Dialogic® CG 6060 PCI Media Board Installation and Developer's Manual

October 2010 64-0481-02 RevA

www.dialogic.com

Copyright and legal notices

Revision history

Revision	Release date	Notes
9000-62417-160	May 2006	EH/LBG Beta
9000-62417-10	July 2006	LBG, Natural Access 2005-1, SP2
9000-62417-12	December 2006	LBZ, Natural Access 2005-1, SP3
9000-62417-13	June 2007	LBZ, Natural Access 2005-1, SP 4
9000-62417-14	October 2007	LBZ, Natural Access 2005-1, SP 5
9000-62417-15	June 2008	LBG/LBZ, Natural Access R8
9000-62417-16	February 2009	LBG, Natural Access R8.1
64-0481-01	October 2009	LBG, NaturalAccess R9.0
64-0481-02 Rev A	October 2010	LBG, NaturalAccess R9.0.4
Last modified: 2010-10-15		

Refer to www.dialogic.com for product updates and for information about support policies, warranty information, and service offerings.

Table Of Contents

1. Introduction

The *Dialogic® CG 6060 PCI Media Board Installation and Developer's Manual* explains how to perform the following tasks:

- Install and configure a CG 6060 board.
- Verify that the board is installed and operating correctly.
- Use the CG 6060 board keywords to configure the board.
- Use the CG 6060 board utilities.

This manual is for programmers and system integrators who develop media server applications. This manual defines telephony terms where applicable, but assumes that the reader is familiar with basic telephony and Internet data communication concepts, switching, and the C programming language.

Terminology

Note: The product to which this document pertains is part of the NMS Communications Platforms business that was sold by NMS Communications Corporation (“NMS”) to Dialogic Corporation (“Dialogic”) on December 8, 2008. Accordingly, certain terminology relating to the product has been changed. Below is a table indicating both terminology that was formerly associated with the product, as well as the new terminology by which the product is now known. This document is being published during a transition period; therefore, it may be that some of the former terminology will appear within the document, in which case the former terminology should be equated to the new terminology, and vice versa.

Former terminology	Dialogic terminology
CG 6060 Board	Dialogic® CG 6060 PCI Media Board
CG 6060C Board	Dialogic® CG 6060C CompactPCI Media Board
CG 6565 Board	Dialogic® CG 6565 PCI Media Board
CG 6565C Board	Dialogic® CG 6565C CompactPCI Media Board
CG 6565e Board	Dialogic® CG 6565E PCI Express Media Board
CX 2000 Board	Dialogic® CX 2000 PCI Station Interface Board
CX 2000C Board	Dialogic® CX 2000C CompactPCI Station Interface Board
AG 2000 Board	Dialogic® AG 2000 PCI Media Board
AG 2000C Board	Dialogic® AG 2000C CompactPCI Media Board
AG 2000-BRI Board	Dialogic® AG 2000-BRI Media Board

Former terminology	Dialogic terminology
NMS OAM Service	Dialogic® NaturalAccess™ OAM API
NMS OAM System	Dialogic® NaturalAccess™ OAM System
NMS SNMP	Dialogic® NaturalAccess™ SNMP API
Natural Access	Dialogic® NaturalAccess™ Software
Natural Access Service	Dialogic® NaturalAccess™ Service
Fusion	Dialogic® NaturalAccess™ Fusion™ VoIP API
ADI Service	Dialogic® NaturalAccess™ Alliance Device Interface API
CDI Service	Dialogic® NaturalAccess™ CX Device Interface API
Digital Trunk Monitor Service	Dialogic® NaturalAccess™ Digital Trunk Monitoring API
MSPP Service	Dialogic® NaturalAccess™ Media Stream Protocol Processing API
Natural Call Control Service	Dialogic® NaturalAccess™ NaturalCallControl™ API
NMS GR303 and V5 Libraries	Dialogic® NaturalAccess™ GR303 and V5 Libraries
Point-to-Point Switching Service	Dialogic® NaturalAccess™ Point-to-Point Switching API
Switching Service	Dialogic® NaturalAccess™ Switching Interface API
Voice Message Service	Dialogic® NaturalAccess™ Voice Control Element API
NMS CAS for Natural Call Control	Dialogic® NaturalAccess™ CAS API
NMS ISDN	Dialogic® NaturalAccess™ ISDN API
NMS ISDN for Natural Call Control	Dialogic® NaturalAccess™ ISDN API
NMS ISDN Messaging API	Dialogic® NaturalAccess™ ISDN Messaging API
NMS ISDN Supplementary Services	Dialogic® NaturalAccess™ ISDN API Supplementary Services

Former terminology	Dialogic terminology
NMS ISDN Management API	Dialogic® NaturalAccess™ ISDN Management API
NaturalConference Service	Dialogic® NaturalAccess™ NaturalConference™ API
NaturalFax	Dialogic® NaturalAccess™ NaturalFax™ API
SAI Service	Dialogic® NaturalAccess™ Universal Speech Access API
NMS SIP for Natural Call Control	Dialogic® NaturalAccess™ SIP API
NMS RJ-45 interface	Dialogic® MD1 RJ-45 interface
NMS RJ-21 interface	Dialogic® MD1 RJ-21 interface
NMS Mini RJ-21 interface	Dialogic® MD1 Mini RJ-21 interface
NMS Mini RJ-21 to NMS RJ-21 cable	Dialogic® MD1 Mini RJ-21 to MD1 RJ-21 cable
NMS RJ-45 to two 75 ohm BNC splitter cable	Dialogic® MD1 RJ-45 to two 75 ohm BNC splitter cable
NMS signal entry panel	Dialogic® Signal Entry Panel
Video Access Utilities	Dialogic® NaturalAccess™ Video Access Toolkit Utilities
Video Mail Application Demonstration Program	Dialogic® NaturalAccess™ Video Access Toolkit Video Mail Application Demonstration Program
Video Messaging Server Interface	Dialogic® NaturalAccess™ Video Access Toolkit Video Messaging Server Interface
3G-324M Interface	Dialogic® NaturalAccess™ Video Access Toolkit 3G-324M Interface

2. Overview of the CG 6060 board

Migration information

The CG 6060 board is based on the CG 6000/C, CG 6100C, and CG 6500C boards. Although the boards are very similar, be aware of the following hardware and software differences as you migrate from one of these boards to the CG 6060 board.

Hardware differences

CG 6000/C, CG 6100C, or CG 6500C board	CG 6060 board
DSP C5420 100 MIPS each. Each chip has two cores	DSP C5441 8 DSPs for 32 cores max (4256 MIPS)
10/100Base-T Ethernet	10/100Base-T Ethernet
Ethernet SPEED LED: • 10Base-T Ethernet (off) • 100Base-T Ethernet (on)	Ethernet SPEED LED: • 10Base-T Ethernet (off) • 100Base-T Ethernet (on)
No echo cancellation hardware	Proprietary echo cancellation chip
Trunk status LEDs for each trunk: CG 6000 – red and yellow CG 6000C – red, yellow, and green CG 6100C – green CG 6500C - green	Red and yellow LEDs per trunk. For more information, refer to Trunk LEDs.
CAS signaling processed in the DSP	CAS signaling processed in the framer
Agere T81***xx***	Proprietary TSI
As many as 8 DSP cores per TSI stream	4 DSP cores per TSI stream
Each TSI stream is 8 MHz with 128 timeslots	Each TSI stream is 8 MHz with 128 timeslots
Supports a 32-bit 33 MHz PCI bus	Supports 32-bit, 33 MHz or 66 MHz PCI bus
No board status LEDs	Board status LEDs were added. For more information, refer to Board status LEDs.

Software differences

The software differences between the CG 6000/C, CG 6100C, CG 6500C boards and the CG 6060 board include changes to the:

- System configuration file
- Board keyword file
- Booting sequence
- Board information

System configuration file

The CG 6060 board product name that appears in the system configuration file:

Product name	Description
CG_6060	A generic name that can be used to refer to any of the CG 6060 board variants.
CG_6060_4	One, two, or four-trunk CG 6060 boards.

The following example shows a CG 6060 board configured using no call control:

```
[CG 6060]
Product = CG_6060
Number = 0
Bus = 2
Slot = 9
File = c6060nocc.cfg
```

For more information, refer to Creating a system configuration file for oamsys.

Board keyword file

The CG 6060 board keyword file differs from the board keyword files of the CG 6000/C, CG 6100C, or CG 6500C boards in the following ways:

- The CG 6060 board uses framers instead of DSPs for signaling.

  You must configure the trunks to perform signaling. Once you configure the trunks, use only channel associated signaling (CAS) to convey signaling information:

  NetworkInterface.T1E1[***x***].SignalingType = CAS

  The CG 6060 board does not require DSPs to do signaling. Do not set the following DSP values:

  DSP.C5x[0].Files = qtsignal

  DSP.C5x[0..1].Files = 8tsignal

- The CG 6060 board merges the core file and the run module to create a file named *cg6060core.ulm*. The CG 6060 board does not require the following DLMFiles[***x***] settings:

  DLMFiles[***x***] = cg6krun

  DLMFiles[***x***] = cg6500run

- In the CG 6060 configuration file, only set the DLMFiles[***x***] keyword when using the board with the following software:

Software	**Setting**
Generic ISDN	DLMFiles[***x***] = c6060igen
Fusion	DLMFiles[***x***] = cg6060fusion
ISDN Management	DLMFiles[***x***] = c6060imgt
DPNSS	DLMFiles[***x***] = c6060dpnss
NaturalFax	DLMFiles[***x***] = cg6060fax

  When you use only one DLM file, ***x*** is always 0 (zero). If using more than one DLM file, number them sequentially starting with 0 (zero). They can appear in any order. For example:

```
DLMFiles[0] = cg6060fusion
DLMFiles[1] = c6060igen
```

- The following keywords are available to set hardware echo cancellation values:
  - HardwareEcho.EchoChipEnabled
  - HardwareEcho.Trunk[x].OnOffTimeslots
  - HardwareEcho.XLaw

  Note: Do not use hardware echo cancellation in conjunction with software echo cancellation (that is, echo cancellation implemented through programs running on CG board DSPs).

  For more information about hardware echo cancellation, refer to Configuring hardware echo cancellation.

- All DSP resources must be managed by the CG 6060 resource management scheme. Use the following keywords to set up appropriate resource management:
  - Resource[***x***].Definitions and Resource[***x***].DSPs instead of DSP.C5x[x].Files to specify DSPs and DSP functions.
  - Resource[***x***].TCPs instead of TCPFiles[***x***] to specify TCPs.

  The NaturalConferene API implements resource management differently than other Natural Access services. For more information, refer to the *Dialogic® NaturalAccess™ NaturalConference™ API Developer's Manual*.

Booting sequence

The CG 6060 board booting sequence differs from the booting sequence of the CG 6000/C, CG 6100C, and CG 6500C boards in the following ways:

- CG 6060 boards do not require that you run the *burnall* script to update the flash memory on any CG 6060 boards in the chassis.
- CG 6060 boards download the run file, *cg6060core.ulm,* directly into SDRAM when the boards boot up using OAM. This file is installed in the Windows *\\nms\cg\load* or UNIX */opt/nms/cg/load* directory.

- On CG 6060 boards, the OAM utility, *oammon*, logs boot-specific information to *cg6x6xboot_bus_slot.txt*. For information about using the *oammon* utility and generating log files, refer to the *Dialogic® NaturalAccess™OAM API System Developer's Manual*.

  The following example shows *oammon* output:

```
Fri Feb 11 11:51:01 - OAMEVN_ALERT ERROR Board 1 "Name1"
1:9 - log cg6x6x boot info to file C:\WINNT\system32\CG_6060boot_1_9.txt
Fri Feb 11 11:51:01 - OAMEVN_ALERT ERROR Board 1 "Name1"
Writing at SDRAM 0xfa2c580 for 4535720 bytes ...
Fri Feb 11 11:51:01 - OAMEVN_ALERT ERROR Board 1 "Name1"
cg6060core.ulm   loaded, version 2.10004, built Feb 11 2005
```

Board information

- The ID for the CG 6060 board is 0x6060. For example, *pciscan* displays the following information:

```
PCI Boards Scanner
Bus Slot ID
--- ---- ------
1    7  0x6060 CG_6060
1    9  0x6060 CG_6060
There were 2 PCI board(s) detected
```

- The OAM product number is 0x630 and is included in the *\nms\include\nmshw.h* file:

```
#define OAM_PRODUCT_NO_CG6060    0x630
```

- CG 6060 boards use the CG 6500C switching model with MVIP-95 DSP streams of 64 to 67 as shown in the following *boardinf* example:

```
Natural Access Board Information Demo V.13 (Jan 25 2005)
MVIP-95
Board  Addr  Type        MIPS  Free memory  Ports  DSP Slots   streams
------ ----- ----------- ----- ------------ ------ ----------- -------
0      1, 7  CG6060      3192  244076180     120   0..119      64-67

1      1, 9  CG6060      3192  244307880     128   0..127      64-67

Total port count = 120
```

- The subsystem ID for a CG board is 0x6060 as shown in the following *cg6ktool* example (use the -A option when you run *cg6ktool*):

```
CG family command line tool, V3.00 (Dec 10 2004) Dialogic Corporation
Board   SubSysID   Bus:Slot   Shelf-Slot    Temp    DSP Cores    Trunks
CG 6060    6060       1:9          0-0       0.0 C       24          0
CG 6060    6060       1:7          0-0       0.0 C       24          4
```

- CG 6060 boards use C5441 DSPs and not C5420 DSPs for applications. The DSP files have *.f41* extensions instead of *.f54* extensions. For example, CG 6000/C, CG 6100C, and CG 6500C boards use *dtmf.f54*, but CG 6060 boards use *dtmf.f41*. The *.f41* files are in the *\\nms\cg\load* directory (or */opt/nms/cg/load* for UNIX), which is the same directory that contains *.f54* files.

  Use *f41info* to display file resource usage instead of the *f54info* utility.

- The Fusion *msppsamp* and *tpktsamp* demonstration programs include the –E option to disable software echo cancellation running on the DSPs. Set this option when using hardware echo cancellation on the CG 6060 board. For information on using *msppsamp*, refer to the *Dialogic® NaturalAccess™ Fusion™ VoIP API Developer's Manual*.

- CG 6060 boards include an echo chip for providing hardware echo cancellation capabilities and to free up DSP resources. For more information, refer to Configuring hardware echo cancellation. To control hardware echo cancellation on a per-timeslot basis, refer to Using echo cancellation control.
- Trunk impedance on the CG 6060 board applies to two trunks and must be identical for trunk pairs. For information about the pairings, refer to NetworkInterface.T1E1[x].Impedance.
- CG 6060 boards do not support the following Switching service functions:
    - **swiConfigStreamSpeed**
    - **swiGetStreamsBySpeed**

  All 32 H.100 streams are automatically configured for maximum capacity (8 Mbit/s or 128 timeslots).

CG 6060 board features

The CG 6060 board is a Dialogic PCI media board. It is a high-density platform for IVR, fax, VoIP, and media server applications. The board uses a high performance PowerPC processor.

Configurations support up to four T1 or E1 digital trunk interfaces and up to two Ethernet 10/100Base-T interfaces.

Refer to the www.dialogic.com/declarations/default.htm for a list of available CG 6060 board configurations, for a list of countries where Dialogic has obtained approval for the CG 6060 board, and for product updates.

CG 6060 boards include the following features:

- DSP resources

  The CG 6060 board has up to 4256 MIPS of media processing DSPs.

- PCI bus connectivity

  Each CG 6060 board is designed to reside in a single PCI bus slot. Each board contains a universal (5 V or 3.3 V signaling) PCI bus interface compliant with the *PCI Local Bus Specification, Revision 2.2*. The PCI interface is a 33 or 66 MHz 32-bit device.

- Trunk connectivity

  Board configurations provide up to four T1 or E1 network interfaces for digital trunk connectivity. You must configure the board for either T1 or E1. For more information, refer to Configuring the T1 or E1 interface and to NetworkInterface.T1E1[x].Type.

- H.100 bus connectivity

  The CG 6060 board fully supports the H.100 bus specification. The H.100 bus enables multiple boards to share data. For example, you can connect two or more CG 6060 boards for applications that perform trunk-to-trunk switching. You can use H.100 compatible products from other manufacturers with the CG 6060 board.

The H.100 interface supports the following stream configurations on the H.100 bus:

- Full mode: 32 streams at 8 MHz each, which provides 128 timeslots each for a total of 4096 timeslots.
- Backward compatibility mode: 16 8-MHz streams, 16 2-MHz streams (total of 2560 timeslots). The H.100 interface can operate with MVIP-90 boards on the same bus. In this configuration, an H.100 board in the system must be the bus master.

- Telephony bus switching

  Switching for the CG 6060 board offers support for the H.100 bus within the H.100 architecture. On the CG 6060 board, switch connections are allowed for a total of 512 full duplex connections between local devices and the H.100 bus.

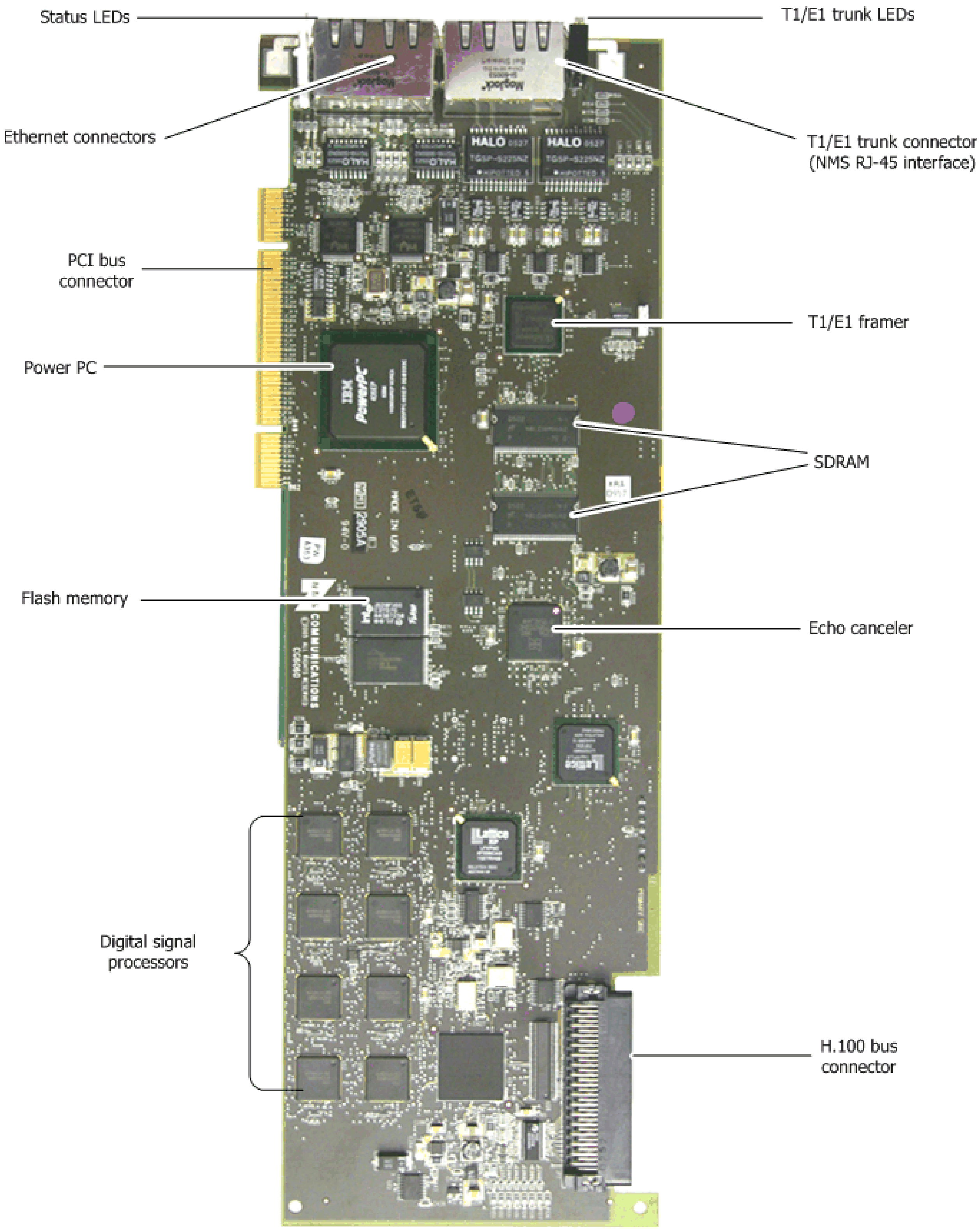
- Ethernet connectivity

  The CG 6060 board contains two 10/100Base-T Ethernet connections for fast Ethernet connectivity and support of both IPv4 and IPv6. For more information, refer to Connecting to an Ethernet network.

- Echo cancellation

  CG 6060 boards support up to 120 ports of line echo cancellation with 64 ms tails.

The following illustration shows where various components are located on a CG 6060 board:



Software components

Natural Access is a development environment that provides such services as call control, system configuration, and voice store and forward. CG 6060 boards require the following software components that are available with Natural Access:

- NaturalAccess OAM API (Operations, Administration, and Maintenance) software and related utilities.
- Configuration files that describe how the board is set up and initialized.
- Runtime software and drivers that control the CG 6060 board.
- One or more trunk control programs (TCPs) that enable applications to communicate with the telephone network using the signaling schemes (protocols) on the trunk.

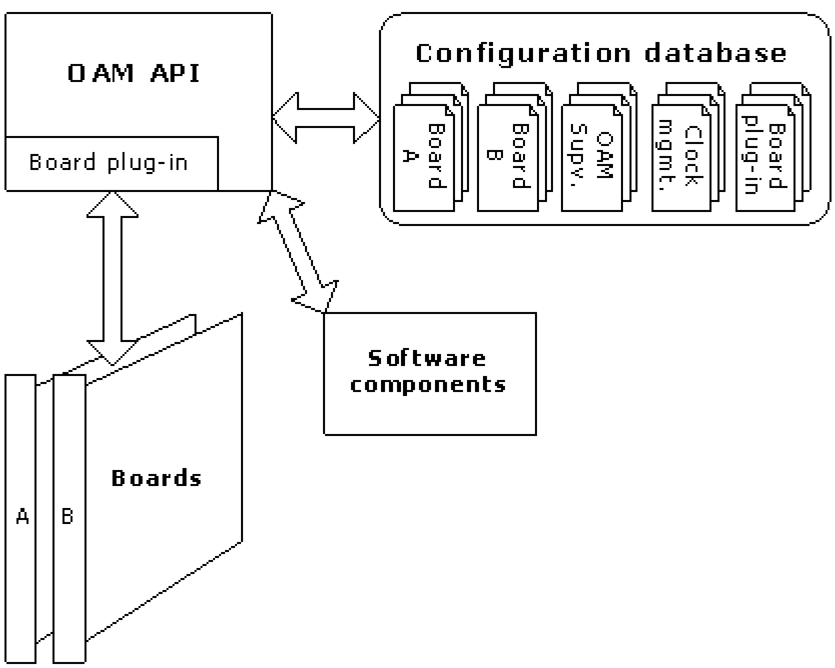
NaturalAccess OAM API

The NaturalAccess OAM API manages and maintains the telephony resources in a system. These resources include hardware components (including CG boards) and low-level board management software modules (such as clock management).

Using the OAM API, you can:

- Create, delete, and query the configuration of a component.
- Start (boot), stop (shut down), and test a component.
- Receive notifications from components.

The OAM API maintains a database containing records of configuration information for each component as shown in the following illustration. This information consists of parameters and values.



Each OAM API database parameter and value is expressed as a keyword name and value pair (for example, Clocking.HBus.ClockMode = MASTER_A). You can query the OAM API database for keyword values in any component. Keywords and values can be added, modified, or deleted.

Before using the NaturalAccess OAM API or any related utility, verify that the NaturalAccess server (*ctdaemon*) is running. For more information about *ctdaemon*, refer to the *Dialogic® NaturalAccess™ Software Developer's Manual*. For general information about the OAM API and its utilities, refer to the *Dialogic® NaturalAccess™ OAM System Developer's Manual*.

CG board plug-in

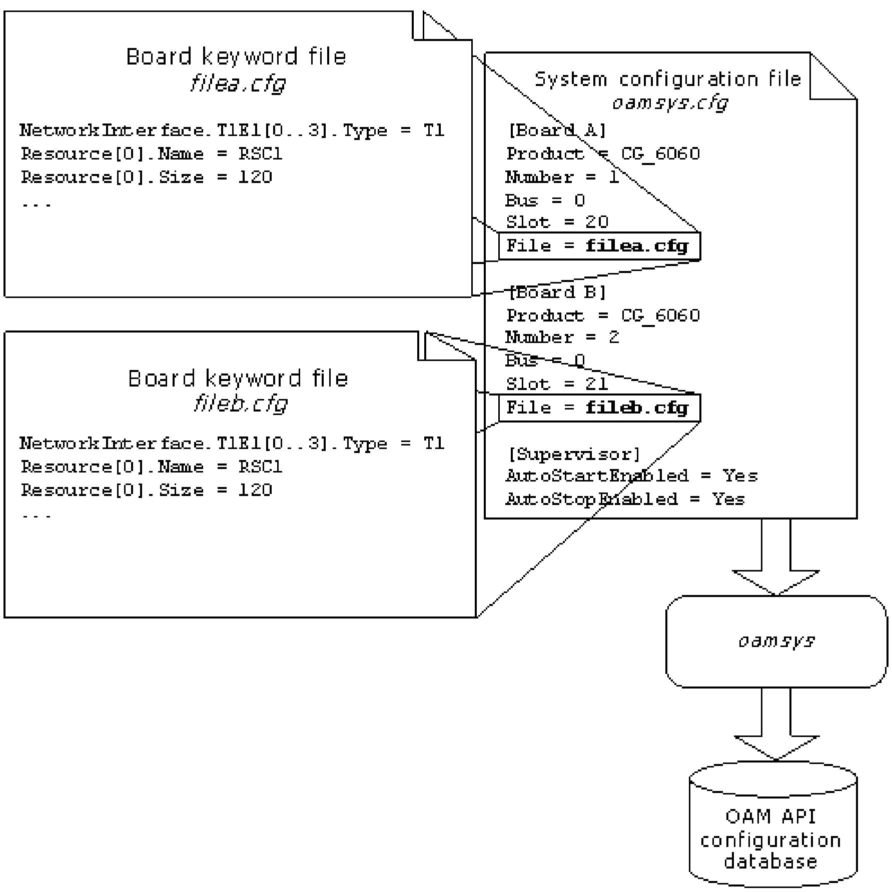
The OAM API uses the CG board plug-in module to communicate with CG boards. The name of the CG plug-in is *cg6kpi.bpi*. This file must reside in the *\nms\bin* directory for Windows or */opt/nms/lib* for UNIX for OAM to load it when it starts up.

Configuration files

The OAM API uses two types of configuration files:

File type	Description
System configuration	Contains a list of boards in the system and the names of one or more board keyword files for each board.
Board keyword	Contains parameters to configure the board. These settings are expressed as keyword name and value pairs.

Several sample board keyword files are installed with Natural Access. You can reference these files in your system configuration file or modify them.



