPX only)

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_RECALL_BUTTON_PRESSED	0-n/a	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_REDIAL_BUTTON_PRESSED

Indicates that the Redial Button has been pressed

Structure: (DChannel event, reported by the NGX and IPX only)

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_REDIAL_BUTTON_PRESSED	0-n/a	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_REDIAL_BUTTON_RELEASED

Indicates that the Recall button has been released (DChannel event, reported by the NGX and IPX only)

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_REDIAL_BUTTON_RELEASED	0-n/a	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_REDIAL_LIGHT_FASTFLASHING
EVT_REDIAL_LIGHT_FLASHING
EVT_REDIAL_LIGHT_OFF
EVT_REDIAL_LIGHT_ON
EVT_REDIAL_LIGHT_QUICKFLASH
EVT_REDIAL_LIGHT_VERY_FASTFLASHING

(DChannel event, reported by the NGX and IPX only)

Purpose:

Indicates that the redial light on the phone is on, off or flashing.

Description:

Observation of various phone models enabled the AudioCodes lab to categorize phone lights as flashing, fast flashing, very fast flashing or quick flashing. The cadence varies depending on PBX models, and this may also be configured. Our definition of flashing lights stems from the signal passed from the PBX to the phone, and is based on factory default configurations for each model. Please refer to your PBX documentation for more information.

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME for-mat	0-n/a	EVT_REDIAL_LIGHT_	*See Below	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

Sub-Reason(this field is populated only when light is flashing):

NOTE: These may vary depending on phone model and PBX.

Bits 0-7 = Light Number

Bits 8 = Green

Bit 9 = Red

Bit 10 = Amber

Bits 11-31 = Reserved

EVT_REJECTED

Not supported. Although this event exists in the header file, it is not currently supported by the SmartWORKS API.

EVT_RELEASE_BUTTON_PRESSED

(DChannel event, reported by the NGX and IPX only)

Indicates that the Release button has been pressed

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME for-mat	0-n/a	EVT_RELEASE_BUTTON_PRESSED	0-n/a	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_RELEASE_BUTTON_RELEASED

(DChannel event, reported by the NGX and IPX only)

Indicates that the Release button has been released

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME for-mat	0-n/a	EVT_RELEASE_BUTTON_RELEASED	0-n/a	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_RELEASE_LIGHT_FASTFLASHING
EVT_RELEASE_LIGHT_FLASHING
EVT_RELEASE_LIGHT_OFF
EVT_RELEASE_LIGHT_ON
EVT_RELEASE_LIGHT_QUICKFLASH
EVT_RELEASE_LIGHT_VERY_FASTFLASHING
EVT_RELEASE_LIGHT_WINK

(DChannel event, reported by the NGX and IPX only)

Purpose:

Indicates that the release light on the phone is on, off or flashing.

Description:

Observation of various phone models enabled the AudioCodes lab to categorize phone lights as flashing, fast flashing, very fast flashing or quick flashing. The cadence varies depending on PBX models, and this may also be configured. Our definition of flashing lights stems from the signal passed from the PBX to the phone, and is based on factory default configurations for each model. Please refer to your PBX documentation for more information.

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File- TIME for- mat	0-n/a	EVT_RELEASE_LIGHT_	*See Below	0-n/a	0-n/a	BoardID	Global- Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

Sub-Reason(this field is populated only when light is flashing):

NOTE: These may vary depending on phone model and PBX.

Bits 0-7 = Light Number
 Bits 8 = Green
 Bit 9 = Red
 Bit 10 = Amber
 Bits 11-31 = Reserved

EVT_REORDER

Indicates that the called destination doesn't exist

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME for-mat	0-n/a	EVT_REORDER	0-n/a	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_RING_LIGHT_FASTFLASHING
EVT_RING_LIGHT_FLASHING
EVT_RING_LIGHT_ON
EVT_RING_LIGHT_OFF
EVT_RING_LIGHT_QUICKFLASH
EVT_RING_LIGHT_VERY_FASTFLASHING

(DChannel event, reported by the NGX and IPX only)

Purpose:

Indicates that the ring light on the phone is on, off or flashing.

Description:

Observation of various phone models enabled the AudioCodes lab to categorize phone lights as flashing, fast flashing, very fast flashing or quick flashing. The cadence varies depending on PBX models, and this may also be configured. Our definition of flashing lights stems from the signal passed from the PBX to the phone, and is based on factory default configurations for each model. Please refer to your PBX documentation for more information.

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME for-mat	0-n/a	EVT_RING_LIGHT_	*See Below	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

Sub-Reason(this field is only populated when the light is flashing):

NOTE: These may vary depending on phone model and PBX.

Bits 0-7 = Light Number
 Bits 8 = Green
 Bit 9 = Red
 Bit 10 = Amber
 Bits 11-31 = Reserved

EVT_RING_OFF

(DChannel event, reported by the NGX and IPX only)

Indicates when the phone stops ringing. The phone ring cadence can be determined by calculating the time interval between EVT_RING_OFF and EVT_RING_ON using the TimeStamp in the event structure

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_RING_OFF	*See Below	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

* indicates the ring type, (may not be used with each PBX that specifies ring type when EVT_RING_ON is reported)

EVT_RING_ON

(DChannel event, reported by the NGX and IPX only)

Indicates when the phone starts ringing. The phone ring cadence can be determined by calculating the time interval between EVT_RING_ON and EVT_RING_OFF using the TimeStamp in the event structure

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_RING_ON	*See Below	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

* indicates the PBX specific ring type.

EVT_RINGS

Indicates an incoming call. The ring detector on the board has detected that the line is ringing.

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_RINGS	total count of rings	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_RINGTONES_BUTTON_PRESSED

(DChannel event, reported by the NGX and IPX only)

Indicates that the ringtones button was pressed

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_RINGTONES_BUTTON_PRESSED	0-n/a	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_RINGTONES_LIGHT_FASTFLASHING
EVT_RINGTONES_LIGHT_FLASHING
EVT_RINGTONES_LIGHT_OFF
EVT_RINGTONES_LIGHT_ON
EVT_RINGTONES_LIGHT_QUICKFLASH
EVT_RINGTONES_LIGHT_VERY_FASTFLASHING

(DChannel event, reported by the NGX and IPX only)

Purpose:

Indicates that the ringtones light on the phone is on, off or flashing.

Description:

Observation of various phone models enabled the AudioCodes lab to categorize phone lights as flashing, fast flashing, very fast flashing or quick flashing. The cadence varies depending on PBX models, and this may also be configured. Our definition of flashing lights stems from the signal passed from the PBX to the phone, and is based on factory default configurations for each model. Please refer to your PBX documentation for more information.

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME for-mat	0-n/a	EVT_RINGTONES_LIGHT_	*See Below	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

Sub-Reason(this field is populated only when light is flashing):

NOTE: These may vary depending on phone model and PBX.

Bits 0-7 = Light Number

Bits 8 = Green

Bit 9 = Red

Bit 10 = Amber

Bits 11-31 = Reserved

EVT_SAVE_BUTTON_PRESSED

(DChannel event, reported by the NGX and IPX only)

Indicates that the save button was pressed

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME for-mat	0-n/a	EVT_SAVE_BUTTON_PRESSED	0-n/a	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_SAVE_LIGHT_FASTFLASHING
EVT_SAVE_LIGHT_FLASHING
EVT_SAVE_LIGHT_OFF
EVT_SAVE_LIGHT_ON
EVT_SAVE_LIGHT_QUICKFLASH
EVT_SAVE_LIGHT_VERY_FASTFLASHING

(DChannel event, reported by the NGX and IPX only)

Purpose:

Indicates that the save light on the phone is on, off or flashing.

Description:

Observation of various phone models enabled the AudioCodes lab to categorize phone lights as flashing, fast flashing, very fast flashing or quick flashing. The cadence varies depending on PBX models, and this may also be configured. Our definition of flashing lights stems from the signal passed from the PBX to the phone, and is based on factory default configurations for each model. Please refer to your PBX documentation for more information.

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME for-mat	0-n/a	EVT_SAVE_LIGHT_	*See Below	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

Sub-Reason(this field is populated only when light is flashing):

NOTE: These may vary depending on phone model and PBX.

Bits 0-7 = Light Number

Bits 8 = Green

Bit 9 = Red

Bit 10 = Amber

Bits 11-31 = Reserved

EVT_SCROLL_BUTTON_PRESSED

(DChannel event, reported by the NGX and IPX only)

Indicates that the scroll button was pressed.

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File- TIME for- mat	0-n/a	EVT_SCROLL_ BUTTON_PRESSED	0-n/a	0-n/a	0-n/a	BoardID	Global- Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_SCROLL_BUTTON_RELEASED

(DChannel event, reported by the NGX and IPX only)

Indicates that the scroll button was released.

