AudioCodes

BLADES BUSINESS LINE

SmartWORKS Utilities Guide

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 0001-004 Build 090828 Rev B

Table Of Contents

Table of Contents

Chapter 1

Introduction

Introduction

This documentation is intended for the developer of CTI application software. This manual assumes the reader is fairly proficient with computer telephony and voice processing components.

Chapter Descriptions

The utilities described in this document are applications designed for the configuration, and maintenance of AudioCodes' SmartWORKS products only. Each utility is available when SmartWORKS is installed onto your system.

The purpose of each chapter is described below:

- "Smart WF" on page 7explains the software utility used to manage the factory installed board firmware.
- "Control Panel" on page 11 provides an overview of the Control Panel applet designed for board configuration.
- "SmartView" on page 27 illustrates the use of the SmartView application used for board testing and troubleshooting purposes.
- "SmartProfiler" on page 89 provides instructions on how to analyze a tone using the SmartProfiler.

Related Documents

For additional information, refer to the following documents:

- The *SmartWORKS Developer's Guide* located on the product CD-ROM.
- The *SmartWORKS User's Guide* located on the product CD-ROM.

Document Version Control

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

TABLE 1: VERSON CONTROL

Page	Description
REV A	
	no documentation changes

Legal Notice

Product Support

This document applies to the following AudioCodes products

NOTE: Lead free boards are referenced by weight.

Product Name	Part Number	Weight	Status 3.10.x
SmartWORKS VR3200	910-0303-001		Retired
SmartWORKS VR3209	910-0303-002		Maintenance
SmartWORKS VR6400	910-0301-001		Retired
SmartWORKS VR6409	910-0321-001		Maintenance
SmartWORKS AT409	910-0328-001		Retired
SmartWORKS AT809	910-0318-001		Retired
SmartWORKS AT1600	910-0309-001		Retired
SmartWORKS AT1609	910-0309-002		Retired
SmartWORKS DP3200	910-0308-001		Retired
SmartWORKS DP3209	910-0308-002	255 g	Released
SmartWORKS DP6400	910-0304-001		Retired
SmartWORKS DP6409	910-0324-001	280 g	Released
SmartWORKS DP3209-eh	910-0703-001	245 g	Released (specical order only)

Product Name	Part Number	Weight	Status 3.10.x
SmartWORKS DP6409-eh	910-0703-002	270 g	Released (specical 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
SmartWORKS MX80 (Expansion)	910-0315-001	68 g	Released
SmartWORKS NGX800A	910-1314-001	TBD#	Released
SmartWORKS NGX1600A	910-1314-002	TBD#	Released
SmartWORKS NGX2400A	910-1314-003	TBD#	Released
SmartWORKS MX80A (Expansion)	910-1315-001	60 g	Released
SmartWORKS NGX800-eh	910-0700-001	208 g	Released
SmartWORKS NGX1600-eh	910-0700-002	269 g	Released
SmartWORKS NGX2400-eh	910-0700-003	208 g	Released
SmartWORKS PT409	910-0307-002		Retired
SmartWORKS PT800	910-0305-001		Retired
SmartWORKS PT809	910-0320-001		Retired
SmartWORKS PT1600	910-0306-001		Retired
SmartWORKS PT1609	910-0319-001		Retired
SmartWORKS LD 101	910-0805-001	130 g	Released
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 (specical order only)
SmartWORKS LD 1609-eh	910-0701-002	460 g	Released (specical order only)
SmartWORKS LD 2409-eh	910-0701-003	575 g	Released (specical order only)
SmartWORKS DT3200	910-0312-001		Retired
SmartWORKS DT3209	910-0325-001		Retired

Product Name	Part Number	Weight	Status 3.10.x
SmartWORKS DT6400	910-0313-001		Retired
SmartWORKS DT6409	910-0323-001		Retired
SmartWORKS DT6409TE	910-0323-002	275 g	Released
SmartWORKS DT3209TE	910-0325-002		Released
SmartWORKS DT3209TE-eh	910-0704-001	240 g	Released (specical order only)
SmartWORKS DT6409TE-eh	910-0704-002	265 g	Released (specical order only)
SmartWORKS PCM 3209	910-0330-001		Retired
SmartWORKS PCM 6409	910-0329-001	270 g	Released (specical order only)
SmartWORKS PCM 3209-eh	910-0702-001	265 g	Released (specical order only)
SmartWORKS PCM 6409-eh	910-0702-002	240 g	Released (specical order only)
SmartWORKS IPX	901-0331-001		Retired
SmartWORKS IPX-C	910-0331-007	250 g	Released

Value not available at time of document publication.

Contacting AudioCodes USA

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.
- E-mail technical support at blade-support@audiocodes.com. Be sure to include a detailed description of the problem along with PC configuration, AudioCodes hardware, driver versions, firmware versions, a sample program that demonstrates the issue, and any other pertinent information.

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:

AudioCodes USA, Inc.

27 World's Fair Drive

Somerset, NJ 08873

Chapter 2

Smart WF

SmartWF - Firmware Update Utility

SmartWF is a software utility used to manage the factory installed board-level firmware. It is typically used to upgrade the firmware image of a board when installing a new software release on a system.The utility is used to complete the following tasks:

- Verification of the firmware version on the SmartWORKS board(s) in the system.
- Saving firmware from the board(s) into image file(s).
- Transference of the firmware from the image file into the on-board FLASH.
- Verification of the FLASH image with binary file.

NOTE: It is required that you upgrade the board firmware whenever new SmartWORKS software is installed.

Automatic Update

As of the SmartWORKS 2.10 release, SmartWF automatically recognizes the board's software version and will update the firmware to match. This only applies to boards running SmartWORKS 2.10 or above. Users running earlier releases must manually update the board's firmware with this utility.

NOTE: Due to the firmware changes associated with the 3.9 release, once an IPX board has been installed on a system running SmartWORKS 3.9, this board can no longer be installed on a system running version 3.8 or earlier. This only applies to IPX and IPX-C products.

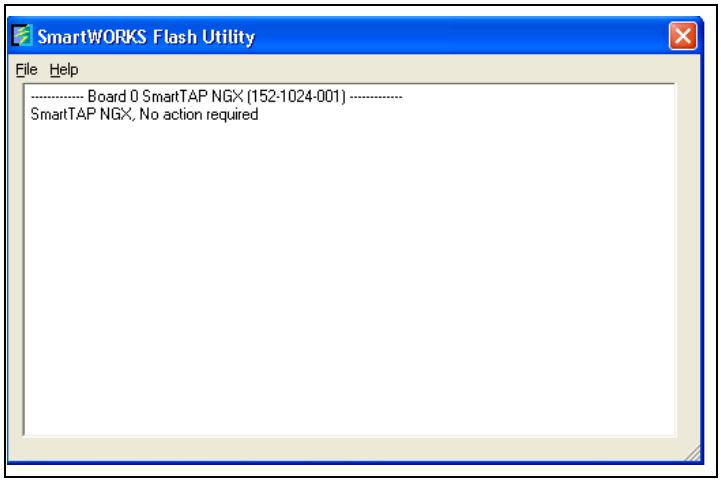
Running the SmartWORKS Flash Utility

From the Windows Start menu, run **Programs > Ai-Logix SmartWORKS > SmartWF**.=

Note for Linux Users

This application does not work in unattended mode. All command line prompts must be typed with lower case letters.

Figure 2-1: SmartWORKS Flash Utility



SmartWORKS Flash Utility Menu Options

For a description of the SmartWORKS Flash Utility menu options, refer to the details below.

Check Flash Version

Selecting this item retrieves the current firmware version of any and all SmartWORKS boards in the system.

Clear Screen

Selecting this item clears the screen beneath ensuring that the latest information is being viewed when using the Check Flash Version function.

Save Flash Image

Selecting this item opens a dialog box that allows you to save the SmartWORKS board's current firmware image to a user specified location.

Write Flash Image

Selecting this item opens a dialog box that allows you to select and download a new firmware image to the SmartWORKS board.

If SmartWORKS was installed with default settings, the location of this file is C:\Program Files\Ai-Logix \SmartWORKS\ Firmware\{SmartWORKS Product Name}. This path will vary depending on your installation.

For the latest firmware, visit the support section of our web site located at http://www.audiocodes.com/blades/support/. Be sure to load the latest version.

Validate Image

Selecting this item opens a dialog box that allows you to select a firmware image and compare it against the firmware currently loaded on the SmartWORKS board.

Chapter 3

Control Panel

SmartControl Panel Applet

The SmartControl applet is used by anyone installing or managing a SmartWORKS board on a telephony network: application developers, system administrators, or end users.

SmartControl provides an interface to many board configuration settings. Users have the choice of configuring the board using SmartWORKS APIs or SmartControl. Once a board is installed and configured for a system, current settings can be viewed through SmartControl.

AudioCodes technical support can also remotely access SmartControl for the purpose of troubleshooting and reviewing board configuration relative to your system.

NOTE: If any changes are made with SmartControl, the board's drivers must be restarted.

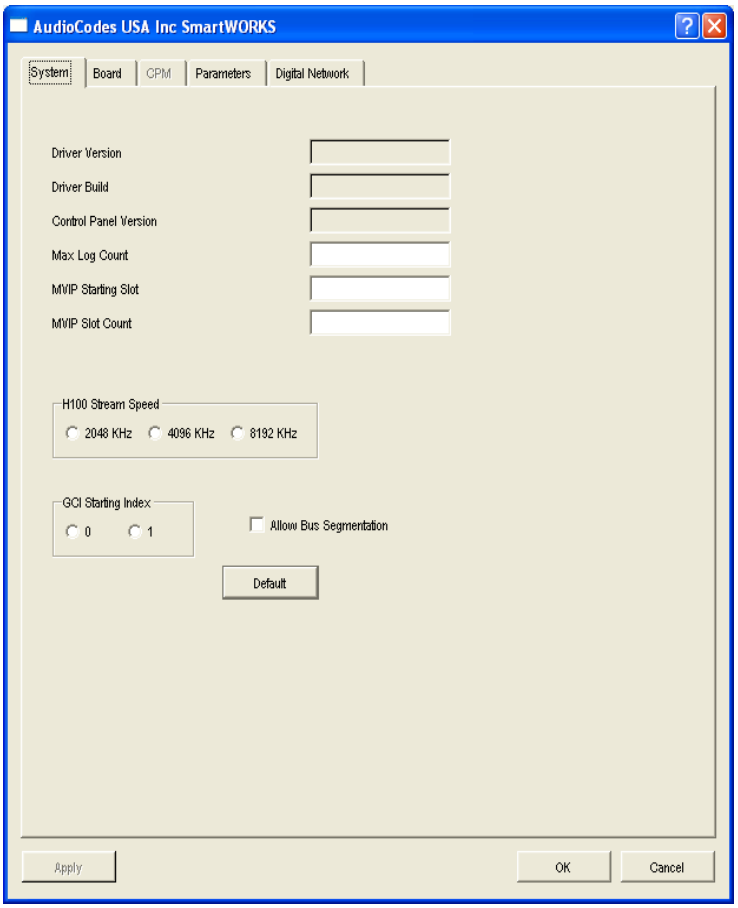
NOTE FOR LINUX USERS

This command line prompt is not supported when using Linux. Users must rely on the control panel's GUI.

System Tab

The following sections describe the parameters of the SmartControl tabs. Default parameters for each tab can be restored by clicking the *Defaults* button.

Figure 3-1: System Tab



NOTE: The Control Panel interface changes depending on the type of card installed. Not each tab is displayed for every SmartWORKS board.

Driver Version

Displays current version of the SmartWORKS driver.

Max Log Count

This parameter specifies the maximum number of event logs the SmartWORKS API is allowed to log into the Windows Event Viewer. The default number is set to 100 to avoid flooding the Event Viewer. **NOTE:** When using Linux, all information is written to a 'messages' file located in the /var/log directory.

MVIP Starting Slot

This is the offset for the TDM slot numbering scheme. It allows you to specify which timeslot to start from for use with commands MTJoin(), MTSetPoint(), and MTSetRoute().

MVIP Slot Count

This parameter allows you to specify the total number of MVIP timeslots available. The sum of TDMStartingSlot and TDMSlotCount cannot exceed 256.

H.100 Stream Speed

Allows you to designate stream speed between 2048, 4096, or 8192 KHz.

GCI Starting Index

Allows the first channel value to be 0 or1. Each time a channel index value is returned, this setting is taken into account.

Halt on Startup Errors

When selected the system will not startup if an error is encountered.

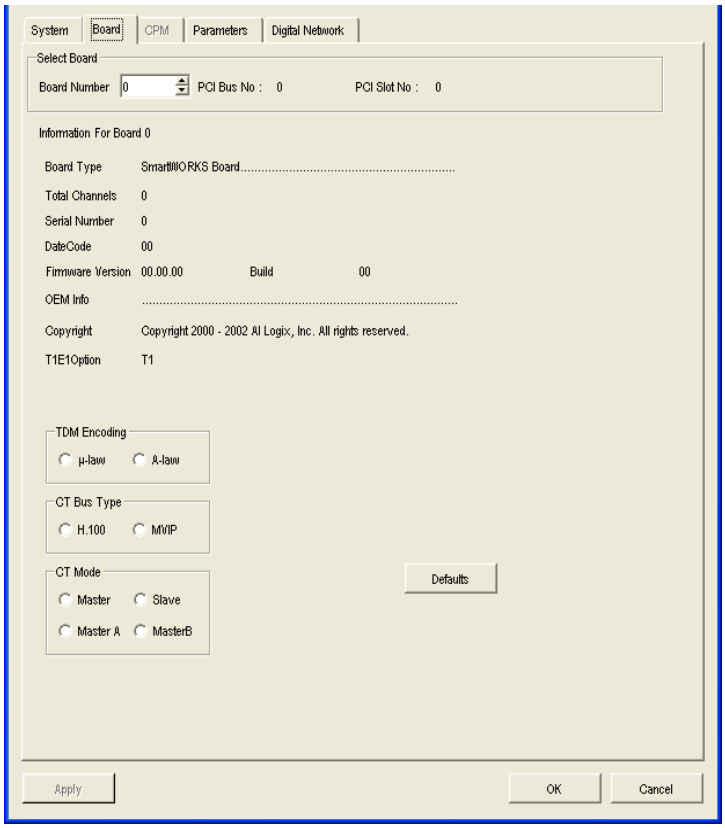
Allow Bus Segmentation

When selected multiple boards can be set to MASTER.

Board Tab

Note that the Board tab for the NGX and the IPX differs from the other boards. This is due to the fact that two daughter cards can be added to the NGX base board to increase port density. If you are using an NGX board, see Figure 3·2 on page 17 and Figure 3·3 on page 18. The IPX requires port configuration, see Figure on page 18 . Otherwise, refer to the image on the next page:

NOTE: All boards can function separately by setting each board to MASTER; or, separate configurations of multiple boards can exist within the same chassis of a system by designating one MASTER for each group of boards



BOARD NUMBER

This parameter allows you to select a specific SmartWORKS board (if multiple SmartWORKS boards are in a single system) and configure the settings for each individual SmartWORKS board.

PCI BUS NUMBER

This parameter displays the PCI bus number that corresponds with the BIOS configuration.

PCI SLOT

This parameter displays the PCI slot number of the currently selected SmartWORKS board.

PBX TYPE (NGX ONLY)

Drop-down menu that allows you to designate a PBX type.

To Select a PBX perform the following steps:

1. Select PBX; click apply. Note, for the changes to take effect you must click apply. If you have daughter cards, be certain to select the proper PBX for each daughter card and click apply for the changes to occur.
2. Restart the board (either under device manager or upon reboot of system). NOTE: A command line program has been provided when running Linux.

TERMINATION (NGX ONLY)

Allows you to designate termination to be 120 Ohm or Hi-Z.

DCHANNEL OPTIONS (NGX ONLY)

Allows you to enable and disable D-Channel and event updates options:

DChannel

Selecting the check-box enables reporting of D-Channel events.

Event Updates

Selecting the check-box enables all events to be reported to the application. By default, event filtering is enabled to prevent redundant information.

BOARD TYPE

Displays the type of board.

TOTAL CHANNELS

This parameter displays the total number of channels per SmartWORKS board.

CHANNELS (NGX ONLY)

This parameter displays the number of channels per NGX board (base or daughter).

SERIAL NUMBER

Displays the serial number of the SmartWORKS board.

OEM INFO

Displays OEM information for the SmartWORKS board.

COPYRIGHT

Displays Copyright information.

FIRMWARE VERSION

This parameter displays the current firmware version on the SmartWORKS board.

TDM ENCODING

Allows you to select encoding type used on the TDM bus:

- µ-Law is used in North America and Japan.
- A-Law is used in Europe and in areas outside of North American influence.

Board Switch ID

Displays the user-defined board value (NGX only). The Control panel reads the value directly from the registry settings which are updated when the board's driver is restarted. Should the value be changed using the board's thumb wheel, and the board's drivers have not been restarted then the information displayed in the Control Panel will *not* reflect the current setting.

T1 E1 Option

Selects between T1 or E1 framer mode. This selection affects all framers on the board.

CT Bus Type

Allows you to designate the bus type as either H.100 or MVIP.