When you run *oamsys*, it creates OAM API database records based on the contents of the specified system configuration file and board keyword files. It then directs the OAM service to start the boards, configured as specified.

Refer to Configuring and starting the system with oamsys for more information.

Runtime software

The runtime software is stored in a run file on the host computer. CG 6060 boards download the run file, *cg6060core.ulm,* directly into SDRAM when the boards boot up using the OAM API. This file is installed in the *\\nms\cg\load* directory for Windows or the */opt/cg/load* directory for UNIX.

DSP files enable the CG 6060 on-board DSPs to perform certain tasks, such as DTMF signaling, voice recording, and playback.

Several run files and DSP files are installed with NaturalAccess. Specify the files to use for your configuration in the board keyword file. Refer to Using board keyword files for more information.

Trunk control programs (TCPs)

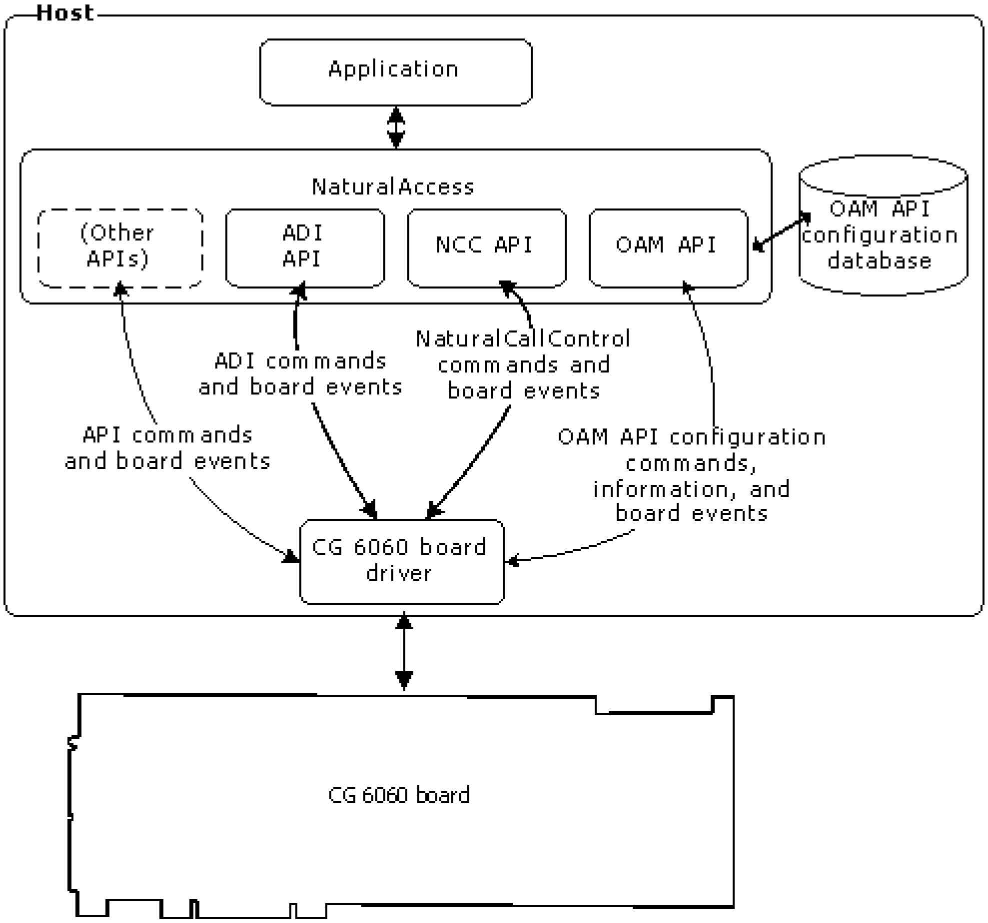
CG 6060 boards are compatible with a variety of PSTN signaling schemes, called protocols. A trunk control program (TCP) performs all of the signaling tasks to interface with the protocol used on a channel. TCPs run on the board, relieving the host computer of the task of processing the protocol directly.

Several different protocol standards are in use throughout the world. For this reason, different TCPs are supplied with NaturalAccess. Each TCP supports various country-specific variations.

For applications that support multiple protocols simultaneously, you can load more than one TCP at a time. Specify the TCPs in the configuration file. OAM downloads the specified TCPs to the board. For more information about TCPs, refer to the *Dialogic® NaturalAccess™ CAS API Developer's Manual*.

NaturalAccess

NaturalAccess is a complete software development environment for voice applications. It provides a standard set of functions grouped into logical services. Each service has a standard programming interface. For more information about standard and optional NaturalAccess services, refer to the *Dialogic® NaturalAccess™ Software Developer's Manual.* The following illustration shows the NaturalAccess software environment as it relates to OAM software and CG 6060 hardware:



Fusion API and the CG 6060 board

Gateway applications provide a way of transferring data between telephone network and packet network interfaces. Fusion provides the ability to develop IP telephony gateway applications that run on CG boards. For more information about Fusion CG 6060 board configurations, refer to the *Dialogic® NaturalAccess™ Fusion™ VoIP API Developer's Manual*.

Fusion configurations use CG 6060 boards to receive and transmit data to PSTN and to IP networks. Fusion applications use NaturalCallControl API (NCC API) functions to place and receive PSTN calls, and Media Stream Protocol Processing API (MSPP API) functions to create and configure media channels between PSTN and IP networks. For more information about MSPP API functions, refer to the *Dialogic® NaturalAccess™ Fusion™ Media Stream Protocol Processing API Developer's Manual.*

Ethernet interfaces

CG 6060 board Ethernet interfaces support IPv4 as well as IPv6 implementations of the Internet protocol. For information about implementing IPv4 and IPv6 Ethernet interface support on CG 6060 boards, refer to the following topics:

For information about...	Refer to...
Configuring IPv4 Ethernet interfaces	Configuring IPv4 Ethernet connections.
Configuring IPv6 Ethernet interfaces	Configuring IPv6 Ethernet connections.
Running the board in dual IPv4/IPv6 stack mode	Running in IPv4/IPv6 dual stack mode.
Gathering statistics from IPv4 and IPv6 Ethernet interfaces (ping also supported)	cg6kcon - Displaying statistics about CG board activity.
Adding, printing, and deleting IPv6 addresses without editing individual board keyword files	cgv6if - Adding, printing, and deleting IPv6 addresses.

3. Installing the hardware

System requirements

To install and use CG 6060 boards, your system must have:

- A chassis with an available PCI bus slot with 5.0 V and 3.3 V of supplied power.
- Natural Access installed.

Use an uninterruptible power supply (UPS) for increased system reliability. The UPS does not need to power the PC video monitor except in areas prone to severe lightning storms.

CG driver software

The following drivers for operating CG boards are installed with Natural Access software:

Operating system	Driver name
Windows	*cg6kwin2k.sys*
UNIX	*cg6k* *cg6ksw*
Red Hat Linux	*cg6k.o* *cg6ksw.o*

Installation summary

The following table summarizes the procedure for installing the hardware and software components:

Step	Action
1	Ensure that your host chassis meets the system requirements.
2	Power down the system if it is running.
3	If necessary, configure the CG 6060 board for H.100 bus termination.
4	Install the CG 6060 board in a PCI bus slot.
5	Power up the system.
6	If NaturalAccess is not already installed, install Natural Access. This also installs the CG board driver and runtime software. The hardware interface drivers are installed with Natural Access software.
7	For each board, create a board keyword file that details the board configuration settings. Find the closest example file in *\nms\cg\cfg* and use it as a baseline.

Step	Action
8	Create a system configuration file that identifies all boards in the system and their associated board keyword files. For more information, refer to Configuring and starting the system with oamsys and the *Dialogic® NaturalAccess™ OAM System Developer's Manual*.
9	Configure an Ethernet connection. For more information, refer to either Configuring IPv4 Ethernet connections or Configuring IPv6 Ethernet connections.
10	Run *oamsys* to configure the boards as specified in the configuration files.
11	Connect the board interfaces to T1 or E1 trunks (if configured) and Ethernet connections.
12	Verify that your installation is operational.

Configuring the hardware

This topic describes the following procedures for configuring the CG 6060 board:

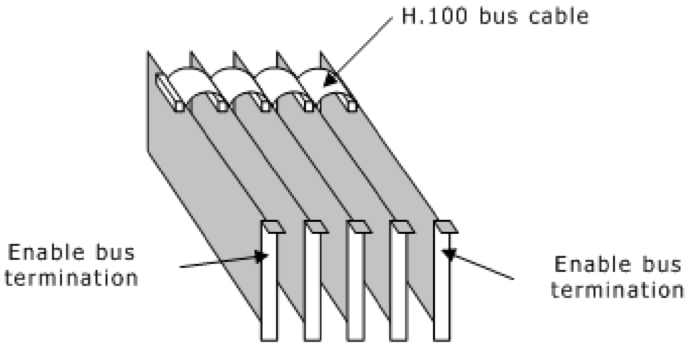
- Configuring H.100 bus termination
- Configuring the DIP switches
- Configuring the T1 or E1 interface
- Configuring hardware echo cancellation

The procedures you follow depend on the CG 6060 board configuration you are installing. Configurations support up to four T1 or E1 digital trunk interfaces and two Ethernet 10/100Base-T interfaces.

Caution:	The CG 6060 board is shipped in a protective anti-static container. Leave the board in its container until you are ready to install it. Handle the board carefully and hold it only by its handles. Wear an anti-static wrist strap connected to a good earth ground whenever you handle the board.

Configuring H.100 bus termination

H.100 boards are connected to one another with an H.100 bus cable. The two boards located at either end of the H.100 bus must have bus termination enabled, as shown in the following illustration. Bus termination is controlled by a DIP switch as explained in Configuring the DIP switches.



Configuring the DIP switches

The CG 6060 DIP switches are located on the board, as shown in the following illustration:

The following table provides a description of the CG 6060 DIP switches:

DIP switch	Description
S1	Set by the manufacturer. Do not change any of the S1 DIP switch settings.
S2	Controls the H.100 bus termination. By default, all S2 switches are set to OFF (H.100 bus termination disabled). Setting all S2 switches to ON enables H.100 bus termination. Set all S2 switches to ON only for the boards that are on the ends of the H.100 bus. **Note:** The switches in the S2 DIP switch must be set to either all ON or all OFF.

Configuring the T1 or E1 interface

To configure the T1 or E1 interface perform the following steps:

Step	Action
1	To disable a trunk interface, set the corresponding NetworkInterface.T1E1[x].Type keyword to NONE. To enable a trunk interface, set the NetworkInterface.T1E1[x].Type keyword in the board keyword file to T1 or E1. You must configure all trunks that are being used as either T1 or E1. Do not specify more than one trunk type per board.
2	Set the NetworkInterface.T1E1[x].Impedance keyword to one of the following values: • DSX1 • G703_75_OHM • G703_120_OHM • HIGH_IMPEDANCE For more information, refer to NetworkInterface.T1E1[x].Impedance.
3	Set the NetworkInterface.T1E1[x].FrameType, NetworkInterface.T1E1[x].LineCode, and NetworkInterface.T1E1[x].SignalingType keywords to values appropriate for your configuration.
4	Ensure that you use the correct I/O cables. Refer to Connecting to an E1 network or Connecting to a T1 network.

Sample T1 trunk configuration

The following example shows a sample T1 configuration for four trunks:

```
NetworkInterface.T1E1[0..3].Type          = T1
NetworkInterface.T1E1[0..3].Impedance     = DSX1
NetworkInterface.T1E1[0..3].LineCode      = B8ZS [other values possible]
NetworkInterface.T1E1[0..3].FrameType     = ESF  [other values possible]
NetworkInterface.T1E1[0..3].SignalingType = CAS  [other values possible]
DSP.C5x[0..47].XLaw                       = MU_LAW
DSPStream.VoiceIdleCode[0..3]             = 0x7F
DSPStream.SignalIdleCode[0..3]            = 0x00
```

Sample E1 trunk configuration

The following example shows a sample E1 configuration for four trunks:

```
NetworkInterface.T1E1[0..3].Type           = E1
NetworkInterface.T1E1[0..3].Impedance      = G703_120_OHM
NetworkInterface.T1E1[0..3].LineCode       = HDB3  [other values possible]
NetworkInterface.T1E1[0..3].FrameType      = CEPT
NetworkInterface.T1E1[0..37].SignalingType = CAS   [other values possible]
DSP.C5x[0..47].XLaw                        = A_LAW
DSPStream.VoiceIdleCode[0..3]              = 0xD5
DSPStream.SignalIdleCode[0..3]             = 0x09
```

Note: The syntax [0..3] specifies that the configuration supports any valid number of trunks within the range of 0 through 3 trunks.

Configuring hardware echo cancellation

Use the HardwareEcho.EchoChipEnabled, HardwareEcho.Trunk[x].OnOffTimeslots, and HardwareEcho.XLaw keywords to set hardware echo cancellation.

When using the hardware echo cancellation capabilities, echo cancellation parameters are fixed. An application cannot change the parameters in the ADI_ECHOCANCEL_PARMS structure with the **adiModifyEchoCanceller** function.

If the application requires flexibility and must modify echo cancellation parameters, use DSP resources to provide software echo cancellation capabilities. For more information, refer to the *ADI Service Developer's Reference Manual*.

Note: Do not operate both hardware echo cancellation and software echo cancellation at the same time on a CG 6060 board.

Installing the board

Warning:	The CG 6060 board powers up and functions only in a PCI chassis that supplies the PCI bus slot with 5.0 V and 3.3 V of power.

Complete the following steps to initially install the CG 6060 board:

Step	Action
1	If necessary, configure the board as described in Configuring the hardware.
2	Power down the chassis and disconnect it from the power source.
3	Remove the cover and set it aside.
4	Arrange the CG 6060 board and other H.100 boards in adjacent PCI bus slots. Make sure each board's PCI bus connector is seated securely in a slot.
5	Secure the end bracket on the board to the chassis.
6	If applicable, connect the H.100 bus cable to the board. If you have multiple H.100 boards, connect the H.100 bus cable to each of the H.100 boards.
7	Replace the cover and connect the computer to its power source.

Connecting to the network

After installing the CG 6060 board, perform the following tasks:

Task	Description
1	Install the CG 6060 software available with Natural Access.
2	Connect the CG 6060 board interfaces to PSTN trunks and Ethernet connections. For more information, refer to the Establishing network connections section, which includes the following topics: • Trunk interfaces • Connecting to a T1 network • Connecting to an E1 network • Testing in loopback mode • Connecting to an Ethernet network

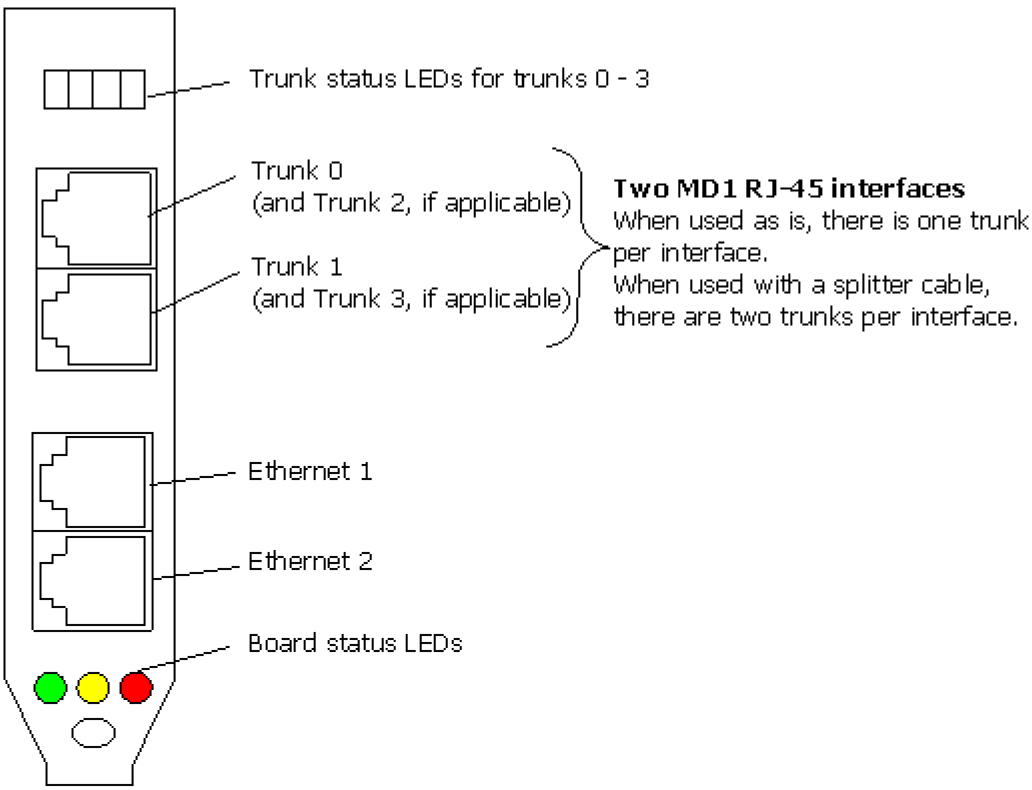
4. Establishing network connections

Trunk interfaces

A CG 6060 board has up to four trunks and provides:

- Four trunk status LEDs
- Two Dialogic® MD1 RJ-45 interfaces
- Two RJ-45 Ethernet connectors
- Three board status LEDs

The following illustration shows a CG 6060 board:



Warning:	**Important safety notes for telephony connections** • Allow only qualified technical personnel to install this board and its associated telephone wiring. • Make sure the host chassis is grounded through the power cord or by other means before connecting the telephone line. • If your system requires an external power supply, make sure it is grounded through the power cord or by other means. • Never install telephone wiring during a lightning storm. • Never install telephone jacks in wet locations. • Telephone companies provide primary lightning protection for their telephone lines. However, if a site connects to private lines that leave the building, make sure that external protection is provided.

For information about the products available for connecting and terminating boards, refer to the *Dialogic® Hardware Connectivity Manual*.

Connecting to a T1 network

Before connecting a CG 6060 board to a T1 network, ensure that you have:

- Specified NetworkInterface.T1E1[x].Type = T1 for the keyword value in the board keyword file.
- Specified NetworkInterface.T1E1[x].Impedance = DSX1 or HIGH_IMPEDANCE for the keyword value in the board keyword file.
- Specified the appropriate keyword values in the board keyword file for the following keywords:

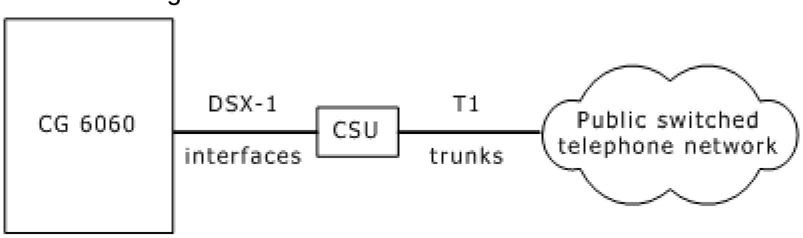
  NetworkInterface.T1E1[x].FrameType
  NetworkInterface.T1E1[x].LineCode
  NetworkInterface.T1E1[x].SignalingType

For more information, refer to Configuring the T1 or E1 interface.

Caution:	You must complete all required performance tests, and a type approval certificate must be granted by the appropriate regulatory authority in the target country before you can connect the CG 6060 board configured as T1 to the public network.

The CG 6060 board has up to four T1 trunk interfaces. For typical T1 communications, each trunk interface connects to a channel service unit (CSU) that is connected to a T1 trunk line. The CSU provides a DSX-1 interface to the T1 line and also contains circuitry that allows the central office (CO) to perform diagnostic tests remotely.

The following illustration shows a CG 6060 trunk interface with a CSU:



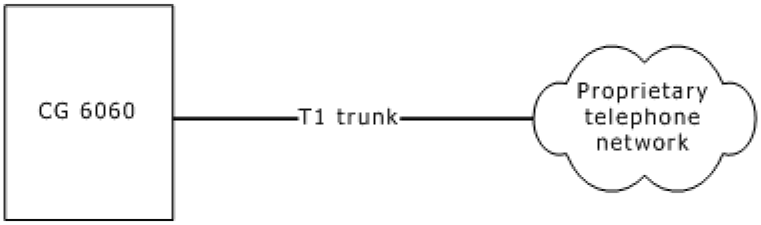
Note: Trunks synchronize when the OAM API boots the board.

You can purchase or lease the CSU from the telephone company or other vendor. The CSU must be compatible with DSX-1 specifications.

Warning:	Important safety note for telephony connections
⚠	The cables attached to this product must be isolated by a channel service unit (CSU) before the cables leave the building.

You can also connect the board directly to the T1 line, without a CSU. This setup is most common in applications where the T1 line is proprietary and is not connected directly to the public network.

The following illustration shows a CG 6060 trunk interface without a CSU:



To avoid causing T1 service provider alarms, make sure that the board always sends a valid signal, either by looping back at the CSU, or by connecting the CSU to a functioning CG 6060 board. The best way to loop back the signal is to unplug the cable from the CSU. The modular connection on most CSUs loops back the transmit signal to the receive signal when nothing is plugged in.

Cable requirements

Refer to www.dialogic.com/declarations/default.htm for information on the cable requirements depending on the target country and the network type.

Connecting to an E1 network

This topic provides procedures for:

- Connecting a CG 6060 to an E1 120 ohm
- Connecting a CG 6060 to an E1 75 ohm

For information about the products available for connecting and terminating boards, refer to the *Dialogic® Hardware Connectivity Manual*.

Before connecting a CG 6060 board to an E1 network, ensure that you have:

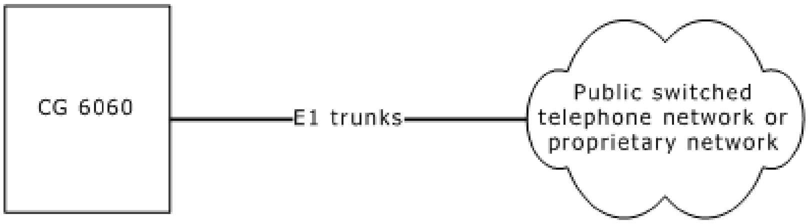
- Specified NetworkInterface.T1E1[x].Type = E1 for the keyword value in the board keyword file.
- Specified NetworkInterface.T1E1[x].Impedance = G703_75_OHM, G703_120_OHM, or HIGH_IMPEDANCE for the keyword value in the board keyword file.
- Specified the appropriate keyword values in the board keyword file for the following keywords:

  NetworkInterface.T1E1[x].FrameType
  NetworkInterface.T1E1[x].LineCode
  NetworkInterface.T1E1[x].SignalingType

For more information, refer to Configuring the T1 or E1 interface.

Caution:	Dialogic obtains board-level approvals certificates for supported countries. Some countries require that you obtain system-level approvals before connecting to the public network. To learn what approvals you require, contact the appropriate regulatory authority in the target country.

The CG 6060 board has four CEPT E1 interfaces. For typical E1 communications, each E1 interface connects directly to an E1 trunk, as shown in the following illustration:



Trunks synchronize when the OAM API boots the board.

Connecting a CG 6060 to an E1 120 ohm

Use a shielded RJ-48 cable to connect a CG 6060 board configured as 120 ohm to an E1 trunk.

If connecting Trunk 2 or Trunk 3, connect a dual T1/E1 120 ohm adapter cable to one or both MD1 RJ-45 interfaces on the board. Connect shielded RJ-48 cables to the trunk connectors on the splitter cables.

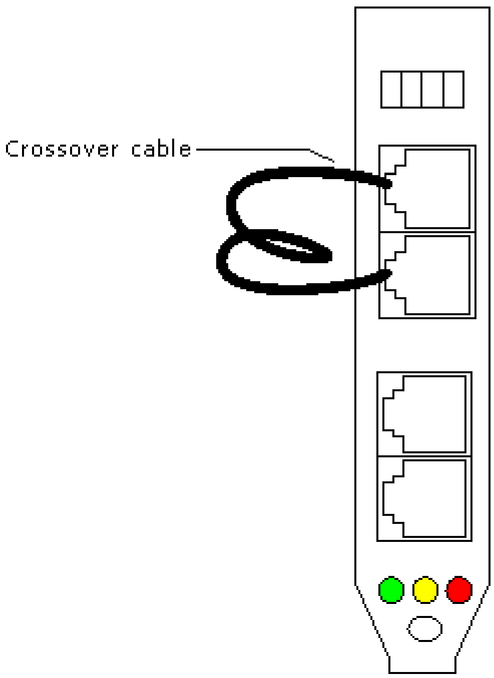
Caution:	Failure to use a shielded cable may negate the Class B approval. Refer to http://www.dialogic.com/declarations/default.htm for more information.

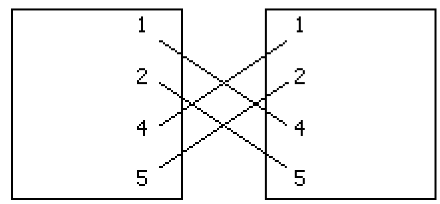
Connecting a CG 6060 to an E1 75 ohm

Use an RJ-48C to 75 ohm BNC adapter cable or a Dialogic® MD1 RJ-45 to 75 ohm BNC adapter cable to connect a CG 6060 board configured as 75 ohm to an E1 trunk.

Testing in loopback mode

To test the digital trunk application without connecting to the telephone network, connect the CG 6060 board in loopback mode. The following illustration shows the loopback configuration using a crossover cable to connect trunk 0 and trunk 1 on a four-trunk CG 6060 board:





Wiring diagram of crossover cable

The cross-over cable connects the transmit signals from one trunk to the receive signals on another trunk as shown.

If your board configuration uses two optional splitter cables to use trunks 2 and 3, you can connect the splitter cables in loopback mode. Use the cross-over cable to connect the transmit signals on one of the splitter cables to receive signals on the second splitter cable.

Connecting to an Ethernet network

To connect a CG 6060 board to an Ethernet network, insert a Category 5 shielded twisted pair (STP) cable into one or both of the board's Ethernet connectors.

To set up IPv4 or IPv6 connections, specify appropriate keyword settings in the board keyword file. For more information, refer to Configuring IPv4 Ethernet connections or Configuring IPv6 Ethernet connections.

5. Configuring the board

Configuring and starting the system with oamsys

To configure and start the boards, specify configuration parameters in the board keyword file for each board. In board keyword files, specify configuration parameters as a keyword name and value pair (for example, Country = USA).

The easiest way to use the board keyword files is to use the *oamsys* utility supplied with the OAM API software. *oamsys* configures and starts boards based on the parameters specified in the system configuration file and the board keyword files. For information about the OAM API utilities, refer to the *Dialogic® NaturalAccess™OAM API System Developer's Manual*.

Applications can use OAM API functions to retrieve and modify configuration parameters. For more information, refer to the *Dialogic® NaturalAccess™OAM API Developer's Manual*.