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File- TIME for- mat	0-n/a	EVT_SCROLL_ BUTTON_RELEASED	0-n/a	0-n/a	0-n/a	BoardID	Global- Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_SCROLL_LIGHT_FASTFLASHING
EVT_SCROLL_LIGHT_VERY_FASTFLASHING
EVT_SCROLL_LIGHT_FLASHING
EVT_SCROLL_LIGHT_OFF
EVT_SCROLL_LIGHT_ON
EVT_SCROLL_LIGHT_QUICKFLASH

(DChannel event, reported by the NGX and IPX only)

Purpose:

Indicates that the scroll light on the phone is on, off, or flashing.

Description:

Observation of various phone models enabled the AudioCodes lab to categorize phone lights as flashing, fast flashing, very fast flashing or quick flashing. The cadence varies depending on PBX models, and this may also be configured. Our definition of flashing lights stems from the signal passed from the PBX to the phone, and is based on factory default configurations for each model. Please refer to your PBX documentation for more information.

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME for-mat	0-n/a	EVT_SCROLL_LIGHT_	*See Below	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

Sub-Reason(this field is populated only when light is flashing):

NOTE: These may vary depending on phone model and PBX.

Bits 0-7 = Light Number

Bits 8 = Green

Bit 9 = Red

Bit 10 = Amber

Bits 11-31 = Reserved

EVT_SELECT_BUTTON_PRESSED

Indicates that the select button has been pressed

Structure: (DChannel event, reported by the NGX and IPX only)

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_SELECT_BUTTON_PRESSED	0-n/a	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_SERVICES_BUTTON_PRESSED

Indicates that the services button has been pressed

Structure: (DChannel event, reported by the NGX and IPX only)

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_SERVICES_BUTTON_PRESSED	0-n/a	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_SERVICES_BUTTON_RELEASED

Indicates that the services button has been released.

Structure: (DChannel event, reported by the NGX and IPX only)

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_SERVICES_BUTTON_RELEASED	0-n/a	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_SET_BUSY

Indicates the “set to busy” feature is active on the phone

Structure: (DChannel event, reported by the NGX and IPX only)

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_SET_BUSY	0-n/a	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_SET_BUSY_CANCELED

Indicates the “set to busy” feature has been canceled.

Structure: (DChannel event, reported by the NGX and IPX only)

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_SET_BUSY_CANCELED	0-n/a	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_SHIFT_BUTTON_PRESSED

Indicates that the shift button has been pressed.

Structure: (DChannel event, reported by the NGX and IPX only)

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_SHIFT_BUTTON_PRESSED	0-n/a	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_SOFT_BUTTON_PRESSED

(DChannel event, reported by the NGX and IPX only)

Indicates that a soft button has been pressed

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_SOFT_BUTTON_PRESSED	0-n/a	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_SOFT_BUTTON_RELEASED

(DChannel event, reported by the NGX and IPX only)

Indicates that a soft button has been released

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_SOFT_BUTTON_RELEASED	0-n/a	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_SOFT_LIGHT_FASTFLASHING
EVT_SOFT_LIGHT_VERY_FASTFLASHING
EVT_SOFT_LIGHT_FLASHING
EVT_SOFT_LIGHT_OFF
EVT_SOFT_LIGHT_ON
EVT_SOFT_LIGHT_QUICKFLASH

(DChannel event, reported by the NGX and IPX only)

Purpose:

Indicates that the soft light on the phone is on, off, or flashing.

Description:

Observation of various phone models enabled the AudioCodes lab to categorize phone lights as flashing, fast flashing, very fast flashing or quick flashing. The cadence varies depending on PBX models, and this may also be configured. Our definition of flashing lights stems from the signal passed from the PBX to the phone, and is based on factory default configurations for each model. Please refer to your PBX documentation for more information.

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME for-mat	0-n/a	EVT_SOFT_LIGHT_	*See Below	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

Sub-Reason(this field is populated only when light is flashing):

NOTE: These may vary depending on phone model and PBX.

Bits 0-7 = Light Number

Bits 8 = Green

Bit 9 = Red

Bit 10 = Amber

Bits 11-31 = Reserved

EVT_SPEAKER_BUTTON_PRESSED

(DChannel event, reported by the NGX and IPX only)

Indicates that a speaker (or hands free) button has been pressed

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME for-mat	0-n/a	EVT_SPEAKER_BUTTON_PRESSED	0-n/a	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_SPEAKER_BUTTON_RELEASED

(DChannel event, reported by the NGX and IPX only)

Indicates that a speaker (or hands free) button has been released

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME for-mat	0-n/a	EVT_SPEAKER_BUTTON_RELEASED	0-n/a	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_SPEAKER_LIGHT_FASTFLASHING
EVT_SPEAKER_LIGHT_FLASHING
EVT_SPEAKER_LIGHT_ON
EVT_SPEAKER_LIGHT_OFF
EVT_SPEAKER_LIGHT_QUICKFLASH
EVT_SPEAKER_LIGHT_VERY_FASTFLASHING
EVT_SPEAKER_LIGHT_MEDIUM_WINK
EVT_SPEAKER_LIGHT_WINK

(DChannel event, reported by the NGX and IPX only)

Purpose:

Indicates that a speaker light on the phone is on, off or flashing.

Description:

Observation of various phone models enabled the AudioCodes lab to categorize phone lights as flashing, fast flashing, very fast flashing, quick flashing or winking. The cadence varies depending on PBX models, and this may also be configured. Our definition of flashing lights stems from the signal passed from the PBX to the phone, and is based on factory default configurations for each model. Please refer to your PBX documentation for more information.

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME for-mat	0-n/a	EVT_SPEAKER_LIGHT_	*See Below	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

Sub-Reason(this field is only populated when the light is flashing or winking):

NOTE: These may vary depending on phone model and PBX.

Bits 0-7 = Light Number

Bits 8 = Green

Bit 9 = Red

Bit 10 = Amber

Bits 11-31 = Reserved

EVT_SPECIAL_BUTTON_PRESSED

Indicates that a special button has been pressed

Structure: (DChannel event, reported by the NGX and IPX only)

TimeStamp	UserStatus	EventCode	SubReason	XtrInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_SPECIAL_BUTTON_PRESSED	0-n/a	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_SPECIAL_BUTTON_RELEASED

Indicates that a special button has been released.

Structure: (DChannel event, reported by the NGX and IPX only)

TimeStamp	UserStatus	EventCode	SubReason	XtrInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_SPECIAL_BUTTON_RELEASED	0-n/a	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_SPEEDDIAL_BUTTON_PRESSED

Indicates that a speeddial button has been pressed

Structure: (DChannel event, reported by the NGX and IPX only)

TimeStamp	UserStatus	EventCode	SubReason	XtrInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_SPEEDDIAL_BUTTON_PRESSED	0-n/a	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_SPEEDDIAL_LIGHT_FASTFLASHING
EVT_SPEEDDIAL_LIGHT_FLASHING
EVT_SPEEDDIAL_LIGHT_ON
EVT_SPEEDDIAL_LIGHT_OFF
EVT_SPEEDDIAL_LIGHT_QUICKFLASH
EVT_SPEEDDIAL_LIGHT_VERY_FASTFLASHING
EVT_SPEEDDIAL_LIGHT_MEDIUM_WINK
EVT_SPEEDDIAL_LIGHT_WINK

(DChannel event, reported by the NGX and IPX only)

Purpose:

Indicates that a speeddial light on the phone is on, off or flashing.

Description:

Observation of various phone models enabled the AudioCodes lab to categorize phone lights as flashing, fast flashing, very fast flashing, quick flashing or winking. The cadence varies depending on PBX models, and this may also be configured. Our definition of flashing lights stems from the signal passed from the PBX to the phone, and is based on factory default configurations for each model. Please refer to your PBX documentation for more information.

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File- TIME for- mat	0-n/a	EVT_SPEEDDIAL_LIGHT_	*See Below	0-n/a	0-n/a	BoardID	Global- Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

Sub-Reason(this field is only populated when the light is flashing or winking):

NOTE: These may vary depending on phone model and PBX.

Bits 0-7 = Light Number

Bits 8 = Green

Bit 9 = Red

Bit 10 = Amber

Bits 11-31 = Reserved

EVT_START_TONE

(DChannel event, reported by the NGX and IPX only)

Indicates that the PBX has commanded the phone to play a tone such as ringback, busy signal, dial tone etc).

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_START_TONE	See Below*	n/a	FuncCode	BoardID	Global-Channel ID (GCI)	pointer to data structure	length of data
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

*Subreason field:

This field passes over the ToneID. When the command to start a tone is passed to the phone, the PBX also informs the type of tone that must be played. The ToneID is specific to the PBX model. Refer to the NGX Integration Guide or the IPX Integration guide for information about a specific PBX.

EVT_STATION_ADDED

(Reported by the IPX and HPX only)

A new VoIP endpoint (phone) has been detected by the IPX or HPX. NOTE: This event is not reported if the board is configured to operate in target mode.