CT Bus Termination (NGX Base Board Only)

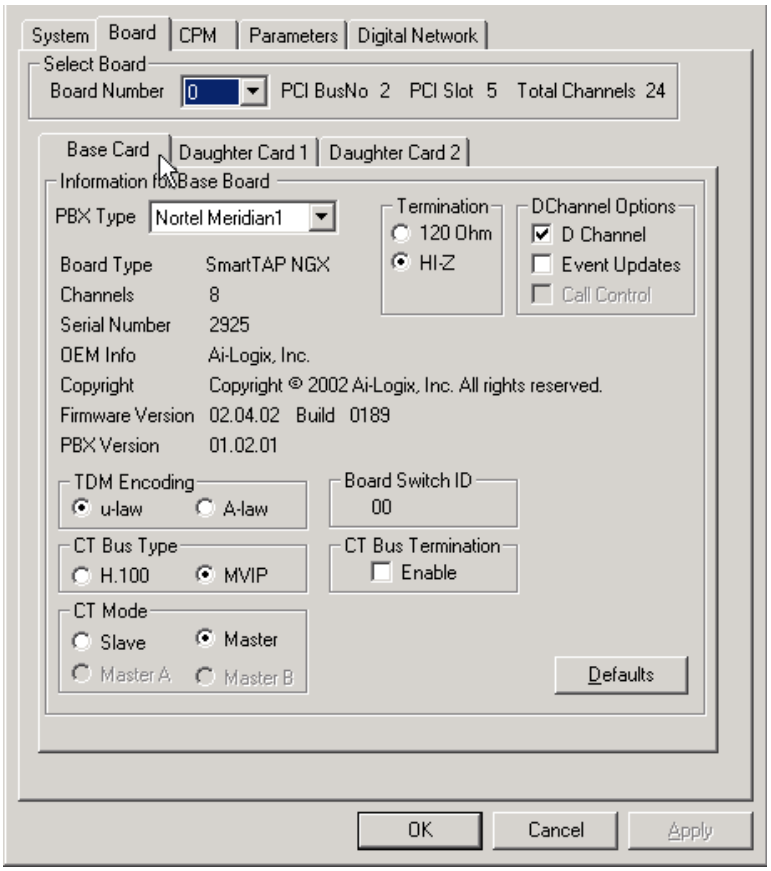
Allows you to enable CT Bus termination.

Master Mode

Allows you to position the clock setting to either SLAVE or MASTER.

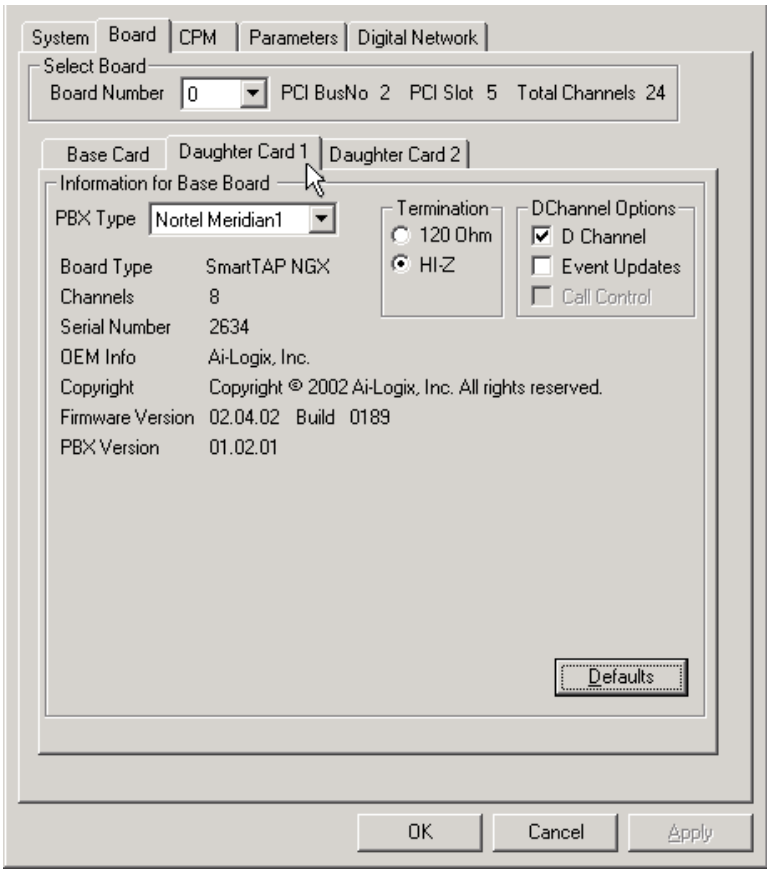
Board Tab NGX Base Board

Figure 3-2: Board Tab NGX Base Board



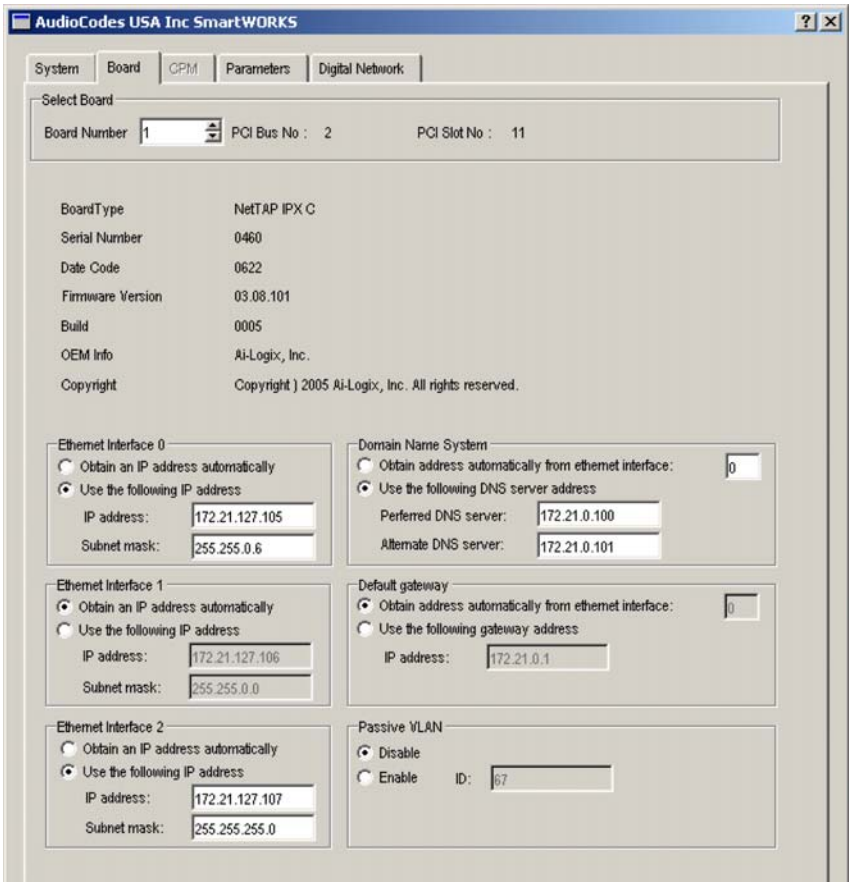
BOARD TAB NGX DAUGHTER CARD

Figure 3-3: Board Tab NGX Daughter Card



BOARD TAB IPX CARD

Figure 3·4:
Figure 3·5: Board Tab IPX Card



IPX CONFIGURATION

All three interfaces on the IPX board can be either passive or active ports. Typically, ports 1&2 are used in promiscuous mode to receive incoming data from the line/s that are monitored. A single port, typically Interface 0, is used to forward RTP packets to the recording apparatus. When the IPX is used in this scenario - only port 0 must be configured with network parameters. Interfaces 1&2, when used in promiscuous mode do not have to be configured, and the default values can be used. **NOTE:** It is important that these default paramters do not match the values used when configuring port 0 as this will cause a routing conflict.

The Default Gateway is a board setting. When setting this value, it is important that the default gateway IP Address is set properly for use by the transmitting port (typically port 0).

DHCP is supported, and can be enabled by using the “Obtain an IP address automatically” radio button on a per port basis, as well as when setting the Default Gateway. When DHCP is used to obtain the IP Address of the default gateway, it is best to enable this on port 0, the port used for transmitting media.

DNS, though visible on the Control Panel, is not yet supported by the IPX. It has been added for future implementations.

HPX Configuration

Users have the ability to select up to two NIC cards for monitoring. These are the NIC cards used for receiving signaling and RTP packets from the network.

General Configuration of both the IPX and HPX

VPN - this field allows the user to identify a VLAN ID used on the tapped network. If this field is enabled, the IPX or HPX products only analyze packets with this VLAN ID tag

RTP Timeout - users can enable or disable this RTP timeout feature (this feature is fully explained in the *IPX/HPX Integration Guide*). By default, this feature is disabled. When enabled, the default setting is 15 seconds.

RTCP QoS reporting - when the media session stopped event is reported, users may also receive a quality report of this session. This report is fully documented in both the *IPX/HPX Integration Guide* as well as were the event is defined in the Function Reference Library (FRL). This report can be used by the application to monitor the total number of RTP packets received by the IPX/HPX boards. If the tapped network supports RTCP, the IPX/HPX decodes these messages and reports any jitter or information about packets sent by the endpoints.

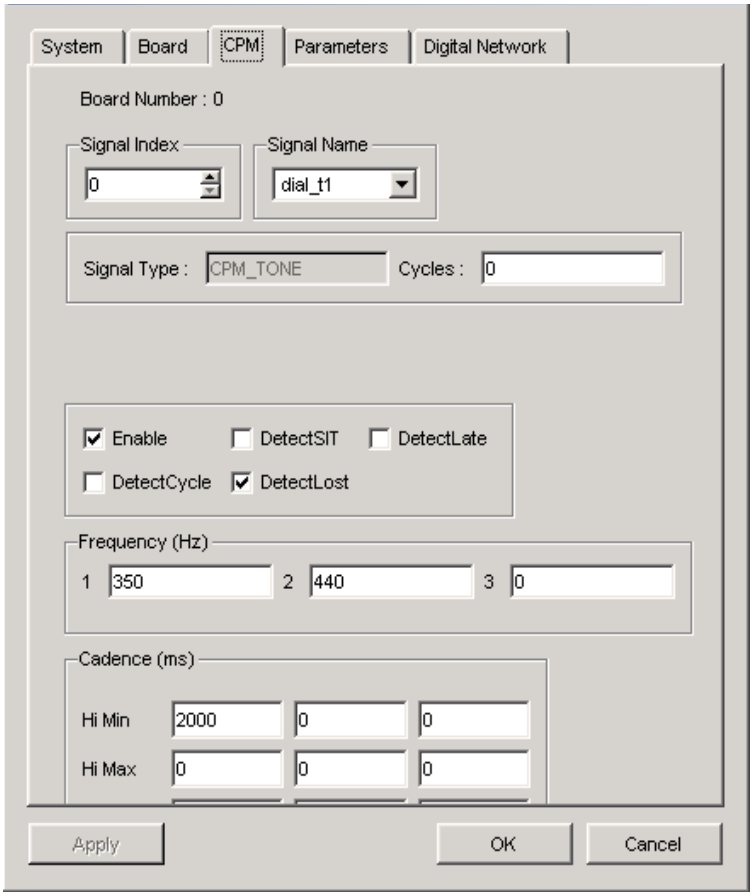
Users can enable/disable this feature here on the Control Panel.

CPM TAB

NOTE: This section of the Control Panel has been temporarily disabled. It will re-enabled with a future release.

Use the CPM TAB for viewing a SmartWORKS board's CPM parameters only. This tab should not be used to configure SmartWORKS boards. User applications should be used to initialize CPM parameters listed on this tab. The reason parameters are left open for configuration on this tab is for compatibility with AudioCodes legacy and T1/E1 products. All SmartWORKS series boards are meant to be initialized from within the user's application.

Figure 3-6: CPM Tab



SIGNAL INDEX

Displays the Index ID of each tone currently set in the tone index array.

SIGNAL NAME

Displays the names of selected signals. There are 7 CPM signals, 4 SIT signals, 2 FAX signals, and a "more" entry for unused signals.

SIGNAL TYPE

Displays the selected signal type (CPM_TONE, USER_TONE, or SIT_TONE).

CYCLES

Displays the number of cycles in the selected signal.

ENABLE

Enables the profile of the selected signal index.

DETECTSIT

Enables the detection of SIT signals. DetectSIT should appear unchecked ("false") when a CPM signal is selected.

DETECTLATE

When DetectLate appears checked, signal detection is delayed until the state machine detects the end of the last cycle of the total cycles specified in Cycles.

DETECTCYCLE

When DetectCycle appears checked, a "CPM signal detected" event is generated every time the state machine reaches the end of a cycle.

DETECTLOST

When DetectLost appears checked, a "Signal lost" event is generated when a previously detected CPM signal has ended.

FREQUENCY

These fields display 3 frequencies per signal (from low to high).

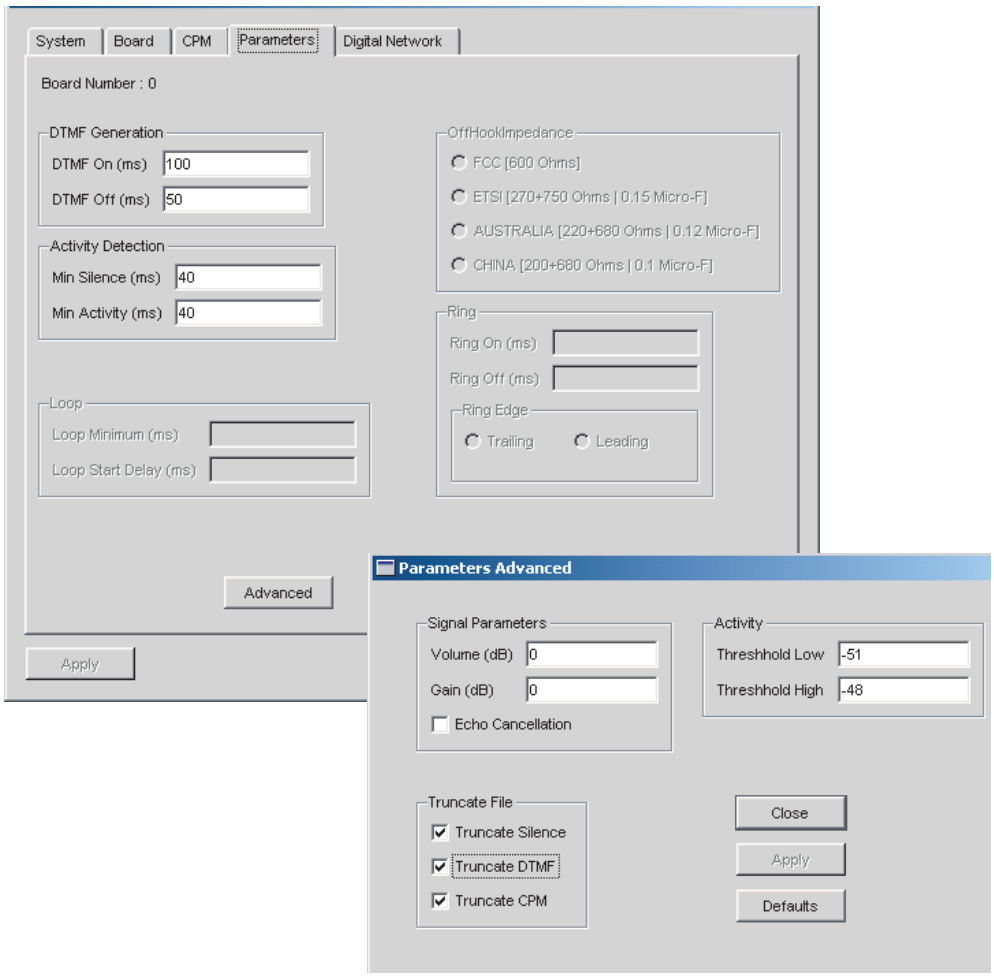
CADENCE

These fields display cadence values (in milliseconds) in up to 3 different pulses.

PARAMETERS TAB

Use the Parameters TAB for viewing a SmartWORKS board's parameters only. This tab should not be used to configure SmartWORKS boards. User applications should be used to initialize parameters listed on this tab. The reason parameters are left open for configuration on this tab is for compatibility with AudioCodes legacy and T1/E1 products. All SmartWORKS series boards are meant to be initialized from within the user's application.

Figure 3·7: Parameters Tab



DTMF Generation

Use this field to view DTMF digit generation time on the SmartWORKS board.

A value of 100 in the DTMF On (ms) field and a value of 50 in the DTMF Off (ms) field means the SmartWORKS board continuously generates DTMF pulses at an interval of 100 milliseconds ON and 50 milliseconds OFF.

Activity Detection

This field displays the SmartWORKS board's activity detection parameters.

The Min Silence (ms) field displays the amount of silence-time in milliseconds required to enter the silence detected state.

The Min Activity(ms) field displays the amount of noise-time in milliseconds required to enter the activity detected state.

Param Advanced for Board Number (Advanced Button)

This field displays more information about activity detection threshold settings including:

Signal Parameters

Used to adjust Volume and Gain settings for all channels on each board.