To configure and start a system using the *oamsys* utility:

Step	Action
1	Install the boards and software as described in the installation summary.
2	Determine which board keyword file to use, or edit one of the sample CG 6060 board keyword files, to specify appropriate configuration information for each board. For more information, refer to Using board keyword files.
3	Determine the CompactPCI bus and slot locations of the boards using the *pciscan* utility. For information about *pciscan*, refer to the *Dialogic® NaturalAccess™OAM API System Developer's Manual*.
4	Create a system configuration file, or edit a sample system configuration file, to point to all the board keyword files for your system. Specify a unique name and board number for each board. For more information, refer to Creating a system configuration file for oamsys.
5	Start *oammon* to monitor the OAM API system and all NaturalAccess boards. For more information about *oammon*, refer to the *Dialogic® NaturalAccess™OAM API System Developer's Manual*. Start *oammon* before running *oamsys*. Keep *oammon* running so that you can see the status of all boards in your system and to view error and tracing messages.
6	Use *oamsys* to start all of the installed boards according to the configuration information specified in the system configuration file and in any associated board keyword files. *ctdaemon* must be running when you use *oamsys*. For more information, refer to Running oamsys.

To determine the physical slot location of a specific board:

Operating system	**Procedure**
Windows	Use *pciscan* to associate the PCI bus assignment to the physical board by flashing an LED on the board. To flash the LED on a board, call *pciscan* with the PCI bus and PCI slot locations. For information about *pciscan*, refer to the *Dialogic® NaturalAccess™OAM API System Developer's Manual*.
UNIX	Use *cg6ktool* to associate the PCI bus assignment to the physical board by flashing an LED on the board. For more information, refer to cg6ktool - Displaying EEPROM and RAM.

Creating a system configuration file for oamsys

The OAM API system configuration files reference all of the boards in your system. System configuration files are typically named *oamsys.cfg* and are located in the *\nms\oam\cfg* directory (*/opt/nms/oam/cfg* for UNIX). When you start *oamsys*, it looks for a system configuration file named *oamsys.cfg*.

The following table describes the board-specific information included in system configuration files:

Keyword	**Description**
[***name***]	Name of the board to be used to refer to the board in the software. The board name must be unique.
Product	Name of the board product: CG_6060 or CG_6060_4.
Number	Board number you use in the Natural Access application to refer to the board.
Bus	PCI bus number. The bus:slot location for each board must be unique. Obtain with *pciscan*.
Slot	PCI slot number. The bus:slot location for each board must be unique. Obtain with *pciscan*.
File	Name of the board keyword file containing settings for the board. Several board keyword files are installed with the CG software.

Refer to the *Dialogic® NaturalAccess™ OAM System Developer's Manual* for specific information about the syntax and structure of the system configuration file.

Sample system configuration file

The following system configuration file provides configuration information for several CG 6060 boards on a particular chassis. Each board has a separate section delimited by a unique user-defined board name (in square brackets).

Modify the Bus and Slot values appropriately for each board to match your chassis configuration. If necessary, add new entries for additional boards.

```
[CG6060PCI]
  Product = CG_6060
  Number  = 0
  Bus     = 1
  Slot    = 7

File = c6060nocc.cfg


#------------------------------------------------------
# Uncomment the following section to boot another board
#-------------------------------------------------------
#[BoardName1]
#  Product = CG_6060
#  Number = 1
#  Bus    = 2
#  Slot   = 14
#  File   = c6060nocc.cfg
```

Running oamsys

To run *oamsys*, enter the following command from the command line:

```
oamsys -f filename
```

where ***filename*** is the name of an OAM system configuration file.

If you invoke *oamsys* without command line options, OAM searches for a file named *oamsys.cfg* in the paths specified in the AGLOAD environment variable.

When you invoke *oamsys* with a valid file name, *oamsys* performs the following tasks:

- Checks the syntax of the system configuration file to make sure that all required keywords are present. *oamsys* discards any unrecognized keywords and reports any syntax errors it finds.

  oamsys verifies the file syntax of the system configuration file, but not of board keyword files.

- Checks for the uniqueness of board names, board numbers, bus numbers, and slot numbers.
- Shuts down all boards recognized by the NaturalAccess OAM Software (if any).
- Deletes all board configuration information currently maintained for the recognized boards (if any).
- Sets up the OAM database and creates all managed objects as described in the system configuration file.
- Attempts to start all boards according to configuration parameters specified in the OAM system configuration file and the board keyword files it references.

The NaturalAccess Server (*ctdaemon*) must be running for *oamsys* to operate. For more information about the NaturalAccess Server, refer to the *Dialogic® NaturalAccess™ Software Developer's Manual*.

Using board keyword files

A board keyword file contains a list of parameters to configure a board. The board keyword file for each board is assigned to the board in a system configuration file. When *oamsys* runs, it creates a record for each board in the OAM database and stores the parameters of the board. It then starts the board and configures it as described in the database.

Refer to the Sample board keyword file. For more information about board keyword files, refer to the *Dialogic® NaturalAccess™ OAM System Developer's Manual.*

Changing configuration parameters

When you run *oamsys*, the OAM API reads parameters provided in the board keyword files. OAM then starts all boards according to these parameters.

You can change board keyword parameters in the following ways:

- Duplicate the sample board keyword file appropriate for your country and board type, modify the new file, specify the name of the new file in the File statement of the *oamsys.cfg* file, and run *oamsys* again. For an example, refer to the sample board keyword file. For information about the syntax used in OAM API board keyword files, refer to the *Dialogic® NaturalAccess™ OAM System Developer's Manual*.
- Create a new board keyword file, either with additional keywords or with keywords whose values override earlier settings.
- Specify parameter settings directly using the *oamcfg* utility. For more information about this utility, refer to the *Dialogic® NaturalAccess™ OAM System Developer's Manual*.
- Specify the settings using OAM service functions. For more information, refer to the *Dialogic® NaturalAccess™ OAM API Developer's Manual*.

For example, this manual describes:

- Configuring the T1 or E1 interface.
- Change which software module files are downloaded to the board at startup. Refer to Specifying configuration file locations for more information.
- Configuring board clocking.
- Connecting to an Ethernet network.
- CG 6060 switch models.

Specifying configuration file locations

Some board keywords require file names as parameters. If the file name keyword value contains a path specification, OAM searches for the file in the specified directory. If the file does not exist in the specified path or if the parameter does not specify a path, OAM searches the current working directory and then the *load_directory* defined by the AGLOAD environment variable.

Configuring board clocking

Note: If you are not using PSTN trunks and if you are not using the CT bus, set Clocking.HBus.ClockMode = STANDALONE, Clocking.HBus.ClockSource = OSC and skip this topic.

When multiple boards are connected to the CT bus, you must set up a bus clock source to synchronize timing between them. In addition, you can configure alternative (or fallback) clock sources to provide the clock signal if the primary source fails.

This topic describes:

- Clocking capabilities
- Clocking configurations

To create a robust clocking configuration, you must understand basic clocking concepts such as clock mastering and fallback. This topic assumes that you have a basic understanding of clocking. For a complete overview of board clocking, refer to the *Dialogic® NaturalAccess™OAM API System Developer's Manual*.

CG 6060 clocking capabilities

This topic describes the rules and limitations that apply to setting up CT bus clocking on CG 6060 boards.

When a CG 6060 board is configured as the system primary clock master:

- The board's first timing reference must be set to a network trunk, a NETREF clock, or OSC.
- The board's fallback timing reference must be set to a network trunk, a NETREF reference, or OSC. Fallback to OSC is not recommended because the transition can cause slave boards to fall back to the secondary clock and create an out-of-sync condition.

When a CG 6060 board is configured as the system secondary clock master:

- The board's first timing reference must be the system's primary clock.
- The board's fallback timing reference must be set to one of the network trunks, a NETREF source, or OSC.

When a CG 6060 board is configured as a clock slave:

- The board's first timing reference must be the system's primary clock.
- The board's fallback timing reference must be the system's secondary clock.

Refer to Other clocking capabilities for more options.

The following tables summarize the CT bus clocking capabilities of the CG 6060 board:

Clocking capabilities as primary master

Capability	Yes/No	Comments
Serve as primary master	Yes	
Drive A_CLOCK	Yes	
Drive B_CLOCK	Yes	
Available primary timing references:		
Local trunk	Yes	The secondary timing reference must also be a local trunk.
NETREF1	Yes	The application must reconfigure the board as soon as possible if NETREF1 fails.
NETREF2	No	This board does not support NETREF2.
OSC	Yes	

Capability	Yes/No	Comments
Fallback to secondary timing reference	Yes	
Available secondary timing references:		
Local trunk	Yes	This is the only valid reference if the primary timing reference is a local trunk.
NETREF1	Yes	
NETREF2	No	This board does not support NETREF2.
OSC	No	
Slave to secondary master if both references fail	Yes	

Clocking capabilities as secondary master

Capability	Yes/No	Comments
Serve as secondary master	Yes	
Drive A_CLOCK	Yes	If the primary master drives B_CLOCK, the secondary master drives A_CLOCK.
Drive B_CLOCK	Yes	If the primary master drives A_CLOCK, the secondary master drives B_CLOCK.
Available secondary timing references:		
Local trunk	Yes	
NETREF1	Yes	
NETREF2	No	This board does not support NETREF2.
OSC	Yes	

Clocking capabilities as slave

Capability	Yes/No	Comments
Serve as slave	Yes	
Slave to A_CLOCK	Yes	
Slave to B_CLOCK	Yes	
Available fallback timing references:		
A_CLOCK	Yes	
B_CLOCK	Yes	
OSC	Yes	The board is synchronized when the application reconfigures the clock.

Other clocking capabilities

Capability	Yes/No	Comments
Drive NETREF1	Yes	
Drive NETREF2	No	This board does not support NETREF2.
Operate in standalone mode	Yes	

Configuring clocking

You can configure board clocking in your system in one of two ways:

Method	Description
Using *clockdemo* application model	Create an application that assigns each board its clocking mode, monitors clocking changes, and reconfigures clocking if clock fallback occurs. A sample clocking application, *clockdemo*, is provided with Natural Access. *clockdemo* provides a robust fallback scheme that suits most system configurations. *clockdemo* source code is included, allowing you to modify the program if your clocking configuration is complex. For more information about *clockdemo*, refer to the *Dialogic® NaturalAccess™ OAM System Developer's Manual*. **Note:** Most clocking applications (including *clockdemo*) require all boards on the CT bus to be started in standalone mode.

Method	Description
Using board keywords (with or without application intervention)	For each board on the CT bus, set the board keywords to determine the board's clocking mode and to determine how each board behaves if clock fallback occurs. This method is documented in this topic. Unlike the *clockdemo* application, which allows you to specify several boards to take over mastery of the clock from one another in a fallback situation, the board keyword method allows you to specify only a single secondary master. For this reason, the board keyword method is best used to implement clock fallback in your system or in test configurations where clock reliability is not a factor. The board keyword method does not create an autonomous clock timing environment. If you implement clock fallback using this method, an application must still intervene when clock fallback occurs to reset system clocking before other clocking changes occur. If both the primary and secondary clock masters stop driving the clocks, and an application does not intervene, the boards default to standalone mode.

Choose only one of these configuration methods across all boards on the CT bus. Otherwise, the two methods interfere with one another, and board clocking may not operate properly.

Configuring CG 6060 board clocking using keywords

CG 6060 board keywords enable you to configure the board in the following ways:

- System primary clock master
- System secondary clock master
- Clock slave
- Standalone board

You can also use board keywords to establish clock fallback sources. Refer to the multiple board system example for a sample configuration.

The following tables describe how to use board keywords to specify the clocking role of each CG 6060 board in a system.

Configuring the CG 6060 as primary clock master

Use the following board keywords to configure the CG 6060 as a primary clock master:

Keyword	Description
Clocking.HBus.ClockSource	Specifies the source from which this board derives its timing. Set this keyword to a network source (NETREF or NETWORK).

Keyword	Description
Clocking.HBus.ClockSourceNetwork	(Optional) Specifies the trunk number that the board uses as an external network clocking source for its internal clock. **Note:** Trunk numbering, in this case, is one-based.
Clocking.HBus.ClockMode	Specifies the CT bus clock that the board drives. Set this keyword to either A_CLOCK (MASTER_A) or B_CLOCK (MASTER_B).
Clocking.HBus.AutoFallBack	Enables or disables clock fallback on the board.
Clocking.HBus.FallBackClockSource	Specifies an alternate timing reference to use when the master clock source fails. Set this keyword to a network timing source (NETREF or NETWORK).
Clocking.HBus.FallBackNetwork	(Optional) Specifies the trunk from which a fallback network timing source (for the clock fallback reference) can be derived when Clocking.HBus.FallBackClockSource = NETWORK. **Note:** Trunk numbering, in this case, is one-based.

If the primary master's first source fails and then returns, the board's timing reference (and consequently, the reference for any slaves) switches back to the first timing source. This is not true for the secondary clock master.

Configuring the CG 6060 as secondary clock master

Use the following board keywords to configure the CG 6060 as a secondary clock master:

Keyword	Description
Clocking.HBus.ClockSource	Specifies the source from which this board derives its timing. Set this keyword to the clock driven by the primary clock master. For example, if the primary master drives A_CLOCK, set this keyword to A_CLOCK.
Clocking.HBus.ClockMode	Specifies the CT bus clock that the secondary master drives. Set this keyword to the clock not driven by the primary clock master (MASTER_A or MASTER_B).
Clocking.HBus.AutoFallBack	Enables or disables clock fallback on the board. Set this keyword to YES.
Clocking.HBus.FallBackClockSource	Specifies an alternate timing reference to use when the master clock does not function properly. Set this keyword to a network source (NETREF or NETWORK).

Keyword	Description
Clocking.HBus.FallBackNetwork	(Optional) Specifies the trunk from which a fallback network timing source (for the clock fallback reference) can be derived. **Note:** Trunk numbering, in this case, is one-based.

If the primary master's timing reference recovers, the secondary master continues to drive the clock referenced by all clock slaves in the system until the application intervenes.

Configuring the CG 6060 as a clock slave

Use the following board keywords to configure the CG 6060 as a clock slave:

Keyword	Description
Clocking.HBus.ClockMode	Specifies the CT bus clock from which the board derives its timing. Set this keyword to SLAVE to indicate that the board does not drive any CT bus clock (although the board can still drive NETREF).
Clocking.HBus.ClockSource	Specifies the source from which this clock derives its timing. Set this keyword to the clock driven by the primary clock master (A_CLOCK or B_CLOCK).
Clocking.HBus.AutoFallBack	Enables or disables clock fallback on the board.
Clocking.HBus.FallBackClockSource	Specifies the alternate clock reference to use when the master clock does not function properly. Set this keyword to the clock driven by the secondary clock master (B_CLOCK or A_CLOCK).

Configuring the CG 6060 as a standalone board

To configure a CG 6060 board in standalone mode so the board references its own clocking information, set Clocking.HBus.ClockMode to STANDALONE. The board can use either its own oscillator or a signal received from a digital trunk as a timing signal reference. However, the board cannot make switch connections to the CT bus.

Multiple board system example

The following example assumes a system configuration in which four CG 6060 boards reside in a single chassis. The boards are configured in the following way using keywords:

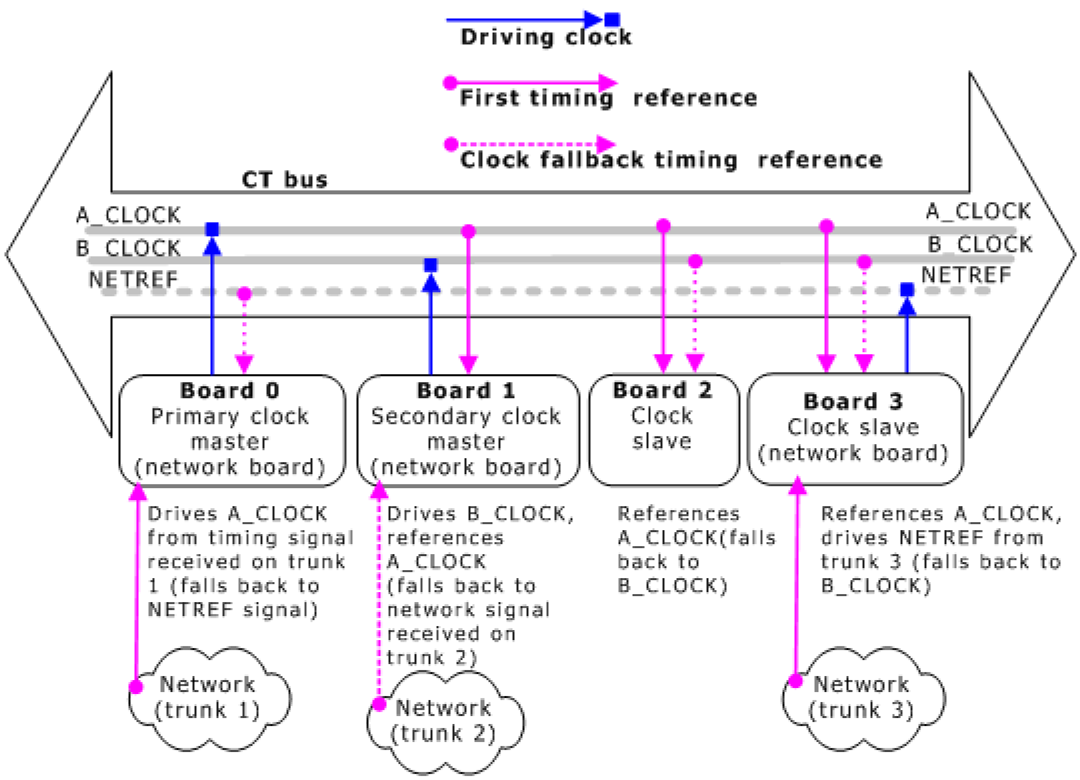
Board	Configuration
0	System primary bus master (driving A_CLOCK)
1	System secondary bus master (driving B_CLOCK)
2	Clock slave (clock fallback enabled)

Board	Configuration
3	Clock slave (clock fallback enabled - drives the NETREF clock)

This configuration assigns the following clocking priorities:

Priority	Timing reference
First	Board 0, digital trunk 1. A network signal from a digital trunk provides the primary master clock source.
Second	Board 3, digital trunk 3. The NETREF signal driven by a digital trunk (on Board 3) acts as the primary master clock fallback source.
Third	Board 1, digital trunk 2. A network signal from a digital trunk provides the secondary master clock fallback source.

The following illustration shows an example of a multiple-board system with a primary and secondary clock master:



The following table shows keywords used to configure the multiple boards according to the configuration shown in the preceding illustration.

Board	Role	Clocking keyword settings
0	Primary clock master	Clocking.HBus.ClockMode = MASTER_A Clocking.HBus.ClockSource = NETWORK Clocking.HBus.ClockSourceNetwork = 1 Clocking.HBus.AutoFallBack = YES Clocking.HBus.FallBackClockSource = NETREF Clocking.HBus.NetRefSpeed = 8K
1	Secondary clock master	Clocking.HBus.ClockMode = MASTER_B Clocking.HBus.ClockSource = A_CLOCK Clocking.HBus.AutoFallBack = YES Clocking.HBus.FallBackClockSource = NETWORK Clocking.HBus.FallBackNetwork = 2
2	Clock slave	Clocking.HBus.ClockMode = SLAVE Clocking.HBus.ClockSource = A_CLOCK Clocking.HBus.AutoFallBack = YES Clocking.HBus.FallBackClockSource = B_CLOCK
3	Slave driving NETREF	Clocking.HBus.ClockMode = SLAVE Clocking.HBus.ClockSource = A_CLOCK Clocking.HBus.AutoFallBack = YES Clocking.HBus.FallBackClockSource = B_CLOCK Clocking.HBus.NetRefSource = NETWORK Clocking.HBus.NetRefSourceNetwork = 3 Clocking.HBus.NetRefSpeed = 8K

In this configuration, Board 0 is the primary clock master, and it drives A_CLOCK. All slave boards on the system use A_CLOCK as their first timing reference. Board 0 references its timing from a network timing signal received on its own trunk 1. Board 0 also uses the NETREF signal (driven based on the digital signal received on trunk 3 of Board 3) as its clock fallback source. If the network timing signal derived from its own digital trunks fails, Board 0 continues to drive A_CLOCK based on NETREF timing reference.

If, however, both of the clocking signals used by Board 0 (the network timing signal and the NETREF signal) fail, Board 0 stops driving A_CLOCK. The secondary clock master (Board 1) then falls back to a timing reference received on its own trunk 3, and uses this signal to drive B_CLOCK. B_CLOCK then becomes the timing source for all boards that use B_CLOCK as their backup timing reference.

For this clock fallback scheme to work, all clock slaves must specify A_CLOCK as the clock source and B_CLOCK as the clock fallback source.

Managing board DSP resources

This topic describes:

- Setting up a single resource pool
- Setting up multiple resource pools
- Using multiple resource pools

The CG 6060 board provides a flexible resource management scheme to allow you to reserve DSP resources at board boot time to ensure deterministic behavior under load. Resources are reserved in one or more pools. Each pool contains a number of DSPs loaded with a set of identical functions, and a number of logical DSP ports running on those DSPs. Each port within a pool is capable of running any of the loaded functions.

You must choose between using a single resource pool or multiple resource pools. Choose multiple resource pools under the following conditions:

- You have two sets of very different functions running on the board (for example, VoIP functions and IVR play/record functions) and you cannot achieve the required port density with a single pool.
- Because of switch blocking limitations, you need to place certain ports on certain physical DSPs.

Refer to DSP resource management keywords for more information.

Setting up a single resource pool

In many cases, a single resource pool is all that is required. With a single pool, all ports on the board have the same capability, and each port uses a physical DSP core chosen by the board. The following example is a board keyword file that uses the resource management keywords in a single pool:

```
Resource[0].Name          = RSC1
Resource[0].Size          = 120
Resource[0].TCPs          = nocc
Resource[0].Definitions   = ( dtmf.det_all & echo.ln20_apt25 &\
                              ptf.det_2f & tone.gen & \
                              callp.gnc & ptf.det_4f & \
                              ((voice.rec_32 & (voice.play_32_100 | voice.play_32_150 | \
                              voice.play_32_200)) | \
                              (gsm_ms.frgsm_rec & gsm_ms.frgsm_play) | \
                              g726.rec_32 | g726.play_32) )
```

Other than setting up these keywords, there is nothing special an application needs to use a single pool. All ports are taken from this pool, and their physical DSPs are chosen automatically.

Setting up multiple resource pools

If you need to configure multiple resource pools, define the pools in the board keyword file and take steps at the application level to use those pools.

The following code sample shows a board keyword file that uses multiple resource pools. The first pool (POOL_A) specifies 120 ports (starting at timeslot 0) of the GSM vocoder and MF signaling, and places this pool on the 15 lowest numbered DSPs. The second pool (POOL_B) specifies 120 ports (starting at timeslot 120) of the G.726 vocoder and MF signaling, and places this pool on the 15 higher numbered DSPs.

```
Resource[0].Name             = POOL_A
Resource[0].Size             = 120
Resource[0].StartTimeSlot    = 0
Resource[0].TCPs             = nocc mfc0
Resource[0].DSPs             = 2 4 5 6 8 9 10 11 12 13 14 15 16 17
Resource[0].Definitions      = ( dtmf.det_all & echo.ln20_apt25 \
                             & ptf.det_2f & tone.gen & \
                             (gsm_ms.frgsm_rec & gsm_ms.frgsm_play))


Resource[1].Name              = POOL_B
Resource[1].Size              = 120
Resource[1].StartTimeSlot     = 120
Resource[1].TCPs              = mfc0 nocc
Resource[1].DSPs              =  3 18 19 20 21 22 23 24 25 26 27 28 29 30 31
Resource[1].Definitions       = ( dtmf.det_all & echo.ln20_apt25 \
                              & ptf.det_2f & tone.gen & \
                              ((voice.rec_32 & (voice.play_32_100 |  \
                              voice.play_32_150 | voice.play_32_200)) | \
                              g726.rec_32 | g726.play_32) )
```

The Resource[x].StartTimeSlot keyword associates each of the pools with a set of logical timeslots that can be used by Natural Access. The pool is assigned to Resource[x].Size number of timeslots, starting with timeslot Resource[x].StartTimeSlot. Logical timeslots associated with a particular pool must be consecutive, and the timeslot ranges for each pool must not overlap.

Using multiple resource pools

Since resource pools are bound to sets of logical timeslots, the pools can be chosen when opening a Natural Access port with the **ctaOpenServices** function. It is the application's responsibility to manage logical timeslot usage. The logical timeslot is specified in the CTA_MVIP_ADDR structure inside the CTA_SERVICE_DESC passed into **ctaOpenServices**. The ADI service must be one of the services opened.

DSP resources are obtained from the resource pool associated with a timeslot when the application calls **ctaOpenServices** to open a port.

Sample board keyword file

The following sample board keyword file for no call control (NOCC) is provided with Natural Access software:

```
#
#     c6060nocc.cfg
#     CG 6060 configuration file
#
#     This file configures the board to run Voice with NOCC in E1
#

Clocking.HBus.ClockMode                          = STANDALONE
Clocking.HBus.ClockSource                        = OSC

# DSP.C5x[x].Os a MUST
```

```
DSP.C5x[0..95].Os                                = dspos6u
#--------------------------
#  NOTE: T1 configuration
#--------------------------
#NetworkInterface.T1E1[0..3].Type                = T1
#NetworkInterface.T1E1[0..3].Impedance           = DSX1
#NetworkInterface.T1E1[0..3].LineCode            = B8ZS
#NetworkInterface.T1E1[0..3].FrameType           = ESF
#NetworkInterface.T1E1[0..3].SignalingType       = RAW
#DSP.C5x[0..95].Libs                             = cg6klibu
#DSP.C5x[0..95].XLaw                             = MU_LAW
#--------------------------
#  NOTE: E1 configuration
#--------------------------
NetworkInterface.T1E1[0..3].Type                 = E1
NetworkInterface.T1E1[0..3].Impedance            = G703_120_OHM
NetworkInterface.T1E1[0..3].LineCode             = HDB3
NetworkInterface.T1E1[0..3].FrameType            = CEPT
NetworkInterface.T1E1[0..3].SignalingType        = RAW
DSP.C5x[0..95].Libs                              = cg6kliba
DSP.C5x[0..95].XLaw                              = A_LAW
#--------------------------
# Hardware Echo Cancellation
# NOTE: it is in bypass by default
# NOTE: uncomment the following two keyword lines to enable and set the XLaw accordingly
#--------------------------
# HardwareEcho.EchoChipEnabled = YES
# HardwareEcho.XLaw = A_LAW
#--------------------------
# Resource management
#--------------------------
Resource[0].Name                                 = RSC1
Resource[0].Size                                 = 120
Resource[0].TCPs                                 = nocc
Resource[0].StartTimeSlot                        = 0
 ################################################################
# Before modifying this resource definition string refer to the CG6060
# Installation and Developers Manual.
# NOTE: echo.ln20_apt25 - echo running on DSP has been removed
#       from resource definitions.  We recommend user to use
#       the hardware echo chip for echo cancellation instead
#################################################################
Resource[0].Definitions         = ( dtmf.det_all & ptf.det_2f & tone.gen & \
callp.gnc & ptf.det_4f & \
( (rvoice.rec_mulaw & rvoice.play_mulaw) | \
(rvoice.rec_alaw & rvoice.play_alaw) | \
(rvoice.rec_lin & rvoice.play_lin) | \
(voice.rec_16 & (voice.play_16_100 | voice.play_16_150 | voice.play_16_200)) | \
(voice.rec_24 & (voice.play_24_100 | voice.play_24_150 | voice.play_24_200)) | \
(voice.rec_32 & (voice.play_32_100 | voice.play_32_150 | voice.play_32_200)) | \
(voice.rec_64 & (voice.play_64_100 | voice.play_64_150 | voice.play_64_200)) | \
(wave.rec_11_16b & wave.play_11_16b) | \
(wave.rec_11_8b & wave.play_11_8b) | \
(oki.rec_24 & (oki.play_24_100 | oki.play_24_150 | oki.play_24_200)) | \
(oki.rec_32 & (oki.play_32_100 | oki.play_32_150 | oki.play_32_200)) | \
(ima.rec_24 & ima.play_24) | \
(ima.rec_32 & ima.play_32) | \
(gsm_ms.frgsm_rec & gsm_ms.frgsm_play) | \
g726.rec_32 | g726.play_32) )
# NOTE: If the DSP cores listed below do not exist on the board, the DSP cores will
# be ignored and will not be booted or used
Resource[0].Dsps = 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 \
            24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 \
            48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 \
            72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95
DebugMask                                        = 0x0
```

6. Configuring Ethernet interfaces

Configuring IPv4 Ethernet connections

The CG 6060 board has two 10/100Base-T Ethernet connections that can be configured in several ways. This topic provides the following information about configuring IPv4 Ethernet interfaces:

- Using IPv4 Ethernet interface keywords
- Setting up the IPv4 Ethernet connections

Using IPv4 Ethernet interface keywords

Use the following board keywords to configure the CG 6060 Ethernet interfaces:

Keyword	Description
IPC.AddRoute[x].DestinationAddress	Specifies the IPv4 address of an Ethernet interface. You can specify up to 32 destination addresses.
IPC.AddRoute[x].GatewayAddress	Specifies the IPv4 address of the network router.
IPC.AddRoute[x].Interface	Specifies the number (1 or 2) of the Ethernet interface you are configuring.
IPC.AddRoute[x].Mask	Specifies the subnet mask for the IPv4 address specified in IPC.AddRoute[***x***].DestinationAddress.
IPC.AddRoute[x].VlanTag	Specifies a VLAN tag to be added to all packets sent to the IPv4 subnet specified by IPC.AddRoute[***x***].DestinationAddress and IPC.AddRoute[***x***].Mask.