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_STATION_ADDED	n/a	see below*	n/a	BoardID	n/a	pointer to data structure	length of data
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

*The station ID and protocol ID.

NOTE: To obtain more information for this VoIP endpoint, use the *MTipGetStationParams()* function.

EVT_STATION_REMOVED

(Reported by the IPX only)

A VoIP endpoint is no longer active on the VoIP network. The station ID is returned to the IPX for re-use.

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_STATION_REMOVED	n/a	see below*	n/a	BoardID	n/a	pointer to data structure	length of data
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

*The station ID and protocol ID.

EVT_STOP_TONE

(DChannel event, reported by the NGX and IPX only)

Indicates that the PBX has commanded the phone to stop playing a tone such as ringback, busy signal, dial tone etc).

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_STATION_ADDED	n/a	n/a	FuncCode	BoardID	Global-Channel ID (GCI)	pointer to data structure	length of data
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_STATION_ADDED

Reported by the IPX only. This event indicates that a VoIP endpoint has been discovered on the tapped network.

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_STATION_ADDED	n/a	Protocol ID/ Station ID	FuncCode	BoardID	Global-Channel ID (GCI)	pointer to data structure	length of data
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_STATION_REMOVED

Reported by the IPX only. This event indicates that a VoIP endpoint has been removed from the tapped network.

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_STATION_REMOVED	n/a	Protocol ID/ Station ID	FuncCode	BoardID	Global-Channel ID (GCI)	pointer to data structure	length of data
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_STOP

Indicates that the specified background function on a channel has stopped when the function **MTStopChannel()** was invoked.

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_STOP	0-n/a	0-n/a	FuncCode	BoardID	Global-Channel ID (GCI)	pointer to data structure	length of data
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_STREAMIN_DROPPED

Indicates input streaming data dropped.

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_STREAMIN_DROPPED	See Below*	0-n/a	FuncCode	BoardID	Global-Channel ID (GCI)	0-n/a	See Below*
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

***Subreason:** This field is populated with EVT_MAXBYTES indicating that the maximum bytes set has been reached

DataLen:

length of data dropped

EVT_STREAMIN_START

Indicates data encode streaming started

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_STREAMIN_START	0-n/a	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_STREAMIN_STOP

Indicates data encode streaming stopped

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_STREAMIN_STOP	See Below*	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

*SubReason:

This field populated with EVT_STREAMIN_COMP indicating that streaming has completed.

EVT_STREAMIN_WATERMARK

Indicates data encode streaming watermark reached. It is fired when the number of bytes in the buffer at least equals the number set in the watermark then writes from the buffer to the file.

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME for-mat	0-n/a	EVT_STREAMIN_WATERMARK	See Below*	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	SEe Beloew*
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

Subreason: This field is populated with EVT_MAXBYTES indicating that the set maximum bytes has been reached

DataLen:
length of data dropped

EVT_STREAMOUT_EMPTY

Indicates output stream buffer empty

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME for-mat	0-n/a	EVT_STREAMOUT_EMPTY	0-n/a	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_STREAMOUT_PAUSED

Indicates output streaming paused

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME for-mat	0-n/a	EVT_STREAMOUT_PAUSED	0-n/a	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_STREAMOUT_RESUMED

Indicates output streaming resumed

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_STREAMOUT_RESUMED	0-n/a	See Below*	0-n/a	BoardID	Global-Channel ID (GCI)	See Below*	See Below*
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

XtraInfo - ERROR_SERVICE_PREQUEST_TIMEOUT

ptrBuffer - pointer to either a file or buffer (depending on which API was called, playing a file or from a buffer)

DataLength - length (bytes) of exercised data at termination

EVT_STREAMOUT_START

Indicates data DEcode streaming started

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_STREAMOUT_START	0-n/a	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_STREAMOUT_STOP

Indicates data DEcode streaming stopped

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME for-mat	0-n/a	EVT_STREAMOUT_STOP	See Below*	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

SubReason:

This field populated with EVT_STREAMOUT_COMP indicating that the data Decode streaming process has completed

EVT_STREAMOUT_WATERMARK

Indicates data DEcode streaming watermark reached. It is fired when the number of bytes in the buffer at least equals the number set in the watermark then writes from the buffer to the file.

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME for-mat	0-n/a	EVT_STREAMOUT_WATERMARK	See Below*	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

Subreason: This field is populated with EVT_MAXBYTES indicating the number set for the watermark

EVT_SUPERVISOR_BUTTON_PRESSED

(DChannel event, reported by the NGX and IPX only)

Indicates that the supervisor button was pressed

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtrInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File- TIME for- mat	0-n/a	EVT_SUPERVISOR_BUTTON _PRESSED	0-n/a	0-n/a	0-n/a	BoardID	Global- Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_SUPERVISOR_BUTTON_RELEASED

(DChannel event, reported by the NGX and IPX only)

Indicates that the supervisor button was released

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtrInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File- TIME for- mat	0-n/a	EVT_SUPERVISOR_BUTTON _RELEASED	0-n/a	0-n/a	0-n/a	BoardID	Global- Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_SUPERVISOR_LIGHT_FASTFLASHING
EVT_SUPERVISOR_LIGHT_FLASHING
EVT_SUPERVISOR_LIGHT_OFF
EVT_SUPERVISOR_LIGHT_ON
EVT_SUPERVISOR_LIGHT_QUICKFLASH
EVT_SUPERVISOR_LIGHT_VERY_FASTFLASHING

(DChannel event, reported by the NGX and IPX only)

Purpose:

Indicates that the supervisor light on the phone is on, off or flashing.

Description:

Observation of various phone models enabled the AudioCodes lab to categorize phone lights as flashing, fast flashing, very fast flashing or quick flashing. The cadence varies depending on PBX models, and this may also be configured. Our definition of flashing lights stems from the signal passed from the PBX to the phone, and is based on factory default configurations for each model. Please refer to your PBX documentation for more information.

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File- TIME for- mat	0-n/a	EVT_SUPERVISOR_LIGHT_	*See Below	0-n/a	0-n/a	BoardID	Global- Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

Sub-Reason(this field is populated only when light is flashing):

NOTE: These may vary depending on phone model and PBX.

Bits 0-7 = Light Number

Bits 8 = Green

Bit 9 = Red

Bit 10 = Amber

Bits 11-31 = Reserved

EVT_SYS_BOARD_ADDED

Indicates that a board has been added to the system (newly initialized) after the user's application began running. It is recommended that the user run **MTGetSystemInfo()** to obtain new total board count. This event is only generated when system events are controlled with **MTSysWaitForEvent()** or **MTSysSetEventCallback()**.

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_SYS_BOARD_ADDED	0-n/a	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_SYS_BOARD_REMOVED

Not implemented yet.

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_SYS_BOARD_REMOVED	0-n/a	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_SYS_ERROR

Indicates a system response timeout error

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_SYS_ERROR	0-n/a	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_SYS_SYNCTIME_NEW

Resynchronization completed. The timestamp of this event displays the new time after resynchronization. This event is only generated when system events are controlled with ***MTSysWaitForEvent()*** or ***MTSysSetEventCallback()***.

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_SYS_SYNCTIME_NEW	0-n/a	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_SYS_SYNCTIME_OLD

Resynchronization completed. The timestamp of this event displays the old time. This event is only generated when system events are controlled with ***MTSysWaitForEvent()*** or ***MTSysSetEventCallback()***.

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_SYS_SYNCTIME_OLD	0-n/a	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_SYS_QUEUE_OVERFLOW

Indicates any channel queue on the board is full or overflowing. This system event will repeat until queue overflow condition is resolved.

Structure:

TimeStamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
In File-TIME format	0-n/a	EVT_SYS_QUEUE_OVERFLOW	0-n/a	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
Large_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_TERMDIGIT

Indicates termination of background function due to a terminating DTMF digit that has been received

Stamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
0-n/a for-	0-n/a	EVT_DIGITENABLE	Termination Digit	0-n/a	FuncCode	BoardID	Global-Channel ID (GCI)	See Below*	See Below
UINT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

ptrBuffer - pointer to either a file or buffer (depending on which API was called, playing a file or from a buffer)
DataLength - length (bytes) of exercised data at termination

EVT_TERMVOLTAGE_NOTOFFHOOK EVT_TERMVOLTAGE_ABOVEORBELOW

Indicates termination of background function on loop voltage changed to notoffhook

Structure:

Stamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
0-n/a for-	0-n/a	EVT_TERMVOLTAGE_NOTOFFHOOK	0-n/a	0-n/a	FuncCode	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
UINT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_TERMSILENCE

Indicates termination of background function due to silence reached on play or record

Structure:

Stamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
0-n/a for-	0-n/a	EVT_TERMSILENCE	See Below*	0-n/a	FuncCode	BoardID	Global-Channel ID (GCI)	0-n/a	length of exercise data
UINT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

*

SubReason:

This field populated with CPM_MAX_SILENCE indicating that the maximum silence is met to terminate play or record

EVT_TERMLVOLTAGE_ABOVEORBELOW EVT_TERMLVOLTAGE_NOTOFFHOOK

Indicates termination of background function due to the loop voltage state that is NOT MIDDLE (LD) or is NOTOFFHOOK (PT). The SmartWORKS LD board generates EVT_LVOLTAGE_ABOVE and EVT_LVOLTAGE_BELOW events when the voltage state is NOT middle (offhook).