Echo Cancellation

Used to reduce the effect of electrical and/or acoustical coupling between the analog destination of the channel's PCM output and the analog source of the channel's PCM input.

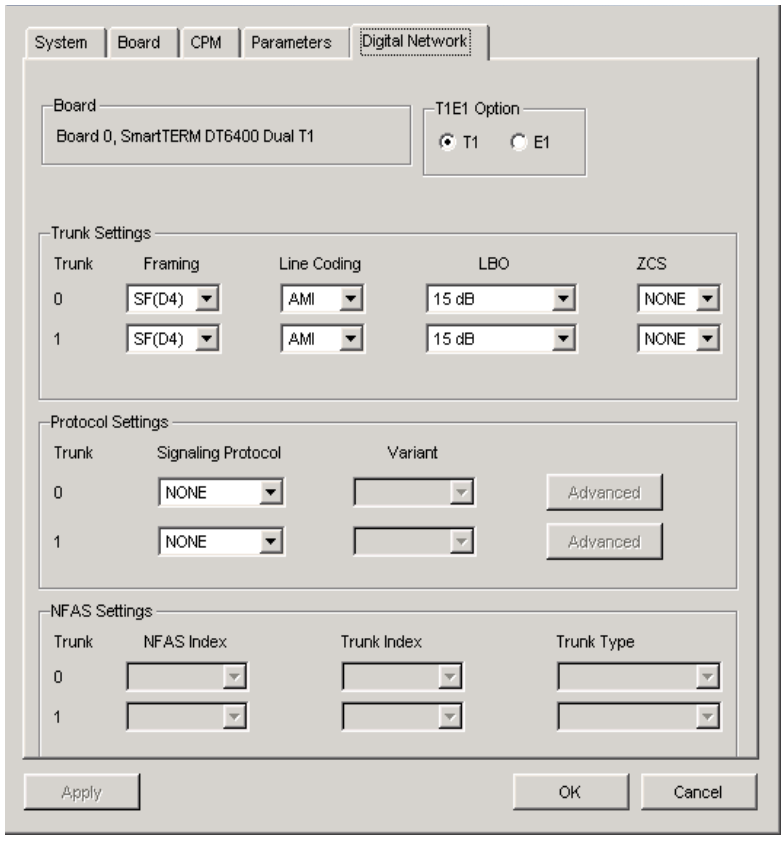
Truncate file

Used to specify how to terminate a file during recording and extract the terminating event.

For a complete explanation of SmartWORKS Activity Detection and a description of its features refer to a complete explanation in the *SmartWORKS Developer's Guide*.

DIGITAL NETWORK TAB

Figure 3-8: Digital Network Tab



Trunk Settings

Framing

When T1 mode is selected, framing is set here. Notice that framing should be selected for each framer. Standard voice trunks typically use Super Frame (SF) framing, also known as D4. Trunks with ISDN signaling use Extended SuperFrame (ESF) framing.

When E1 mode is selected, framing is set here. Notice that framing should be selected for each framer. Two framing options are available: Basic G.704 and CRC-4. The CRC-4 format adds additional protection to basic G.704 frame structure.

Line Coding

When T1 mode is selected, line coding is set here. Notice that line coding should be selected for each framer. Standard voice trunks typically use AMI (alternate mark inversion). Trunk with ISDN signaling use B8ZS (binary eight zero substitution) line coding.

When E1 mode is selected, line coding is set here. Notice that line coding should be selected for each framer. Two options are available: AMI and HDB3.

LBO

Electrical Line Build Out. This option is only available on DT 3209 or 6409.

ZCS (DT only)

Zero Code Suppression. This option is only available on SmartWORKS DT 3209, 6409. 3209TE, 6409TE.

Protocol Settings

Signaling Protocol

Options vary depending on board.
SmartWORKS DT: None, RBS, ISDN
SmartWORKS DP: None or ISDN

Variant

The variant for RBS (only available for DT).

ADVANCED DIGITAL NETWORK SETTINGS (ADVANCED BUTTON)

RBSMaxDigit

Number of digits to be processed by the RBS state machine. If this number is greater than the actual number of digits received, the RBS state machine waits the amount of time designated in the "RBS InterDigitTime" before terminating digit processing.

Setting this parameter to "0" results in an immediate call answer. During an incoming call, the RBS state machine processes the digits (as they arrive) up to the amount specified in the RBSMaxDigit parameter. These digits are used to construct the dialed number in the MT_CCINFOIND event, but do not appear in the DTMF digit cue.

RBS InterDigitTime

Time-out value at which the RBS state machine stops collecting digits.

Chapter 4

SmartView

SmartView

SmartView is an application provided by AudioCodes used to test the functionality of SmartWORKS products. Developers or System Administrators, by using the commands accessible via the SmartView interface, can test whether the board is installed and configured properly for the network, as well as view events generated.

This document has been organized into two sections:

- SmartView Setup
- Command menu - an explanation of each command available through the SmartView interface, including a details section that defines each parameter associated with the API.

SmartView Setup

Starting SmartView

From the Start menu select:
Programs > Ai-Logix > SmartWORKS >SmartVIEW.
This launches the SmartView application.

Note for Linux Users

When launching this program use lower case letters when typing into the command line prompt.

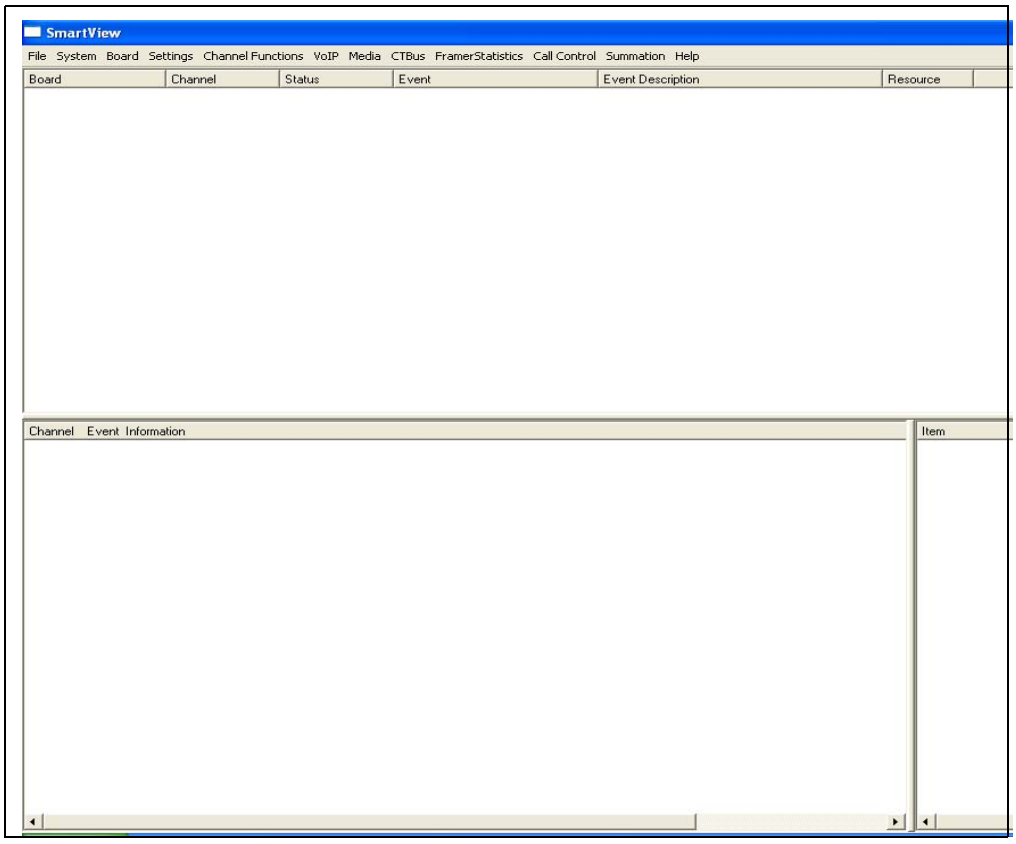
Using SmartView

The SmartVIEW application has 3 panes for viewing board information:

- Channel View (double-clicking a channel toggles the state of the channel between Open and Closed)
- Event View
- Current Channel Status View (Figure 4·1 on page 29)

These three panes are illustrated in the figure on the following page:

Figure 4-1: SmartView Main Window



VIEWING CHANNEL INFORMATION

To view current channel information in any of these panes you must first select a channel by highlighting it in the Channel View. Once the channel is selected you can then execute a command from the menu. The command may cause events to be generated in the event view which updates in real-time.

Following is a list and explanation of available commands on the Application menu.

Command Menu

The command window displayed on the top of the screen enables the user to access commands. The menu is organized logically with the following headings

File - provides the user the ability to store board configuration for later use

System - from here the user can access System and board Information, and access commands to open/close boards and channels.

Board - access board information and settings, as well as control the PCM interface on SmartWORKS PCM boards.

Settings - basic board functionality is controlled here: IO control, AVC, AGC, tone generation, CPM tone settings, echo cancellation and more.

Channel Functions - from here the user can access commands to force a channel into the following states: Off Hook, On Hook, Dialing, etc. The user can also stop a channel or stop/start pending functions.

VoIP- all VoIP releated commands are accessed via this menu option

Media - the commands here can be used to control all media functions such as file playback, recording, streaming and buffering. Use these commands to control board communications with the CTBus (either MVIP or H.100). The user can apply clock settings, stop/start the Mux, and control the direction of data.

CTBus - The commands under this heading enable the user to manage the use of the CTBus. This section allows the user to start/stop the Bus, set Master clock and NET references, as well as control the flow of data when passing data over the CTBus.

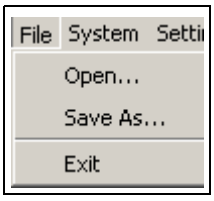
Framer Statistics - when applied these commands can be used for obtaining framer statistics (T1, E1, or digital statistics used by the NGX card), and alarms plus enabling ABCD signalling.

Call Control- these commands apply to basic ISDN and RBS call control. The user can execute basic call controls (accept, place, or release a call), obtain call statistics, and line signalling statistics.

Summation- summation services are applied using these commands. The user can start/stop summation, as well as control volume, gain and manage summation inputs

Details - displays version information about SmartView.

FILE



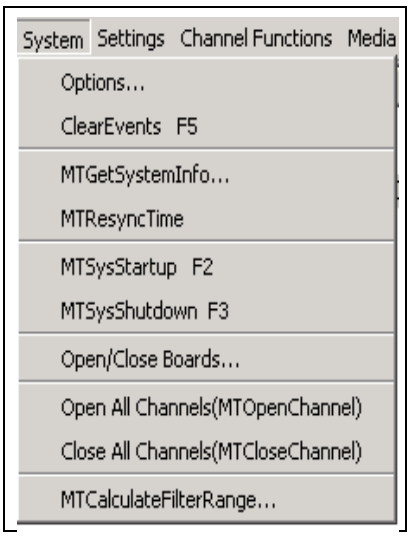
OPEN

Any saved SmartView configuration files can be accessed through this interface and loaded for use by the application.

SAVE AS

You can save your SmartView configuration settings to a file for later use. This is useful when testing multiple boards on one system, each with unique configuration settings.

SYSTEM



OPTIONS

Provides access to SmartView configuration options. Here the user controls whether time stamp and call control events are displayed, enable/disable auto-accept, and the location of the event log file. Details...

CLEAR EVENTS

When this is selected, all events currently displayed in the event view screen are cleared from view.

MTGETSYSTEMINFO()

System information is displayed such as: Driver and Dll version and build numbers, and CTBus data such as the number of masters by type(MVIP or H.100), Clock Source, stream speed (H.100 only) and Bus segmentation. Use the drop down to select a board number and display information such as: board type and status, number of channels, firmware information, bus type and clock source, summation and audio jack total.

When the board number is changed in the ***MTGetSystemInfo()*** area, a call to ***MTBlinkBoard()*** is executed. This will blink the LED of the physical board that matches the board ID.

MTRESYNCTIME()

By default, the timestamp is synced when a board/channel is opened 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.

MTSYSSTARTUP()

When this option is selected all SmartWORKS boards on the system will be opened, all SmartWORKS channels will be opened, board communications between DLL and driver will be opened, and MVIP MUX will be enabled.

MYSYSSHUTDOWN()

When this option is selected, it closes all boards. Channels are closed 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.

OPEN/CLOSE BOARDS

MTOpenBoard() opens the specified board for access. The channels will remain closed.

OPEN ALL CHANNELS (MTOPENCHANNEL())

This command opens all channels on the entire system. To open one specific channel double click on the channel.

When using the API ***MTOpenChannel()*** in your application it opens only the specified channel for access.

CLOSE ALL CHANNELS (MTCLOSECHANNEL())

This command closes all channels on the entire system.

When using the API ***MTCloseChannel()*** in your application it closes only the specified channel.

MTCALCULATEFILTERRANGE()

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.

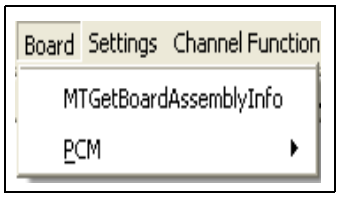
- this function accepts upto three nominal frequencies

- 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*.

BOARD



MTBOARDRESET() - NOT IMPLEMENTED YET

When invoked, the board is put into a reset state. This allows users to load new board configuration settings using ***MTBoardConfig()*** without stopping and starting the driver.

To invoke this API make certain that no other applications are currently accessing this board. The board must be re-enabled using the ***MTBoardEnable()*** API.

MTBOARDCONFIG() - NOT IMPLEMENTED YET

This API can only be used when running SmartWORKS on the Windows OS. This API gets values from the windows registry setting and loads them to the board's DLL. Before using this API, ***MTBoardReset()*** must first be invoked. When finished, use ***MTBoardEnable()***. Board parameters are modified in the registry settings using the API ***MTSetAdapterConfig()***.

MTBOARDENABLE() - NOT IMPLEMENTED YET

This API enables the board after is has been put into a reset state by the API ***MTBoardReset()***.

MTGETBOARDASSEMBLYINFO()

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

MTBOARD(GET/SET)IDLECODING()

This API is only used with SmartWORKS PCM cards. Sets the coding format for the idle packet. In the event of sychronization loss, the framer replaces the data stream normally passed to the TDM with an idle packet signal. By default, μ-Law idle packet is used.

MTBOARD(GET/SET)TERMINATIONIMPEDANCE()

This API is used with SmartWORKS PCM cards only. Sets the termination impedance for the identified trunk. By default, all trunks are set to Hi-Impedance (passive). Use the Get button to determine a trunk's current settings.

MTBOARDPCM(GET/SET)SIGNALCONFIG()

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.

Transmit: pins 1&2

Receive: pins 4&5

Frame Sync: pins 3&6

MTBoardPCMTrunkFrameSync(Status/Control)()

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.

MTBoardPCMTrunkSlotTx(Status/Control)()

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 33th channel is connected to the first timeslot on the second trunk. This enable field is a bit-map for each timslot - bit value 0 for disable, 1 for enable with bit 0 for first slot on the specified trunk.

Settings

MT(Get/Set Volume)()

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. The value displayed in the dB field shows the current setting of the selected channel.

If output data is routed using ***MTCTSetRoute()*** the DSP is bypassed, therefore Volume settings do not alter this data.

MT(Get/Set) Gain()

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. The value displayed in the dB field shows the current setting of the selected channel.

If output data is routed using ***MTCTSetRoute()*** the DSP is bypassed, therefore Gain settings do not alter this data.

MTChInputStepGain()

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:
-1Adjust gain down by 1 dB
1Adjust gain up by 1 dB
0Reset 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.

MTChInput(Get/Set)Gain()

MTChInputSetGain() sets the input gain of the specified input per channel. ***MTChInputGetGain()*** retrieves the current value. 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.

MT (Get/Set) AGC()

Automatic Gain Control (AGC) settings can be modified here.

MTSetAGC() sets the input AGC parameters.

- Parameter TMA (target maximum amplitude) ranges from 1 to -45 dBm. The default is -6 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 4 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 30 dB.

The values displayed in the fields show the current settings. By default, AGC is disabled.

If output data is routed using ***MTCTSetRoute()*** the DSP is by-passed, therefore AGC settings do not alter this data.

MTChInput(Get/Set)AGC()

MTChInputSetAGC() sets the input Automatic Gain Control parameters on a per input basis for each channel. ***MTChInputGetAGC()*** retrieves the current settings.

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.

MT (Get/Set) AVC()

MTSetAVC() sets the output Automatic Volume Control (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 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 30 dB.

The values displayed in the fields show the current settings. By default, AVC is disabled.

If output data is routed using ***MTCTSetRoute()*** the DSP is by-passed, therefore AVC settings do not alter this data.

MT(Enable/Disable) Caller ID()

CallerID detection is enabled by default with a sensitivity level set to 0x80.
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. The value displayed in the field represents the current setting.

MT(Get/Set) Alert Tone Params()

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. Use this interface to set the values for the tone that will be played. To enable alert tone, use the Encoding IO_Control interface and enable Alert Tone in the Start Control field. Details...

MT(Enable/Disable) Mixing()

This command enables mixing of the secondary audio input with the primary audio input before it is sent on for further processing. On SmartWORKS DP and SmartWORKS NGX cards, the mixing of the primary and secondary inputs is enabled by default. For most applications, this setting must remain unchanged from its default state.