Note: For these keywords, ***x*** represents an entry in the routing table.

CG 6060 Ethernet interfaces must be configured for Fusion systems. For more information about Fusion software, configurations, and programming models, refer to the *Fusion Developer's Manual*.

Setting up the IPv4 Ethernet connections

Use the IPC.AddRoute keywords to set the IPv4 addressing and static IPv4 routing table information for the CG 6060 board. You can configure up to 32 separate routing table entries for the CG 6060 board.

Depending upon the desired mode of operation, you can configure each Ethernet interface on the CG 6060 board with its own IP addressing information. To do this, specify the IPv4 address, the IPv4 subnet mask, and the particular Ethernet interface.

IP addresses

To specify the IPv4 address of an Ethernet interface on the CG 6060 board, define the following keywords:

Keyword	Description
IPC.AddRoute[***x***].DestinationAddress	IPv4 address of Ethernet interface.
IPC.AddRoute[***x***].Mask	Subnet mask for IPv4 destination address.
IPC.AddRoute[***x***].Interface	Ethernet interface number (1 or 2).
IPC.AddRoute[***x***].VlanTag	VLAN tag (optional).

The combination of IPC.AddRoute[***x***].Mask and IPC.AddRoute[***x***].DestinationAddress defines a subnet. Multiple different subnets can be configured on the same interface.

VLAN configuration

If a VLAN tag is specified, then all packets destined for the subnet will be tagged with a VLAN (IEEE 802.1Q Virtual LAN) ID. This does not affect handling of incoming packets; the board ignores the VLAN tag on incoming packets.

Static IP routes

In addition, the CG 6060 board allows you to configure multiple static IPv4 routes by specifying the addresses of routers on the subnet. Once configured, the static IP routes manage the transfer of packets between the IPv4 subnet associated with the CG 6060 board and the IP network. The IP stack on the CG 6060 board uses standard IPv4 routing algorithms to determine how to route outbound packets.

To specify a static IPv4 route that the CG 6060 board IP stack uses, define the following keywords:

Keyword	Description
IPC.AddRoute[***x***].DestinationAddress	IPv4 address of Ethernet interface.
IPC.AddRoute[***x***].Mask	Subnet mask for IPv4 destination address.
IPC.AddRoute[***x***].GatewayAddress	IPv4 address of router.

Example

The following example shows how to use IPC.Addroute statements to specify the board's IPv4 address, subnet mask, and gateway IPv4 address:

```
#CG 6060 Board IP Address, subnet mask, and gateway IPv4 address.
IPC.AddRoute[0].DestinationAddress = 10.102.64.151
IPC.AddRoute[0].Mask               = 255.255.255.0
IPC.AddRoute[0].Interface          = 1

#Gateway IP Address, subnet mask, and gateway IPv4 address.
IPC.AddRoute[1].DestinationAddress = 0.0.0.0
IPC.AddRoute[1].Mask               = 0.0.0.0
IPC.AddRoute[1].GatewayAddress     = 10.102.64.1
```

In this example, the first three IPC entries specify the IPv4 address and mask of the CG 6060 board. The second three entries configure the address of the gateway.

The IPv4 addressing and gateway configuration information for each CG 6060 board resides in the board keyword file. Every time you reboot the CG 6060 board with *oamsys*, *oamsys* reconfigures the IPv4 addressing information for the specified board.

The *cgroute* utility provides an alternative way to configure specific IPv4 addressing information without editing the CG 6060 board keyword file. *cgroute* is similar to the standard *route* utility found on most systems with IP processing capabilities. *cgroute* allows you to add, delete, and display routing information from the CG 6060 board. For more information, refer to cgroute - Setting up CG board IPv4 routing tables.

Configuring IPv6 Ethernet connections

This topic provides the following information about IPv6 Ethernet interfaces:

- IPv6 Ethernet interface keywords
- IPv6 addresses and routing
- IPv6 and neighbor discovery
- IP security and IPv6
- IPv6 path redundancy
- Example configuration
- IPv6 standards

IPv6 Ethernet interface keywords

Use the following board keywords to configure the CG board Ethernet interfaces for IPv6:

Keyword	Description
IPv6.Link[x].Enable	Enables or disables IPv6 on the specified Ethernet interface.
IPv6.Link[x].IPSec	Enables or disables IPSec for IPv6 on the specified Ethernet interface.
IPv6.Link[x].MTU	Specifies the IPv6 maximum transmission unit (MTU) for the Ethernet interface.
IPv6.Link[x].HopLimit	Specifies the default IPv6 hop limit value (that is, the number of routers through which a datagram will travel) for the Ethernet interface.
IPv6.Link[x].EnablePing	Enables or disables IPv6 PING on the specified Ethernet interface.
IPv6.Link[x].ICMPRateLimit	Specifies the IPv6 ICMP rate limit (that is, the maximum amount of ICMP error messages per second that can be sent) for the Ethernet interface.

Keyword	Description
IPv6.Link[x].NDAttempts	Specifies the neighbor discovery attempt (NDA) limit for the Ethernet interface.
IPv6.Link[x].NDRetranTimer	Specifies the neighbor discovery re-transmission timer for the Ethernet interface in milliseconds.
IPv6.Link[x].NDReachabilityTimer	Specifies the neighbor discovery reachability timer duration for the Ethernet interface in milliseconds.

IPv6 addresses and routing

Use the IPv6.Link[x].Enable keyword to enable IPv6 on a particular Ethernet interface. Unlike IPv4, IPv6 addressing and routing information are not explicitly configured. The IPv6 addresses and routing information are automatically configured using the Stateless Address Autoconfiguration protocols and procedures as specified in RFC 2461, RFC 2462, and RFC 2464.

This address autoconfiguration procedure is initiated for each Ethernet interface independently. The default setting for IPv6.Link[x].Enable is NO, meaning IPv6 is disabled by default.

When enabled, the CG board IPv6 stack automatically configures itself with the following IPv6 addresses:

Address	Definition
Link-local unicast	FE80::EUI-64
Link local scope all nodes multicast address	FF02::1
Each unicast address	Solicited node multicast address
Loopback address	1
Multiple site local or global unicast addresses	Added based on the contents of any router advertisements received. These addresses take the following form: `prefix/64:EUI-64.`

Refer to RFC 2373 and RFC 2464 for more information about EUI-64 addresses.

IPv6 and neighbor discovery

The neighbor discovery protocol as defined in RFC 2461 manages the interactions between different nodes by exchanging messages that enable hosts to communicate with each other and implement autoconfiguration. Use the IPv6.Link[x].NDAttempts, IPv6.Link[x].NDRetranTimer, and IPv6.Link[x].NDReachabilityTimer keywords to configure the neighbor discovery protocol.

Neighbor discovery uses ICMPv6 as its base protocol and replaces ARP, ICMPv4 Router Discovery, and ICMPv4 Redirect. In addition, the neighbor discovery protocol explicitly defines mechanisms for determining neighbor reachability on an ongoing basis.

Neighbor discovery keywords configure the following settings:

Keyword	Description
IPv6.Link[x].NDAttempts	Configures the number of neighbor solicitations sent to a particular neighbor address prior to determining that the neighbor is unreachable.
IPv6.Link[x].NDRetranTimer	Configures the amount of time in milliseconds between re-transmission of neighbor solicitations when a corresponding neighbor advertisement has not been received.
IPv6.Link[x].NDReachabilityTimer	Configures the amount of time in milliseconds between reverifications that a particular neighbor is reachable.

IP security and IPv6

Use the IPv6.Link[x].IPSec keyword to enable or disable IP Security (IPSec) for IPv6 on a particular Ethernet interface. You can enable or disable IPSec independently for each Ethernet interface. The default setting is NO, so that IPSec is disabled by default. There is a minor performance impact on the system when IPSec is enabled.

For more information about implementing IPSec on CG boards, refer to cgsetkey - Configuring IPv6 security keys and policies.

IPv6 path redundancy

The IPv6 neighbor discovery protocol provides a mechanism for discovering faults in the network between a source system and either another link local system or a router into the larger IPv6 network. The fault detection extends beyond the directly-connected Ethernet cable and includes all network components between the source and its exit point to the global IPv6 network. When using the CG board IPv6 stack, you can configure the board to implement this type of path redundancy to supplement the single link redundancy capabilities built into the board’s IPv4 stack.

The CG 6060 IPv4 stack can detect link failures between the board Ethernet port and its directly connected link partner (typically an Ethernet switch), but not component failures that occur elsewhere on the network (for example, link failures between the Ethernet switch and either another Ethernet switch or router). You can configure the CG board IPv6 stack to use the IPv6 neighbor discovery protocol to determine whether or not it can reach each link's local IPv6 destination. Regardless of where a component failure occurs, the board can notify the application of any link failures, and the application can take corrective actions based on this information.

Path redundancy and Fusion

To implement path redundancy for CG board IPv6 Ethernet interfaces, you must enable the passage of Fusion route availability events. These events notify the application when the Ethernet interface associated with a particular endpoint experiences a change of status.

Based on the information provided by the event, the application can then change the network path associated with the Ethernet interface. Applications enable this feature on an endpoint-by-endpoint basis when creating Fusion RTP and UDP endpoints. For more information about using Fusion route availability events, refer to the *Fusion Developer's Manual*.

Example configuration

The following example shows IPv6.Link keywords that configure two CG board IPv6 Ethernet interfaces:

```
########################################
#  Enables both Ethernet interfaces for IPv6
########################################
IPv6.Link[0].Enable = YES
IPv6.Link[0].IPSec  = NO
IPv6.Link[0].MTU    = 1500
IPv6.Link[0].HopLimit = 64
IPv6.Link[0].EnablePing = YES
IPv6.Link[0].ICMPRateLimit = 100
IPv6.Link[0].NDAttempts = 3
IPv6.Link[0].NDRetranTimer = 1000
IPv6.Link[0].NDReachabilityTImer = 30000

IPv6.Link[1].Enable = YES
IPv6.Link[1].IPSec  = NO
IPv6.Link[1].MTU    = 1500
IPv6.Link[1].HopLimit = 128
IPv6.Link[1].EnablePing = YES
IPv6.Link[1].ICMPRateLimit = 100
IPv6.Link[1].NDAttempts = 3
IPv6.Link[1].NDRetranTimer = 1000
IPv6.Link[1].NDReachabilityTImer = 30000
```

For more information about implementing IPv6 functionality on CG boards, refer to the Fusion documentation.

IPv6 standards

The following table lists some of the standards from the IETF that are relevant to IPv6:

Document	Title
RFC 2460	*Internet Protocol, Version 6 (IPv6) Specification*
RFC 2373	*IP Version 6 Addressing Architecture*
RFC 2463	*Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6)*
RFC 2401	*Security Architecture for the Internet Protocol*
RFC 2461	*Neighbor Discovery for IP Version 6 (IPv6)*
RFC 2462	*IPv6 Stateless Address Autoconfiguration*
RFC 2464	*A Method for Transmission of IPv6 Packets over Ethernet Networks*

Running in IPv4/IPv6 dual stack mode

CG boards implement an IPv4/IPv6 dual stack that allows applications to configure the board in any of the following modes:

IP stack mode	Description
IPv4 only	Default mode of operation for CG boards. You configure specific IPv4 addresses and routing information with the IPC.AddRoute keywords.
IPv6 only	Enable by setting the IPv6.Link[x].Enable keyword to YES for a particular Ethernet interface. The IPv4 stack remains passive if no IPv4 addresses are configured with IPC.AddRoute keywords.
IPv4/IPv6 dual stack	Add IPv4 addresses with IPC.AddRoute keywords, and enable IPv6 capability on a particular interface with the IPv6.Link[x].Enable keyword.

You can configure the board Ethernet links in any of the following ways:

- Configure the Ethernet link separately for IPv4 and IPv6 support.
- Configure separate protocols for separate Ethernet links.
- Configure both protocols for either one or both links.

The only interaction between the IPv4 and IPv6 protocol stacks occurs when a CG board is configured in redundant Ethernet mode.

The CG board IPv6 stack does not support redundant Ethernet configurations in the manner supported by the IPv4 stack. To implement IPv4 redundant Ethernet capabilities while using IPv6, you must enable the IPv6 stack only on the first Ethernet interface. Enabling IPv6 on the second Ethernet link places the Ethernet interfaces in dual Ethernet mode rather than redundant Ethernet mode.

Setting up multi-homed configurations

On CG boards, each Ethernet interface can be configured to a different IP subnet, and associated with a separate default router. This type of configuration is called a multi-homed configuration. Applications can use the Fusion MSPP service to direct the flow of data to the separate IP subnets associated with multi-homed configurations.

The MSPP service enables applications to create MSPP endpoints that act as transmission and reception points for data at the CG board's network interfaces. Applications can join endpoints together with MSPP channels that perform processing tasks with voice or fax data as it moves from endpoint to endpoint. For more information about Fusion MSPP endpoints and channels, refer to the *Fusion Developer's Manual*.

In multi-homed configurations running Fusion gateways, applications can use the MSPP service to set endpoint address parameters (for RTP and UDP endpoints) that control the inbound demultiplexing and outbound routing behavior of the CG board IP stack. On an endpoint-by-endpoint basis, applications set the data transfer characteristics that apply to RTP or UDP sessions by setting the source IP addresses (specified through endpoint address parameters) of the RTP or UDP endpoints associated with these sessions. In this context, endpoint source IP addresses and destination IP addresses are oriented from the perspective of the CG board when transmitting data. That is, the source IP address is the IP address and port number of the local CG board, while the destination IP address is the IP address and port number of a remote system.

This topic provides the following examples of how applications can set up multi-homed configurations:

- Load balancing in dual subnet configurations
- UDP port numbers in multi-homed configurations

RFC 1122 describes the requirements for multi-homed end systems (ES). It outlines two models for accomplishing this, the strong ES model and the weak ES model. For more information about the strong and weak ES models, refer to RFC 1122.

Load balancing in dual subnet configurations

In multi-homed, multi-router configurations, applications can balance the amount of data transferred through each Ethernet interface by using the Fusion MSPP service to specify which Ethernet interface an RTP or UDP endpoint uses to transmit data. Applications specify which Ethernet interface to use by setting the endpoint source IP addresses to match the IP address assigned to one of the CG board's Ethernet interfaces.

In the following example, the CG board's OAM board keyword file assigns IP addresses and subnet masks for each of the board's Ethernet interfaces, and defines default routes for these interfaces:

```
/* Ethernet #1: IP Address 198.62.139.27, Subnet 255.255.255.0 */
IPC.AddRoute[1].Interface          = 1
IPC.AddRoute[1].DestinationAddress = 198.62.139.27
IPC.AddRoute[1].Mask               = 255.255.255.0

/* Ethernet #2: IP Address 139.37.200.43, Subnet 255.255.255.0 */
IPC.AddRoute[2].Interface          = 2
IPC.AddRoute[2].DestinationAddress = 139.37.200.43
IPC.AddRoute[2].Mask               = 139.37.200.43

/* Default Route #1: 0.0.0.0/0.0.0.0 Router IP Address: 198.62.139.1 */
IPC.AddRoute[3].DestinationAddress = 0.0.0.0
IPC.AddRoute[3].Mask               = 0.0.0.0
IPC.AddRoute[3].GatewayAddress     = 198.62.139.1

/* Default Route #2: 0.0.0.0/0.0.0.0 Router IP Address: 139.37.200.1 */
IPC.AddRoute[4].DestinationAddress = 0.0.0.0
IPC.AddRoute[4].Mask               = 0.0.0.0
IPC.AddRoute[4].GatewayAddress     = 139.37.200.1
```

In this case, an application can implement load balancing by creating MSPP service endpoints in the following way:

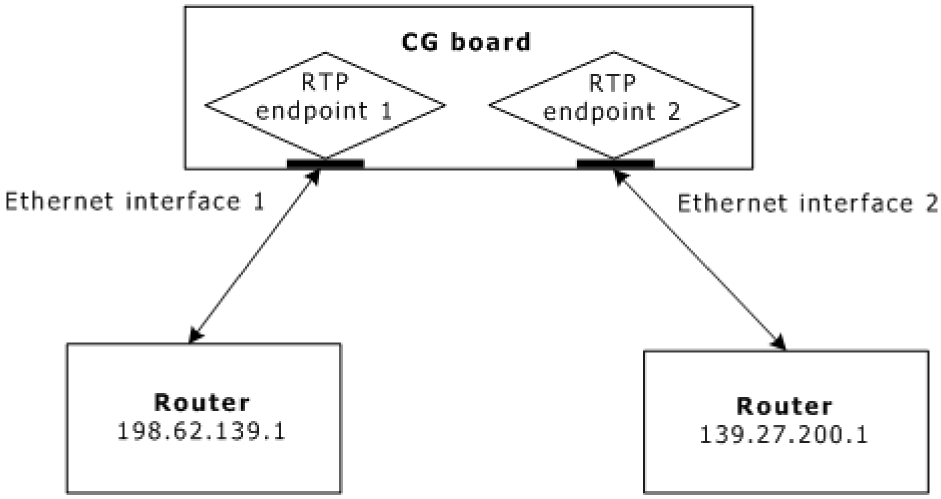
- Creating an endpoint (RTP endpoint 1) and specifying the following endpoint's source IP address:

  198.62.139.27

- Creating an endpoint (RTP endpoint 2) and specifying the following endpoint's source IP address:

  139.37.200.43

The following illustration shows the relationship between the RTP endpoints and the CG board's Ethernet interface:



If the CG board is configured in redundant Ethernet mode, or if it does not matter which CG board Ethernet interface the application uses for transferring packet data, the application can set the endpoint source address parameter to 0.0.0.0. In this case, the CG board directs the outbound packets to the first valid route in its IP routing table.

If the application specifies an invalid SourceIpAddress parameter, the CG board defaults to standard IP routing and sends the outbound packets to the first valid route found in the CG board's IP routing table.

UDP port numbers in multi-homed configurations

In a multi-homed IP environment, the application can treat the UDP port number range as a single port number domain, a multiple port number domain qualified by the local IP address, or a combination of both. When the application creates RTP and UDP endpoints, the way the application sets endpoint source IP addresses determines whether the UDP port number uses a single port number domain or multiple port number domain.

If the application sets the endpoint's source IP address to 0.0.0.0, the IP address of any inbound RTP or UDP packets is not used to qualify the UDP port number domain.

If the application sets the endpoint's source IP address to an IP address corresponding to one of the CG board's Ethernet interfaces, then the destination IP address of any incoming packets is used to further qualify the RTP or UDP session to which the packet is bound.

Configuring the board in redundant Ethernet mode

By default, the Ethernet subsystem on the CG 6060 board initializes in redundant Ethernet mode. In this mode, Ethernet 1 provides the primary Ethernet connection and Ethernet 2 operates in a standby mode.

All IPv4 configuration and routing information applies to both the primary and secondary Ethernet connection. If the primary connection loses link integrity, the secondary connection takes over. Once link integrity returns to the primary connection, all Ethernet traffic converts back to the primary Ethernet connection.

While in standby mode, the secondary Ethernet connection establishes link integrity, but remains passive. It does not send or receive packets to or from the IP network. When the primary connection loses link integrity, the secondary connection enables its transmitter and receiver and takes over for the primary connection. The fallback process is automatic, occurring in less than 1 ms, and is transparent to both the network and the application.

If you explicitly configure the secondary Ethernet connection with any IP addressing information or enable IPv6 on the secondary Ethernet connection, you disable the board's redundant Ethernet capability.

Example

The following example shows how to configure a CG 6060 board in redundant Ethernet mode. This example shows a CG 6060 board on a Class C subnet (198.62.139.***x***) with a single router providing access to the external IP network.

When you specify 0.0.0.0 as the router destination address and subnet mask, all IP addresses not on the local subnet (198.62.139.***x***) are forwarded to the router 198.62.139.1 (typically referred to as the default route).

```
IPC.AddRoute[0].DestinationAddress = 198.62.139.32
IPC.AddRoute[0].Mask               = 255.255.255.0
IPC.AddRoute[0].Interface          = 1

IPC.AddRoute[1].DestinationAddress = 0.0.0.0
IPC.AddRoute[1].Mask               = 0.0.0.0
```

IPv6 connections support a type of path redundancy not supported on IPv4 connections. For information about setting up path redundancy in configurations that support IPv6 connections, refer to IPv6 path redundancy.

Configuring the board in dual subnet mode

To direct different types or classes of IP traffic to separate IP networks, you can associate each CG 6060 board Ethernet interface with a separate IP address. When you configure the secondary Ethernet connection with an IPv4 address or enable IPv6 on the secondary Ethernet connection, the board operates in dual subnet mode rather than redundant Ethernet mode.

It is possible to configure both Ethernet connections into the same IPv4 subnet, but this is not recommended. When configured in this manner, the CG 6060 board receives packets for both addresses. Based on standard IPv4 routing practice, outbound packets take the first route that matches. Therefore, the CG board sends all outbound IPv4 packets to the Ethernet connection associated with the first IPv4 address in the routing table.

Example

The following example shows how to configure a CG 6060 board in dual subnet mode. Each Ethernet interface is configured for a separate Class C subnet, and each specifies a separate router. However, the second router is configured so that only IPv4 addresses in the Class A subnet of 10.x.y.z are forwarded to the second router. All other external IPv4 addresses are forwarded to the first router.

```
#  Specify IPv4 address
IPC.AddRoute[0].DestinationAddress = 198.62.139.32
IPC.AddRoute[0].Mask               = 255.255.255.0
IPC.AddRoute[0].Interface          = 1

#  Specify route
IPC.AddRoute[1].DestinationAddress = 0.0.0.0
IPC.AddRoute[1].Mask               = 0.0.0.0
IPC.AddRoute[1].GatewayAddress     = 198.62.139.1

#  Specify IPv4 address
IPC.AddRoute[2].DestinationAddress = 198.62.140.75
IPC.AddRoute[2].Mask               = 255.255.255.0
IPC.AddRoute[2].Interface          = 2

#  Specify route
IPC.AddRoute[3].DestinationAddress = 10.0.0.0
IPC.AddRoute[3].Mask               = 255.0.0.0
IPC.AddRoute[3].GatewayAddress     = 198.62.140.1
```

Monitoring Ethernet link status events

The *oammon* utility displays Ethernet link status events. The OAM API also provides a way for applications to monitor board level events associated with CG board Ethernet interfaces. To monitor these events, the application must register with the OAM API, then use the NaturalAccess function **ctaWaitEvent** to retrieve the appropriate OAM API events.

ctaWaitEvent returns event information that describes what event occurred on what context. The buffer field of the OAM API event provides a pointer to an OAM_MSG structure that provides the following information:

```
typedef struct oam_msg_tag
{
    DWORD dwMsgLen;     // Msg length, including appended name & msg strings
    DWORD dwCode;       // Msg event code (use OAMEVN_xxx)
    DWORD dwSeverity;   // Msg severity
    DWORD dwOfsSzName;  // Offset to name string of source managed object
    DWORD dwOfsSzMessage; // Offset to text msg string
    DWORD dwValue;      // Possible additional event-specific data
                        // string data is appended here
} OAM_MSG;
```

When state transitions occur at the CG board's Ethernet interfaces (that is, when one of the Ethernet interfaces goes out of service or returns to service), the dwCode field in this structure contains an OAMEVN_ALERT message code, and the dwValue field returns one of the following values:

dwValue	Description
0x121B	The CG board's Ethernet link 1 has gone out of service.
0x121D	The CG board's Ethernet link 1 has returned to service.
0x121C	The CG board's Ethernet link 2 has gone out of service.

dwValue	Description
0x121E	The CG board's Ethernet link 2 has returned to service.

For information about processing OAM API events, refer to the *Dialogic® NaturalAccess™ OAM API Developer's Manual*.