Structure:

Stamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
0-n/a for-	0-n/a	EVT_TERMLVOLTAGE_	0-n/a	0-n/a	FuncCode	BoardID	Global-Channel ID (GCI)	0-n/a	length of exercise data
UINT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_TONEP_DONE

Indicates tone generation complete

Structure:

Stamp	UserStatus	EventCode	SubReason	Xtralnfo	FuncCode	Board	Channel	ptrBuffer	DataL
-for-	0-n/a	EVT_TONEP_DONE	0-n/a	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n
_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULC

EVT_TONEPERR

Indicates tone generation template error

Structure:

Stamp	UserStatus	EventCode	SubReason	Xtralnfo	FuncCode	Board	Channel	ptrBuffer	DataL
-for-	0-n/a	EVT_TONEPERR	0-n/a	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n
_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULC

EVT_TRANSFER_BUTTON_PRESSED

(DChannel event, reported by the NGX and IPX only)

Indicates that the Transfer button has been pressed

Structure:

Stamp	UserStatus	EventCode	SubReason	Xtralnfo	FuncCode	Board	Channel	ptrBuffer	DataL
-for-	0-n/a	EVT_TRANSFER_BUTTON_PRESSED	0-n/a	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n
_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULC

EVT_TRANSFER_BUTTON_RELEASED

(DChannel event, reported by the NGX and IPX only)

Indicates that the Transfer button has been released

Structure:

Stamp	UserStatus	EventCode	SubReason	Xtralnfo	FuncCode	Board	Channel	ptrBuffer	DataL
-for-	0-n/a	EVT_TRANSFER_BUTTON_RELEASED	0-n/a	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n
_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULC

EVT_TRANSFER_LIGHT_FASTFLASHING

EVT_TRANSFER_LIGHT_FLASHING

EVT_TRANSFER_LIGHT_OFF

EVT_TRANSFER_LIGHT_ON

EVT_TRANSFER_LIGHT_QUICKFLASH

EVT_TRANSFER_LIGHT_VERY_FASTFLASHING

EVT_TRANSFER_LIGHT_WINK

(DChannel event, reported by the NGX and IPX only)

Purpose:

Indicates that the transfer light on the phone is on, off or flashing.

Description:

Observation of various phone models enabled the AudioCodes lab to categorize phone lights as flashing, fast flashing, very fast flashing, quick flashing or winking. The cadence varies depending on PBX models, and this may also be configured. Our definition of flashing lights stems from the signal passed from the PBX to the phone, and is based on factory default configurations for each model. Please refer to your PBX documentation for more information.

Stamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
0-n/a for-	0-n/a	EVT_TRANSFER_LIGHT_	*See Below	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
UINT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

Sub-Reason(this field is populated only when light is flashing or winking):

NOTE: These may vary depending on phone model and PBX.

Bits 0-7 = Light Number
Bits 8 = Green
Bit 9 = Red
Bit 10 = Amber
Bits 11-31 = Reserved

EVT_TS16RAI_ALARM

Indicates Timeslot 16 Remote Alarm Indication alarm (E1 only)

Structure:

Stamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
0-n/a for-	0-n/a	EVT_TS16RAI_ALARM	An octet containing the number of the framer reporting the event	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
UINT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_UP_DOWN

Indicates that the volume up/down or view contrast buttons have been pressed

Structure:

Stamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
0-n/a for-	0-n/a	EVT_UP_DOWN	0-n/a	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
UINT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_UTONE_OFF

Indicates detection of a user defined tone has ended. The Signal Index ID is passed into the Xtralnfo field.
Structure:

Stamp	UserStatus	EventCode	SubReason	Xtralnfo	FuncCode	Board	Channel	ptrBuffer	DataLength
0-n/a for-	0-n/a	EVT_UTONE_OFF	n/a	Index ID	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
UINT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_UTONE_ON

Indicates detection of a user defined tone. The Signal Index ID is passed into the Xtralnfo field.
Structure:

Stamp	UserStatus	EventCode	SubReason	Xtralnfo	FuncCode	Board	Channel	ptrBuffer	DataLength
0-n/a for-	0-n/a	EVT_UTONE_ON	n/a	Index ID	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
UINT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_VOICE_RECORD

Indicates recording started on detection of activity
Structure:

Stamp	UserStatus	EventCode	SubReason	Xtralnfo	FuncCode	Board	Channel	ptrBuffer	DataLength
0-n/a for-	0-n/a	EVT_VOICE_RECORD	See Below*	0-n/a	FuncCode	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
UINT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

*The field is populated with CPM_MON_VOICE which indicates that voice activity was detected on the line.

EVT_WINKDONE

Indicates winks protocol complete
Structure:

Stamp	UserStatus	EventCode	SubReason	Xtralnfo	FuncCode	Board	Channel	ptrBuffer	DataLength
0-n/a for-	0-n/a	EVT_WINKDONE	0-n/a	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
UINT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_WRAPUP_BUTTON_PRESSED

(DChannel event, reported by the NGX and IPX only)

Indicates that the wrapup button was pressed

Structure:

Timestamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
0-n/a	0-n/a	EVT_WRAPUP_BUTTON_PRESSED	0-n/a	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
ULONGLONG	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_WRAPUP_BUTTON_RELEASED

(DChannel event, reported by the NGX and IPX only)

Indicates that the wrapup button was released

Structure:

Timestamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataLength
0-n/a	0-n/a	EVT_WRAPUP_BUTTON_RELEASED	0-n/a	0-n/a	0-n/a	BoardID	Global-Channel ID (GCI)	0-n/a	0-n/a
ULONGLONG	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULONG

EVT_WRAPUP_LIGHT_FASTFLASHING EVT_WRAPUP_LIGHT_FLASHING EVT_WRAPUP_LIGHT_OFF EVT_WRAPUP_LIGHT_ON EVT_WRAPUP_LIGHT_QUICKFLASH EVT_WRAPUP_LIGHT_VERY_FASTFLASHING

(DChannel event, reported by the NGX and IPX only)

Purpose:

Indicates that the wrapup light on the phone is on, off or flashing.

Description:

Observation of various phone models enabled the AudioCodes lab to categorize phone lights as flashing, fast flashing, very fast flashing or quick flashing. The cadence varies depending on PBX models, and this may also be configured. Our definition of flashing lights stems from the signal passed from the PBX to the phone, and is based on factory default configurations for each model. Please refer to your PBX documentation for more information.

stamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataL
- for-	0-n/a	EVT_WRAPUP_LIGHT_	*See Below	0-n/a	0-n/a	BoardID	Global- Channel ID (GCI)	0-n/a	0-n
_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULC

Sub-Reason(this field is populated only when light is flashing):

NOTE: These may vary depending on phone model and PBX.

- Bits 0-7 = Light Number
- Bits 8 = Green
- Bit 9 = Red
- Bit 10 = Amber
- Bits 11-31 = Reserved

EVT_YELLOW_ALARM

Indicates yellow alarm(T1/E1 only). The framer alarm must be enabled using the *MTArmFramerAlarm()* API.
Structure:

Stamp	UserStatus	EventCode	SubReason	XtraInfo	FuncCode	Board	Channel	ptrBuffer	DataL
- for-	0-n/a	EVT_YELLOW_ALARM	An octet containing the number of the framer reporting the event	0-n/a	0-n/a	BoardID	Global- Channel ID (GCI)	0-n/a	0-n
_INT	ULONG	ULONG	ULONG	ULONG	ULONG	USHORT	USHORT	PVOID	ULC

Appendix A

MT_IO_CONTROL

The MT_IO_CONTROL Data Structure

[MT_IO_CONTROL](#) is an important structure used by most background functions. It specifies most termination conditions and contains pointers to buffers and a file offset. Each [MT_IO_CONTROL](#) takes effect when the associated background function is active. The channel setting is not affected.