The ***MTEnableMixingDetect()*** API is invoked if the Mixing Detect option is selected. This API enables the signal detecter associated with the secondary input on the second audio data stream. Detectors involved are for DTMF, and activity.

MTHighpassFilterControl()

Use this command to enable/disable high pass filtering. The highpass filter is enabled by default. Setting the input highpass filter ON directs primary and secondary PCM streams to pass through a highpass filter before further processing.

It is highly recommended that highpass filtering remain enabled, especially when using SmartWORKS DT and SmartWORKS DP boards.

MT(Get/Set)ReverseLoopPolarity()

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 loop polarity is reversed. This API enables you to reverse the loop polarity through a software change rather than a wiring change.

MT(Get/Set)LV(Params/Detect)()

These APIs are only supported on the SmartWORKS LD boards.
MTGetLVDetect() is used to measure the voltage on any particular channel. First select the channel by highlighting it in the Channel View screen, then select the *Update* button on this interface.

MT(Get/Set)LVParams() are used to provide the user the capability to obtain current thresholds settings, and modify them if necessary. 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). Details...

Use the **Generate** button to save all information to a comma deliminated (.csv) file.

MT(Get/Set)EventFilters()

The user can switch event reporting of a specific event on or off by setting a bit with ***MTSetEventFilters()***. To view events in the Event View screen, they must be enabled here. Details...

MT(Arm/Disarm)(CT, Framer,Master)Alarm()

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. H.100 bus related errors are reported as events EVT_H100_MASTER_A_ALARM and EVT_H100_MASTER_B_ALARM.

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. Details...

MTArmMasterAlarm() arms the report of master clock synchronization related errors through EVT_MASTER_PLL_ALARM. Enable the Master PLL option on this interface.

MT(GET/SET)MONI()

Monitor event enable bits control which monitoring events are reported to the user applicatioin. **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. (more on next page)

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.

This API is further explained, along with Call Progress Monitoring, in the Application Note, Managing CPM Events, available on the AudioCodes website Details...

MT(GET/SET)TERM()

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. Details...

MT(GET/SET)PSTNPARAMS()

MTSetPSTNParams() sets the PSTN parameters used to define values for terminate applications This API applies settings held in the MT_PSTN structure. Details...

MT(GET/SET)ACTDPARAMS()

MTSetACTDParams() sets the silence/activity detection configuration. This API applies settings held in the MT_ACTDParams structure. Details...

MTCHINPUT(GET/SET)ACTDPARAMS()

MTChInputSetACTDParams() sets the silence/activity detection configuration on the specified input per channel. Details...

MTChInputACTD(Status/Control)()

MTChInputACTDStatus() retrieves the ACTD status for the specified channel input. When enabled, a check is present in the ACTD On checkbox.

MTChInputACTDControl() enables or disables activity detection on the specified input per channel. Activity detection is enabled by default.

MT(Get/Set)DTMFDetectParams()

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.

min_detect_time - minimum time to detect a DTMF tone, 40 - 100 ms

max_report_time - maximum time to report leading edge detection of a DTMF tone. Value must fall within the range of min_detect_time to 100

MTChInput(Get/Set)DTMFDetectParams()

This API is only used when the tone detection trigger is set to LEADING_EDGE detection. This is set by using the ***MTChInputSetToneDetectParams()*** API and set the Trigger field in the MT_DetectRegs structure.

This API sets the parameters which control tone detection and reporting.

min_detect_time - minimum time to detect a DTMF tone, 40 - 100 ms

MTGetTimeSlotRxSignalling()

This API is only available when using the SmartWORKS DT board.

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. Details...

MT(Get/Set)Tone(Gen/Detect)Params()

MTSetToneGenParams() sets the DTMF/MF tone generation parameters defined in MT_GENREGS structure. **Note:** MF is currently not supported. Details...

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. Details...

MTChInput(Get/Set)Tone(Gen/Detect)Params()

MTSetToneGenParams() sets the DTMF/MF tone generation parameters defined in MT_GENREGS structure. **Note:** MF is currently not supported. Details...

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. Details...

MTChInputToneDetect(Status/Control)()

MTChInputToneDetectStatus() retreives the queue control status of each input per channel.

Possible values for queue control are as follows:

Q_DISABLEDTone detection on the specified queue is disabled; no event reporting and tone queuing performed.
Q_FLUSHTone detection on the specified queue is enabled without tones queued. Tones are reported through channel events.
Q_KEEPTone detection on the specified queue is enabled with tones queued. Tones are also reported through channel events.

MTChInputToneDetectControl() specifies whether this is to control DTMF queue or MF queue per each input of a channel.

MT(GET/SET/RESET)CPMTONEPARAMS()

MTSetCpmToneParams() sets CPM tone parameters (profile). There are maximum of 15 CPM tone entries and the index ranges from 0 to 14. Call Progress Monitoring is explained in the Application Notes: *Call Progress Monitoring*, or *Understanding Signal Profiles*, available on the AudioCodes website.. Details...

MT(GET/SET/CLEAR)USRTONEPARAMS()

MTSetUsrToneParams() adds a user defined tone into the tone profiler table for later detection. Details...

MTCH(GET/SET/RESET)CPMSIGNALPARAMS()

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.

MTChResetCpmSignalParams() either disables all CPM signal profiles or sets all CPM signal profiles to default settings.

MTCH(GET/SET/CLEAR)USRSIGNALPARAMS()

MTChSetUsrSignalParams() adds a user defined tone into the tone profiler table for later detection. More information is available in the application notes: *Call Progress Monitoring* and *Understanding Signal Profiles*. Both are available on the AudioCodes website.

MTChClearUsrSignalParams() clears are user defined signals from the channel's configuration.

MTCHCPMSIGNAL(STATUS/CONTROL)

MTChCPMSignalControl() allows users to disable/enable any individual signal profile. Once disabled, the signal is not reported to the user application when present on the line.

MTChCPMSignalStatus() retrieves the current status of an individual signal.

MTCHVOICEDETECT(STATUS/CONTROL)

MTChVoiceDetectControl() is used to enable/disable voice detection on a channel. When enabled MONI_VOICE must also be selected to generate the monitor events associated with voice/machine detection.

MTChVoiceDetectStatus() retrieves the current status.

MTChCPM(Status/Control)

MTChCPMControl() enables and disables Call Progress Monitoring on an individual channel.

MTChCPMStatus() retrieves the current status.

MT(Get/Set)OutputSource()

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)

NOTE: In this scenario, the channel output is not returned through the Network Interface (NI). This setup also by-passes the tone generator when using boards that support this feature. This is the recommended output method for the SmartWORKS NGX and DP boards. Refer to the section that explains the API ***MTSetOutputSource()*** in the SmartWORKS Developer's Guide for a logical diagram.

MT(Get/Set)ECControl()

MTSetECControl() allows the enabling or disabling of echo cancellation for the specified channel. This API is only supported on terminate boards with 09 DSPs (AT409, AT809, AT1609, DT3209, DT6409, LD409, LD809).

MT(Get/Set)ECFreezeAdaptation()

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 canellation is reset (using **MTECControl()**, then the adaptive capabilities of echo cancellation is automatically restarted.

MT(Get/Set)ECParams()

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. Details...

MT(Get/Set)LimiterControl()

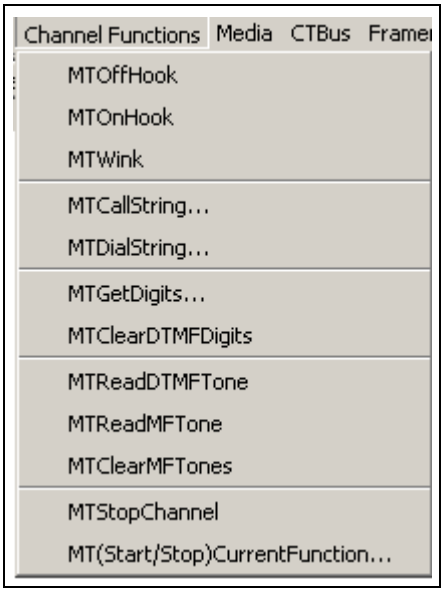
MT(Get/Set)VoiceDetectParams()

This API controls the detection of human voice or answering machine when the application is running the Call Progress Monitoring (CPM) feature. Call Progress Monitoring is explained in the Application Note, *Understanding Call Progress Monitoring* available on the AudioCodes website. Details...

MT(GET/SET)SLIC

This API is only used on LD101 boards. This feature is only used on station channels configured for FXS. Station channel carries a control called Subscriber Line Interface Circuit (SLIC). SmartWORKS currently supports two modes of operation: power-down and active-normal. To enable the SLIC, set the bit to true = 1.

CHANNEL FUNCTIONS



MTOFFHOOK()

MTOffHook() takes a specified channel off hook (provided it 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 ***MTSetEventFilter()***]. If loop current starts flowing by going off-hook, EVT_OFFHOOK will be followed by EVT_LOOP_ON, if enabled. This API is only available onANALOG_VFXO_LS _E channels on the SmartWORKS LD.

MTONHOOK()

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 ***MTSetEventFilter()***).

This API is only available on ANALOG_VFXO_LS_E type channels on the SmartWORKS LD card.

MTWINK()

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.

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.

MTCALLSTRING()

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. ***MTCallString()*** is only supported on ANALOG_VFXO_LS_E type channels on the SmartWORKS LD card.

MTDialString()

MTDialString() is a background function that dials the string specified in the dial string field. This function does not perform Call Progress Monitoring. Use the ms field to set the maximum action timeout period allowed for this call.

Control characters are as follows:

& - Generates a hook flash

, (comma) - Inserts a pause

T - Switches back to DTMF dialing mode

W - Signals that the SmartWORKS board must wait for a dial tone before dialing the remainder of the string.

H - Take the line off hook

NOTE: Option '&', and 'H' are supported only on ANALOG_FXO_LS and *ANALOG_VFXO_LS_E* channels. The MT_IO_CONTROL.MaxTime field is used for caller to specify the maximum time to wait for option 'W' or 'L'.

MTGetDigits()

Set the parameters in this API to enable digit termination. Details...

MTClearDTMFDigits()

MTClearDTMFDigits() flushes the digit queue of all DTMF digits received in the specified channel.

MTReadDTMFTone()

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

MTReadMFTone()

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

MTClearMFTones()

MTClearMFTones() flushes the MF tone queue of all tones received in the specified channel.

MTStopChannel()

MTStopChannel() stops a single channel without affecting any other channel in the system. This function returns a return code of MT_RET_OK. If there is a current function, event EVT_STOP will be issued when that function is aborted. 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.

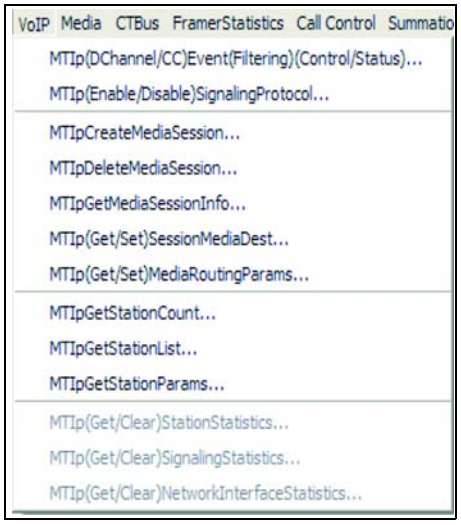
MT(Start/Stop)CurrentFunction()

All record and playback tasks are controlled through the MT_IO_CONTROL.StartControl field. ***MTStartCurrentFunction()*** suppresses the start control of the current pending task and forces the task to start immediately. Use this API to override any StartControl options set in the IO_CONTROL interface.
Note: Currently, the start control is applicable only when recording.

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.

VoIP

The SmartVIEW Utility has been modified to incorporate the IPX. A new menu on the Task Bar "VoIP" has been added:



MTIP(DCHANNEL/CC)EVENT(FILTERING)(CONTROL/STATUS)()

D-Channel event reporting must be enabled per each protocol stack running per board. **NOTE:** Users must first enable the protocol stack (***MTIpEnableSignalingProtocol()***) before D-channel event reporting is possible.

Protocol ID - select the protocol stack that you are controlling. Once selected, more fields appear that are protocol specific. The IPX Integration Guide provides examples of the information that must be passed in with this function call. Refer to the appropriate chapter of the IPX Integration Guide for more informaion.

Status - once a control button is used (**Enable** or **Disable** button) the current status is displayed in this field.

The following are a few examples of the information that must be passed to the board when this function is invoked. All protocol information is available in the *IPX Integration Guide* in the appropriate chapter per each protocol.

CISCO

Port: The port number used by the Call Manager to listen for signaling data from the telephones

Type: Whether signaling data is transmitted in UDP or TCP.

AVAYA

CSPort: The port number used by Avaya's Call Agent to listen for signaling data from the telephones

Type: Whether signaling data is transmitted in UDP or TCP.

RASPort: The port number used by Avaya's Call Agent to listen for phone initialization requests from the telephones.

Type: Whether signaling data is transmitted in UDP or TCP.

MTIPCREATEMEDIASESSION()

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. **NOTE:** General routing rules to not apply to this session. The user must also invoke ***MTIpSetSessionMediaDest()***. The following parameters are configured when creating a media session:

PrimaryIPAddress - the IP address of the station endpoint on the tapped network.

PrimaryUDPPort - the UDP port used for media messages (RTP) by the VoIP endpoint on the tapped network

SecondaryIPAddress - the IP address of the other endpoint associated with this call.

SecondaryUDPPort - the UDP port used for media messages (RTP) by the VoIP endpoint which received this call

Codec - the codec of the RTP media used during this call

NOTE: Once the **Create** button is used, then the Session ID is returned to the user in the Session ID field.

MTIPDELETEMEDIASESSION()

The user is required to delete any media sessions created with the ***MTIpCreateMediaSession()*** API. Once the session is deleted the Session ID is returned to the IPX board for re-use.

MTIPGETMEDIASESSIONINFO()

When this API is used (Session ID must be provided), information is returned to the user about a current media session. The user is provided with Primary and Secondary IP addresses and UDP ports of the VoIP endpoints.

MTIP(GET/SET)SESSIONMEDIADEST()

This command is used to set destination parameters for all media packets associated with a specified Session ID. The following parameters must be set when using the IPX (along with Board and Session ID):

Target - NETWORK. This specifies that all media packets are forwarded to a network device. **NOTE:** Targets, LOCAL_CHANNEL and HOST, are reserved for the IPR product.

DestPrimaryIPAddress - The IP address of the recording device that receives media packets that the **primary** station is transmitting..

DestPrimaryStationBaseID - the port on the recording device where packets associated with the primary station are sent

DestSecondaryIPAddress - The IP address of the recording device that receives media packets that the **secondary** station is transmitting

DestSecondaryStationBaseID - the port on the recording device where packets associated with the secondary station are sent.

OptFilter - this field is not required.

MTIP(GET/SET)MEDIAROUTINGPARAMS()

This command sets routing rules for all media packets entering the IPX board. NOTE: These rules are not applied to media packets when the user creates a media session with ***MTIpCreateMediaSession()***.

NOTE: The use of this API is discussed in detail in the Developers Reference chapter located in the IPX Integration Guide.

Name	Description
RuleNo	Rule index number up to MT_IP_MAX_MEDIA_ROUTING_RULES (the rule with the lowest number is given the highest priority)
Target	Flag indicating the destination target type. only NETWORK is supported at this time
RuleType	ROUTE_BY_NONE = (rule is canceled) ROUTE _BY_SESSION ROUTE _BY_STATION ROUTE _BY_IP

Name	Description
Min Station/Session/IP Address Max Station/Session/IP Address	The range indicating what types of packets will be forwarded. For example, if forwarding by session one could set a range of session IDs from 0-32.
Primary Station IP Address	The IP address of the recording device that receives media packets that the **primary** station is transmitting.
Secondary Station IP Address	The IP address of the recording device that receives media packets that the **secondary** station is transmitting.
Primary Station BaseID	The base port number used to calculate an available port on this device. (The port on the recording device where packets associated with the primary station are sent)
Secondary Station BaseID	The base port number used to calculate an available port on this device. (The port on the recording device where packets associated with the secondary station are sent)
Primary Inc Offset	Value required for calculating port number.
Secondary Inc Offset	Value required for calculating port number.

MTIPGETSTATIONCOUNT()

Returns the total number of station devices presently detected by the board.

NOTE: The "phone removed" functionality is not supported in the IPX Beta release. When a VoIP endpoint is removed from the tapped network, it is still counted when this API is invoked.

MTIPGETSTATIONLIST()

Returns information about detected VoIP endpoints. When using SmartVIEW this command exhibits different behavior than when using the API in a user application. To obtain information about station endpoints, supply the board and then use the **Get** button. The *Count* field shows the total number of stations. All Station IDs and Protocol IDs are displayed.

NOTE: The "phone removed" functionality is not supported in the Beta release. When a VoIP endpoint is removed from the tapped network, information is still presented when this API is invoked.

MTIPGETSTATIONPARAMS()

Use this interface to obtain information for a specified VoIP endpoint on the tapped network. Select the board number and type the Station ID, then use the **Get** button. The protocol associated with this station and the IP Address are displayed. Protocol specific information is displayed.