7. Verifying the installation

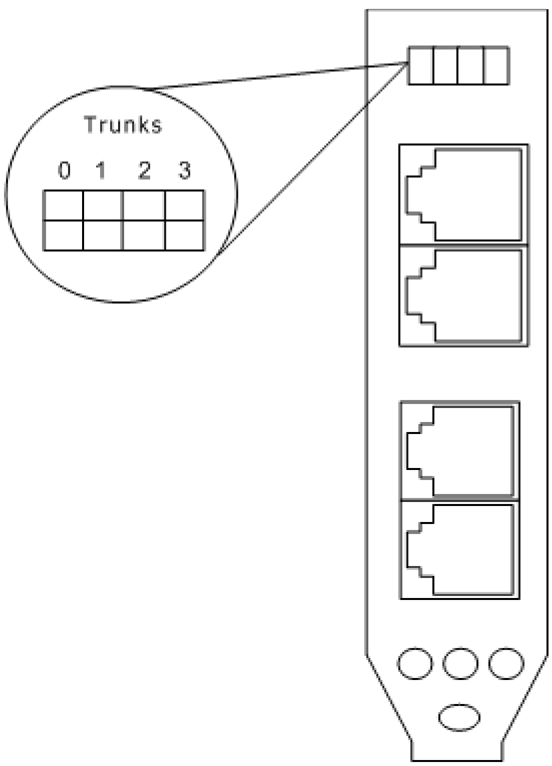
Status indicator LEDs

The CG 6060 board contains the following types of LEDs:

- Trunk LEDs
- Ethernet LEDs
- Board status LEDs

Trunk LEDs

The CG 6060 board has two-color LED indicators (yellow and red) to indicate the status of each trunk. The following table illustrates the location of the trunk status LEDs on the end bracket of the board and describes what each color indicates:

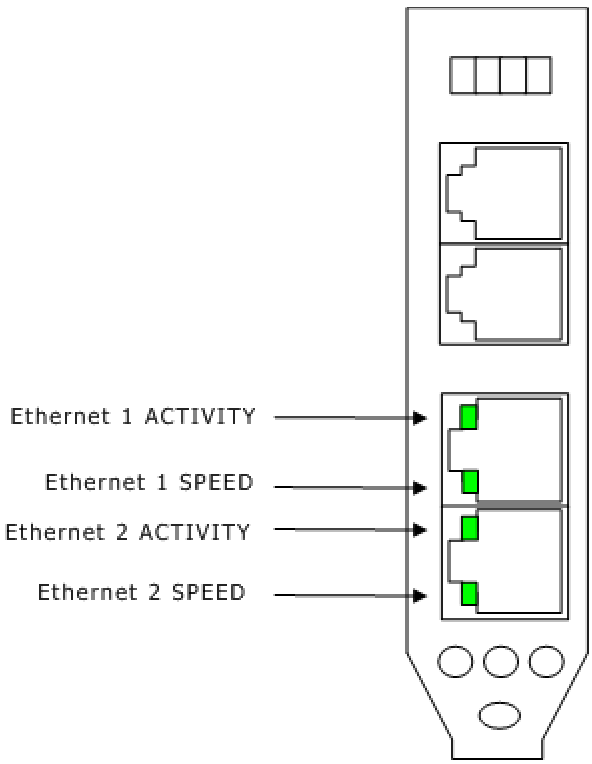
Trunk status LEDs	Color	Description
	Red	Loss of frame, loss of signal, or bit error rate.
	Yellow	Remote loss of frame or remote loss of signaling multiframe.
	Off	Trunk is not configured, or the trunk is not in alarm.

If the NetworkInterface.T1E1[x].Type keyword is set to NONE, no trunk LEDs are illuminated.

For more specific diagnostic information about the current state of trunk synchronization, run the Natural Access trunk monitoring utility *trunkmon*. Refer to the *Dialogic® NaturalAccess™ OAM System Developer's Manual* for more information about *trunkmon*.

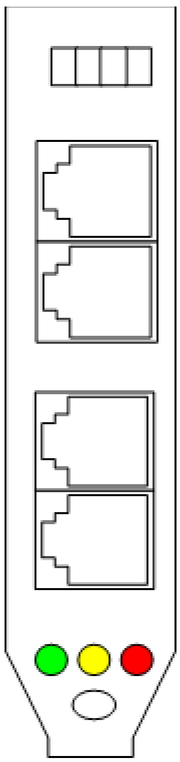
Ethernet LEDs

The CG 6060 board provides two LEDs to indicate the status of each Ethernet interface. The following table illustrates the location of the Ethernet LEDs and describes what the lights indicate:

<table>
<tr><th>Ethernet LEDs</th><th>Label</th><th>Description</th></tr>
<tr><td rowspan="2">
</td><td>ACTIVITY</td><td>There is activity on the Ethernet link. When the Ethernet has established link integrity, and there is transmit or receive activity on the link, the LED flickers on.</td></tr>
<tr><td>SPEED</td><td>Rate at which data is being transmitted. The LED is used only when a reliable Ethernet connection has been established and the ACTIVITY LED is on.
<table>
<tr><th>Data rate</th><th>LED</th></tr>
<tr><td>10 Mb</td><td>Off</td></tr>
<tr><td>100 Mb</td><td>On</td></tr>
</table></td></tr>
</table>

Board status LEDs

The CG 6060 board provides three LEDs that are used to indicate the status of the board. The following table illustrates the location of the board status LEDs and describes what the colors indicate:

Board status LEDs	LED	Description
	Green only, blinking slowly	Board booted successfully and is running.
	Red only	Fatal software error.
	Red and yellow	Board is resetting (transient).
	Red, yellow, and green	Board resets are released (transient).
	Yellow and green	Boot loader has started.
	Not lit	Power is off or the hardware is starting up.

Verifying the board installation

Complete the following steps to verify that you have installed the board correctly:

Step	Action
1	Create a board keyword file to boot the CG 6060 board by copying or editing one of the sample board keyword files to match your specific configuration. Refer to Using board keyword files for more information.
2	Use the *pciscan* utility to determine the bus and slot number. Refer to the *Dialogic® NaturalAccess™ OAM System Developer's Manual* for information about *pciscan*.

Step	Action
3	Edit the system configuration file, *oamsys.cfg*, to reflect the board locations in your system. You can use the *oamgen* utility (included with the OAM API software) to create a sample system configuration file for your system. The system configuration file created by *oamgen* may not be appropriate for your configuration. You may need to make further modifications to the file before running *oamsys* to configure your boards based on the file. For information about *oamgen*, refer to the *Dialogic® NaturalAccess™ OAM System Developer's Manual*.
4	Run *oammon* to monitor the status of all boards. The BootDiagnosticLevel keyword in the board keyword file determines the type of board diagnostic tests that take place when you boot the board. If a test fails, the test number is reported as an error code. You must be running *oammon* to view diagnostic results. For information about board level error messages, refer to the *Dialogic® NaturalAccess™ Board and Driver Error Reference.*
5	Use the *oamsys* command to boot the board.

Verifying trunk connections

Complete the following steps to verify that the board is working correctly:

Step	Action
1	Set the following keyword values in the board keyword file: Clocking.HBus.ClockSource = OSC Clocking.HBus.ClockMode = STANDALONE
2	Use the *oamsys* command to boot the board.
3	Run the digital trunk monitor utility, *trunkmon*. This utility monitors alarms and gathers performance statistics for T1 and E1 trunks. A red trunk LED indicates an alarm state when *trunkmon* detects a local or remote loss of frame or excessive bit errors. If no T1/E1 trunk cables are connected to the CG 6060 board, *trunkmon* shows a loss of frame and an alarm state on all trunks. The red alarm LED on the end bracket lights up for all trunks. Refer to the *Dialogic® NaturalAccess™ OAM System Developer's Manual* for information about *trunkmon*.

Step	Action
4	Connect a cross-over cable between any two trunks of the CG 6060 board. The Frame Sync status changes to OK and the red/yellow LEDs go out. The remote alarm (yellow) LEDs light up to show that the trunk is indicating an alarm state to the other side. About 15 seconds (for T1 trunks, immediately for E1 trunks) after frame synchronization has been acquired, both trunks leave the alarm state. *trunkmon* indicates NONE for the alarm status and the yellow/red alarm LEDs go out.

Demonstration programs

The following demonstration programs are provided with Natural Access and can be used to verify that the CG 6060 board is operating correctly:

Program	Description
ctatest	Demonstrates Natural Access functions.
incta	Demonstrates handling inbound calls.
outcta	Demonstrates establishing outbound calls.
prt2prt	Demonstrates call transfer from an incoming line to an outgoing line and uses the Switching service to make connections and to send patterns.
vceplay	Demonstrates using the Voice Message service to play messages in voice files.
vcerec	Records one or more messages to a voice file.

Note: Executables for *incta*, *outcta*, and *prt2prt* are in the respective sub-directories under *\nms\ctaccess\demos* for Windows and */opt/nms/ctaccess/*demos for UNIX.

Running these demonstration programs requires a connection to either a live T1/E1 trunk or a connection to T1/E1 test equipment that supports call generation and voice path testing. You can use the T1/E1 cross-over cable to loop back one trunk to another trunk. One trunk can then receive calls placed on the other trunk.

To run these demonstration programs on the CG 6060 board, specify the MVIP-95 timeslot number of the local DSP resource on which to run the program.

For example, on a CG 6060 board configured as an E1 CAS board and booted with default switch connections, the DSP resources on stream 64, timeslots 0..29 are connected to the first trunk. Timeslots 30..59 are connected to the second trunk, and so on. Assume that the board number is 0.

To run *ctatest* on the first channel of the first E1 trunk, enter the following command:

```
ctatest -s0 -b 0
```

To run *ctatest* on the first channel of the second E1 trunk, enter the following command:

```
ctatest -s30 -b 0
```

8. CG 6060 switching

Switch blocking

The CG 6060 board:

- Can simultaneously connect (simplex) to all 4096 timeslots on the H.100 bus.
- Does not support switching signaling from trunk-to-trunk or from trunk-to-bus. Signaling must terminate on the board.

If a connection is made to or from a CT bus timeslot, any existing connection in the other direction on that timeslot is disconnected.

Signaling streams cannot be switched to the H.100 bus. They are hard wired to the framer.

CG 6060 switch models

The CG 6060 supports the following switch models:

- Channel associated signaling (CAS)
- Primary Rate Interface (PRI)
- RAW

To define a switch model for CG 6060 boards configured for T1/E1, use the NetworkInterface.T1E1[x].SignalingType keyword.

CAS mode switching

This topic contains the following CAS mode switching information:

- CAS switching limitations
- CAS mode switch model
- H.100 and local streams
- Voice and signaling information routing on T1 trunks
- Voice and signaling information routing on E1 trunks
- Default connections

CAS switching limitations

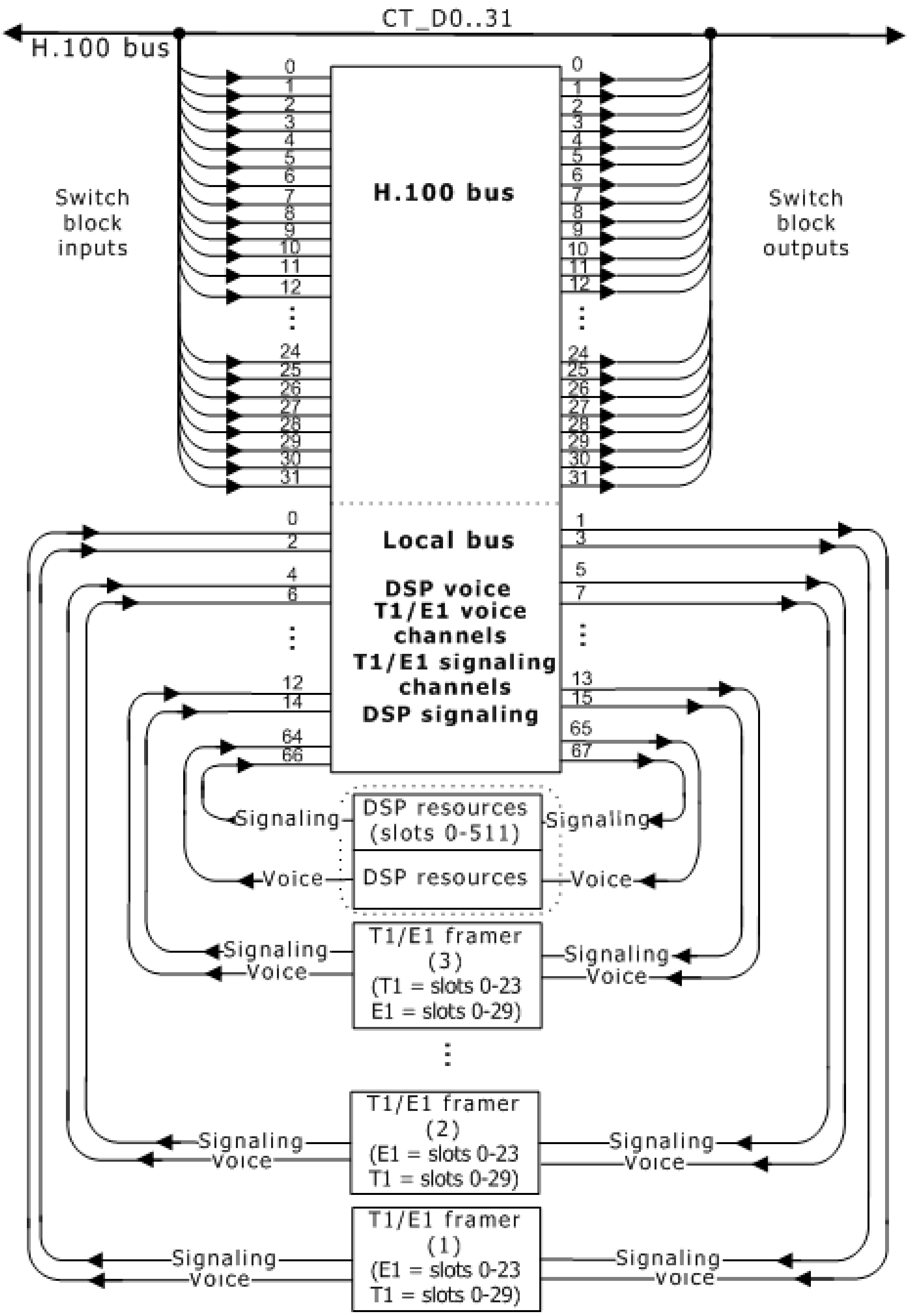
CG 6060 boards terminate CAS signaling on local T1 and E1 trunks. In CAS mode, CG 6060 boards support signaling streams on the DSPs and the framers. These streams are provided for backward compatibility with applications that switch signaling streams.

The CG 6060 switch model supports full duplex connections between DSP signaling and trunk signaling. However, no other CAS signaling connections are supported by the CG 6060 board.

CG 6060 boards do not support DSP-to-DSP signaling connections (such as local stream:timeslot 66:0 to local stream:timeslot 67:4), trunk-to-trunk signaling connections (such as local stream:timeslot 2:0 to local stream:timeslot 7:3), or signaling-to-bus connections (such as local stream:timeslot 2:0 to MVIP stream:timeslot 0:0 or local stream:timeslot 66:0 to MVIP stream:timeslot 0:0).

CAS mode switch model

The following illustration shows the CG 6060 switching model in CAS mode:



H.100 and local streams

The following tables list the specific use of each stream in the CG 6060 CAS switching model:

H.100 streams

H.100 bus	Streams 0..31, timeslots 0..127 (Streams clocked at 8 MHz)

Local streams

Trunk voice information (T1 trunks)	Trunk 1: Streams 0 and 1 Trunk 2: Streams 4 and 5 Trunk 3: Streams 8 and 9 Trunk 4: Streams 12 and 13	timeslots 0..23 timeslots 0..23 timeslots 0..23 timeslots 0..23
Trunk voice information (E1 trunks)	Trunk 1: Streams 0 and 1 Trunk 2: Streams 4 and 5 Trunk 3: Streams 8 and 9 Trunk 4: Streams 12 and 13	timeslots 0..29 timeslots 0..29 timeslots 0..29 timeslots 0..29
Trunk signaling information (T1 trunks)	Trunk 1: Streams 2 and 3 Trunk 2: Streams 6 and 7 Trunk 3: Streams10 and 11 Trunk 4: Streams 14 and 15	timeslots 0..23 timeslots 0..23 timeslots 0..23 timeslots 0..23
Trunk signaling information (E1 trunks)	Trunk 1: Streams 2 and 3 Trunk 2: Streams 6 and 7 Trunk 3: Streams 10 and 11 Trunk 4: Streams 14 and 15	timeslots 0..29 timeslots 0..29 timeslots 0..29 timeslots 0..29
DSP voice information	Streams 64 and 65, timeslots 0.. up to 899	
DSP signaling information	Streams 66 and 67, timeslots 0..511	

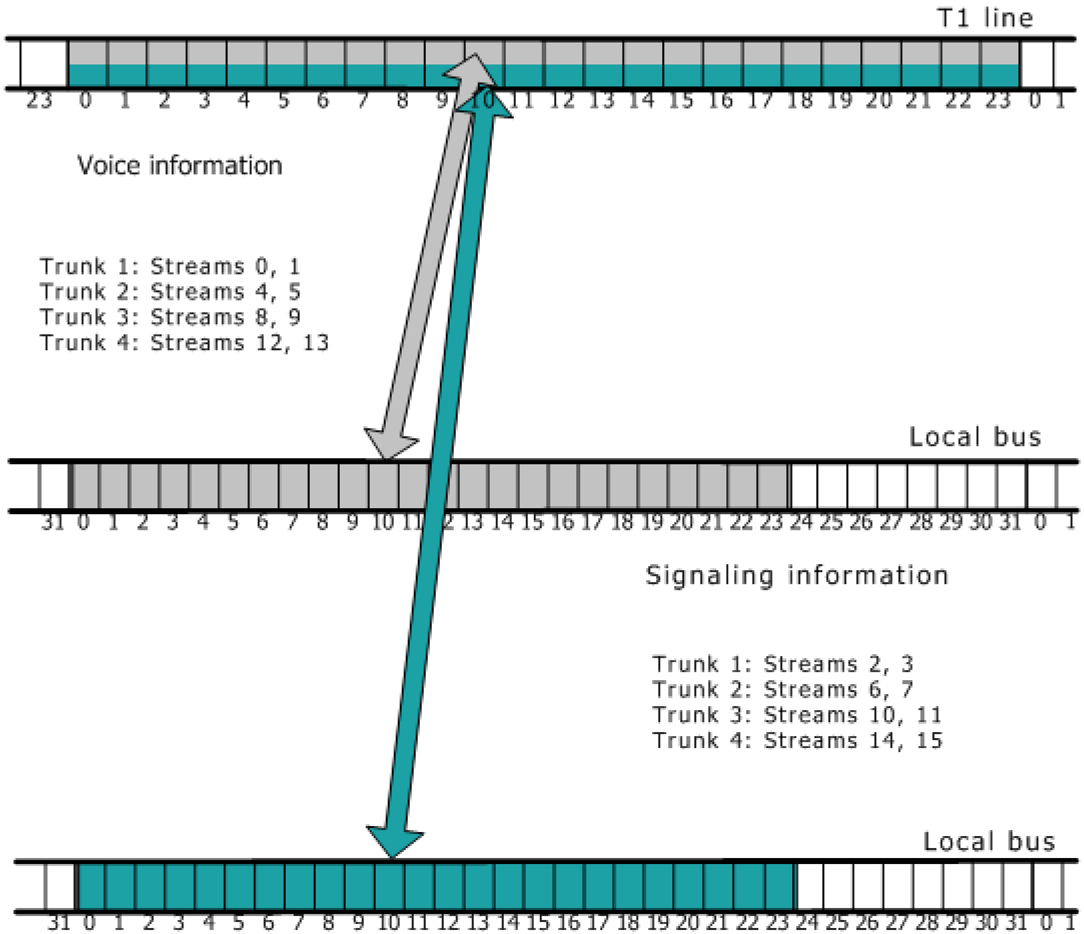
Voice and signaling information routing on T1 trunks (CAS mode)

If NetworkInterface.T1E1[x].SignalingType = CAS (the default setting), voice and signaling information is routed to accommodate a T1 channel associated signaling configuration. Voice information is transmitted in each channel on the T1 trunk and each channel is placed in a corresponding timeslot on the local bus.

Signaling information is transmitted in each channel using robbed-bit signaling. The signaling information is broken out and placed on the corresponding signaling stream for that trunk. The signaling information for a given channel is placed in the same timeslot number as the voice information for that channel.

Note: The CG 6060 board does not allow signaling streams to be connected to the CT bus.

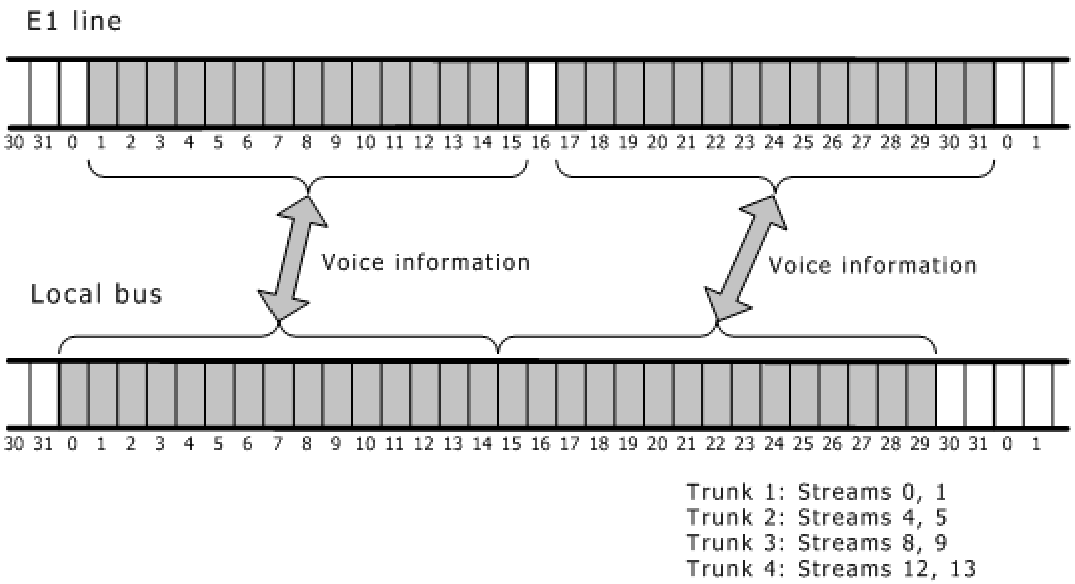
The following illustration shows how data is assigned to timeslots on a T1 trunk:



Voice and signaling information routing on E1 trunks (CAS mode)

Regardless of the NetworkInterface.T1E1[x].SignalingType setting, the CG 6060 board routes voice information by assigning E1 timeslots 1 through 15 to the local bus timeslots 0..14. E1 timeslots 17 through 31 are assigned to the local bus timeslots 15..29. Timeslot 0 on the E1 line carries framing data.

The following illustration shows how voice channel data is assigned to timeslots:

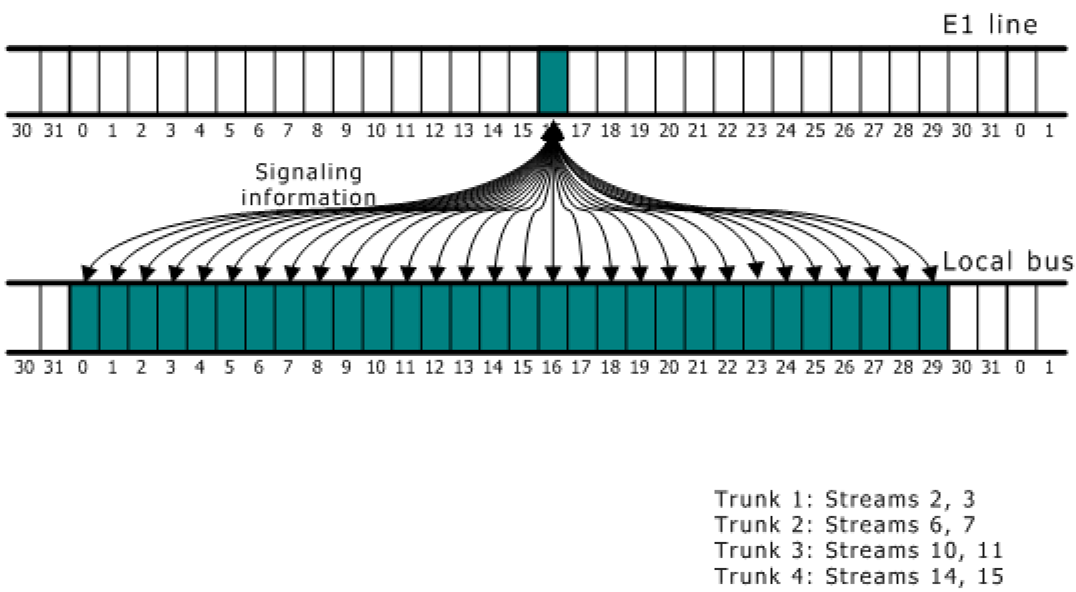


If NetworkInterface.T1E1[x].SignalingType = CAS (the default setting), signaling information is routed to accommodate channel associated signaling. Line channel 16 carries the signaling information for all channels.

Signaling information is broken out and placed on the corresponding signaling stream for that trunk. The signaling information for a given channel is placed in the same timeslot number as the voice information for that channel.

Note: The CG 6060 board does not allow signaling streams to be connected to the CT bus.

The following illustration shows how signaling data is distributed (although the streams are shown here, they cannot be switched to the CT bus):



Default connections (CAS mode)

If a board is configured for standalone operation (Clocking.HBus.ClockMode = STANDALONE), the DSPs and trunks are connected.

Note: The SwitchConnections keyword can override this setting.

The exact settings for CG 6060 boards configured as T1 or E1 depend upon the setting of the NetworkInterface.T1E1[x].SignalingType keyword.

The Voice information and DSP resources table and the Signaling information and DSP resources table show the default routing for CG 6060 boards in CAS mode.