TABLE 1: MT_IO_CONTROL

Type	Name	Purpose
ULONG	PreACTDuration	ms of data to record before triggering on ACTD
ULONG	CtrlData	# of activity events to trigger recording for START_ACTD control
		GENERAL TERMINATION CONTROL
ULONG	MaxTime	Action timeout period in milliseconds
ULONG	MaxBytes	Action terminate on size of bytes
		SIGNAL TERMINATION CONTROL
ULONG	MaxSilence	Max. silence time for termination (ms)
ULONG	MaxActivity	Max activity time for termination (ms)
ULONG	CpmMode	CPM Mode control
ULONG	TermEnable	Termination conditions
ULONG	MoniEnable	Monitoring conditions
		DIGIT TERMINATION CONTROL
ULONG	MaxDigits	Number of digits for termination
MT_DIGIT_ENABLE	DigitEnable	Enable bits for termination digits
ULONG	EndDigit	Termination digit (the character '@' terminates on any digit)
ULONG	DigitTime	Inter-digit timeout when MaxDigits is not zero
		MISCELLANEOUS CONTROL DATA
ULONG	StartControl	Start Control flag field
ULONG	ToneDuration	Prompt tone duration in milliseconds
ULONG	TermControl	Flush queued playback(s) on termination other than EVT_EOF, EVT_MAXBYTE, EVT_MAXTIME
		PLAY/RECORD SPECIFIC DATA
ULONG	FileIndex	File starting offset for play/record APIs (not required for index APIs)
ULONG	FileHandle	file handle; Device handle for user specified media access
ULONG	FileType	Type of voice file
		Data Portion
ULONG	Length	Length of the following data buffer
PUCHA	Buffer	Data buffer pointer

GENERAL TERMINATION CONTROL

MAXTIME

The maximum time, in milliseconds, that a background function can be active. This parameter causes an [EVT_MAXTIME](#) event. **NOTE:** This termination condition is not supported when streaming.

MAXBYTES

The maximum number of bytes to read/write. This parameter causes an [EVT_MAXBYTES](#) event. **NOTE:** This termination condition is not supported when streaming.

SIGNAL TERMINATION CONTROL

MAXSILENCE

For termination of background functions (play function only), this variable specifies the maximum silence allowed in milliseconds. If this period of no signal is exceeded, the function terminates. If a recording is terminated by this event, an [EVT_TERMSILENCE](#) event is generated. This parameter cannot be used for play functions.

MAXACTIVITY

This parameter specifies the maximum time for a continuous signal or activity before the function terminates(play function only). It is similar to [MaxSilence](#). Like [MaxSilence](#), [MaxActivity](#) can not be used for play functions. An [EVT_MAX_ACTIVITY](#) event is generated if this termination condition occurs.

CPMMODE

Users control the CPM mode for a particular background function. This field controls whether monitoring events ([EVT_MON_signalname](#)) are reported to the user application or whether this task is stopped because a termination condition is detected. The following table lists each option when setting this field in the [MT_IO_CONTROL](#) structure:

Mode	Description
MONI_USE_CHANNEL_ONLY	Report monitoring events as set at the channel level using MTSetMoni() .
MONI_USE_NONE	Do not report any monitoring events. Ignore the options set at the channel level using MTSetMoni() . *
MONI_USE_TASK_ONLY	Report only the options set using the <i>MoniEnable</i> field of the IO_CONTROL structure. Ignore the options set at the channel level using MTSetMoni() .
MONI_USE_CHANNEL_AND_TASK	Report the monitoring events set with the <i>MoniEnable</i> field of the IO_CONTROL structure and at the channel level using MTSetMoni() .
TERM_USE_NONE	To not terminate this function. Ignore the termination conditions set at the channel level using MTSetTerm() .
TERM_USE_CHANNEL_AND_TASK	Terminate this function using the termination conditions set with the <i>TermEnable</i> field of the IO_CONTROL structure and at the channel level using MTSetTerm() .

TERM_USE_CHANNEL_ONLY	Terminate this background function using the termination conditions set at the channel level using MTSetTerm() .
TERM_USE_TASK_ONLY	Terminate this function using the termination conditions set with the <i>TermEnable</i> field of the IO_CONTROL structure. Ignore the termination conditions set at the channel level with MTSetTerm() .

*If other background tasks are running with a mode that supports event reporting, monitoring events are still generated.

Backwards Compatibility

To support older versions of the SmartWORKS API, (3.2 and earlier) the following modes are still accepted:

Mode - 3.2 or earlier	Later Releases
CPMMODE_NONE	MONI_USE_CHANNEL_ONLY TERM_USE_NONE
CPMMODE_MONI	MONI_USE_CHANNEL_AND_TASK TERM_USE_NONE
CPMMODE_COMP	MONI_USE_CHANNEL_ONLY TERM_USE_CHANNEL_AND_TASK

TERM ENABLE

This field allows users to set the same termination conditions available using the **MTSetTerm()** API. This bit field specifies line conditions that will behave as termination conditions while this background function is running. Once this line condition is detected, this task is automatically stopped by the SmartWORKS DLL. This field can be used in combination with **MTSetTerm()** or configured to work alone. The CPM_MODE field of the MT_IO_CONTROL structure controls whether the channel, task or a combination of both settings are used.

When a condition is met, the event EVT_CPM_STOP is generated with the Subreason and XtraInfo fields of the MT_EVENT structure identifying the line condition, and the FuncCode field identifying the type of media function terminated. The FuncCode IDs are defined in the [FuncCode](#) chapter of this manual.

The table on the following page explains each option of the TermEnable field and the respective subreason and XtraInfo field generated when EVT_CPM_STOP is reported.

MTSetTerm() Control	Subreason	XtralInfo
TERM_LOOP_DROP	CPM_MON_LOOP_DROP	
TERM_MON_SIL	CPM_MON_SILENCE	
TERM_MON_ACT	CPM_MON_ACTIVITY	
TERM_MON_DIAL*+	CPM_MON_DIAL(1-3)	
TERM_MON_BUSY*+	CPM_MON_BUSY	The Index of the signal (normal busy or trunk busy)
TERM_CONNECT+	(Using MTCallString()) CPM_MON_CONNECT ----- (Using MTDialString()) CPM_MON_HUMAN CPM_MON_MACHINE	CPM_MON_HUMAN, CPM_MON_MACHINE -----
TERM_NO_ANSWER+	CPM_NO_ANSWER	
TERM_INTERCEPT*+	CPM_MON_INTERCEPT	
TERM_NOR_SILENCE	CPM_MON_MAX_SILENCE CPM_MAX_SILENCE	
TERM_NOR_ACTIVITY	CPM_MON_MAX_ACTIVITY CPM_MAX_ACTIVITY	
TERM_LVOLTAGE_ABOVEORBELOW	CPM_MON_LVOLTAGE_DROP	
TERM_LVOLTAGE_NOTOFFHOOK	(same as TERM_LVOLTAGE_ABOVE_OR_BELOW	
TERM_MON_UTONE+	CPM_MON_UTONE	Index Id
TERM_MON_CALLWAITING+	CPM_MON_CALLWAITING	
TERM_MON_RECEIVEOFF+	CPM_MON_RECEIVEOFF	
TERM_MON_DIAL1+	CPM_MON_DIAL1	
TERM_MON_DIAL2+	CPM_MON_DIAL2	
TERM_MON_DIAL3+	CPM_MON_DIAL3	
TERM_MON_BUSY1+	CPM_MON_BUSY1	
TERM_MON_BUSY2+	CPM_MON_BUSY2	
TERM_MON_FAX1+	CPM_MON_FAX1	
TERM_MON_FAX2+	CPM_MON_FAX2	
TERM_MON_SIT1+	CPM_MON_SIT1	
TERM_MON_SIT2+	CPM_MON_SIT2	
TERM_MON_SIT3+	CPM_MON_SIT3	
TERM_MON_SIT4+	CPM_MON_SIT4	
TERM_MON_SIT5+	CPM_MON_SIT5	

* These bits are maintained for backwards compatibility. New development should not use these options.

+ Call Progress Monitoring (CPM) must be enabled to detect these conditions

MONIENABLE

This field allows users to set the same monitoring conditions available using the **MTSetMoni()** API. This bit field controls the types of MON events that will be reported while this background function is running. This field can be used in combination with **MTSetMoni()** or configured to work alone. The CPM_MODE field of the MT_IO_CONTROL structure controls this. The following table lists all options available when setting the *MoniEnable* field:

Signal Name	MTSetMoni() Control	Corresponding Event
BUSY1	MONI_BUSY1	EVT_MON_BUSY1
BUSY1	MONI_BUSY*	EVT_MON_BUSY1 (EVT_MON_BUSY)
BUSY2	MONI_BUSY2	EVT_MON_BUSY2
BUSY2	MONI_TBUSY*	EVT_MON_BUSY2 EVT_MON_TBUSY)
DIAL1-3	MONI_DIAL*	EVT_MON_DIAL1-3
CALLWAITING	MONI_CALLWAITING	EVT_MON_CALLWAITING
RECEIVEOFF	MONI_RECEIVEOFF	EVT_MON_RECEIVEOFF
RINGBACK1	MONI_RINGBACK1	EVT_MON_RINGBACK1
RINGBACK1	MONI_RINGBACK*	EVT_MON_RINGBACK1 (EVT_MON_RINGBACK)
RINGBACK2	MONI_RINGBACK2	EVT_MON_RINGBACK2
RINGBACK2	MONI_DRINGBACK*	EVT_MON_RINGBACK2 EVT_MON_DRINGBACK)
SIT 1-5	MONI_SIT*	EVT_MON_SIT1-5
FAX1 & FAX2	MONI_FAX*	EVT_MON_FAX
FAX1	MONI_FAX1	EVT_MON_FAX1
FAX2	MONI_FAX2	EVT_MON_FAX2
	MONI_UTONE	EVT_UTONE_ON+
	MONI_SIGNAL_CYCLE	EVT_MON_SIGNAL_CYCLE
SIT1	MONI_SIT1	EVT_MON_SIT1
SIT2	MONI_SIT2	EVT_MON_SIT2
SIT3	MONI_SIT3	EVT_MON_SIT3
SIT4	MONI_SIT4	EVT_MON_SIT4
SIT5	MONI_SIT5	EVT_MON_SIT5
DIAL1	MONI_DIAL1	EVT_MON_DIAL1
DIAL2	MONI_DIAL2	EVT_MON_DIAL2
DIAL3	MONI_DIAL3	EVT_MON_DIAL3
	MONI_VOICE	EVT_MON_VOICE EVT_MON_HUMAN EVT_MON_MACHINE EVT_MON_NOVOICE
	MONI_REVERSAL	EVT_MON_REVERSAL

*These options have been maintained for backwards compatibility. They are not recommended for new development.