MTIP(GET/CLEAR)STATIONSTATISTICS()

NOTE: This API is not supported.

MTIpGetStationStatistics() retrieves data collected for a specified station. Once a station endpoint is discovered by the IPX, data is collected until the endpoint is no longer transmitting (station removed). The user can also clear this data by using the ***MTIpClearStationStatistics()*** command. The following information is displayed to the user:

Name	Description
StationID	Bit field containing ProtocolID | Station ID. Station ID is the unique number assigned to the station by the IPX station manager
Reserved	
XchngedPackets	The total number of packets passed on the network to the specified station.
NonHandledPackets	The number of packets passed to this station that are not handled by the IPX.
RcvdTcpPackets	The total number of TCP packets received by this station
NonHandledTcpPackets	The number of TCP packets passed to this phone, that are not handled by the IPX as they are not relevant to a tapping solution (data packets, etc).
RcvdUdpPackets	The total number of UDP packets received by this station

MTIP(GET/CLEAR)SIGNALINGSTATISTICS()

NOTE: This API is not supported.

Once a protocol stack is enabled, signaling statistics are collected until the stack is no longer running (disabled) or until the user clears the count by using ***MTIpClearSignalingStatistics()***. The following data is collected:

ProtocolID	Protocol ID for the signaling protocol
Reserved	
XchngedPackets	The total number of packets received by the IPX.
NonHandledPackets	The number of packets dropped by the IPX which do not apply to a VoIP solution (data packets, or web packets, etc)
CmPackets	The total number of packets originating from the Call Manager
StationPackets	The total number of packets originating from stations supporting this protocol.

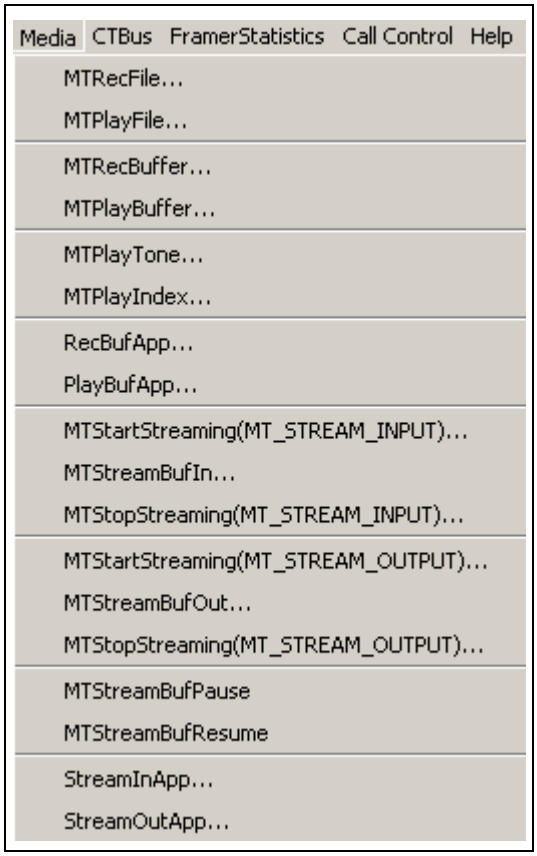
MTIP(GET/CLEAR)NETWORKINTERFACESTATISTICS()

NOTE: This API is not supported.

Retrieves the current statistical data maintained by the board, for the specified port. The following data is collected:

NetID	The ID of the specified network device. 1-2 are receiving ports while device 0 is used for actively transmitting RTP packets to the recording device.
Reserved	
XchngedPackets	The total number of packets received by this device.
NonHandledPackets	The number of packets received by this device, which are not handled by the IPX (they are not relevant to a tapping solution).
RcvdTcpPackets	The total number of received TCP packets by this device
NonHandledTcpPackets	The total number of TCP packets which are not handled by the IPX
RcvdUdpPackets	The total number of UDP packets received by this device

MEDIA



ENCODING IO CONTROL

Basic encoding IO control is set through this interface. The user has the ability to control the Call Progress Monitoring (CPM) monitoring or termination conditions, General Termination conditions, plus Digit Termination. File format of the encoded data is set by using the GSM and the MS Wave Option fields. These parameters are used by the MT_IO_Control structure. Details...

DECODING IO CONTROL

Basic Decoding IO control is set through this interface. The user has the ability to control the Call Progress Monitoring (CPM) monitoring or termination conditions, General Termination conditions, plus Digit Termination. File format of the encoded data is set by using the GSM and the MS Wave Option fields. These parameters are used by the MT_IO_Control structure. Details...

MTRecFile()

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. Details...

MTPlayFile()

MTPlayFile() is a background function that plays data from a file. Details...

MTRecBuffer()

MTRecBuffer() is a background function that starts recording to the application buffer. Details...

MTPlayBuffer()

MTPlayBuffer() is a background function that plays back data from an application buffer. 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...

Details...

MTPlayTone()

Use this API function to play dial tone, SIT tone, ring-back tone, etc. Details...

MTPlayIndex()

Use this API to play back a file. Details...

RecBufApp()

This interface provides access to many settings that can be used to record files while buffering. Set the buffer size and select a format to record the file in. Use the *Advanced* button to modify any settings in the MT_IO_Control structure.

PlayBufApp()

This interface provides access to many settings that can be used to playback data from a buffer. Use the *Advanced* button to modify any settings in the MT_IO_Control structure.

MTStartStreaming()(MT_STREAM_INPUT)

Use this API to begin streaming the data on the specified channel. Use the *Advanced* button to modify any settings in the MT_IO_Control structure.

MTStreamBufIn()

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.

Set the buffer size in bytes.

Set the timeout period:

- 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.
- any positive integer - 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.
- -1 - this function returns when the specified data length is satisfied. Setting timeout period to -1 is equivalent to setting no timeout period.

MTStopStreaming()(MT_STREAM_INPUT)

Stops the streaming of data.
The parameter Stop Mode specifies how it should be stopped.

- Stop Now- aborts streaming immediately.
- Stop Delay - stops streaming after the currently queued streaming buffer function completes.

MTStartStreaming()(MT_STREAM_OUTPUT)

Use this API to begin streaming the data on the specified channel. Use the *Advanced* button to modify any settings in the MT_IO_Control structure.

MTStreamBufOut()

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.

Set the buffer size in bytes.

Set the timeout period:

- 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.
- any positive integer - 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.

- -1 - this function returns when the specified data length is satisfied. Setting timeout period to -1 is equivalent to setting no timeout period.

MTStopStreaming()(MT_STREAM_OUTPUT)

Stops the streaming of data.
The parameter Stop Mode specifies how it should be stopped.

- Stop Now - aborts streaming immediately.
- Stop Delay - stops streaming after the currently queued streaming buffer function completes.

MTStreamBufPause()

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.

MTStreamBufResume()

MTStreamBufResume() resumes any paused output streaming on a specified channel.

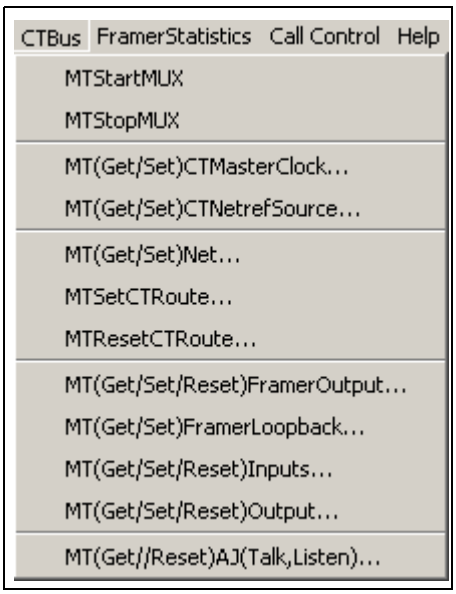
StreamInApp

This interface provides access to many settings that can be used to stream in data. Details...

StreamOutApp

This interface provides access to many settings that can be used to stream out data. Details...

CTBus



The commands under this heading enable the user to manage the use of the CTBus. This section allows the user to start/stop the Bus, set Master clock and NET references, as well as control the flow of data when passing data over the CTBus.

MTStartMUX()

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. Currently, ***MTSysStartup()*** will start the MUX as part of system initialization.

MTStopMUX()

MTStopMUX() clears all existing MUX connections on all channels grabbed by the user application and shuts the MUX down.

MT(Get/Set)CTMasterClock()

MTSetCTMasterClock() changes the master mode and clock source settings for MVIP/H.100 bus to any of the following:

- H.100 - MASTER_A, MASTER_B, or SLAVE for H100
- MVIP - MASTER or SLAVE

Details...

MT(Get/Set)CTNetrefSource()

Sets the Netref clock source on an H.100 bus or the SEC8K clock source on an MVIP bus. A board must be in slave mode to use this API on an MVIP bus. Details...

MT(Get/Set)Net()

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). Details...

MTSetCTRoute()

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_SUPER is used for monitoring.

NOTE: In this 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 DT boards. Refer to the section that explains ***MTSetCTRoute()*** in the Developer's Guide for a diagram of the SmartWORKS logical model when using this API. Details...

MTResetCTRoute()

MTResetCTRoute() resets the MVIP/H.100 link. See explanation above for more information.

MT(Get/Set/Reset)FramerOutput()

The API gives the User the ability to route the data from the output of the framer to the CT Bus. This is the same input data that is routed to the DSP by default. Details...

MT(GET/SET/RESET)FRAMERLOOPBACKMODE()

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.* Details...

MT(GET/SET/RESET)INPUTS()

MTSetInputs() sets the CT bus connection of the primary and secondary input of the specified channel. **Note:** Only one channel can transmit (talk) on a time slot, however, multiple channels can receive (listen) on a time slot.

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.

H.100 - the number of timeslots available per streams (0 - 31) depends on which stream speed is selected in the SmartControl panel.

- 2048 KHz - 32 timeslots per stream are available.
- 4096 KHz - 64 timeslots per stream are available.
- 8192 KHz - 128 timeslots per stream are available.

Details...

MT(Get/Set/Reset)Output()

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()***).

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.

H.100 - the number of timeslots available per streams (0 - 31) depends on which stream speed is selected in the SmartControl panel.

- 2048 KHz - 32 timeslots per stream are available.
- 4096 KHz - 64 timeslots per stream are available.
- 8192 KHz - 128 timeslots per stream are available.

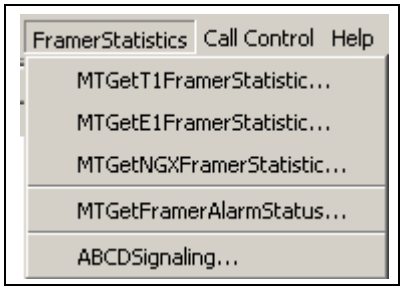
Details...

MT(GET/RESET)AJ(TALK,LISTEN)()

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. Details...

MTAJListen() sets the audio jack to receive its input from the specified CT bus time slot. Details...

FRAMER STATISTICS



The commands in this section provides access to framer statistics when monitoring T1/E1 or digital networks with the NGX card. These statistics are useful when troubleshooting network connectivity issues. ABCD signalling parameters can also be set here.

MTGETT1FRAMERSTATISTICS()

Gets error statistics for the specified framer. A total count of bit errors, framing bit errors, OOF and line code violations is provided for performance monitoring. Use the *Clear* button to clear all error counts then use the *Get* button to display new errors. Details...

MTGETE1FRAMERSTATISTICS()

Gets error statistics for the specified framer.
In E1 mode, the total count of line code errors, frame alignment signal (FAS) bit errors, far end block error and line code violations is provided for performance monitoring. Use the *Clear* button to clear all error counts then use the *Get* button to display new errors. Details...

MTGETNGXFRAMERSTATISTICS()

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. Use the *Clear* button to clear all error counts then use the *Get* button to display new errors. Details...

Use the **Generate** button to save all information to a comma deliminated (.csv) file. Framer statistics are saved for all channels on the NGX board.

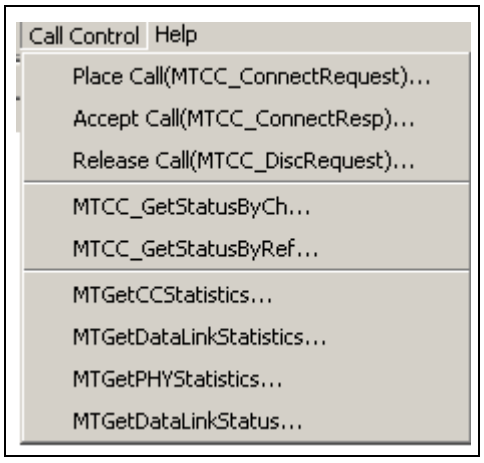
MTGETFRAMERALARMSTATUS()

Retrieves whether the specified framer is armed, and displays the total count and type of alarms present on the line. Use the *Refresh* button to update the display. 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 been cleared. Details...

ABCDSignalling

This function enables the reporting and queuing of ABCD signaling change. 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 must be enabled manually on this interface.

CALL CONTROL



PLACE CALL (MTCC_CONNECTREQUEST())

Initiates an outgoing call. Type in phone number to call, and include a unique ConID which will be used as a call reference number. Select the **Advanced** button for more options. When using the Advanced features the **Default** button returns all the options to default values used by the SmartVIEW application.

ACCEPTCALL(MTCC_CONNECTRESP())

When in incoming call is present on the channel, use this to accept the call. Select the channel this call is coming in on, add a unique ConID number (call reference number) and click the *Accept* button. If Auto Answer is enabled (System>Options) the call will be automatically answered.

RELEASECALL(MTCC_DISCREQUEST())

Release an already connected call, to refuse an incoming call or to abort an outgoing call.

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. Details...

MTCC_GETSTATUSBYREF()

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. Details...

MTGETCCSTATISTICS()

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.

MTGETDATALINKSTATISTICS()

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. Details...

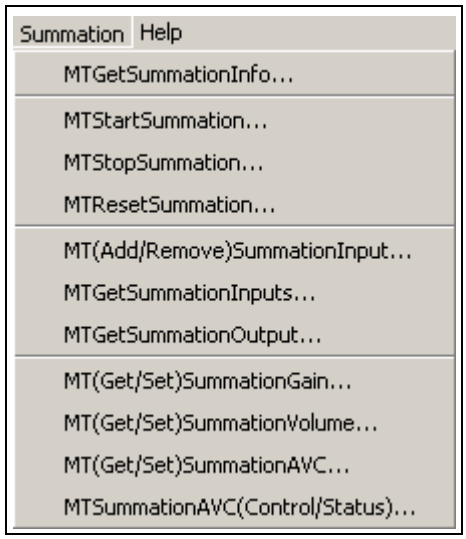
MTGetPHYStatistics()

MTGetPHYStatistics() returns the collected physical layer statistics for the trunk specified by the parameters nBoard and nTrunk. These statistics concerns the CRC errors counter, the Receiver Overflow counter, etc…
The counts returned correspond to the total number since the channel was last opened. Details...

MTGetDataLinkStatus()

MTGetDataLinkStatus() returns the data link state for the trunk specified by the board number and trunk number parameters. The Data Link state is used for passive call control monitoring. Details...

Summation



MTGetSummationInfo()

This API is used to return the total number of inputs the specified summation resource supports. Details...

MTStartSummation()

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.

MTStopSummation()

MTStopSummmation() disconnects the previously connected summation output. No other settings are changed.

MTResetSummation()

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).

MTAdd/RemoveSummationInput()

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. Details...

MTRemoveSummationInput() disconnects the previously connected summation input. When using this command, only the iSummation field is used. The Stream, Gain and Slot fields can be ignored.

MTGetSummationInputs()

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.

MTGetSummationOutput()

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.

MTGet/SetSummationGain()

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).

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.

MTGet/SetSummationVolume()

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. Volume can be set within the range of -50dB to +24dB.

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.

MTGET/SETSUMMATIONAVC()

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.

MTGetSummationAVC() retrieves the AVC settings for the output on the specified summation function.

MTSUMMATIONAVC(CONTROL/STATUS)()

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. Use the **SetControl** button to modify the AVC setting.

MTSummationAVCStatus() reads the AVC status for the output on the specified summation function. Use the drop down to select the summation index. If AVC is currently enabled, a check in displayed in the ON field.

Details

This section provides a detailed interpretation of parameters displayed on the SmartView interface. Not every API viewed on the SmartView interface is described here. For an explanation of each API refer to the section that explains each of the Command Menu options.

SYSTEM

OPTIONS

The following parameters can be set to configure the SmartVIEW application for use:

Decode Call State Events	Once enabled, ISDN call control events are displayed. This is used for SmartWORKS DP, DT and NGX cards.
Event Time Stamp	Provides the ability to display event timestamps.
Auto Accept Calls	If Auto Accept is enable, when board gets connection ID event it will automatically issue connection response and set up the call. This occurs without the need to assign call control information such as call reference ID (ConnID).
Auto Call Again	This is used with the SmartWORKS DT card. Once a call is generated, the application will release this call and automatically initiate a new connection.
Log Events to File	All events displayed in the Event View screen can be saved into a log file. This sets the location of the file. By default, the file is saved in root where SmartVIEW has been installed.