Voice information and DSP resources

Trunk type	Full duplex connection between the trunk voice information and the DSP resources...	
T1	Trunk 1: 0:0..23 => 65:0..23 Trunk 2: 4:0..23 => 65:24..47 Trunk 3: 8:0..23 => 65:48..71 Trunk 4: 12:0..23 => 65:72..95	64:0..23 => 1:0..23 64:24..47 => 5:0..23 64:48..71 => 9:0..23 64:72..95 => 13:0..23
E1	Trunk 1: 0:0..29 => 65:0..29 Trunk 2: 4:0..29 => 65:30..59 Trunk 3: 8:0..29 => 65:60..89 Trunk 4: 12:0..29 => 65:90..119	64:0..29 => 1:0..29 64:30..59 => 5:0..29 64:60..89 => 9:0..29 64:90..119 => 13:0..29

Signaling information and DSP resources

Trunk type	Full duplex connection between trunk signaling information and the DSP resources...	
T1	Trunk 1: 2:0..23 => 67:0..23 Trunk 2: 6:0..23 => 67:24..47 Trunk 3: 10:0..23 => 67:48..71 Trunk 4: 14:0..23 => 67:72..95	66:0..23 => 3:0..23 66:24..47 => 7:0..23 66:48..71 => 11:0..23 66:72..95 => 15:0..23
E1	Trunk 1: 2:0..29 => 67:0..29 Trunk 2: 6:0..29 => 67:30..59 Trunk 3: 10:0..29 => 67:60..89 Trunk 4: 14:0..29 => 67:90..119	66:0..29 => 3:0..29 66:30..59 => 7:0..29 66:60..89 => 11:0..29 66:90..119 => 15:0..29

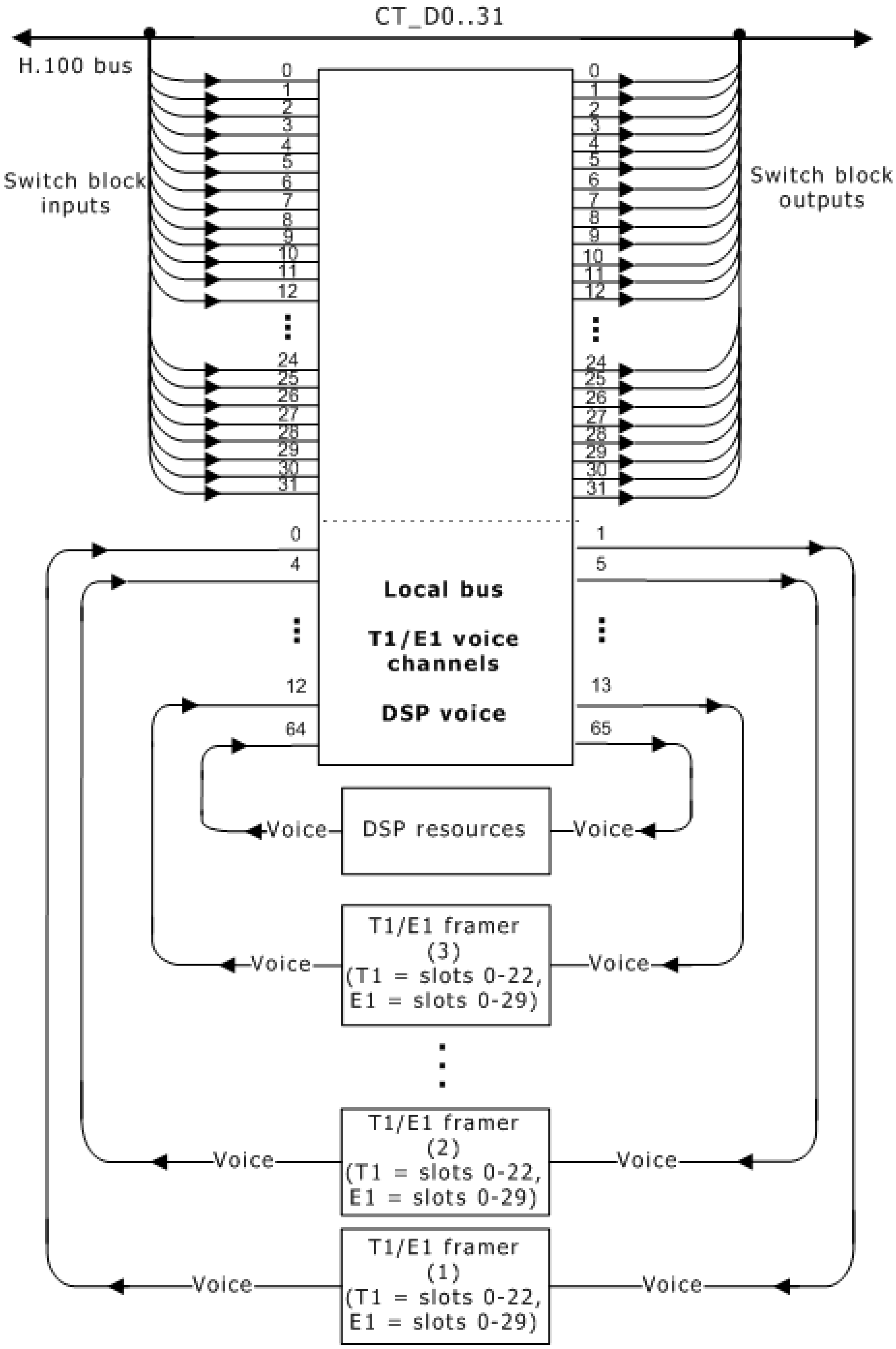
PRI mode switching

This topic contains the following PRI mode switching information:

- PRI mode switch model
- H.100 and local streams
- Voice information routing on T1 trunks
- Voice information routing on E1 trunks
- T1/E1 signaling information routing
- Default connections

PRI mode switch model

The following illustration shows the CG 6060 switching model in PRI mode:



H.100 and local streams

The following tables list the specific use of each stream in the CG 6060 PRI switch model:

H.100 streams

H.100 bus	Streams 0..31, timeslots 0..127 (Streams clocked at 8 MHz)

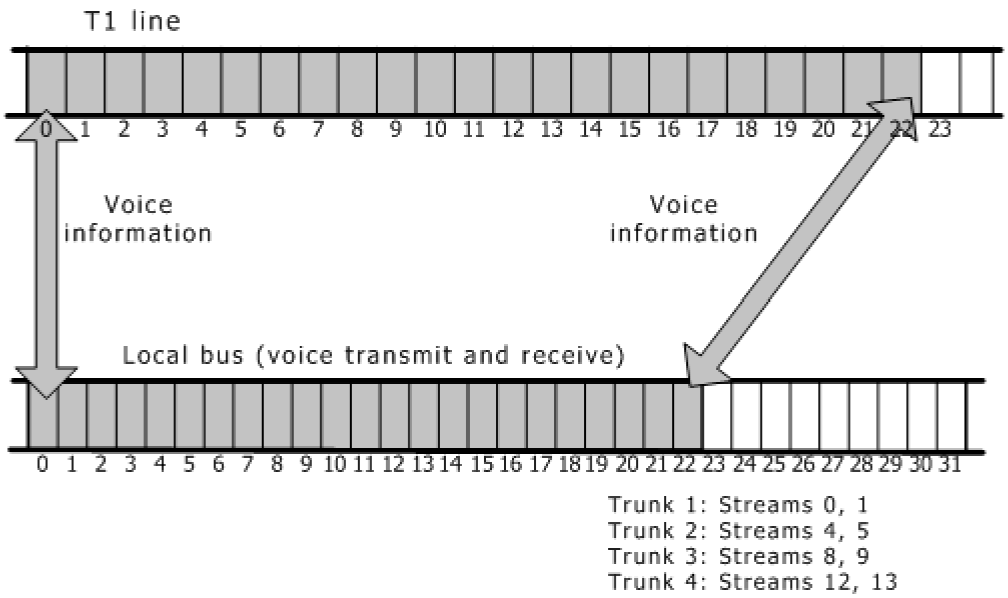
Local streams

Trunk voice information (T1 trunks)	Trunk 1: Streams 0 and 1 Trunk 2: Streams 4 and 5 Trunk 3: Streams 8 and 9 Trunk 4: Streams 12 and 13	timeslots 0..22 timeslots 0..22 timeslots 0..22 timeslots 0..22
Trunk voice information (E1 trunks)	Trunk 1: Streams 0 and 1 Trunk 2: Streams 4 and 5 Trunk 3: Streams 8 and 9 Trunk 4: Streams 12 and 13	timeslots 0..29 timeslots 0..29 timeslots 0..29 timeslots 0..29
DSP voice information (T1 and E1 trunks)	Streams 64 and 65, timeslots 0.. up to 899	

In PRI mode, the D channel signaling is automatically terminated by an internal HDLC controller.

Voice information routing on T1 trunks (PRI mode)

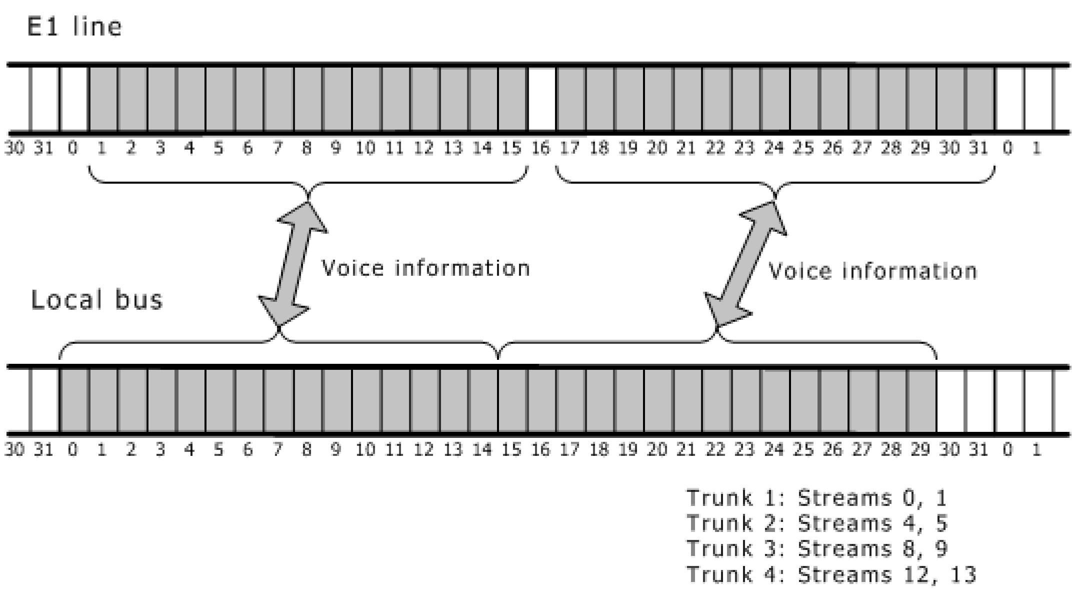
If NetworkInterface.T1E1[x].SignalingType = PRI, signaling information is routed to accommodate the T1 ISDN common channel signaling configuration, where voice information is transmitted in the first 23 channels. Each voice channel on the T1 trunk is placed in a corresponding timeslot on the local bus in the following streams:



Voice information routing on E1 trunks (PRI mode)

Regardless of the NetworkInterface.T1E1[x].SignalingType setting, the CG 6060 board routes the voice information by assigning E1 timeslots 1 through 15 to the local bus timeslots 0..14. E1 timeslots 17 through 31 are assigned to the local bus timeslots 15..29. Timeslot 0 on the E1 line carries framing data.

The following illustration shows how voice channel data is assigned to timeslots:



T1/E1 signaling information routing (PRI mode)

If NetworkInterface.T1E1[x].SignalingType = PRI, signaling information is routed differently to accommodate an ISDN common channel signaling configuration, where CCS signaling packets are transmitted in the D channel.

In PRI mode, the trunk signaling streams are not used. In this case, each trunk signaling stream has zero timeslots.

On CG 6060 boards, framer signaling is hard wired to internal HDLCs when the board runs in PRI mode. Refer to the PRI mode switch model. HDLC signaling is automatically terminated by an internal HDLC.

Default connections (PRI mode)

If a board is configured for standalone operation (Clocking.HBus.ClockMode = STANDALONE), the DSPs and trunks are connected as shown in the following table.

Note: The SwitchConnections keyword can override this setting.

The exact settings for CG 6060 boards configured as T1 or E1 depend upon the NetworkInterface.T1E1[x].SignalingType keyword setting.

The following table shows the default routing for CG 6060 boards in PRI mode:

Trunk type	Full duplex connection between trunk voice information and the DSP resources...	
T1	Trunk 1: 0:0..22 => 65:0..22 Trunk 2: 4:0..22 => 65:24..46 Trunk 3: 8:0..22 => 65:48..70 Trunk 4: 12:0..22 => 65:72..94	64:0..22 => 1:0..22 64:24..46 => 5:0..22 64:48..70 => 9:0..22 64:72..94 => 13:0..22
E1	Trunk 1: 0:0..29 => 65:0..29 Trunk 2: 4:0..29 => 65:30..59 Trunk 3: 8:0..29 => 65:60..89 Trunk 4: 12:0..29 => 65:90..119	64:0..29 => 1:0..29 64:30..59 => 5:0..29 64:60..89 => 9:0..29 64:90..119 => 13:0..29

On CG 6060 boards, the framer signaling is hard wired to internal HDLCs when the board runs in PRI mode.

Note: The CG 6060 board does not allow signaling streams to be connected to the CT bus.

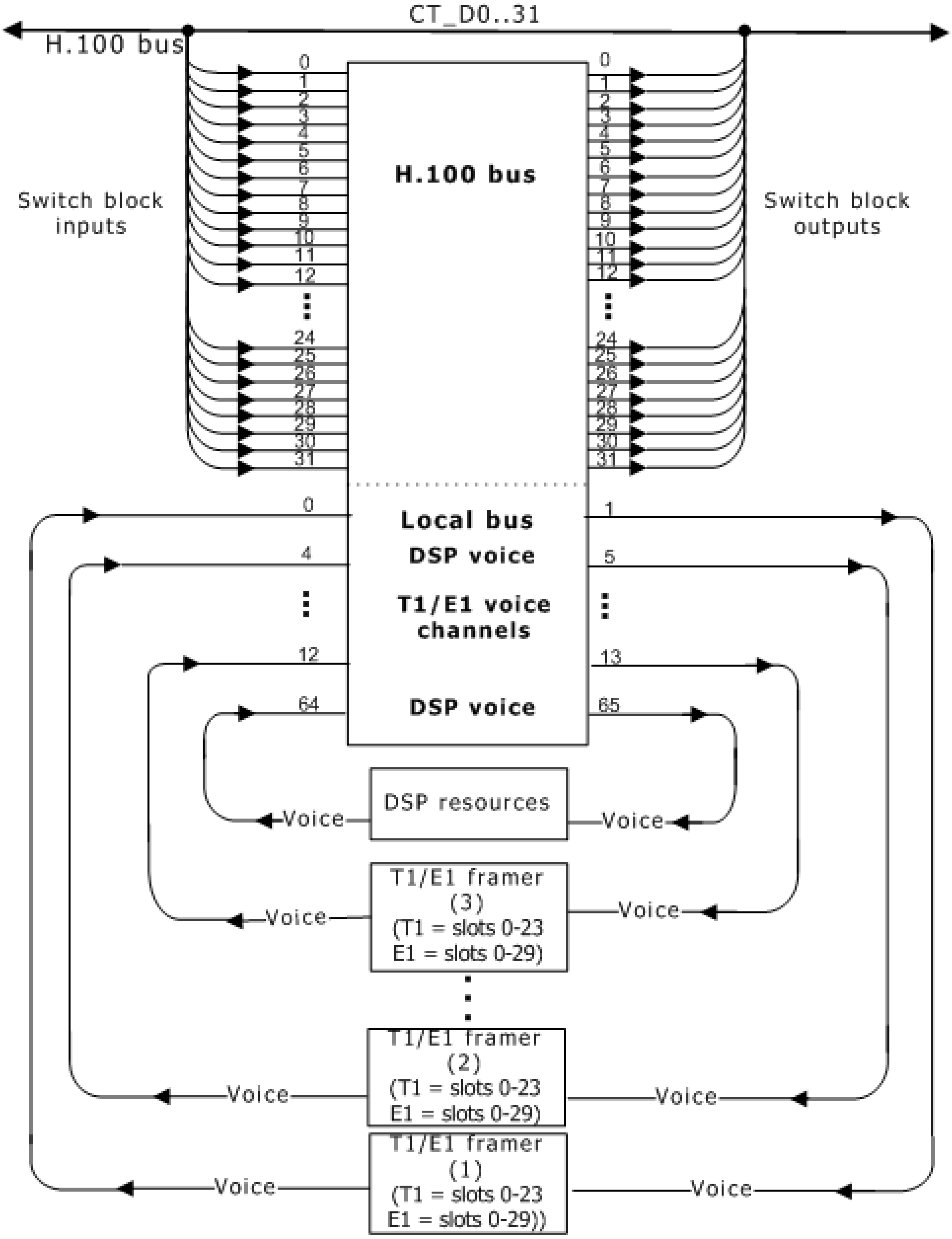
RAW mode switching

This topic contains the following RAW mode switching information:

- RAW mode switch model
- H.100 and local streams
- Voice information routing on T1 trunks
- Voice information routing on E1 trunks
- T1/E1 signaling information routing
- Default connections

RAW mode switch model

The following illustration shows the CG 6060 switching model in RAW mode:



H.100 and local streams

The following tables list the specific use of each stream in the CG 6060 RAW switching model:

H.100 streams

H.100 bus	Streams 0..31, timeslots 0..127 (Streams clocked at 8 MHz)

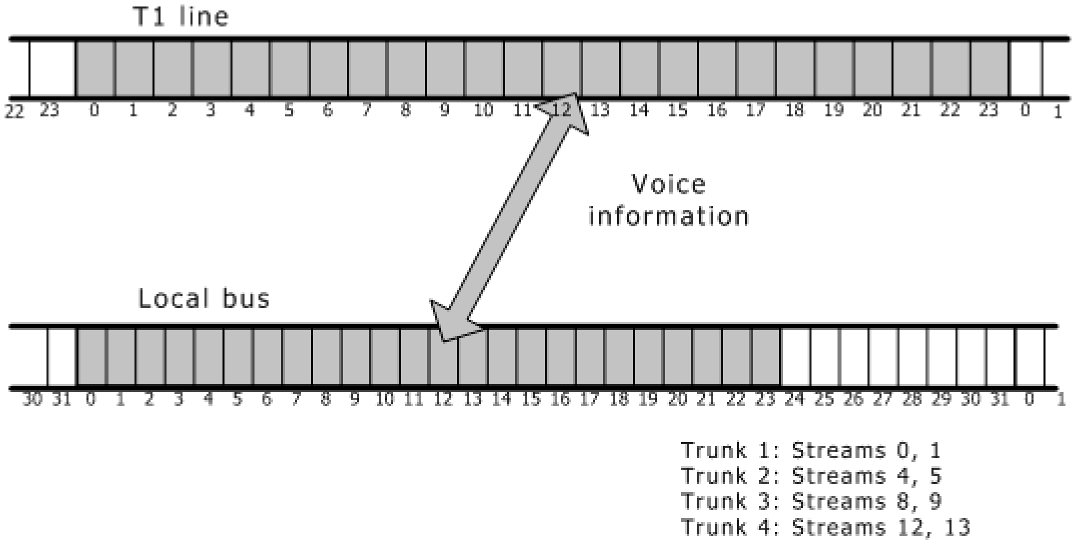
Local streams

Trunk voice information (T1 trunks)	Trunk 1: Streams 0 and 1 Trunk 2: Streams 4 and 5 Trunk 3: Streams 8 and 9 Trunk 4: Streams 12 and 13	timeslots 0..23 timeslots 0..23 timeslots 0..23 timeslots 0..23
Trunk voice information (E1 trunks)	Trunk 1: Streams 0 and 1 Trunk 2: Streams 4 and 5 Trunk 3: Streams 8 and 9 Trunk 4: Streams 12 and 13	timeslots 0..30 timeslots 0..30 timeslots 0..30 timeslots 0..30
DSP voice information (T1 and E1 trunks)	Streams 64 and 65, timeslots 0.. up to 899	

Voice information routing on T1 trunks (RAW mode)

If NetworkInterface.T1E1[x].SignalingType is set to RAW, information is routed to accommodate a configuration where no signaling is present on the T1 trunk. Voice information is transmitted in all 24 channels.

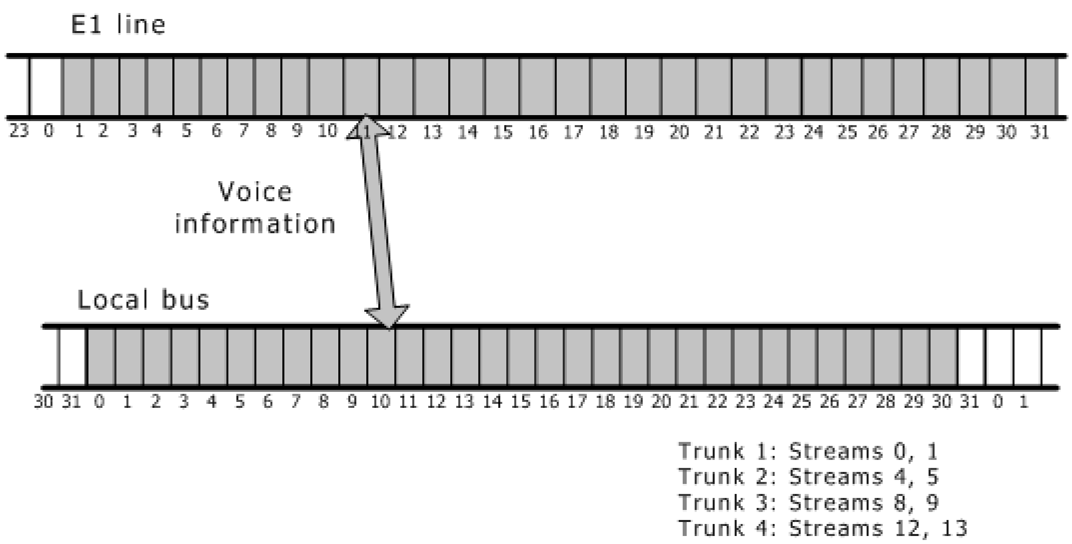
Each voice channel on the T1 trunk is placed in a corresponding timeslot on the local bus in the following streams:



Voice information routing on E1 trunks (RAW mode)

If NetworkInterface.T1E1[x].SignalingType is set to RAW, voice information is transmitted in 31 channels. Timeslot 0 on the E1 line carries framing data.

The following illustration shows how voice channel data is assigned to timeslots:



T1/E1 signaling information routing (RAW mode)

No signaling information is transmitted in RAW mode. It is assumed that another E1 trunk is carrying a D channel containing signaling for all trunks. In this mode, the local trunk signaling streams have zero timeslots.

Default connections (RAW mode)

If a board is configured for standalone operation (that is, Clocking.HBus.ClockMode = STANDALONE), the DSPs and trunks are connected as shown in the following table.

Note: The SwitchConnections keyword can override this setting.

The exact settings for CG 6060 boards configured as E1 depend upon the NetworkInterface.T1E1[x].SignalingType keyword setting.

The following table shows the default routing for CG 6060 boards in RAW mode:

Trunk type	Full duplex connection between trunk voice information and DSP resources...	
T1	Trunk 1: 0:0..23 => 65:0..23 Trunk 2: 4:0..23 => 65:24..47 Trunk 3: 8:0..23 => 65:48..71 Trunk 4: 12:0..23 => 65:72..95	64:0..23 => 1:0..23 64:24..47 => 5:0..23 64:48..71 => 9:0..23 64:72..95 => 13:0..23
E1	Trunk 1: 0:0..30 => 65:0..30 Trunk 2: 4:0..30 => 65:31...61 Trunk 3: 8:0..30 => 65:62..92 Trunk 4: 12:0..30 => 65:93..123	64:0..30 => 1:0..30 64:31..61 => 5:0..30 64:62..92 => 9:0..30 64:93..123 => 13:0..30

9. Echo cancellation control

Using echo cancellation control

The CG 6060 board includes a hardware echo cancellation feature that offloads DSP processing time for running other functions. The hardware echo canceler is located between the switch and the trunking framers, placing the echo canceler feature at the trunks. The control API allows you to modify hardware echo cancellation features for a trunk port.

Applications can query and configure features of the hardware echo cancellation feature on a per-timeslot basis. Since the echo chip is wired in series between the switch and trunking framers, individual echo cancellation channels are identified by specifying the MVIP address of the associated trunk port. Refer to the CG 6060 switch models for your board, to identify the MVIP address.

Echo cancellation control uses the Switching service for configuring and querying the board-specific hardware parameters. **swiConfigLocalTimeslot** and **swiGetLocalTimeslotInfo** allow applications to configure or query a device on a given local stream and timeslot by specifying a particular parameter and providing a data structure specific to that parameter. The syntax for these functions is repeated here for your convenience.

Refer to the *Switching Service Developer's Reference Manual* for details about the Switching service.

Syntax

The syntax of **swiConfigLocalTimeslot** and **swiConfigLocalTimeslotInfo** is:

DWORD **swiConfigLocalTimeslot** (SWIHD ***swihd***, SWI_LOCALTIMESLOT_ARGS ***** ***args***, void ****buffer***, unsigned ***size***)

DWORD **swiGetLocalTimeslotInfo** (SWIHD ***swihd***, SWI_LOCALTIMESLOT_ARGS ***** ***args***, void ****buffer***, unsigned ***size***)

Argument	Description
swihd	Switch handle.

args	Pointer to a SWI_LOCALTIMESLOT_ARGS structure. This structure indicates the specific parameter to be configured or queried on the device indicated by localstream and localtimeslot.
	`typedef struct`
	`{`
	`      DWORD localstream;`
	`      DWORD localtimeslot;`
	`      DWORD deviceid;`
	`      DWORD parameterid;`
	`} SWI_LOCALTIMESLOT_ARGS;`

buffer	Pointer to a structure that is specific to the parameterid.
size	Size of ***buffer***, in bytes.

For more information, refer to Setting or retrieving the echo cancellation bypass state, Setting or retrieving the nonlinear processing state, or Setting or retrieving the acoustic echo control state.

Setting or retrieving the echo cancellation bypass state

The bypass control allows you to enable or disable the hardware echo cancellation feature for a trunk port. Use **swiConfigLocalTimeslot** to set the echo canceler bypass state and **swiGetLocalTimeslotInfo** to retrieve the echo canceler bypass state. Set the arguments for these functions as follows:

Argument	Field	Value
swihd		Handle returned by **swiOpenSwitch**.
args	localstream	Identifies the target trunk on the local bus. Specify the number of either the transmit or receive voice stream. Refer to CG 6060 switch models for more information.
	localtimeslot	Identifies the target timeslot on the trunk. Specify the timeslot number of the target trunk port on the local bus. Refer to CG 6060 switch models for more information.
	deviceid	Device type on the local bus. The deviceid is hardware dependent. For example, MVIP95_T1_TRUNK_DEVICE or MVIP95_E1_TRUNK_DEVICE. For information about the deviceid, refer to the *Dialogic® NaturalAccess™ Switching Interface API Developer's Manual*.
	parameterid	Data item to configure or query. Set to NMS_ECHO_CHANNEL_BYPASS (0x80000007).
buffer		Points to the NMS_ECHO_CHANNEL_BYPASS_PARMS structure. Valid values are: 0 = NMS_ECHO_BYPASS_DISABLE 1 = NMS_ECHO_BYPASS_ENABLE
size		Size of ***buffer***, in bytes.

The NMS_ECHO_CHANNEL_BYPASS_PARMS structure is:

```
typedef  struct
{
     DWORD bypass;  /*  NMS_ECHO_BYPASS_DISABLE or  NMS_ECHO_BYPASS_ENABLE */

}    NMS_ECHO_CHANNEL_BYPASS_PARMS;
```

The value returned from the NMS_ECHO_CHANNEL_BYPASS_PARMS structure indicates whether the echo cancellation bypass state is enabled or disabled for the specified device.

For information about the echo cancellation bypass feature, refer to Using echo cancellation control. For more information about **swiConfigLocalTimeslot** or **swiGetLocalTimeslotInfo**, refer to the *Dialogic® NaturalAccess™ Switching Interface API Developer's Manual*.