DIGIT TERMINATION CONTROL

MaxDigits

The background function terminates if the DTMF queue contains `MaxDigits` or more digits, resulting in an `EVT_MAXDIGITS` event. If there are digits in the DTMF queue this can cause the `MTGetDigits()` to terminate immediately. If this is undesirable, call the `MTClearDTMFDigits()` function first.

Only DTMF digits detected on the primary channel are used for termination control. On NGX and DP, the primary channel corresponds to the incoming direction, i.e. from CO to CPE.

DIGITENABLE

This 16-bit field specifies the termination digits. If a digit matches one of the specified digits, an `EVT_DIGITENABLE` event is generated. The bits are assigned as follows:

Bit:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Digit:	C	B	A	#	*	0	9	8	7	6	5	4	3	2	1	D

Within the SmartWORKS API, these digits are defined with the prefix `MT_DIGIT_`. For example, digit "1" is defined as `MT_DIGIT_1`, digit # is defined as `MT_DIGIT_POUND`.

ENDDIGIT

If one of the digits in the queue matches this character, the function terminates with an `EVT_TERMDIGIT` event. If this character is set to @, any digit will terminate the function. This field uses ASCII values such as '#' or '1' and not the bit values defined for the DigitEnable field.

To disable this feature, the value must be set to 0x0. If a value other than the above acceptable values is passed to the board, SmartWORKS automatically sets this value to 0x0 and disables this feature.

DIGITTIME

This specifies the maximum time, in milliseconds, between digits received. If the maximum time is exceeded, it causes an `EVT_INTERDIGIT` event.

Note: If a recording is terminated with an `EVT_MAXDIGITS`, `EVT_TERMDIGIT`, or `EVT_DIGITENABLE` event, the last 200ms (the last digit) of the recording is truncated. If any background function is terminated with an `EVT_MAXDIGITS`, `EVT_TERMDIGIT` or `EVT_DIGITENABLE` event, the terminating digit can be found in the SubReason field of the `MT_EVENT` structure. ‘

MISCELLANEOUS CONTROL DATA

STARTCONTROL

This field specifies the controls used for record or play.

TABLE 2: STARTCONTROL

Bit	Control	Definition
0	Reserved	Set to 0
1	PROMPT_TONE*	A prompt tone will be played before recording. ToneDuration field sets the length of this tone
2	INTER_DIGIT_DELAY	1 - Inter-digit delay starts after the first digit 0 - Inter-digit delay starts immediately
3	REPEAT_TONE	Repeat the playing of tones through MTPlayTone()
4	START_ACTD*	Activity triggered Recording. Automatically starts recording upon activity detection, event EVT_ACTD_RECORD will be issued
5.	START_LOOPON	Loop Recording: Automatically starts recording upon loop current detection and issues EVT_LOOPON_RECORD. If CPM termination for loop drop is <i>not</i> enabled, then recording automatically stops upon loop drop and EVT_LOOP_STOP is reported. If CPM termination is enabled, then EVT_CPM_STOP is reported.
6	N/A	Previously reserved for NOAGC which has been obsoleted
7	ALERTTONE_ON*	Plays Alert tone during Call logging
8	SILENCE_TRUNCATION*	Not to record the terminating silence
9	SILENCE_SUPPRESSION	Suppress silence during recording. This is to be added later
10	START_LVOLTAGE_MIDDLE* ----- START_LVOLTAGE_OFFHOOK*	Start recording on detection of loop voltage MIDDLE state. This is available on SmartWORKS LD only. If CPM termination for voltage is <i>not</i> enabled, then recording automatically stops when voltage is above or below thresholds and EVT_TERM_VOLTAGE_ABOVEORBELOW is reported. If CPM termination is enabled, then EVT_CPM_STOP is reported. ----- Recording starts on detection of loop voltage and event EVT_LVOLTAGE_RECORD will be issued (PT boards only). If CPM termination for voltage is <i>not</i> enabled, then recording automatically stops with an onhook condition and EVT_TERM_VOLTAGE_NOTOFFHOOK is reported. If CPM termination is enabled, then EVT_CPM_STOP is reported.
11	Reserved	Set to 0
12	GSM_FRAME34	Use NVDSP compatible 34-byte GSM frame length
13.	WAVE_RIFFFMTDATA_CHUNK	Add wave header, not available on streaming and DeviceIO functions

TABLE 2: STARTCONTROL (CONTINUED)

14.	WAVE_DATACHUNK_ONLY	Add wave data chunk descriptor only, not available on streaming and DeviceIO functions
15.	FRAME_HEADER (TBD)*	(TBD) Add a 2-byte frame size header to each frame. Set to 0.
16-31	Reserved	Set to 0.

*These fields are reserved for record functions only.

TONEDURATION

For a prompt tone to be played, the lowest four bits of this variable must specify the length of the tone in milliseconds.

TERMCONTROL

If TermControl is set to "1," all commands in queue will be deleted when a PlayBack task is terminated by any termination event set in [MT_IO_CONTROL](#).

[EVT_EOF](#) is the only termination event which cannot clear the command queue even if the TermControl field is set to "1."

This setting allows termination of the current **MTPlayFile()** command in the queue when the current play command is terminated by an event like [EVT_MAXDIGIT](#) or [EVT_TERMDIGIT](#). This control avoids the use of the **MTStopChannel()** command which resets all commands in the queue. [EVT_TERMDIGIT](#) immediately terminates only the playback currently in progress.

PLAY/RECORD SPECIFIC DATA

FILEINDEX

The file starting offset for record/play APIs. This field does not need to be populated for for playindex APIs.

FILEHANDLER

The user I/O access file descriptor for record/play.

FILETYPE

The media format.

DATA PORTION

LENGTH

This specifies the length of the data buffer.

BUFFER

This is required when using API functions that require a buffer from the user application. Pointer to a data buffer for the media data in the format specified in the FileType field.

A

API's Presented Alphabetically 10
API's Presented with PBX Support 10
APIs Presented Logically 23
Audio Jack Functions 61

B

Board Information Functions 126

C

Channel Functions 233
Contact Ai-Logix 8
Control Data viii
CT Bus Functions 443

D

D-Channel Event Codes 587

E

event codes
 user defined 593
Event Codes Presented Alphabetically 593
Event Functions 263
EVT_AIS_ALARM 594
EVT_ALERTING_AUTO_ANSWER 597
EVT_ANSWER_BUTTON_PRESSED 595
EVT_ANSWER_LIGHT_FASTFLASHING 596, 672, 699, 764
EVT_ANSWER_LIGHT_FLASHING 597
EVT_ANSWER_LIGHT_OFF 597, 609
EVT_ANSWER_LIGHT_ON 597
EVT_ANSWER_LIGHT_Q UICKFLASH 597
EVT_ANSWER_LIGHT_VERY_FASTFLASHING 597
EVT_AUDIO_CHANGE 597
EVT_AUDIO_OFF 599
EVT_AUDIO_ON 599
EVT_AUTO_ANSWER 599, 600
EVT_AUTO_ANSWER_CANCELED 599
EVT_BOARD_PANIC_ERROR 602
EVT_BREAK_BUTTON_PRESSED 602
EVT_CALL_ABANDONED 602
EVT_CALL_CONNECTED 602
EVT_CALL_HELD 603
EVT_CALL_REJECTED 603
EVT_CALL_RELEASED 603
EVT_CALL_RESUMED 603
EVT_CALL_RETRIEVED 603
EVT_CALL_SUSPENDED 603
EVT_CALLID_DROPPED 603
EVT_CALLID_MSGLEN 605
EVT_CALLID_START 605
EVT_CALLID_STOP 605
EVT_CC_ALERT_IND 609
EVT_CC_CALL_ABANDONED 613
EVT_CC_CALL_CONNECTED 614
EVT_CC_CALL_HELD 616
EVT_CC_CALL_REJECTED 618
EVT_CC_CALL_RELEASED 620