Command Menu | < Back

SETTINGS

ENCODING/DECODING IO_CONTROL

CPM Mode	Must be set for both Termination and Monitoring control: _USE_NONE _USE_TASK_ONLY _USE_CHANNEL_ONLY _USE_TASK_AND_CHANNEL
Termination Conditions	This background function terminates when any of these conditions are met
Monitoring Conditions	Once detected, these events are reported to the user application.
General Termination	More than one of these parameters may be set a a time. The threshold that is reached first determines the end of recording/playback.
MAX Silence	Maximum silence time before recording is terminated or playback is stopped. (this value is ignored by the DLL with play functions)
MAX Activity	Action timeout period in milliseconds before recording is terminated or playback is stopped. (this value is ignored by the DLL with play functions)
MAX Bytes	Recording is terminated when this number of bytes has been received
MAX Time	Recording is terminated when this time limit is reached
Digit Termination	
Max Digits	Number of digits for termination
Inter-Digit Time	Inter-digit timeout when MaxDigits is not zero
End Digit	Termination digit (@terminates on any digit)
Digit Enable	
	When any one is selected, the board is set to automatically terminate on detection of selected digit.
Termination Control	
	When the Flush play option is enabled the board will flush queued playback(s) on termination other than EVT_EOF, EVT_MAXBYTE, EVT_MAXTIME
Start Control	
Prompt Tone	A prompt tone will be played before recording. ToneDuration field sets the length of this tone. Used on the AT, LD, and DT boards (this value is ignored by the DLL with play functions)
Inter Digit Delay	1 - Inter-digit delay starts after the first digit 0 - Inter-digit delay starts immediately Used by all boards
Alert Tone On	Plays Alert tone during Call logging. Used on the PT and LD boards (this value is ignored by the DLL with play functions)

Activity Start	Activity triggered Recording. Automatically starts recording upon activity detection, event EVT_ACTD_RECORD will be issued. Used by all boards.
Loop on Start	Loop Recording: Automatically starts recording upon loop current detection and issues EVT_LOOPON_RECORD. Automatically stops recording when loop current is dropped and issues EVT_LOOP_STOP. Used by the LD and AT boards
Loop Voltage OffHook	Start recording on detection of loop voltage MIDDLE state. This is available on SmartWORKS LD only. --- Recording starts on detection of loop voltage and event EVT_LVOLTAGE_RECORD will be issued. . used by the LD boards
Silence Truncation	Not to record the terminating silence, used by all boards. (this value is ignored by the DLL with play functions)
Frame Header	When enabled, a 2-byte frame header is added to each frame. This will be deployed in the future for use with variable sized frames. Used by all boards. (this value is ignored by the DLL with play functions)
Pre-Activity Duration	ms of data to record before triggering on ACTD. This field must be used only if Activity Start is enabled. (this value is ignored by the DLL with play functions)
Tone Duration	Sets the duration of the prompt tone in milliseconds. (this value is ignored by the DLL with play functions)
GSM Option	
	Use NVDSP compatible 34-byte GSM frame length. This is required for backwards compatibility with the Passport boards. This field is obsolete.
MS Wave Option	
	Select the type of header file that will be added to recording files. This option is only available with the only media formats: GSM, all linear, and unsigned 8-bit.

Command Menu

MTSetAlertTone()

Amplitude	Amplitude in dBm, 3.0 dBm to -60.0 dBm
Frequency	Frequency in Hz
Cycle Duration	Alert tone interval, Repeat duration in unit of 20ms; eg. 750->15sec
Tone Duration	Duration in 20 ms unit for this tone
First Cycle Duration	Duration in 20 ms unit for the first cycle only. Use the *Update* button to obtain current reading.

Command Menu | **< Back**

MT(Get/Set)LV(Params/Detect)()

Threshold low	-60V to 60V, default = -16V
Threshold high	60V to –60V, default = +16V
Deglitch Time (ms)	De-bouncing time in unit of 10ms, valid range is 10 ms to 2550 ms, default 50ms
Current Voltage	Current voltage reading in increments of 2V. Use the *Update* button to obtain current reading.

Command Menu | **< Back**

MTSetEventFilters()

Enable the events will be generated by SmartVIEW when a network event is detected on the line.

SE_LCURRENT_CHANGE	enables the board to generate an event each time a change in loop current is detected, used by LD and PT boards only.
SE_RING	enables the board to generate an event each time an incoming call is detected, used by LD and PT boards only.
SE_ACT	enables the board to generate an event each time activity is detected, used by LD and PT boards only.
SE_SIL	enables the board to generate an event each time silence is detected, used by LD and PT boards only.
SE_OFFHOOK	enables the board to generate an event each time a phone goes off hook, used by all boards except PTs and DTs.
SE_ONHOOK	enables the board to generate an event each time a phone goes on hook, used by all boards except PTs and DTs.
SE_LVOLTAGE_CHANGE	enables the board to generate an event each time a change in loop voltage is detected, used by LD and PT boards only.
SE_WKRECV	enables the board to generate an event each time an incoming wink is detected, used by LD and PT boards only.
SE_LREV	enables the board to generate an event each time a loop reversal is detected, used by LD and PT boards only.

Command Menu | < Back

MTArmFramerAlarm()

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

The possible framer alarm enables are

T1 Framer Alarm Enables:

Loss of Signal (LOS)	for Loss of Signal alarm
Loss of Frame (LOF)	for Loss of Frame alarm
AIS	for Alarm Indication Signal alarm
YELLOW	for YELLOW alarm (Remote Alarm Indication - RAI)

E1 Framer Alarm Enables:

Loss of Signal (LOS)	Loss of Signal alarm
YELLOW	for YELLOW alarm (Remote Alarm Indication - RAI)
LOS_MF	Loss of Signaling Multiframe Alignment alarm. This alarm is not used when configured for Basic E1.

CRC-4	Loss of CRC-4 multi-frame alignment. This alarm is not used when configured for Basic E1.
TS16RAI	for Time Slot 16 Remote Alarm Indication alarm(enable AIS alarm). This alarm is not used when configured for ISDN and DASS2.

NGX Framer Alarm Enables:

Loss of Signal (LOS)	for Loss of Signaling Multiframe Alignment alarm

Command Menu | < Back

MT(GET/SET)MONI()

First use the IO_CONTROL interface to verify that CPM Mode is set to "1" CPM_MONI Then enable the monitoring of specific tones by using this interface. Each tone profile is defined in the CPM profile table. The following table lists each option, plus provides the event that is generated when the monitored event occurs.

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

Command Menu | < Back

MT(GET/SET)TERM()

NOTE: Termination conditions can also be set for a specific task using the *TermEnable* field of the MT_IO_CONTROL structure.

Refer to the table on the next page for a list of all termination conditions.

The following bits can be put together with "OR" to enable termination. The following table lists each termination conditions, plus the correcsponding Subreason and XtraInfo field generated when the EVT_CPM_STOP event is reported:

MTSetTerm() Control	**Subreason**	**XtraInfo**
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)..

Command Menu | < Back

MTSetPSTNParams()

Flash Char	The character used to designate hook-flash. The default setting is "&". This field accepts any ascii character.
Flash Time	The length of hook-flash in ms. The default is 500ms.
Wink Delay	The wink delay in ms. The default is 150ms.
Wink Time	The wink time in ms. The default is 150ms.
Pause Char	The character used to designate a pause. The default setting is " ,". This field accepts any ascii character.
Pause Time	The length of the pause in ms. The default is 2000ms.
Pulse On	The pause dialing on-hook length in ms. The default is 40ms.
Pulse Off	The pause dialing off-hook length in ms. The default is 60ms.
Pulse Delay	The pause dialing inter-digit delay in ms. The default is 1000ms.
Loop Deglich	The minimum loop glitch time ranged from 10 to 2550 ms. The default is 1000ms.
Loop Delay	With MTCallString(), the channel dials a number and goes into CPMMODE_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 in ms. The default is 10ms.
Ring On	The ring on duration in ms. The default is 300ms.
Ring Off	The ring off duration in ms. The default is 500ms.
Ring Reset	The duration (ms) to reset ring count to 0
Ring Count	The minimum ring cycle required to report this event. The default is one (1) cycle.
Ring Trigger	This sets whether ring detection is on the leading or trailing signal cycle. The default is 0 - leading
Ring Auto	This sets whether Auto-answer is on or off. (0=disable, 1=enable). By default, this is set to 0.
CPM Start Delay	The delay time to report a CPM event after dialing in ms. The default is 1000ms.
Off-Hook Delay	The delay time to report the EVT_OFFHOOK event when a channel goes to the off-hook state in ms. The default is 500 ms.
Min rings no answer	The number of ringbacks or double ringbacks that are detected before a no answer is reported. By default this is set to 4.
Max noise no answer	the maximum continuous signal time before no-ringback is reported. This also sets the activity timeout for CPM_NO_RINGBACK, default is 6500
Max silence no answer	Maximum continuous silence time, default is 40000

Command Menu | < Back

MTSetACTDParams(

Thresh Low	Value for Activity High threshold, ranged from 0.0 to -60.0 dBm. **Must be greater than or equal to threshold_low**.
Thresh High	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. Default 40 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. Default 30000 ms
Min Noise	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. Default 40 ms
Max Noise	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. Default 10000 ms
Current Power	The channels current reading.

Command Menu | < Back

MTChInputSetACTDParams]

Thresh Low	Value for Activity High threshold, ranged from 0.0 to -60.0 dBm. **Must be greater than or equal to threshold_low**.
Thresh High	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. Default 40 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. Default 30000 ms
Min Noise	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. Default 40 ms
Max Noise	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. Default 10000 ms
Current Power	The channels current reading.

Command Menu | < Back

MTGetTimeSlotRxSignaling()

Name	Purpose
Framer	Framer index
TimeSlot	Time slot index
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
PrevSignal	Previous signaling
ElapsedTime	Duration (in 10ms unit) between Signal and PrevSignal

Command Menu | < Back

MTSetToneGenParams()

Type	Choose DTMF or MF.
Width	The length of a DTMF or MF tone in ms. The default is 100ms. 75 for AT and LD boards.
Gap	The inter-digit length between two tones in ms. The default is 50ms. 75 for AT and LD boards.
Low Gain	The amplitude for the low frequency. The default is 10156 (-7dBm). 9500 (-7.6dBm) for AT and LD boards.
High Gain	The amplitude for the high frequency. The default is 10156 (-7dBm). 12000 (-5.6dBm) for AT and LD boards.

NOTE: If any values are set to 0, the application will replace them with the default

Command Menu | < Back

MTSetToneDetectParams()

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.

Type	Choose tone type: DTMF or MF.
Sub Type	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

Command Menu | < Back

MTSetCPMTone()

Name	Description
Signal Type	Tone type (CPM_TONE, USR_TONE, SIT_TONE). The options for this field are defined in the NtiData.h file.
Cycles	The number of cycles of the defined tone that must completed before it is detected
Signal Index	Tone table index (0-19) 0-14: CPM tone, 15-19: USR tone
Signal Name	Tone ID: 0-199 for USR tone; or CPM_DIAL1, etc. The options for this field are defined in the NtiData.h file.
signal	Enable - indicates whether this tone profile is active or not, it should always be enabled when a tone profile is defined DetectSIT - always disable for CPM signals DetectLate - detection is delayed until the minimum duration for the last gap in the cycle following the completion of the number of cycles specified by ncycDetect. DetectCycle - when enabled, an end of cycle event will be reported with both detection and then with each pulse for use as a counting tool DetectLost -
Frequency	Set the tone frequency
Pulse	Input the tone's pulse values

Command Menu | < Back

MTSetUsrTone()

The MT_USRTONE structure is defined as follows:

Type	Name	Description
MT_CPMTONE	Tone	Tone specifications as defined in the MT_CPMTONE structure. Refer to the NtiData.h file for a structure definition.

Type	Name	Description
UCHAR	asCPMTone	Treat this tone as the specified CPM tone 0 = USR_NONE,Report EVT_UTONE_ON on detection 1 = USR_DIAL, Report EVT_MON_DIAL on detection 2 = USR_RINGBACK 3 = USR_BUSY 4 = USR_FAX 5 = USR_SIT 6 = USR_LOOPOFF 7 = USR_DIGIT, Report EVT_DIGIT and the specified digit through DTMF queue
UCHAR	useDigit	The digit if asCPMTone is set to USR_DIGIT

Command Menu | <Back

MTSetECParams()

The MT_ECPARAMS structure is defined as follows:

Type	Name	Description
SHORT	max_comfort_noise	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

Command Menu | <Back

MT(Set/Get)VoiceDetectParams()

The MT_VOICEDETECTPARAMS structure is defined as follows:

Type	Name	Description
ULONG	threshold_power	the minimum voice-band power required for voice detection
USHORT	relative_threshold	the minimum proportion of band pass filter power required for detection in any given filter
USHORT	threshold_filter_count	the number of band-pass filters where the relative_threshold must be met for voice detection
USHORT	no_voice_min	length of time there is no voice activity to determine the voice has stopped speaking

Type	Name	Description
USHORT	reset	the minimum length silence (no voice detection) required to reset CPM voice detect. If a voice is detected, new events are generated.
USHORT	voice_min	the minimum time voice activity is required before the application generates a voice detected event
USHORT	machine_min	the minimum voice activity required to determine that an answer machine has been detected

Command Menu | <Back

MTGETDIGITS()

General Termination	
Max Time	The maximum time, in milliseconds, that a background function can be active.
Digit Termination	
MaxDigits	The background function terminates if the DTMF queue contains `MaxDigits` or more digits. Only DTMF digits detected on the primary channel are used for termination control.
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 accepts 0-9, A, B, C, D, #, *, or @.
Inter Digit Time	This specifies the maximum time, in milliseconds, between digits received.

Command Menu | < Back

MEDIA

MTRECFILE()

File Name	Locate the file
Data Format	Select the format of the data
Start Offset	offset, from the beginning of the file, to start the play from Set to -1 to indicate the end of the table
Advanced	Use the Advanced button to set any MT_IO_CONTROL paramaters.

Command Menu | <Back

MTPLAYFILE()

File Name	Locate the file
Data Format	Select the format of the data
Index	An MT_INDEX_TABLE structure that specifies the data to play
Start Offset	offset, from the beginning of the file, to start the play from Set to -1 to indicate the end of the table
Advanced	Use the Advanced button to set any MT_IO_CONTROL paramaters.

Command Menu | < Back

MTRECBUFFER()

Data Format	Select the format of the data
Buffer Size	The size of the application buffer
Advanced	Use the Advanced button to set any MT_IO_CONTROL paramaters.

Command Menu | <Back

MTPLAYBUFFER()

Data Format	Select the format of the data
Buffer Size	The size of the application buffer
Advanced	Use the Advanced button to set any MT_IO_CONTROL paramaters.

Command Menu | <Back

MTPLAYTONE()

ampl1	Amplitude 1 in dBm, 3.0 dBm to -60.0 dBm
freq1	Frequency 1 in Hz (300 Hz to 3400 Hz, 0 is also valid)
ampl2	Amplitude 2 in dBm (+3.0 dbm to -60.0 dbm)

freq2	Frequency 2 in Hz ((300 Hz to 3400 Hz, 0 is also valid)
durtone	Duration in 125 us of generated tone
dursilence	Duration in 125 us of post tone silence
Repeat Tone	Enable this option to generated the same tone repeatedly.

Command Menu | < Back

MTPLAYINDEX()

File Name	Locate the file
Data Format	Select the format of the data
Index	An MT_INDEX_TABLE structure that specifies the data to play
Offset	offset, from the beginning of the file, to start the play from Set to -1 to indicate the end of the table
Length	Length of data to play. Set to 0 to indicate a skipping of index entry.
Advanced	Use the Advanced button to set any MT_IO_CONTROL parameters.

Command Menu | < Back

MTSTREAMIN/OUTAPP()

File Name	Locate the file
Data Format	Select the format of the data
DLL Stream Size	DLL buffer size, the size of the buffer set by the application for the DLL to buffer into
Watermark	The watermark byte count. 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. The watermark byte count can be specified as 0 to disable watermark reporting.
Buffer Size	application buffer size
Buffer Timeout	timeout (in ms) when queue is full. This value must be at least 20 ms. (-1= wait until buffer space is available)
Advanced	Select the Advance button to set any parameters used by the MT_IO_CONTROL structure.

Command Menu | < Back

CTBUS

MTSetCTMasterClock()

Board	Board index number
Master Mode	MODE_MASTER_A, MODE_MASTER_B, or MODE_SLAVE for MUX_H100; MODE_MASTER or MODE_SLAVE for MUX_MVIP
Clock Source	Clock source must be set if the clock is Master, otherwise it can be ignored.

Command Menu | < Back

MTSetCTNetrefSource()

Board	Board index number
Master Mode	This option is only available if the board is in slave mode.
Clock Source	Clock source must be set if the clock is Master, otherwise it can be ignored.

Command Menu | < Back

MTSetNet()

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

Command Menu | < Back

MTSetCTRoute()

Channel	channel number
Direction	DIR_TALK, DIR_LISTEN or DIR_SUPER
CT Stream	CT bus stream index: 0-15 for MVIP, 0-31 for H100
CT 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

NOTE: In this 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 DT boards.Refer to the section that explains ***MTSetCTRoute()*** in the Developer's Guide for a diagram of the SmartWORKS logical model when using this API.