Setting the bypass state example

The following example shows how to set the echo canceler bypass state:

```
#include " swidef.h"      /*  Switching service                        */
#include "mvip95.h"      /*  MVIP-95 definitions                      */
#include " nmshw.h"       /*   NMS hardware-specific definitions       */

DWORD  mySetBypass(SWIHD  swihd,  SWI_TERMINUS terminus,  int  bBypassEnabled)
{
      SWI_LOCALTIMESLOT_ARGS         args;
      NMS_ECHO_CHANNEL_BYPASS_PARMS parms;

      args.localstream      =  terminus.stream;     /* from board switch model */
      args.localtimeslot    =  terminus.timeslot;   /* from board switch model */
      args.deviceid         = MVIP95_T1_TRUNK_DEVICE;             /* mvip95.h */
      args.parameterid      =  NMS_ECHO_CHANNEL_BYPASS;            /*  nmshw.h  */

     if ( bBypassEnabled)
         parms.bypass =  NMS_ECHO_BYPASS_ENABLE;            /*  nmshw.h  */
     else
         parms.bypass =  NMS_ECHO_BYPASS_DISABLE;           /*  nmshw.h  */

     return  swiConfigLocalTimeslot (
                 swihd,            /* switch handle                         */
              &  args,             /* target device and  config item         */
      ( void*) & parms,            /* buffer (defined by  parameterid)       */
                sizeof(parms));   /* buffer size in bytes                  */

}
```

Retrieving the bypass state example

The following example shows how to retrieve the echo canceler bypass state:

```
#include " swidef.h"      /*  Switching service                        */
#include "mvip95.h"      /*  MVIP-95 definitions                      */
#include " nmshw.h"       /*   NMS hardware-specific definitions       */


DWORD  myGetBypass(SWIHD  swihd,  SWI_TERMINUS terminus,  int*  pbBypassEnabled)
{
      SWI_LOCALTIMESLOT_ARGS         args;
      NMS_ECHO_CHANNEL_BYPASS_PARMS parms;
      DWORD                          swi =  SWI_SUCCESS;

      args.localstream      =  terminus.stream;     /* from board switch model */
      args.localtimeslot    = terminus.timeslot;   /* from board switch model */
     args.deviceid         = MVIP95_T1_TRUNK_DEVICE;             /* mvip95.h */
     args.parameterid      = NMS_ECHO_CHANNEL_BYPASS;            /* nmshw.h  */

     swi = swiGetLocalTimeslotInfo (
               swihd,            /* switch handle                         */
             & args,             /* target device and config item         */
     (void*) & parms,            /* buffer (defined by parameterid)       */
               sizeof(parms));   /* buffer size in bytes                  */


     if (parms.bypass == NMS_ECHO_BYPASS_ENABLE)           /* nmshw.h  */
         *pbBypassEnabled = 1;  // true
     else
         *pbBypassEnabled = 0;  // false

     return swi;
```

```
}
```

Setting or retrieving the nonlinear processing state

Nonlinear processing (NLP) removes residual echo by attenuating the return path when there is no far-end speech. Natural-sound, adaptive-threshold, NLP technology provides excellent voice quality while avoiding signal clipping. The NLP control allows you to enable or disable the NLP feature for a trunk port. NLP is enabled by default.

Use **swiConfigLocalTimeslot** to enable or disable NLP and **swiGetLocalTimeslotInfo** to retrieve the current setting. Set the arguments for these functions as follows:

Argument	Field	Value
swihd		Handle returned by **swiOpenSwitch**.
args	localstream	Identifies the target trunk on the local bus. Specify the number of either the transmit or receive voice stream. Refer to CG 6060 switch models for more information.
	localtimeslot	Identifies the target timeslot on the trunk. Specify the timeslot number of the target trunk port on the local bus. Refer to CG 6060 switch models for more information.
	deviceid	Device type on the local bus. The deviceid is MVIP95_T1_TRUNK_DEVICE or MVIP95_E1_TRUNK_DEVICE. For information about the deviceid, refer to the *Dialogic® NaturalAccess™ Switching Interface API Developer's Manual.*
	parameterid	Data item to configure or query. Set to NMS_ECHO_CHANNEL_NLP (0x80000020).
buffer		Points to the NMS_ECHO_CHANNEL_NLP_PARMS structure. Valid values are: 0 = NMS_ECHO_DISABLE_NLP 1 = NMS_ECHO_ENABLE_NLP
size		Size of ***buffer***, in bytes.

The NMS_ECHO_CHANNEL_NLP_PARMS structure is:

```
typedef struct
{
    DWORD enable_NLP;  // 0=no NLP, 1=NLP enabled

}   NMS_ECHO_CHANNEL_NLP_PARMS;
```

For **swiGetLocalTimeslotInfo**, the value returned from the NMS_ECHO_CHANNEL_NLP_PARMS structure indicates whether echo cancellation nonlinear processing is enabled or disabled for the specified device.

For more information about **swiConfigLocalTimeslot** or **swiGetLocalTimeslotInfo**, refer to the *Dialogic® NaturalAccess™ Switching Interface API Developer's Manual*.

Setting the NLP state example

The following example shows how to enable or disable NLP:

```
#include "swidef.h"      /*  Switching service                        */
#include "mvip95.h"      /*  MVIP-95 definitions                      */
#include "nmshw.h"       /*  NMS hardware-specific definitions        */

DWORD mySetNLP(SWIHD swihd, SWI_TERMINUS terminus, int bNLPEnabled)
{
     SWI_LOCALTIMESLOT_ARGS      args;
     NMS_ECHO_CHANNEL_NLP_PARMS  parms;

     args.localstream      = terminus.stream;     /* from board switch model */
     args.localtimeslot    = terminus.timeslot;   /* from board switch model */
     args.deviceid         = MVIP95_T1_TRUNK_DEVICE;              /* mvip95.h */
     args.parameterid      = NMS_ECHO_CHANNEL_NLP;                /* nmshw.h  */

     if (bNLPEnabled)
         parms.enable_NLP = NMS_ECHO_ENABLE_NLP;            /* nmshw.h  */
     else
         parms.enable_NLP = NMS_ECHO_ENABLE_NLP;            /* nmshw.h  */

     return swiConfigLocalTimeslot (
                swihd,            /* switch handle                        */
              & args,             /* target device and config item        */
      (void*) & parms,            /* buffer (defined by parameterid)      */
                sizeof(parms));   /* buffer size in bytes                 */

}
```

Retrieving the NLP state example

The following example shows how to retrieve the echo canceler NLP setting:

```
#include "swidef.h"      /*  Switching service                        */
#include "mvip95.h"      /*  MVIP-95 definitions                      */
#include "nmshw.h"       /*  NMS hardware-specific definitions        */


DWORD myGetNLP(SWIHD swihd, SWI_TERMINUS terminus, int* pbNLPEnabled)
{
     SWI_LOCALTIMESLOT_ARGS      args;
     NMS_ECHO_CHANNEL_NLP_PARMS  parms;
     DWORD                         swi = SWI_SUCCESS;

     args.localstream      = terminus.stream;     /* from board switch model */
     args.localtimeslot    = terminus.timeslot;   /* from board switch model */
     args.deviceid         = MVIP95_T1_TRUNK_DEVICE;              /* mvip95.h */
     args.parameterid      = NMS_ECHO_CHANNEL_NLP;                /* nmshw.h  */

     swi = swiGetLocalTimeslotInfo (
               swihd,            /* switch handle                         */
             & args,             /* target device and config item         */
     (void*) & parms,            /* buffer (defined by parameterid)       */
               sizeof(parms));   /* buffer size in bytes                  */


     if (parms.enable_NLP == NMS_ECHO_ENABLE_NLP)           /* nmshw.h  */
         *pbNLPEnabled = 1;  // true
     else
         *pbNLPEnabled = 0;  // false
```

```
    return swi;
}
```

Setting or retrieving the acoustic echo control state

The acoustic echo control (AEC) state is a proprietary adaptive suppression algorithm that provides far end acoustic echo control for up to 400 ms of flat delay. It works in the opposite direction from the trunk, for example, the IP direction in a PSTN to IP gateway. AEC is disabled by default.

Use **swiConfigLocalTimeslot** to enable or disable AEC and **swiGetLocalTimeslotInfo** to retrieve the current setting. Set the arguments for these functions as follows:

Argument	Field	Value
swihd		Handle returned by **swiOpenSwitch**.
args	localstream	Identifies the target trunk on the local bus. Specify the number of either the transmit or receive voice stream. Refer to CG 6060 switch models for more information.
	localtimeslot	Identifies the target timeslot on the trunk. Specify the timeslot number of the target trunk port on the local bus. Refer to CG 6060 switch models for more information.
	deviceid	Device type on the local bus. The deviceid is MVIP95_T1_TRUNK_DEVICE or MVIP95_E1_TRUNK_DEVICE. For information about the deviceid, refer to the *Dialogic® NaturalAccess™ Switching Interface API Developer's Manual*.
	parameterid	Data item to configure or query. Set to NMS_ECHO_CHANNEL_AEC (0x80000021).
buffer		Points to the NMS_ECHO_CHANNEL_AEC_PARMS structure. Valid values are: 0 = NMS_ECHO_DISABLE_AEC 1 = NMS_ECHO_ENABLE_AEC
size		Size of ***buffer***, in bytes.

The NMS_ECHO_CHANNEL_AEC_PARMS structure is:

```
typedef struct
{
    DWORD enable_AEC;  // 0=no AEC, 1=AEC enabled

}   NMS_ECHO_CHANNEL_AEC_PARMS;
```

For **swiGetLocalTimeslotInfo**, the value returned from the NMS_ECHO_CHANNEL_AEC_PARMS structure indicates whether echo cancellation acoustic echo control is enabled or disabled for the specified device.

For more information about **swiConfigLocalTimeslot** or **swiGetLocalTimeslotInfo**, refer to the *Dialogic® NaturalAccess™ Switching Interface API Developer's Manual*.

Setting the acoustic echo control state example

The following example shows how to enable or disable AEC:

```
#include "swidef.h"      /*  Switching service                       */
#include "mvip95.h"      /*  MVIP-95 definitions                     */
#include "nmshw.h"       /*  NMS hardware-specific definitions       */

DWORD mySetAEC(SWIHD swihd, SWI_TERMINUS terminus, int bAECEnabled)
{
     SWI_LOCALTIMESLOT_ARGS      args;
     NMS_ECHO_CHANNEL_AEC_PARMS  parms;

     args.localstream      = terminus.stream;     /* from board switch model */
     args.localtimeslot    = terminus.timeslot;   /* from board switch model */
     args.deviceid         = MVIP95_T1_TRUNK_DEVICE;             /* mvip95.h */
     args.parameterid      = NMS_ECHO_CHANNEL_AEC;               /* nmshw.h  */

     if (bAECEnabled)
         parms.enable_AEC = NMS_ECHO_ENABLE_AEC;            /* nmshw.h  */
     else
         parms.enable_AEC = NMS_ECHO_ENABLE_AEC;            /* nmshw.h  */

     return swiConfigLocalTimeslot (
                swihd,             /* switch handle                        */
              & args,              /* target device and config item        */
      (void*) & parms,             /* buffer (defined by parameterid)      */
                sizeof(parms));    /* buffer size in bytes                 */

}
```

Retrieving the acoustic echo control state example

The following example shows how to retrieve the echo canceler AEC setting:

```
#include "swidef.h"      /*  Switching service                       */
#include "mvip95.h"      /*  MVIP-95 definitions                     */
#include "nmshw.h"       /*  NMS hardware-specific definitions       */


DWORD myGetAEC(SWIHD swihd, SWI_TERMINUS terminus, int* pbAECEnabled)
{
     SWI_LOCALTIMESLOT_ARGS      args;
     NMS_ECHO_CHANNEL_AEC_PARMS  parms;
     DWORD                         swi = SWI_SUCCESS;

     args.localstream      = terminus.stream;     /* from board switch model */
     args.localtimeslot    = terminus.timeslot;   /* from board switch model */
     args.deviceid         = MVIP95_T1_TRUNK_DEVICE;             /* mvip95.h */
     args.parameterid      = NMS_ECHO_CHANNEL_AEC;               /* nmshw.h  */

     swi = swiGetLocalTimeslotInfo (
               swihd,             /* switch handle                        */
             & args,              /* target device and config item        */
     (void*) & parms,             /* buffer (defined by parameterid)      */
               sizeof(parms));    /* buffer size in bytes                 */


     if (parms.enable_AEC == NMS_ECHO_ENABLE_AEC)           /* nmshw.h  */
         *pbAECEnabled = 1;  // true
     else
         *pbAECEnabled = 0;  // false
```

```
    return swi;
}
```

10. Keyword summary

Keyword types

The keywords for a CG 6060 board describe that board's configuration. Some keywords are read/write; others are read-only:

Keyword type	Description
Read/write (editable)	Determines how the board is configured when it starts up. Changes to these keywords become effective after the board is rebooted.
Read-only (informational)	Indicates the board's current configuration. Read-only keywords cannot be modified.

A keyword has the general syntax:

keyword = ***value***

Keywords are not case sensitive except where operating system conventions prevail (for example, file names under UNIX). All values are strings or strings that represent integers:

- Integer keywords require a fixed range of legal numeric values.
- String keywords either require a fixed set of legal values or accept any string.

Setting keyword values

There are several ways to set the values of read/write keywords:

- Duplicate the board keyword file corresponding to your country and board type, modify the new file, specify the name of this new file in the File statement in the system configuration file, and run *oamsys* again. For information about board keyword file syntax, refer to the *Dialogic® NaturalAccess™ OAM System Developer's Manual*.
- Create a new board keyword file, either with additional keywords, or keywords whose values override earlier settings.
- Specify parameter settings directly with the *oamcfg* utility. For more information about *oamcfg*, refer to the *Dialogic® NaturalAccess™ OAM System Developer's Manual*.
- Specify the settings with OAM service functions. For more information, refer to the *Dialogic® NaturalAccess™ OAM API Developer's Manual*.

Keyword values in the CG board keyword files take effect when the board is rebooted.

Retrieving keyword values

To retrieve the values of read/write and read-only keywords:

- Run the *oaminfo* sample program. Specify the name of the board with the -b option (specifying the board number) on the command line:

```
oaminfo -b boardnumber
```

  oaminfo returns a complete list of keywords and values for the specified board.

- Use the OAM service.

For more information, refer to the *Dialogic® NaturalAccess™ OAM API Developer's Manual*.

Editable keywords

The following table summarizes the keywords that you can change:

If you want to...	Use these keywords...
Specify information about the board	AutoStart AutoStop EnableMonitor MaxChannels Name Number
Specify CG 6060 line interfaces as T1 or E1	NetworkInterface.T1E1[x].Type
Set up trunk information for the board	NetworkInterface.T1E1[x].FrameType NetworkInterface.T1E1[x].Impedance NetworkInterface.T1E1[x].Length NetworkInterface.T1E1[x].LineCode NetworkInterface.T1E1[x].SignalingType NetworkInterface.T1E1[x].Type NetworkInterface.T1E1[x].CRCMFMode
Set up trunk information specific to ISDN	NetworkInterface.T1E1[x].D_Channel NetworkInterface.T1E1[x].ISDN.D_Channel_Backup_Trunk NetworkInterface.T1E1[x].ISDN.NFAS_Member[y].Board NetworkInterface.T1E1[x].ISDN.NFAS_Member[y].NAI NetworkInterface.T1E1[x].ISDN.NFAS_Member[y].Trunk NetworkInterface.T1E1[x].ISDN.NFASGroup
Modify memory allocation	Buffers[x].Size DynamicRecordBuffers

If you want to...	Use these keywords...
Set IPv4 addressing and static IPv4 routing table information for the board	IPC.AddRoute[x].DestinationAddress IPC.AddRoute[x].GatewayAddress IPC.AddRoute[x].Interface IPC.AddRoute[x].Mask
Set IPv6 addressing for the board	IPv6.Link[x].Enable IPv6.Link[x].EnablePing IPv6.Link[x].HopLimit IPv6.Link[x].ICMPRateLimit IPv6.Link[x].IPSec IPv6.Link[x].MTU IPv6.Link[x].NDAttempts IPv6.Link[x].NDReachabilityTimer IPv6.Link[x].NDRetranTimer
Set up clocking information	Clocking.HBus.ClockMode Clocking.HBus.ClockSource Clocking.HBus.ClockSourceNetwork
Configure clock fallback	Clocking.HBus.AutoFallBack Clocking.HBus.FallBackClockSource Clocking.HBus.FallBackNetwork
Set up clocking information specific to NETREF	Clocking.HBus.NetRefSource Clocking.HBus.NetRefSourceNetwork Clocking.HBus.NetRefSpeed Clocking.HBus.SClockSpeed Clocking.HBus.Segment
Set up switching information	SwitchConnections SwitchConnectMode DSPStream.SlotCount DSPStream.SignalIdleCode[x] DSPStream.VoiceIdleCode[x]
Enable conferencing streams for switching	ConferencingStream.Enable ConferencingStream.SlotCount

If you want to...	Use these keywords...
Manage the DSP resources on the board	Resource[x].Definitions Resource[x].DSPs Resource[x].Name Resource[x].Size Resource[x].StartTimeSlot Resource[x].TCPs
Set up debug level information	DebugMask EnableMonitor
Configure HDLC signaling	Hdlc[x].Boot Hdlc[x].RxTimeSlot Hdlc[x].TxTimeSlot
Configure DSPs	DSP.C5x[x].CmdQSize DSP.C5x[x].CmdQStart DSP.C5x[x].DataInQSize DSP.C5x[x].DataInQStart DSP.C5x[x].DspOutQSize DSP.C5x[x].DspOutQStart DSP.C5X[x].Image DSP.C5x[x].Libs[y] DSP.C5x[x].NumRxTimeSlots DSP.C5x[x].NumTxTimeSlots DSP.C5x[x].OS DSP.C5x[x].XLaw
Control switching on the echo canceler reference stream (applies to software echo cancellation only)	Echo.AutoSwitchingRefSource Echo.EnableExternalPins
Specify the board location	Location.PCI.Bus Location.PCI.Slot

If you want to...	Use these keywords...
Implement ThroughPacket packet multiplexing	TPKT.ComplexForward.Count TPKT.ComplexForward[x].DestinationPacketSize TPKT.ComplexForward[x].LifeTimeTicks TPKT.ComplexRxPort TPKT.ComplexTxPort TPKT.Enable TPKT.NumberOfComplexForwardConditions TPKT.SimpleRxPort TPKT.SimpleTxPort
Modify the hardware echo cancellation settings	HardwareEcho.EchoChipEnabled HardwareEcho.Trunk[x].OnOffTimeslots HardwareEcho.XLaw

Informational keywords

This topic describes read-only keywords for retrieving information. Do not edit the keywords listed in this topic. Use these keywords to retrieve information about the:

- Board
- Driver
- Miscellaneous board information
- EEPROM

Retrieving board information

Keyword	Type	Description
State	String	State of the physical board.

Retrieving driver information

Keyword	Type	Description
Driver.BoardID	Integer	Number used by the CG board driver to refer to this board. Two boards accessed by different drivers can have the same driver ID number.
Driver.Name	String	Operating system independent name (the root name) of the board driver.
SwitchDriver.Name	String	Operating system independent name (the root name) of the board switching driver.

Retrieving miscellaneous information

Keyword	Type	Description
NetworkInterface.T1E1[x].ISDN.NFAS_Member.Count	Integer	Number of interfaces in this NFAS group. The value is calculated based on the number of T1E1[x].ISDN.NFAS_MEMBER[y] structures specified.
NetworkInterface.Ethernet[x].MAC_Address	String	Specifies the MAC address.

Retrieving EEPROM information

The data type for all EEPROM keywords is Integer.

Keyword	Description
Eeprom.AssemblyRevision	Hardware assembly level.
Eeprom.ATETestBit	Indicates whether the ATE test was successful. A non-zero value indicates success.
Eeprom.DSPExtClk	Oscillator used to trigger the DSP.
Eeprom.DSPSpeed	DSP processor speed in MHz.
Eeprom.DSPType	Type of DSP on the board (for example, TI C5441).
Eeprom.EthernetType	Type of Ethernet connection on the board (for example, 10/100Base-T).
Eeprom.Family	Family ID of the board.
Eeprom.FlashBlkSz	Size of the Flash.
Eeprom.FlashID	Type of Flash chip ID used on the board.
Eeprom.HostBusType	Type of host bus used on the board (for example, PCI).
Eeprom.MFGWeek	Week of the last full test.
Eeprom.MFGYear	Year of the last full test.
Eeprom.MSBType	Type of CT bus used on the board.
Eeprom.NumCPU	Number of CPUs on the board.

Keyword	Description
Eeprom.NumDaughterCard	Number of daughterboards attached to the main board.
Eeprom.NumDSPCores	Number of DSP cores on the board.
Eeprom.NumEthernet	Number of Ethernet connections on the board.
Eeprom.NumSwitch	Number of switches on the board.
Eeprom.NumTrunk	Number of PSTN line interfaces on the board.
Eeprom.Product	OAM API product ID number associated with the board. This number is factory configured and unique to each board type. The product ID for the CG 6060 board is 0x630.
Eeprom.SerialNum	Board's serial number.
Eeprom.SoftwareCompatibility	Minimum software revision level.
Eeprom.SwitchType	Type of switch on the board (for example, T8100).
Eeprom.TrunkType	Type of line interfaces on the board (for example, digital).

Plug-in keywords

CG plug-in keywords provide specific board family information for CG boards. All CG 6060 plug-in keywords (as opposed to board keywords) except BootDiagnosticLevel are read-only.

The following table lists CG plug-in keywords:

Keyword	Description
Boards[x]	Retrieves the name of the board object.
BootDiagnosticLevel	Sets the board diagnostic level.
DetectedBoards[x]	Retrieves the board names of detected boards.
Products[x]	Retrieves board product types.

11. Keyword reference

Using the keyword reference

The keywords are presented in detail in the following topics. Each keyword description includes the following information:

Syntax	The syntax of the keyword
Access	Read/write or read-only
Type	The data type of the value: string or integer
Default	Default value
Allowed values	A list of all possible values
Example	An example of usage
Details	A detailed description of the keyword's function
See also	A list of related keywords

AutoStart

Specifies whether the board automatically starts when *ctdaemon* is started.

Syntax

AutoStart = ***start***

Access

Read/Write

Type

String

Default

NO

Allowed values

YES | NO

Example

```
AutoStart = YES
```

Details

The Supervisor-level keyword AutoStartEnabled enables or disables the autostart feature. If AutoStartEnabled is set to YES, the Supervisor starts each board whose AutoStart keyword is set to YES when *ctdaemon* is started. If AutoStartEnabled is set to NO, no boards are started automatically, regardless of the setting of the AutoStart keyword in the board keyword files.

For more information, refer to the *Dialogic® NaturalAccess™ OAM System Developer's Manual*.

See also

AutoStop

AutoStop

Specifies whether the board automatically stops when *ctdaemon* is stopped.

Syntax

AutoStop = ***stop***

Access

Read/Write

Type

String

Default

NO

Allowed values

YES | NO

Example

```
AutoStop = YES
```

Details

The Supervisor-level keyword AutoStopEnabled enables or disables the autostop feature. If AutoStopEnabled is set to YES, the Supervisor stops each board whose AutoStop keyword is set to YES when *ctdaemon* is stopped. If AutoStopEnabled is set to NO, no boards are stopped automatically, regardless of the setting of the AutoStop keyword in the board keyword files.

For more information, refer to the *Dialogic® NaturalAccess™ OAM System Developer's Manual*.

See also

AutoStart

Boards[x]

Indicates the name of the board object.

Syntax

Boards[***x***] = ***name***

Access

Read-only (CG plug-in level)

Type

String

Default

Not applicable.

Allowed values

Not applicable.

See also

Name, Number

BootDiagnosticLevel

Specifies the level of diagnostics during initialization of the board.

Syntax

BootDiagnosticLevel = ***level***

Access

Read/Write (CG plug-in level)

Type

Integer

Default

1

Allowed values

0 | 1

Example

```
BootDiagnosticLevel = 1
```

Details

When disabled (set to 0) the board ignores any diagnostic errors returned while it is being initialized. The valid values for ***level*** are 0 and 1. Zero (0) indicates that no diagnostics are performed. The maximum level is 1.

If a test fails, the test number is reported back as the error code.

Some tests can pass back more than one error code, depending on the options selected and the mode of failure.

You must be running *oammon* to view diagnostic results.

Buffers[x].Size

Specifies the size in bytes of the board's buffer pool.

Syntax

Buffers[***x***].Size = ***size***

x = 0 | 1 (buffer pool index)

x represents a buffer pool index. Buffers[0].Size is used for large play and record buffers. Buffers[1].Size is used for ISDN messages, dynamic record buffers, and play and record of small buffers.

Access

Read/Write

Type

Integer

Default

16400

Allowed values

0 - 1000000

Example

```
Buffers[0].Size = 16400
```

Details

The CG 6060 has been optimized for the following values:

Buffers[0].Size = 16400

Buffers[1].Size = 1000

See also

DynamicRecordBuffers

Clocking.HBus.AutoFallBack

Enables clock fallback on the board.

For more information about setting up CT bus clocking and about the rules and restrictions that apply to setting up clocking with CG 6060 boards, refer to Configuring board clocking.

Syntax

Clocking.HBus.AutoFallBack = ***mode***

Access

Read/Write

Type

String

Default

NO

Allowed values

YES | NO

Example

```
Clocking.HBus.AutoFallBack = YES
```

Details

Set to YES to specify that the board automatically switches to an alternative timing reference (specified with the Clocking.HBus.FallBackClockSource keyword) when the first timing reference (specified with the Clocking.HBus.ClockSource keyword) fails.

This keyword applies for all modes specified by the Clocking.HBus.ClockMode keyword.

The physical timing references specified with the Clocking.HBus.ClockSource and Clocking.HBus.FallBackClockSource keywords must be present and not in an ALARM state when the CG 6060 board's clocking is set up.

Specifying NO indicates that the system does not fall back to a backup timing reference.

Use the *swish* command **queryBoardClck** to determine what timing reference the board is actively using.

If the board is configured as the primary master or in standalone mode, Clocking.HBus.AutoFallBack enables the board to switch to the fallback timing reference when the first source goes into an ALARM state.

In addition:

- If the primary clock master's first timing reference fails and then returns, the primary master's timing reference (and consequently the timing reference for any clock slaves) tries to switch back to the first source.
- If the primary clock master's first timing reference and fallback timing reference fail, the secondary clock master begins to drive the CT bus clock for all clock slaves. If either of the primary clock master's timing references then recover, the CT bus does not switch back to either of these sources. The secondary master continues to drive the CT bus clock until directed otherwise.
- If the board is configured as the primary clock master and both timing references fail, the board reconfigures itself to become a slave to the secondary H.100/H.110 timing reference.
- If the board is configured in standalone mode and both the first timing reference and fallback timing references fail, the board automatically switches to OSC.

See also

Clocking.HBus.FallBackNetwork

Clocking.HBus.ClockMode

Specifies the board's control of the H.110 clock.

For more information about setting up CT bus clocking and about the rules and restrictions that apply to setting up clocking with CG 6060 boards, refer to Configuring board clocking.

Syntax

Clocking.HBus.ClockMode = ***clockmode***

Access

Read/Write

Type

String

Default

STANDALONE

Allowed values

MASTER_A | MASTER_B | SLAVE | STANDALONE

Example

```
Clocking.HBus.ClockMode = MASTER_A
```

Details

Valid entries include:

Value	Description
MASTER_A	The board drives the CT bus A_CLOCK based on the timing information derived from a specified timing reference.
MASTER_B	The board drives the CT bus B_CLOCK based on the timing information derived from a specified timing reference.
SLAVE	The board acts as a clock slave, deriving its timing from the primary bus master. Connections are allowed to the board's CT bus timeslots in slave mode.
STANDALONE	The board behaves like a primary clock master, but does not drive any CT bus clocks. Connections are not allowed to the board's CT bus timeslots in standalone mode.