EVT_CC_CALL_RESUMED 622
EVT_CC_CALL_RETRIEVED 624
EVT_CC_CALL_SUSPENDED 626
EVT_CC_CONNECT_CONF 628
EVT_CC_DISC_CONF 632
EVT_CC_DISC_IND 634
EVT_CC_INFO_IND 638
EVT_CC_PROGRESS_IND 642
EVT_CC_USER_INFO 647
EVT_CC_USERINFO_IND 652
EVT_CFW 654
EVT_CFW_CANCELED 655
EVT_CONF_BUTTON_PRESSED 659
EVT_CONF_BUTTON_RELEASED 660, 782, 814, 825
EVT_CONFEREENCE_LIGHT_FASTFLASHING 656, 661
EVT_CONFEREENCE_LIGHT_FLASHING 656, 661
EVT_CONFEREENCE_LIGHT_OFF 662
EVT_CONFEREENCE_LIGHT_ON 656, 661
EVT_CONFEREENCE_LIGHT_QUICKFLASH 662
EVT_CONFEREENCE_LIGHT_VERY_FASTFLASHING 662
EVT_CONNECTED 662
EVT_CPM_STOP 663
EVT_DEVICE_READ_ERROR 666
EVT_DEVICE_SEEK_ERROR 666
EVT_DEVICE_WRITE_ERROR 667
EVT_DIALED 667
EVT_DIGIT 668
EVT_DIGIT_DROPPED 668
EVT_DIGIT_PRESSED 669
EVT_DIGIT_RELEASED 669
EVT_DIGITENABLE 670
EVT_DISCONNECTED 673
EVT_DISK_FULL 673
EVT_DISK_WRITE_ERROR 673
EVT_DISPLAY 674
EVT_DISPLAY_CLEAR 674
EVT_DISPLAY_CLOCK 674
EVT_DISPLAY_MESSAGE 675
EVT_DSPRESET 681
EVT_EOF 682
EVT_ERROR 682
EVT_EXIT_BUTTON_PRESSED 683, 684
EVT_FAX_ERROR 684
EVT_FAX_PHASED_COMP 685
EVT_FAXRECVEOP 685
EVT_FEATURE_BUTTON_PRESSED 685
EVT_FEATURE_HEADER_ERROR 687
EVT_FEATURE_LIGHT_FASTFLASHING 686
EVT_FEATURE_LIGHT_FLASHING 687
EVT_FINISHED_PLAY 687
EVT_FUNCTION_LIGHT_FASTFLASHING 693
EVT_H100_MASTER_A_ALARM 697
EVT_H100_MASTER_B_ALARM 697
EVT_HELD 701

EVT_HELP_BUTTON_PRESSED 702
EVT_HOLD_BUTTON_PRESSED 705
EVT_HOLD_BUTTON_RELEASED 705
EVT_HOLD_LIGHT_FLASHING 707
EVT_IDLE 707
EVT_INCOMING_SIGCHANGE 707
EVT_INDEX_PLAYED 708
EVT_INTERDIGIT 709
EVT_LIGHT_FASTFLASHING 710
EVT_LIGHT_FLASHING 711
EVT_LINE_BUTTON_PRESSED 711
EVT_LINE_LIGHT_FASTFLASHING 712
EVT_LOCRMF_ALARM 714
EVT_LOF_ALARM 714
EVT_LOGON_BUTTON_PRESSED 715
EVT_LOOP_DROP 715
EVT_LOOP_ON 715
EVT_LOOP_REVERSE 716
EVT_LOOP_STOP 716
EVT_LOOPON_RECORD 717
EVT_LOS_ALARM 718
EVT_LOSMF_ALARM 719
EVT_LVOLTAGE_NOTOFFHOOK 721
EVT_LVOLTAGE_OFFHOOK 720, 722
EVT_LVOLTAGE_RECORD 722
EVT_MASTER_PLL_ALARM 724
EVT_MAX_ACTIVITY 724
EVT_MAX_SILENCE 725
EVT_MAXBYTES 725
EVT_MAXDIGITS 726
EVT_MAXTIME 726
EVT_MENU_BUTTON_PRESSED 727, 729, 730
EVT_MENU_LIGHT_FASTFLASHING 729, 730
EVT_MESSAGE_BUTTON_PRESSED 730
EVT_MESSAGE_CHANGE 730
EVT_MESSAGE_LIGHT_FASTFLASHING 732
EVT_MFTONE 733
EVT_MFTONE_DROPPED 733
EVT_MIC_LIGHT_FASTFLASHING 735
EVT_MON_ACTIVITY 736
EVT_MON_BUSY 737
EVT_MON_BUSY_TRUNK 761
EVT_MON_DIAL 741, 742, 824
EVT_MON_DRINGBACK 745
EVT_MON_FAX 746, 747, 748
EVT_MON_RINGBACK 750
EVT_MON_SILENCE 754
EVT_MON_SIT 754, 755, 756, 757, 758, 759, 760
EVT_MUTE_BUTTON_PRESSED 763, 765
EVT_MVIP_ALARM 765
EVT_NEXT_BUTTON_PRESSED 766
EVT_NO_EFFECT 767
EVT_NOT_AVAILABLE 767
EVT_NOT_SUPPORTED 767
EVT_OFFHOOK 768
EVT_ONHOOK 768
EVT_OUTGOING_SIGCHANGE 770
EVT_PBX_DCH 775
EVT_PHN_DCH 775
EVT_PREVIOUS_BUTTON_PRESSED 776, 778
EVT_RECALL_BUTTON_PRESSED 784
EVT_REDIAL_BUTTON_PRESSED 784
EVT_REDIAL_BUTTON_RELEASED 784
EVT_REJECTED 785
EVT_RELEASE_BUTTON_PRESSED 786
EVT_RELEASE_BUTTON_RELEASED 786
EVT_RELEASE_LIGHT_FASTFLASHING 787
EVT_RELEASE_LIGHT_FLASHING 787
EVT_RELEASED 787
EVT_REORDER 788
EVT_RESUMED 789
EVT_REVERSAL 789
EVT_RING_LIGHT_FASTFLASHING 789
EVT_RING_LIGHT_FLASHING 789
EVT_RING_OFF 790
EVT_RING_ON 790
EVT_RINGS 791
EVT_RX_HDLC_ACTIVE 793
EVT_SET_BUSY 798
EVT_SET_BUSY_CANCELED 798
EVT_SETPAR_DONE 798
EVT_SOFT_BUTTON_PRESSED 799
EVT_SOFT_LIGHT_FASTFLASHING 801, 803
EVT_SPEAKER_BUTTON_PRESSED 801, 803
EVT_SPEAKER_BUTTON_RELEASED 801
EVT_SPEAKER_LIGHT_FASTFLASHING 802, 804
EVT_STOP 804
EVT_STREAMIN_DROPPED 808
EVT_STREAMIN_START 809
EVT_STREAMIN_STOP 809
EVT_STREAMIN_WATERMARK 809
EVT_STREAMOUT_COMP 811
EVT_STREAMOUT_EMPTY 811
EVT_STREAMOUT_PAUSED 811
EVT_STREAMOUT_RESUMED 812
EVT_STREAMOUT_START 812
EVT_STREAMOUT_STOP 813
EVT_STREAMOUT_WATERMARK 813
EVT_SUPERVISOR_LIGHT_FLASHING 815
EVT_SUSPENDED 816
EVT_SYS_ERROR 816
EVT_TERMSILENCE 820
EVT_TERMVOLTAGE_NOTOFFHOOK 819
EVT_TONEP_DONE 821
EVT_TONEPERR 821
EVT_TRANSFER_BUTTON_PRESSED 821
EVT_TRANSFER_BUTTON_RELEASED 821
EVT_TRANSFER_LIGHT_FASTFLASHING 822

EVT_TRANSFER_LIGHT_FLASHING 822
EVT_TS16RAI_ALARM 822
EVT_UP_DOWN 823
EVT_USER_INFO 824
EVT_UTONE_OFF 824
EVT_UTONE_ON 824
EVT_UTONE_TON 824
EVT_VOICE_RECORD 824
EVT_VOX_OFF 824
EVT_WORK_BUTTON_PRESSED 824
EVT_WRAPUP_BUTTON_PRESSED 824
EVT_WRAPUP_LIGHT_FLASHING 827
EVT_YELLOW_ALARM 827