Command Menu | < Back

MTSetFramerOutput()

Number	Index of framer on the specified board: 0-3 for the DP, 0-1 DT, and 0-23 for the NGX (24 channel card)
Slot	Index of framer slot: 1-24 for T1, 0-31 for E1
Stream	Index of TDM stream: 0-15 for MVIP, 0-31 for H100
TimeSlot	Index of TDM slot on a stream: 0-31 for MVIP

Before developing applications that work in 4-wire scenarios, it is important to understand how framing and timeslot numbering was implemented on SmartWORKS products.

Diagram and mapping tables are provided in the Developer's Guide where the API ***MTSetFramerOutput()*** is explained.

Command Menu | < Back

MTSetInputs()

Channel Number	Select a channel by highlighting it on the SmartVIEW interface.
TDM Stream	channel input connection stream, Index of TDM stream: 0 - 15 for MVIP, 0-31 for H.100
TDM Timeslot	channel input connection timeslot: 0 to 31 for MVIP number varies for H.100, depending on the stream speed -1 for not connected

Command Menu | < Back

MT(Get/Set/Reset)FramerLoopbackMode()

nBoard	Board index number
nFramer	Framer number
LoopbackMode	Valid mode Id: 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.

Command Menu | < Back

MTSetOutput()

Channel Number	The channel number used for output
TDM Stream	channel connection stream, Index of TDM stream: 0 - 15 for MVIP, 0-31 for H.100
TDM Timeslot	channel connection timeslot: 0 to 31 for MVIP number varies for H.100, depending on the stream speed -1 for not connected

Command Menu | < Back

MTAJTALK/LISTEN()

MTAJTalk()
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

Command Menu | < Back

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.

- 2048 KHz - 32 timeslots per stream are available.
- 4096 KHz - 64 timeslots per stream are available.
- 8192 KHz - 128 timeslots per stream are available.

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*

Command Menu | < Back

MTGetT1FramerStatistics()

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.
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.
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.
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.

Command Menu | < Back

MTGetE1FramerStatistics()

FASErr;	Framing bit error In E1 mode, the count is either number of FAS (frame alignment signal) bits.
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.
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.

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.

Command Menu | < Back

MTGETNGXFRAMERSTATISTICS()

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.
TEErr	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.
SyncErr	Indicates a complete loss of signal
NTAmpl	PBX amplitude in volts. This is not an error indication, rather it is a true reading of current voltage levels.
TEAmpl	Phone amplitude in volts. This is not an error indication, rather it is a true reading of current voltage levels.
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.
Clipping	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.

Command Menu | <Back

MTGETFRAMERALARMSTATUS()

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

Type	Name
nFramer	Framer index
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. (This field is also used to indicate a loss of sync. alarm on the SmartWORKS PCM).
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.
AIS_Alarm	Alarm Indication Signal (AIS) alarm detected. Tranmitted to the NI upon a loss of originating signal or a sever disruption.
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.
SMFAE	Not applicable, E1 only
SMFCRC4E	Not applicable, E1 only
TS16RAI_alarm	Not applicable, E1 only

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

nFramer	Framer index
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.
LOF_Alarm	not applicable
AIS_Alarm	not applicable
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.
SMFAE	Indicates signaling MF alignment error. The received alignments 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.

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.
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:

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

Command Menu | < Back

MTCC_GETSTATUSBYCH()

Channel Number	The specified channel to obtain statistics
Source	Only 2 values are possible INCOMING or OUTGOING
Trunk	The trunk where the call occurred.
State	State of the call. Call states are defined in the NtiDataCC.h file.
Caller	If no number has been obtained, the number of digits field = 0.
Called	If no number has been obtained, the number of digits field = 0.
Connected	If no number has been obtained, the number of digits field = 0.
Redirecting	If no number has been obtained, the number of digits field = 0.

Command Menu | < Back

MTCC_GETSTATUSBYREF()

Reference	A unique value assigned by the user's application for each call.
Source	Only 2 values are possible INCOMING or OUTGOING
Trunk	The trunk where the call occurred.
State	State of the call. Call states are defined in the NtiDataCC.h file.
Caller	If no number has been obtained, the number of digits field = 0.
Called	If no number has been obtained, the number of digits field = 0.
Connected	If no number has been obtained, the number of digits field = 0.
Redirecting	If no number has been obtained, the number of digits field = 0.

Command Menu | < Back

MTGETDATALINKSTATISTICS()

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/Data Link	
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.

Command Menu | < Back

MTGETPHYSTATISTICS()

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.
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.

Command Menu | < Back

MTGETDATALINKSTATUS()

Data link status can be returned with the following values:

- 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

Command Menu | <Back

SUMMATION

MTGETSUMMATIONINFO()

Structure

The MT_SUMMATION_INFO structure is defined as follows:

iSummation	Index of summation resource: 0 for the first one

tResource	Summation resource sub-type, if any
iResource	Summation resource system index
iChGCI	GCI index of the channel which controls the summation resource
iBoard	System index of the board the summation resides
iOnBoard	Summation index within the board it resides
nInputs	Number of inputs for the summation

Command Menu | < Back

MTADDSUMMATIONINPUT()

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.

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.

Command Menu | <Back

Chapter 5

SmartProfiler

SmartProfiler

The SmartProfiler performs a frequency and cadence analysis of call progress signals. The output is based on a 256 DFT (Discrete Fourier Transform) using Hamming-windowed input samples. This output produces a group of bins with the center frequency of each bin being a multiple of 31.25 Hz. When a signal is analyzed both the nominal frequencies of each sine wave is returned along with the bin numbers that represent this sine wave. Cadence is measured by both how long (ms) the tone is detected with a PULSE ON and then a PULSE OFF.

This section explains how to set up and run the SmartProfiler when operating on the Windows OS.

GETTING STARTED

NOTE: This utility is not supported when running on Linux.

The SmartProfiler requires two phone lines connected to the phone network. The phones must also be connected through a PBX or CO. Please make note of any dialing protocols required for this network (for example, it is necessary to dial a '9' to reach an outside line).

SmartProfiler operates with terminate cards only. Verify that one card, with appropriate drivers has been installed. The profiler works with the following SmartWORKS boards:

- SmartWORKS LD (409, 409H, 809, 1609, 2409)

The procedure used to create tone profiles is a four step process:

1. Enter the system configuration and phone numbers to dial.
2. Call the numbers and record the tones.
3. Analyze and create a profile of the recorded tones.
4. Set the information to a file and test.

The SmartProfiler works in two modes:

- Automatic
- From a File

Automatic

The Profiler automatically performs a sequence of calls to record busy, ringback and dial signals.

Dial Tone - this is done by going off-hook, dialing a number to get an outside line and then recording the dial tone for approximately 5 seconds.

Ringback Tone - this is done by dialing the number of another voice port. As soon as the board hears the signal it starts recording the tone for a maximum of 30 seconds.

Busy Tone - SmartProfiler "busy's" the voice port specified in configuration. It then calls it to record the busy signal.

Hang Up Tone - produced by the CO after a far side hang up.

From a File

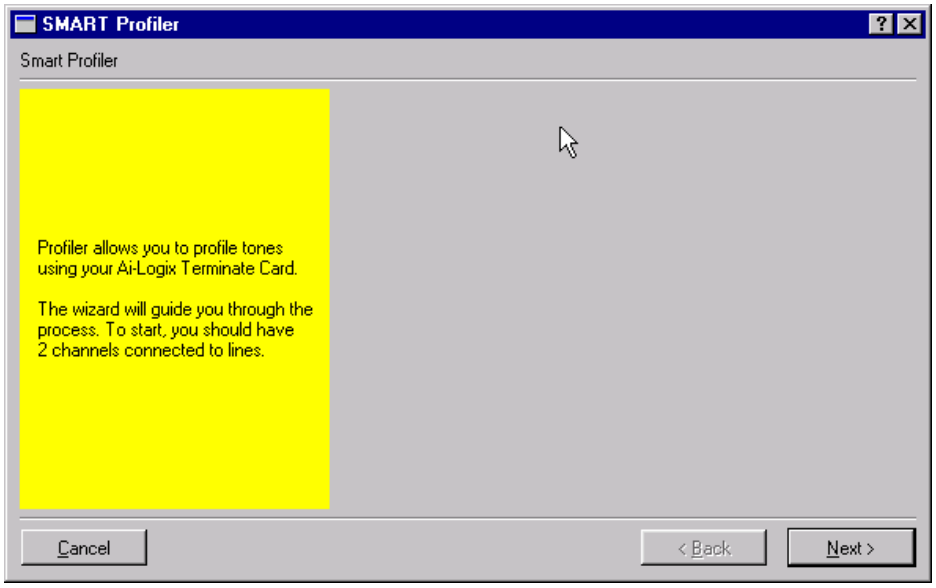
Users can record any signal from the line and use the Profile to analyze the signal.

PROFILE SETUP

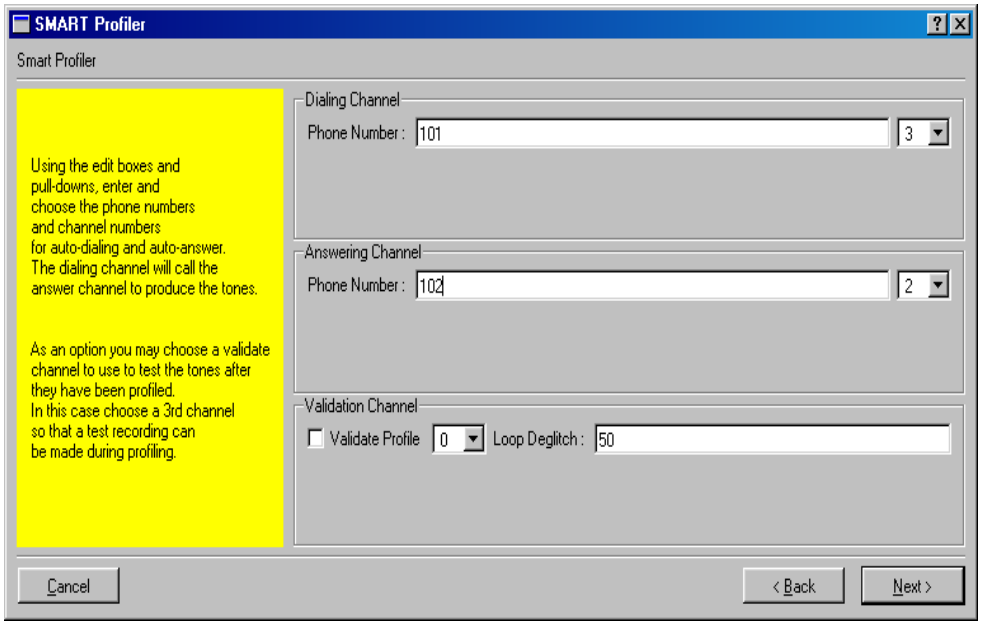
The SmartProfiler is installed when you install all AudioCodes software. To run the application go the Start bar then > Programs > Ai-Logix > SmartProfiler. The profiler, when executed, automatically discovers the terminate card. If no terminate card is installed, an error message is displayed. When the Profile is first started users have the option of selecting to work in Auto or File mode.

AUTO MODE

Select the ***Auto*** option to begin the wizard.



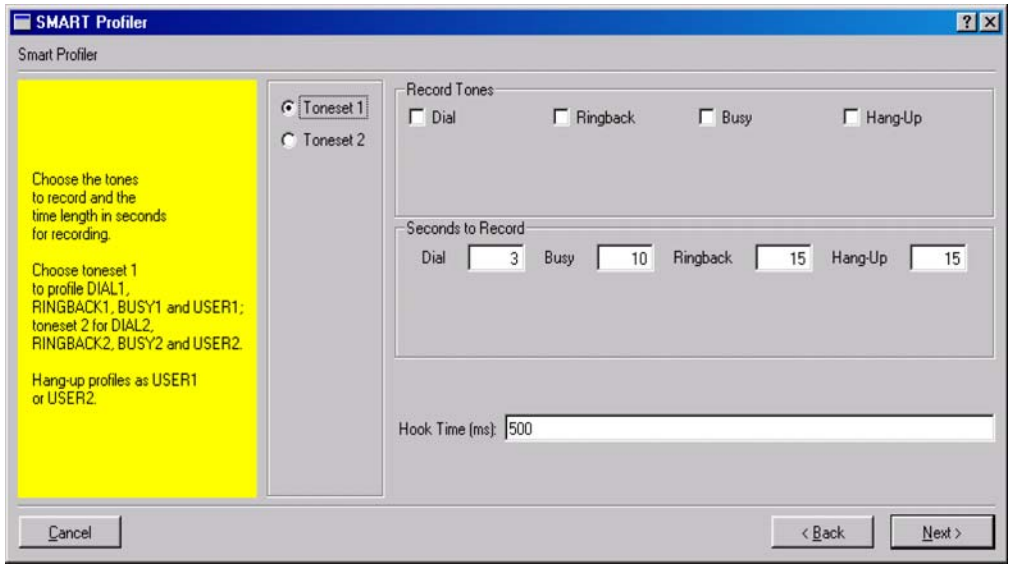
This is the first screen that appears. Click on the ***Next*** button to set up the dialing options.



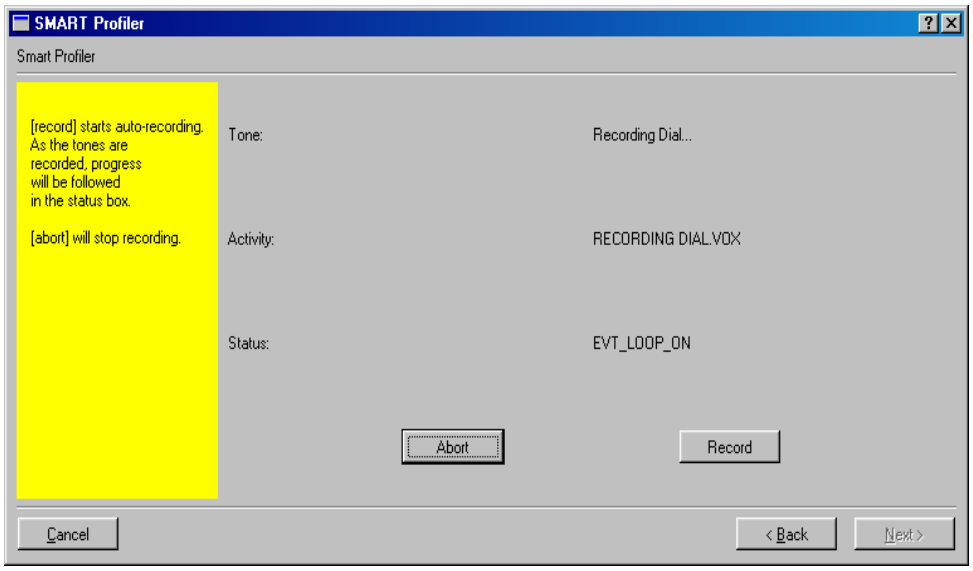
1. Identify the phone number the SmartProfiler will use to make outbound calls. In the above example, only the extension number of the phone was required. Add any prefixes required for dialing, such as '9' for an outside line.

NOTE: It is important to check whether the channels you are using have been configured to call forward or roll over. If so, they will not generate a busy signal.

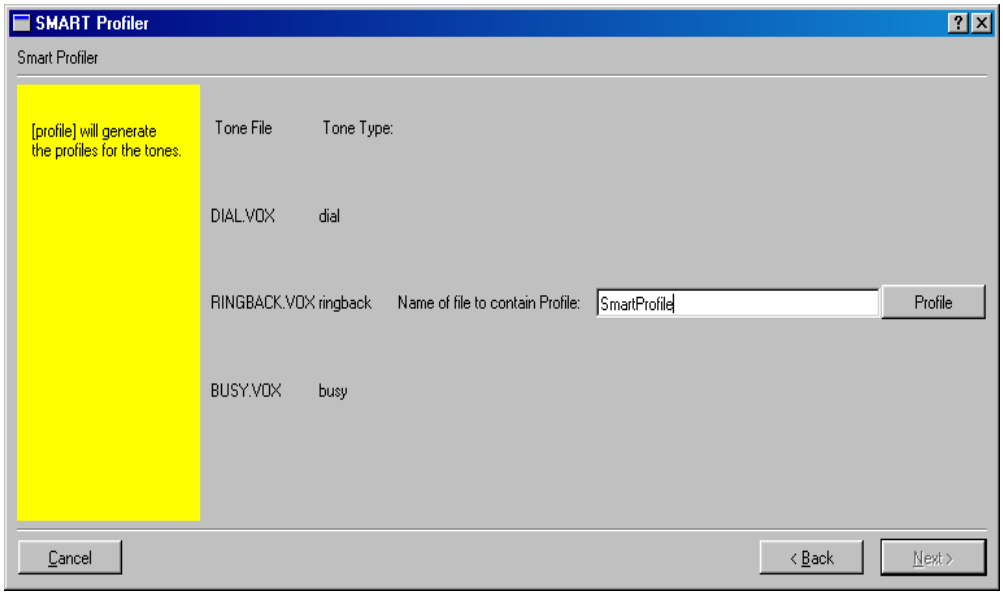
2. Add the channel number on the board that is used to monitor this phone number or extension.
3. Identify the phone number or extension that the SmartProfiler will dial out to.
4. Add the channel number on the board that is used to monitor the dialed phone.
5. Select Validate Profile and then another channel on the board than can be used to validate results. This option is used after the Profiler has completed analyzing the signals. The user can test the operation of the profiles if these options are selected.
6. Set Loop Deglitch - the default is 53.
7. Click the ***Next*** button to continue.