G

Gain/Volume Functions 266
General Termination Control iii

H

HIGHPASS Filter Functions 293

L

Loop Voltage Functions 467

M

MakeWaveGSMHeader 503
MakeWavePCMHeader 504
Media Access with Device IO Functions 296
Misc. Functions 24
MT_EVENT ix
MTAddSummationInput 483
MTAGCCControl 281, 282
MTAGCStatus 283, 284
MTAJListen 65
MTAJTalk 63
MTArmCTAlarm 447
MTArmFramerAlarm 31
MTArmMasterAlarm 444
MTAVCCControl 290
MTAVCStatus 292
MTBlinkBoard 134
MTCallStringA 234
MTCallStringAsync 234
MTCallStringEx 234
MTCC_ConnectRequest 211
MTCC_ConnectResp 213
MTCC_DiscRequest 215
MTCC_FacilityRequest 218
MTCC_GetStatusByCh 223
MTCC_GetStatusByRef 225
MTCC_ResumeRequest 221
MTCC_SuspendRequest 219
MTCC_UserInfoRequest 220
MTClearBoardEventCallback 94
MTClearDigits 259
MTClearDTMFDigits 259
MTClearEventCallback 81, 98, 99
MTClearFramerStatistic 47, 49, 50, 51

MTClearMFTones 260
MTClearPriorityEventCallback 84
MTClearUsrTone 401, 402
MTClearUsrToneParams 401
MTCloseBoard 123
MTCloseChannel 182
MTControlToneQ 261, 262
MTDialString 238
MTDialStringAsync 238
MTDialStringEx 238
MTDisableCallerID 241
MTDisableFramerABCDSignalingEvent 54
MTDisableMixing 377
MTDisarmCTAlarm 448
MTDisarmFramerAlarm 33
MTDisarmMasterAlarm 445
MTEnableACTD 373, 374
MTEnableCallerID 242, 244, 246, 247, 248
MTEnableFramerABCDSignalingEvent 52
MTEventControl 77
MTFlushCallerID 250
MTFlushEvents 78, 190
MTGetACTDParams 371
MTGetACTDRegs 371
MTGetAdapterDescription 132
MTGetAdapterInfo 126, 131, 136, 176
MTGetAdapterXInfo 129
MTGetAGC 271, 272
MTGetAJListen 69
MTGetAJMap 62
MTGetAJTalk 68
MTGetAlertTone 424
MTGetAlertToneParams 424
MTGetAVC 285
MTGetBoardAssemblyInfo 137
MTGetBoardOEMInfo 169
MTGetBoardOpenStatus 124
MTGetCallerID 249
MTGetCallerIDStatus 243
MTGetCallStatus 251
MTGetCCStatistics 230
MTGetChannelEvent 85
MTGetChannelInfo 201
MTGetChannelOpenStatus 195
MTGetChannelOwner 203
MTGetChannelStatus 196
MTGetCpmTone 388
MTGetCpmToneParams 388
MTGetCTAlarmStatus 449
MTGetCTMasterClock 152
MTGetCTNetrefSource 156
MTGetCTRoute 452
MTGetCTSlaveClockMode 158
MTGetCTSlotConnection 309

MTGetDataLinkStatistics 228
MTGetDataLinkStatus 227
MTGetDetectRegs 357
MTGetDigits 253
MTGetDigitsAsync 253
MTGetDigitsEx 253
MTGetDLLVersion 116, 118
MTGetE1FramerStatistic 41
MTGetEventFilters 74, 75
MTGetFramerABCDSignalingEventStatus 55
MTGetFramerAlarmArmStatus 35
MTGetFramerAlarmStatus 36
MTGetFramerOutput 462
MTGetFraming 45
MTGetGain 276, 277
MTGetGCI 192
MTGetGCIMap 193
MTGetGenRegs 349
MTGetHookState 500
MTGetInputs 456
MTGetLastErrorA 26, 28, 29
MTGetLastErrorW 27
MTGetLastTerm 435
MTGetLCDetectState 469
MTGetLVDetectState 468
MTGetMasterAlarmStatus 446
MTGetMixingStatus 378
MTGetMoni 442
MTGetMVIPAlarmStatus 449
MTGetNET 159
MTGetNGXFramerStatistic 43
MTGetOutput 460
MTGetOutputSource 265
MTGetPHYStatistics 231
MTGetPSTN 341
MTGetPSTNParams 341
MTGetRingDetectState 470
MTGetStatistics 205
MTGetStatus 196
MTGetStreamingConfig 329
MTGetSummationAVC 494
MTGetSummationGain 490
MTGetSummationInfo 478
MTGetSummationInputs 486
MTGetSummationOutput 488
MTGetSummationVolume 492
MTGetSysInfo 110
MTGetSystemInfo 110
MTGetT1FramerStatistic 39
MTGetTerm 439
MTGetTimeSlotRxSignaling 56
MTGetTimeSlotTxSignaling 60
MTGetToneDetectParams 357
MTGetToneGenParams 349, 359, 361, 363
MTGetUsrTone 397, 399
MTGetUsrToneParams 397
MTGetVersion 114
MTGetVolume 287
MTHighpassFilterControl 294
MTHighpassFilterStatus 295
MTOffHook 498
MTOOnHook 499
MTOpenBoard 122
MTOpenChannel 181
MTPlayBuffer 311
MTPlayBufferAsync 311
MTPlayBufferEx 311
MTPlayDevice 300
MTPlayDeviceAsync 300
MTPlayDeviceEx 300
MTPlayFile 313
MTPlayFileAsync 313
MTPlayFileEx 313
MTPlayIndex 317
MTPlayIndexAsync 317
MTPlayIndexEx 317
MTPlayTone 431
MTPlayToneAsync 431
MTPlayToneEx 431
MTPlayUsrTone 431
MTPutBoardEvent 91
MTPutChannelEvent 89
MTPutEvent 89
MTReadDigit 256
MTReadDTMFTone 257
MTReadMFTone 258
MTRecBuffer 321
MTRecBufferAsync 321
MTRecBufferEx 321
MTRecDevice 302
MTRecDeviceAsync 302
MTRecDeviceEx 302
MTRecFile 323
MTRecFileAsync 323
MTRecFileEx 323
MTRemoveSummationInput 485
MTResetAJListen 71
MTResetAJTalk 70
MTResetBoardStatistics 209
MTResetCpmToneParams 393, 414, 417, 419
MTResetCTRoute 453
MTResetFramerOutput 466
MTResetInputs 458
MTResetOutput 461
MTResetRoute 453
MTResetStatistics 207
MTResetSummation 480
MTReturnEventBuffer 505

MTSetACTDParams 366, 369
MTSetACTDRegs 366, 369
MTSetAGC 267, 269
MTSetAlertTone 426
MTSetAlertToneParams 426
MTSetCpmTone 390, 411, 413
MTSetCpmToneParams 390
MTSetCTMasterClock 150
MTSetCTNetrefSource 154
MTSetCTRoute 450
MTSetCTSlaveClockMode 157
MTSetDetectRegs 351
MTSetDeviceIO 297
MTSetECControl 379, 380, 381, 382, 383, 385
MTSetEventCallback 79
MTSetFramerOutput 463
MTSetGain 274
MTSetGenRegs 347
MTSetInputs 454
MTSetMoni 440
MTSetNET 160
MTSetNET1, MTSetNET2 177
MTSetOutput 459
MTSetOutputSource 263
MTSetPriorityEventCallback 82
MTSetPSTN 343
MTSetPSTNParams 343
MTSetPxdEcho 379
MTSetRoute 450
MTSetStreamingConfig 328
MTSetSummationAVC 493
MTSetSummationGain 489
MTSetSummationVolume 491
MTSetTerm 436
MTSetTimeSlotTxSignaling 58
MTSetToneDetectParams 351, 354
MTSetToneGenParams 347
MTSetUsrTone 403, 405
MTSetUsrToneParams 403
MTSetVolume 286
MTStartAlertTone 428
MTStartLink 183
MTStartMUX 305
MTStartMUXOnChannel 163, 164, 166, 167, 507
MTStartStreaming 330
MTStartSummation 481
MTStepGain 278, 279
MTStepVolume 288
MTStopAlertTone 429
MTStopBlinkBoard 135
MTStopChannel 186
MTStopCurrentFunction 188
MTStopLink 184
MTStopMUX 308

MTStopMUXOnChannel 307
MTStopStreaming 339
MTStopSummation 482
MTStreamBufIn 334
MTStreamBufOut 332
MTStreamBufPause 336
MTStreamBufResume 337
MTSummationAVCControl 495
MTSummationAVCStatus 496
MTSysShutdown 184
MTSysStartup 183
MTWaitForAdapterEvent 95
MTWaitForChannelEvent 87, 100
MTWink 501
Mux/MVIP bus Routines 304

P

Play Data ix
Play Functions 310
Play/Record Data ix

R

Record Data ix
Record Functions 320
Ring Detection Status Functions 467

S

Signal Termination Control iii
Stream Functions - Wave Streaming 326
Streaming - Stream Control 338
Structure Functions 340
Summation Functions 477

T

Terminate cards (T1 RBS, ISDN) 210, 217, 222
Termination Control iii
Termination/Monitor Functions 434
Tone Functions - Alert Tones 423
Tone Functions - CPM 387
Tone Functions - Play Tones 430
Trunk Specific APIs 497

W

Wave Header Utility 502