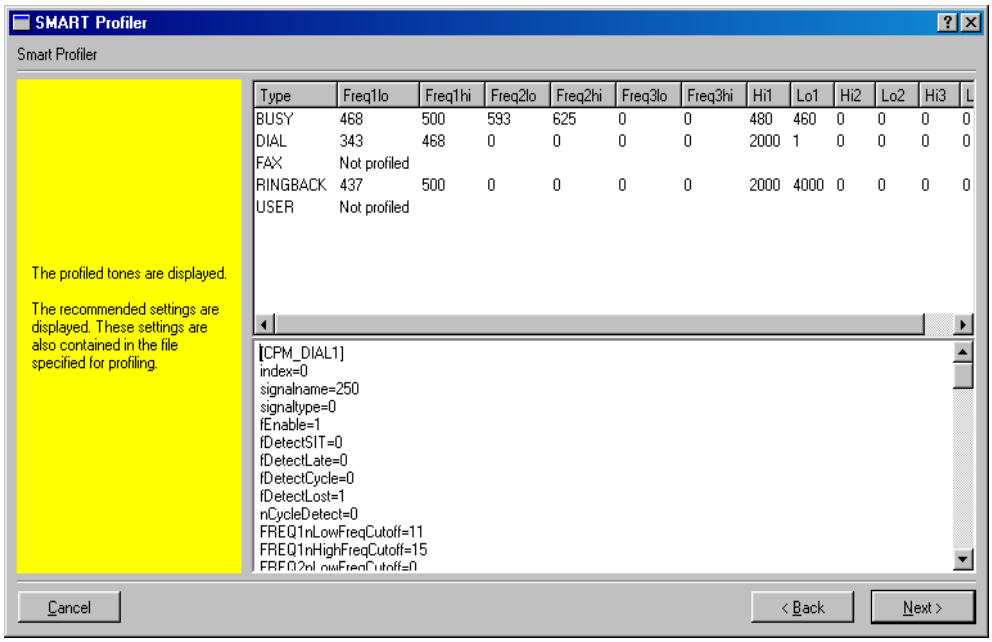
1. *Record Tones* - Select the tones that the SmartProfiler will generate and record.
2. *Seconds to Record* - once the tone is detected, this value sets the time (in seconds) of the recording.
3. *Hook Time* - the time the application waits after a phone is taken off hook before detecting tones
4. When finished, click the ***Next*** button to continue.
5. Use the next screen to Verify the tones. Click on the ***Verify*** button to begin. Here the profiler performs quick test by checking the phone numbers selected. Any errors are reported.
6. When the test is complete, the ***Next*** button becomes activated so you can continue.



1. Click the ***Record*** button to begin the recording process. Each recording is placed in the root folder where the SmartProfiler has been installed. All recordings are saved as *.vox* files. Please note, the Profiler will write over any existing recordings. To save those, rename or move the files.
2. When finished, the ***Next*** button becomes active and you can continue.



1. Select a file name for this set of profiles. Do not appended a file extension. All signal profiles are added to one file.
2. Select the ***Profile*** button when completed.
3. When finished, the ***Next*** button becomes active and you can continue.



When finished, the signal profiles are displayed. **NOTE:** The hang up tone is presented as a User tone.

The upper section shows the nominal frequencies and cadence pattern of each signal. The lower section displays the complete signal profiles. Note bin numbers are used and not the nominal frequencies. To configure a channel use ***MTLoadSignalProfile()*** which is used to pull one signal profile from the file. Then invoke ***MTChSetCPMSignalParams()*** to set the parameter values to the channel.

NOTE: Default Index ID's and signal types are set for each signal profile. When using MTChSetCPMSignalParams() these can be modified.

The following shows a Dial tone signal profile. A complete explanation of all CPM APIs is available in an application notes: *Call Progress Monitoring* and *Understanding Signal Profiles* available on the AudioCodes website.

```
[CPM_DIAL1]
index=0
signalname=250
signaltype=0
fEnable=1
fDetectSIT=0
fDetectLate=0
fDetectCycle=0
fDetectLost=1
nCycleDetect=0
FREQ1nLowFreqCutoff=11
FREQ1nHighFreqCutoff=15
FREQ2nLowFreqCutoff=0
FREQ2nHighFreqCutoff=0
FREQ3nLowFreqCutoff=0
FREQ3nHighFreqCutoff=0
PULSE1ONMIN=2000
PULSE1ONMAX=0
PULSE1OFFMIN=1
PULSE1OFFMAX=1
PULSE2ONMIN=0
PULSE2ONMAX=0
PULSE2OFFMIN=0
PULSE2OFFMAX=0
PULSE3ONMIN=0
PULSE3ONMAX=0
PULSE3OFFMIN=0
PULSE3OFFMAX=0
```

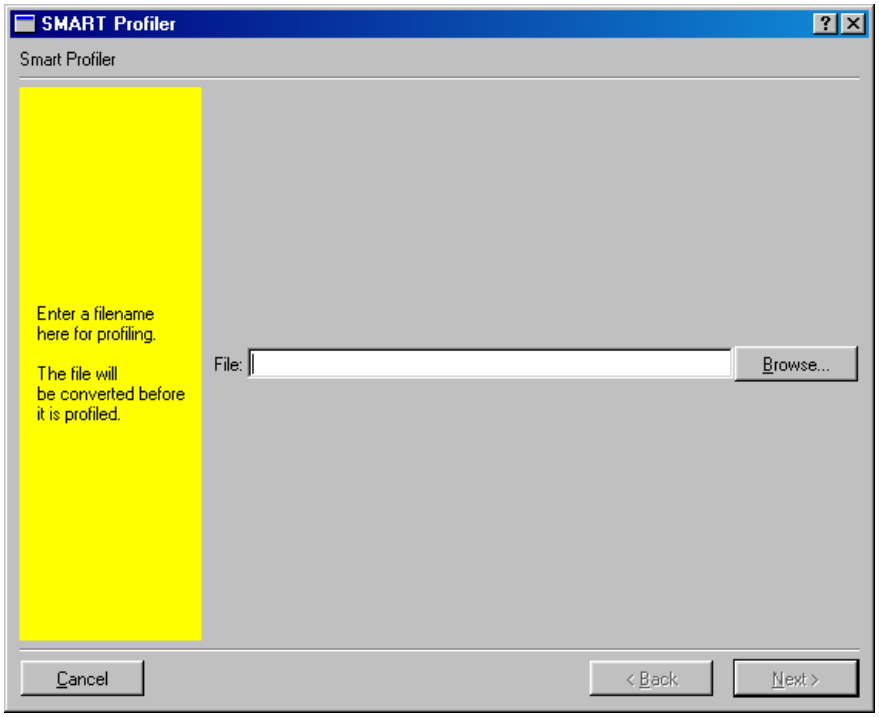
File Mode

Any recorded file (except SIT tones) can be passed into the Profiler and analyzed. To use the Profiler, files must be recorded in one of the following formats:

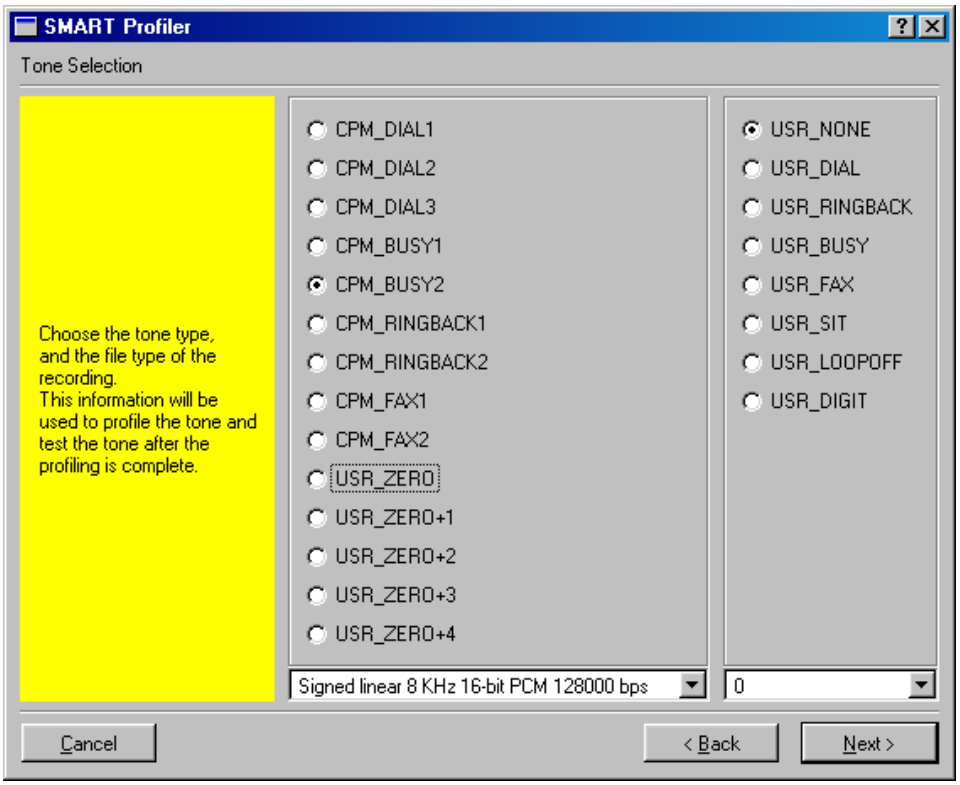
- u-law 8KHz 8-bit PCM 64000 bps
- A-law 8KHz 8-bit PCM 64000 bps
- signed linear 8KHz 16-bit PCM 128000 bps
- signed linear 8KHz 8-bit PCM 64000 bps
- unsigned linear 8KHz 8-bit PCM 64000 bps
- unsigned linear 8KHz 16-bit PCM 128000 bps
- signed linear 6KHz 16-bit PCM 96000 bps

To run the application go the Start bar then > Programs > Ai-Logix > SmartProfiler. The Profiler, when executed, automatically discovers the terminate card. If no terminate card is installed, an error message is displayed. When the Profiler is first started users have the option of selecting to work in Auto or File mode.

Select the ***File*** option to begin.



1. Use the ***Browse*** button to locate the recording that will be analyzed.
2. Use the ***Next*** button to continue.

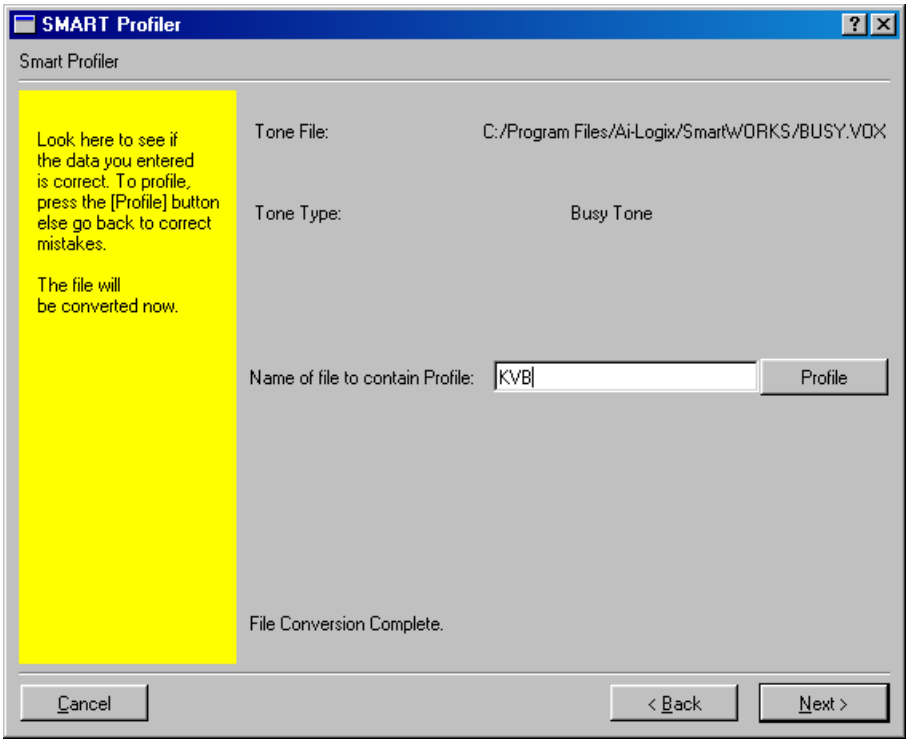


1. Select the signal name from the left side. If a Usr Tone is selected, then an option must be also selected from the list on the right.
   a) If this tone is not aliased, then select USR_NONE
   b) If this tone is aliased, select the signal name from the options:

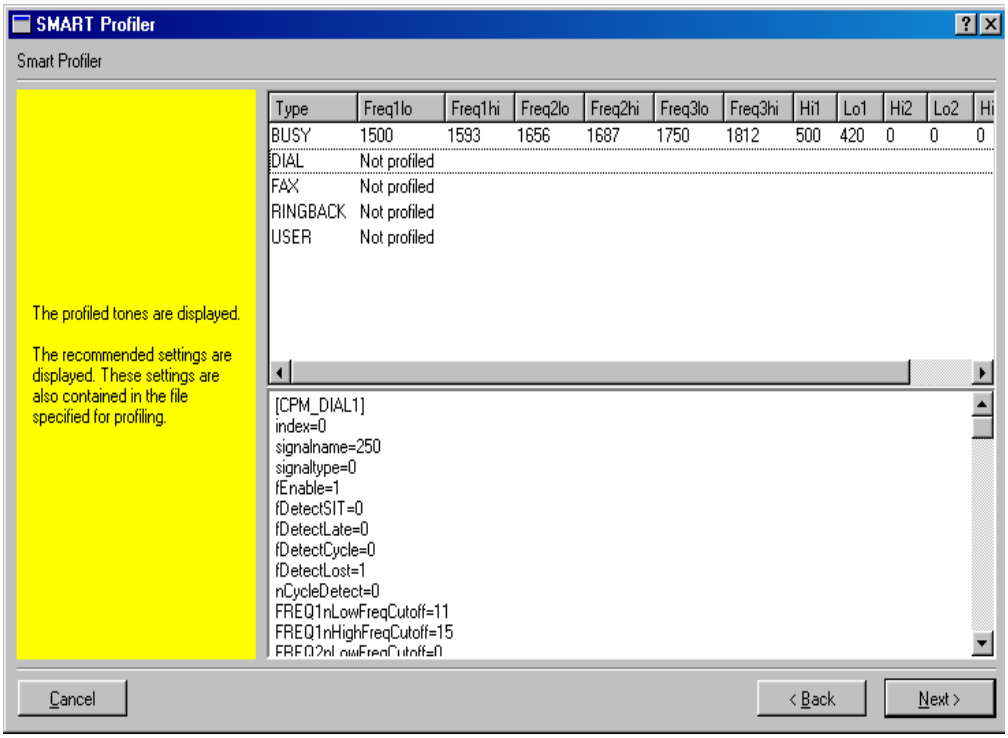
TABLE 1: USR TONES MAPPED TO CPM SIGNAL NAMES

USR_DIAL	CPM_DIAL1
USR_RINGBACK	CPM_RINGBACK1
USR_BUSY	CPM_BUSY1
USR_FAX	CPM_FAX1
USR_SIT	CPM_SIT1
USR_LOOPOFF	CPM_RECEIVEOFF

   c) If this signal is aliased to a digit (USR_DIGIT), then a digit must also be selected from the drop down list. When this signal is detected, a digit is placed in the DTMF queue alerting the user application.
2. Select the format used to record the signal.
3. Use the ***Next*** button to continue.



1. Verify that the correct recording has been selected. This is displayed in the Tone File field.
2. The type of the signal that is being profiled is displayed here.
3. Type the name of the file which will hold the completed signal profile. Do not append this name with a file extension. There are two options:
   a) Type in the name of a new file. This one signal profile is saved in this file and is identified by the signal name (CPM_DIAL1) that has been selected. When retrieving this information using ***MTLoadSignalProfile()*** both the file name and signal name must be used.
   b) Select the name of a file that already exists. This signal profile is appended to the end of this file. If a signal profile already exists with the same signal name, then it is replaced with this new profile.
4. Use the ***Profile*** button to begin the signal analysis.
5. When completed, use the ***Next*** button to continue.



The analysis has been completed. The upper half of the screen displays the nominal frequency and measured cadence pattern. The lower half of the screen displays the values that have been saved into the file.

A default index ID is added to the file. The user can modify the file by going to the copy located in the Ai-Logix/SmartWORKS/Profiles folder.

To configure a channel with these settings use the ***MTLoadSignalProfile()*** API to obtains all parameter values, then use ***MTChSetCPMSignalParams()*** to configure the channel. When using ***MTChSetCPMSignalParams()*** any parameters can be modified per the needs of your application.

To test the accuracy of the profile, select the ***Next*** button.