For more information, refer to the *Switching Service Developer's Reference Manual.*

See also

Clocking.HBus.AutoFallBack, Clocking.HBus.ClockSource, Clocking.HBus.FallBackClockSource, Clocking.HBus.FallBackNetwork, SwitchConnections

Clocking.HBus.ClockSource

Specifies the timing reference for the board to use based on the Clocking.HBus.ClockMode setting.

For more information about setting up CT bus clocking and about the rules and restrictions that apply to setting up clocking with CG 6060 boards, refer to Configuring board clocking.

Syntax

Clocking.HBus.ClockSource = ***clock_source***

Access

Read/Write

Type

String

Default

OSC

Allowed values

OSC | A_CLOCK | B_CLOCK | NETREF | NETWORK

Example

```
Clocking.HBus.ClockSource = NETWORK
```

Details

Valid entries include:

Value	Description
OSC	Drives the T1 or E1 line transmit clock using the on-board oscillator. Use this setting only when the board's T1 or E1 connection is isolated from the network. Apply this setting, for example, if you use a T1 or E1 connection as a link between two computers, or if you use one board to simulate network traffic to another. The on-board oscillator is accurate to 32 ppm (parts per million) and meets the requirements for a Stratum 4E clock.
A_CLOCK	Uses CT_C8_A and CT_FRAME_A timing signals as the board's first timing reference.
B_CLOCK	Uses CT_C8_B and CT_FRAME_B timing signals on the H.100 bus.
NETREF	Uses NETREF as the board's first timing reference. The NETREF reference source is set with Clocking.HBus.NetRefSource. The source may be a trunk on another board.
NETWORK	Uses a digital trunk as the board's first timing reference. The trunk must be specified with the Clocking.HBus.ClockSourceNetwork keyword.

The board returns an error if you select a CT bus clock source and no source is detected.

Clocking.HBus.ClockSourceNetwork

Specifies the trunk to use as an external network timing reference for the board's internal clock.

For more information about setting up CT bus clocking and about the rules and restrictions that apply to setting up clocking with CG 6060 boards, refer to Configuring board clocking.

Syntax

Clocking.HBus.ClockSourceNetwork = ***trunk_number***

Access

Read-only

Type

Integer

Default

1

Allowed values

1 to 4 (1-based trunk number)

Example

```
Clocking.HBus.ClockSourceNetwork = 1
```

Details

The board must have multiple external network connections and the Clocking.HBus.FallBackClockSource keyword must be set to NETWORK to take effect.

The Clocking.HBus.ClockSourceNetwork entry is a one-based number, while the ***x*** entry in the NetworkInterface.T1E1[x].Type keyword is a zero-based number.

If the Clocking.HBus.ClockSource keyword is not set to NETWORK, this keyword is ignored.

Clocking.HBus.FallBackClockSource

Specifies the alternate timing reference to use when the first timing reference does not function properly.

For more information about setting up CT bus clocking and about the rules and restrictions that apply to setting up clocking with CG 6060 boards, refer to Configuring board clocking.

Syntax

Clocking.HBus.FallBackClockSource = ***clock_source***

Access

Read/Write

Type

String

Default

OSC

Fallback to OSC is not recommended because the transition can cause slave boards to fall back to the secondary clock and create an out-of-sync condition.

Allowed values

OSC | A_CLOCK | B_CLOCK | NETREF | NETWORK

Example

```
Clocking.HBus.FallBackClockSource = B_CLOCK
```

Details

Valid entries include the following values:

Value	Description
OSC	Causes the board to drive the T1 or E1 line transmit clock using the on-board oscillator. Use this setting only when the board's T1 or E1 connection is isolated from the network. Apply this setting, for example, if you use a T1 or E1 connection as a link between two computers, or if you use one board to simulate network traffic to another. The on-board oscillator is accurate to 32 ppm (parts per million) and meets the requirements for a Stratum 4E clock.
A_CLOCK	Uses CT_C8_A and CT_FRAME_A timing signals as the board's fallback timing reference.
B_CLOCK	Uses the CT_C8_B and CT_FRAME_B timing signals as the board's fallback timing reference.
NETREF	Uses NETREF as the board's fallback timing reference. The NETREF reference source is set with Clocking.HBus.NetRefSource. The source can be a trunk on another board.
NETWORK	Uses a digital trunk as the board's fallback timing reference. This trunk is specified with the Clocking.HBus.ClockSourceNetwork keyword.

When this keyword is set to NETWORK, you must also specify the fallback network timing reference source with the Clocking.HBus.FallBackNetwork keyword.

If the Clocking.HBus.AutoFallBack keyword is set to NO, this keyword is ignored.

See also

Clocking.HBus.ClockMode, Clocking.HBus.ClockSource

Clocking.HBus.FallBackNetwork

Specifies the trunk to use as an external network timing reference if the clock source defined with Clocking.HBus.ClockSource fails.

For more information about setting up CT bus clocking and about the rules and restrictions that apply to setting up clocking with CG 6060 boards, refer to Configuring board clocking.

Syntax

Clocking.HBus.FallBackNetwork = ***trunk_number***

Access

Read/Write

Type

Integer

Default

1

Allowed values

1 to 4 (1-based trunk number)

Example

```
Clocking.HBus.FallBackNetwork = 1
```

Details

The board must have multiple external network connections and the Clocking.HBus.FallBackClockSource keyword must be set to NETWORK to take effect.

The Clocking.HBus.FallBackNetwork entry is a one-based number, while the ***x*** entry in the NetworkInterface.T1E1[x].Type keyword is a zero-based number.

See also

Clocking.HBus.AutoFallBack

Clocking.HBus.NetRefSource

Specifies a source to drive the NETREF timing signal on the CT bus.

For more information about setting up CT bus clocking and about the rules and restrictions that apply to setting up clocking with CG 6060 boards, refer to Configuring board clocking.

Syntax

Clocking.HBus.NetRefSource = ***source***

Access

Read/Write

Type

String

Default

OSC

Allowed values

OSC | NETWORK | STANDALONE

Example

```
Clocking.HBus.NetRefSource = STANDALONE
```

Details

Valid entries include the following values:

Value	Description
OSC	Specifies that the oscillator uses the board's local clock (for diagnostics only).
NETWORK	Specifies that the timing signal is derived from a device source (digital trunk). When using this keyword, you must also specify the trunk number with Clocking.HBus.NetRefSourceNetwork.
STANDALONE	Specifies that the NETREF clock is not driven.

See also

Clocking.HBus.NetRefSpeed

Clocking.HBus.NetRefSourceNetwork

Specifies the trunk used to drive the NETREF timing signal on the CT bus.

For more information about setting up CT bus clocking and about the rules and restrictions that apply to setting up clocking with CG 6060 boards, refer to Configuring board clocking.

Syntax

Clocking.HBus.NetRefSourceNetwork = ***trunk_number***

Access

Read/Write

Type

Integer

Default

1

Allowed values

1 to 4 (1-based trunk number)

Example

```
Clocking.HBus.NetRefSourceNetwork = 1
```

Details

You must specify a value with this keyword when the Clocking.HBus.NetRefSource keyword is set to NETWORK. If the Clocking.HBus.NetRefSource keyword is not set to NETWORK, this keyword is ignored.

See also

Clocking.HBus.NetRefSpeed

Clocking.HBus.NetRefSpeed

Specifies the speed of the NETREF timing signal on the CT bus.

For more information about setting up CT bus clocking and about the rules and restrictions that apply to setting up clocking with CG 6060 boards, refer to Configuring board clocking.

Syntax

Clocking.HBus.NetRefSpeed = ***speed***

Access

Read/Write

Type

String

Default

8K

Allowed values

8K

Example

```
Clocking.HBus.NetRefSpeed = 8K
```

See also

Clocking.HBus.NetRefSource, Clocking.HBus.NetRefSourceNetwork

Clocking.HBus.SClockSpeed

Specifies the speed (in MHz) of the driven Sclock when a board acts as primary master.

For more information about setting up CT bus clocking and about the rules and restrictions that apply to setting up clocking with CG 6060 boards, refer to Configuring board clocking.

Syntax

Clocking.HBus.SClockSpeed = ***speed***

Access

Read/Write

Type

String

Default

8M

Allowed values

8M

Example

```
Clocking.HBus.SClockSpeed = 8M
```

See also

Clocking.HBus.Segment

Clocking.HBus.Segment

Specifies the CT bus segment to which the board is connected. In most cases, the chassis contains only one segment.

For more information about setting up CT bus clocking and about the rules and restrictions that apply to setting up clocking with CG 6060 boards, refer to Configuring board clocking.

Syntax

Clocking.HBus.Segment = ***number***

Access

Read/Write

Type

Integer

Default

1

Allowed values

Positive integer

Example

```
Clocking.HBus.Segment = 1
```

See also

Clocking.HBus.SClockSpeed

ConferencingStream.Enable

Determines if the conferencing stream is available for switching. Set this keyword to YES when using NaturalConference.

Syntax

ConferencingStream.Enable = ***setting***

Access

Read/Write

Type

String

Default

NO

Allowed values

NO | YES

Example

```
ConferencingStream.Enable = NO
```

Details

Setting this keyword to YES enables the conferencing members to be switched. They appear on output stream 68 and input stream 69.

For more information, refer to the *NaturalConference Service Developer's Reference Manual*.

See also

ConferencingStream.SlotCount

ConferencingStream.SlotCount

Specifies the number of logical timeslots allocated to logical conferencing streams 68 and 69.

Syntax

ConferencingStream.SlotCount = ***slotcount_number***

Access

Read/Write

Type

Integer

Default

128

Example

```
ConferencingStream.SlotCount = 128
```

Allowed values

0 - 1500

Details

The number of reserved timeslots varies by user configuration. Refer to the *NaturalConference Service Developer's Reference Manual* for more information about conferencing. For information about streams and timeslots on CG boards, refer to the CAS mode switch model, the PRI mode switch model, and the RAW mode switch model.

See also

ConferencingStream.Enable

DebugMask

Specifies the type and level of tracing that the board performs.

Syntax

DebugMask = ***mask***

Access

Read/Write

Type

Integer

Default

0

Allowed values

A value shown in the following table.

Example

```
DebugMask = 0x00000200
```

Details

To view the results of CG 6060 tracing, you must be running *oammon*.

You can specify the following DebugMask parameters:

Value	Description
0x00000001	Additional initialization messages.
0x00000002	Legacy initialization messages.
0x00000008	Total resources for each DSP and calculate resource string.
0x00000100	Host interface up and down messages.
0x00000200	Inter-manager messages.
0x00000400	All manager messages.
0x00000800	High speed memory usage tracing messages.
0x00001000	Memory usage tracing messages while rechecking allocated usage.
0x00002000	Framers global tracing level.
0x00004000	General clock tracing messages.
0x80000000	Available memory.

DetectedBoards[x]

Indicates the user-defined name of the detected board.

Syntax

DetectedBoards[***x***] = ***boardid***

x = index of the board name

Access

Read-only (CG plug-in level)

Type

String

Allowed values

Not applicable.

DLMFiles[x]

Specifies an optional runtime component (modular extension to the core file) to be transferred to the board by the configuration file.

Syntax

DLMFiles[***x***] = ***filename***

x = 0..63 Index of the file name. The first value is always 0 (zero) with additional values numbered sequentially.

Access

Read/Write

Type

File name

Default

None.

Allowed values

A valid file name.

Example

```
DLMFiles[1] = cg6060fusion
```

Details

A *.dlm* file is a type of run module. For some CG boards, the software that runs on the board co-processor consists of the core file and any run modules. For the CG 6060 board, the core file and the run module are merged to create a file named *cg6060core.ulm*. In the CG 6060 configuration file, only set DLMFiles[***x***] when using the following software:

Software	Value
Generic ISDN	DLMFiles[***x***] = c6060igen
Fusion	DLMFiles[***x***] = cg6060fusion
ISDN Management	DLMFiles[x] = c6060imgt
DPNSS	DLMFiles[x] = c6060dpnss

Software	Value
NaturalFax	DLMFiles[x] = cg6060fax

When you use only one DLM file, ***x*** is always 0 (zero). If using more than one DLM file, number them sequentially starting with 0 (zero). They can appear in any order. For example:

```
DLMFiles[0] = cg6060fusion
DLMFiles[1] = c6060igen
```

DSP.C5x[x].CmdQSize

Specifies the size of the command queue in the DSP memory. The command queue sends commands to the DSP.

Syntax

DSP.C5x[***x***].CmdQSize = ***size***

x = 0-(***n***-1) (A range of DSP cores where ***n*** equals the total number of DSP cores available.)

Access

Read/Write

Type

Integer

Default

0x110

Allowed values

0x0000 - 0xFFFF

Example

```
DSP.C5x[0].CmdQSize = 0x110
```

Details

Changing the queue location or increasing the queue size can reduce the memory available to DSP functions. To change the DSP.C5x[***x***].CmdQSize settings, you must be familiar with the resource allocation in the CG board DSPs.

Caution:	Due to a DSP architectural limitation, DSPs in a pair core must maintain the same HPI queue setting. Therefore, keep the settings for CmdQ, DataInQ, and DspOutQ the same in a pair core. For example: `DSP.C5x[0..1].CmdQSize = 0x110`

See also

DSP.C5x[x].CmdQStart, DSP.C5x[x].DataInQSize, DSP.C5x[x].DataInQStart, DSP.C5x[x].DspOutQSize, DSP.C5x[x].DspOutQStart

DSP.C5x[x].CmdQStart

Specifies the start of the command queue in the DSP memory. The command queue sends commands to the DSP.

Syntax

DSP.C5x[***x***].CmdQStart = ***address***

x = 0-(***n***-1) (A range of DSP cores where ***n*** equals the total number of DSP cores available.)

Access

Read/Write

Type

Integer

Default

0x2000

Allowed values

0x0000 – 0xFFFF

Example

```
DSP.C5x[0].CmdQStart = 0x2000
```

Details

Changing the queue location or increasing the queue size can reduce the memory available to DSP functions. To change the DSP.C5x[***x***].CmdQStart settings, you must be familiar with the resource allocation in the CG board DSPs.

<table>
<tr>
<td>Caution:</td>
<td>Due to a DSP architectural limitation:

DSPs in a pair core must maintain the same HPI queue setting. Therefore, keep the settings for CmdQ, DataInQ, and DspOutQ the same in a pair core. For example:
<code>DSP.C5x[0..1].CmdQStart = 0x2000</code>
Conferencing DSPs must use the value 0xE800. All other types of DSPs can use the default value. For example:
<code>DSP.C5x[0..1].CmdQStart = 0xE800</code>

</td>
</tr>
</table>

See also

DSP.C5x[x].CmdQSize, DSP.C5x[x].DataInQSize, DSP.C5x[x].DataInQStart, DSP.C5x[x].DspOutQSize, DSP.C5x[x].DspOutQStart

DSP.C5x[x].DataInQSize

Specifies the size of the DataIn queue in the DSP memory. The DataIn queue sends the data to the DSP.

Syntax

DSP.C5x[***x***].DataInQSize = ***size***

x = 0-(***n***-1) (A range of DSP cores where ***n*** equals the total number of DSP cores available.)

Access

Read/Write

Type

Integer

Default

0x300

Allowed values

0x0000 – 0xFFFF

Example

```
DSP.C5x[0].DataInQSize = 0x300
```

Details

Changing the queue location or increasing the queue size can reduce the memory available to DSP functions. To change the DSP.C5x[***x***].DataInQSize settings, you must be familiar with the resource allocation in the CG board DSPs.

Caution:	Due to a DSP architectural limitation, DSPs in a pair core must maintain the same HPI queue setting. Therefore, keep the settings for CmdQ, DataInQ, and DspOutQ the same in a pair core. For example: `DSP.C5x[0..1].DataInQSize = 0x300`

See also

DSP.C5x[x].CmdQSize, DSP.C5x[x].CmdQStart, DSP.C5x[x].DataInQStart, DSP.C5x[x].DspOutQSize, DSP.C5x[x].DspOutQStart

DSP.C5x[x].DataInQStart

Specifies the start of the DataIn queue in the DSP memory. The DataIn queue sends the data to the DSP.

Syntax

DSP.C5x[***x***].DataInQStart = ***address***

x = 0-(***n***-1) (A range of DSP cores where ***n*** equals the total number of DSP cores available.)

Access

Read/Write

Type

Integer

Default

0x2280

Allowed values

0x0000 – 0xFFFF

Example

```
DSP.C5x[0].DataInQStart = 0x2280
```

Details

Changing the queue location or increasing the queue size can reduce the memory available to DSP functions. To change the DSP.C5x[***x***].DataInQStart settings, you must be familiar with the resource allocation in the CG board DSPs.

<table>
<tr>
<td>Caution:</td>
<td>Due to a DSP architectural limitation, DSPs in a pair core must maintain the same HPI queue setting. Therefore, keep the settings for CmdQ, DataInQ, and DspOutQ the same in a pair core. For example:
<code>DSP.C5x[0..1].DataInQStart = 0x2280</code>
Conferencing DSPs must use the value 0xF800. All other types of DSPs can use the default value. For example:
<code>DSP.C5x[0..1].DataInQStart = 0xF800</code></td>
</tr>
</table>

See also

DSP.C5x[x].CmdQSize, DSP.C5x[x].CmdQStart, DSP.C5x[x].DataInQSize, DSP.C5x[x].DspOutQSize, DSP.C5x[x].DspOutQStart

DSP.C5x[x].DspOutQSize

Specifies the size of the DspOut queue in the DSP memory. The DspOut queue retrieves information from the DSP.

Syntax

DSP.C5x[***x***].DspOutQSize = ***size***

x = 0-(***n***-1) (A range of DSP cores where ***n*** equals the total number of DSP cores available.)

Access

Read/Write

Type

Integer

Default

0x300

Allowed values

0x0000 – 0xFFFF

Example

```
DSP.C5x[0].DspOutQSize = 0x300
```

Details

Changing the queue location or increasing the queue size can reduce the memory available to DSP functions. To change the DSP.C5x[**x**].DspOutQSize settings, you must be familiar with the resource allocation in the CG board DSPs.

Caution:	Due to a DSP architectural limitation, DSPs in a pair core must maintain the same HPI queue setting. Therefore, keep the settings for CmdQ, DataInQ, and DspOutQ the same in a pair core. For example: `DSP.C5x[0..1].DspOutQSize = 0x300`

See also

DSP.C5x[x].CmdQSize, DSP.C5x[x].CmdQStart, DSP.C5x[x].DataInQSize, DSP.C5x[x].DataInQStart, DSP.C5x[x].DspOutQStart

DSP.C5x[x].DspOutQStart

Specifies the start of the DspOut queue in the DSP memory. The DspOut queue retrieves information from the DSP.

Syntax

DSP.C5x[***x***].DspOutQStart = ***address***

x = 0-(***n***-1) (A range of DSP cores where ***n*** equals the total number of DSP cores available.)

Access

Read/Write

Type

Integer

Default

0x2580

Allowed values

0x0000 – 0xFFFF

Example

```
DSP.C5x[0].DspOutQStart = 0x2580
```

Details

Changing the queue location or increasing the queue size can reduce the memory available to DSP functions. To change the DSP.C5x[***x***].DspOutQStart settings, you must be familiar with the resource allocation in the CG board DSPs.

Caution:	Due to a DSP architectural limitation, DSPs in a pair core must maintain the same HPI queue setting. Therefore, keep the settings for CmdQ, DataInQ, and DspOutQ the same in a pair core. For example:
	`DSP.C5x[0..1].DspOutQStart = 0x2580`
	Conferencing DSPs must use the value 0xFB00. All other types of DSPs can use the default value. For example:
	`DSP.C5x[0..1].DspOutQStart = 0xFB00`

See also

DSP.C5x[x].CmdQSize, DSP.C5x[x].CmdQStart, DSP.C5x[x].DataInQSize, DSP.C5x[x].DataInQStart, DSP.C5x[x].DspOutQSize

DSP.C5x[x].Image

Specifies a pre-linked DSP image file for CG 6060 boards.

Syntax

DSP.C5x[***x***].Image = ***filename***

x = 0-(***n***-1) (A range of DSP cores where ***n*** equals the total number of DSP cores available.)

Access

Read/Write

Type

File name

Default

None.

Allowed values

Valid DSP image file name.

Example

```
DSP.C5x[1].Image = name.c41
```

Details

Use this keyword to specify DSP images that you create. The naming convention for DSP image files is ***filename****.c41*.

Setting DSP.C5x[x].Image = NULL leaves the specified DSPs in an unbooted state.

See also

DSP.C5x[x].Libs[y], DSP.C5x[x].OS, DSP.C5x[x].NumRxTimeSlots, DSP.C5x[x].NumTxTimeSlots, DSP.C5x[x].XLaw

DSP.C5x[x].Libs[y]

Specifies the DSP library file name.

Syntax

DSP.C5x[***x***].Lib[***y***] = ***filename***

x = 0-(***n***-1) (A range of DSP cores where ***n*** equals the total number of DSP cores available.)

y = 0..15 (An index of a DSP library.)

Access

Read/Write

Type

File name

Default

cg6kliba

Allowed values

A valid DSP library file name.

Example

```
DSP.C5x[0..19].Lib[0] = cg6kliba.r41
```

Details

All DSPs must be set to either A-law or mu-law. There are two DSP operating system service libraries: *cg6klibu.r41* and *cg6kliba.r41*.

Library	Function
cg6klibu	mu-law conversion
cg6kliba	A-law conversion

See also

DSP.C5x[x].OS, DSP.C5x[x].NumRxTimeSlots, DSP.C5x[x].NumTxTimeSlots, DSP.C5x[x].XLaw

DSP.C5x[x].NumRxTimeSlots

Specifies the number of timeslots on which the DSP can receive data.

Syntax

DSP.C5x[***x***].NumRxTimeSlots = ***numberslots***

x = 0-(***n***-1) (A range of DSP cores where ***n*** equals the total number of DSP cores available.)

Access

Read/Write

Type

Integer

Default

32

Allowed values

16 | 32

Example

```
DSP.C5x[0].NumRxTimeSlots = 16
```

Details

Conferencing applications can require 16 timeslots per DSP. Decreasing the number of timeslots per DSP allows more DSP MIPS per conference call. For more information about using this keyword in conjunction with conferencing applications and determining which streams are attached to logical DSP numbers, refer to the *Dialogic® NaturalAccess™ NaturalConference™ API Service Developer's Manual*.

Caution:	Due to a DSP architectural limitation, DSPs in a pair core must maintain the same timeslot setting. For example: `DSP.C5x[0..1].NumRxTimeslots = 16`

See also

DSP.C5x[x].OS, DSP.C5x[x].NumTxTimeSlots, DSP.C5x[x].XLaw

DSP.C5x[x].NumTxTimeSlots

Specifies the number of timeslots on which the DSP can transmit data.

Syntax

DSP.C5x[***x***].NumTxTimeSlots = ***numberslots***

x = 0-(***n***-1) (A range of DSP cores where ***n*** equals the total number of DSP cores available.)

Access

Read/Write

Type

Integer

Default

32

Allowed values

16 | 32

Example

```
DSP.C5x[0].NumTxTimeSlots = 16
```

Details

Conferencing applications can require more than 16 timeslots per DSP. Decreasing the number of timeslots per DSP allows more DSP MIPS per conference call. For more information about using this keyword in conjunction with conferencing applications and determining which streams are attached to logical DSP numbers, refer to the *Dialogic® NaturalAccess™ NaturalConference™ API Service Developer's Manual.*

Caution:	Due to a DSP architectural limitation, DSPs in a pair core must maintain the same timeslot setting. For example: `DSP.C5x[0..1].NumRxTimeslots = 16`

See also

DSP.C5x[x].OS, DSP.C5x[x].NumRxTimeSlots, DSP.C5x[x].XLaw

DSP.C5x[x].OS

Specifies the digital signal processor (DSP) operating system to use on the DSP core of the current board(s).

Syntax

DSP.C5x[***x***].OS = ***filename***

x = 0-(***n***-1) (A range of DSP cores where ***n*** equals the total number of DSP cores available.)

Access

Read/Write

Type

File name

Default

dspos6u

Allowed values

Name of a valid DSP processor operating system file.

Example

```
DSP.C5x[0..31].OS = dspos6u
```

See also

DSP.C5x[x].Libs[y], DSP.C5x[x].NumRxTimeSlots, DSP.C5x[x].NumTxTimeSlots, DSP.C5x[x].XLaw

DSP.C5x[x].XLaw

Determines the DSP hardware companding mode.

Syntax

DSP.C5x[***x***].XLaw = ***mode***

x = 0-(***n***-1) (A range of DSP cores where ***n*** equals the total number of DSP cores available.)

Access

Read/Write

Type

String

Default

A_LAW

Allowed values

A_LAW | MU_LAW | NO_LAW

Example

```
DSP.C5x[0..31].XLaw = MU_LAW
```

Details

For A_LAW and MU_LAW modes, 8-bit data sent and received to or from the TSI circuit switch is converted to or from the 16-bit linear form used internally.

NO_LAW uses bits 0 - 7 of the 16-bit word for the 8-bit timeslot and fills zeros into bits 8 - 15.

The hardware companding mode must match the DSP operating system service library used. Therefore, A_LAW must use *cg6kliba.r41*, and MU_LAW must use *cg6klibu.r41*.

All DSPs within a resource pool must have the same value for this keyword.

See also

DSP.C5x[x].Libs[y]

DSPStream.SignalIdleCode[x]

Sets the idle code for timeslots on DSP signaling streams.

Syntax

DSPStream.SignalIdleCode[***x***] = ***signal_idlecode***

x = 0-(***n***-1) (0-based trunk number or range of trunk numbers where ***n*** equals the number of trunks on the board.)

Access

Read/Write

Type

Integer

Default

0x0D

Allowed values

0x00 - 0xFF

Example

```
DSPStream.SignalIdleCode[0..7] = 0x00
```

Details

The CG 6060 board signaling DSP uses this value to generate the idle pattern on the outbound signaling trunk.

All trunks must be configured with the same DSPStream.SignalIdleCode setting.

See also

DSPStream.SlotCount, DSPStream.VoiceIdleCode[x]

DSPStream.SlotCount

Specifies the number of logical timeslots on logical streams. Refer to CG 6060 switch models for more information.

Syntax

DSPStream.SlotCount = ***slotcount_number***

Access

Read/Write

Type

Integer

Default

900

Example

```
DSPStream.SlotCount = 128
```

Allowed values

1 - 1500

See also

DSPStream.SignalIdleCode[x], DSPStream.VoiceIdleCode[x]

DSPStream.VoiceIdleCode[x]

Sets the voice idle code for timeslots on the specified DSP voice streams.

Syntax

DSPStream.VoiceIdleCode[***x***] = ***voice_idlecode***

x = 0-(***n***-1) (0-based trunk number or range of trunk numbers where ***n*** equals the number of trunks on the board.)

Access

Read/Write

Type

Integer

Default

0xD5

Allowed values

0x00 - 0xFF

Example

```
DSPStream.VoiceIdleCode[0..7] = 0x7F
```

Details

All trunks must be configured with the same DSPStream.VoiceIdleCode setting.

See also

DSPStream.SignalIdleCode[x], DSPStream.SlotCount

DynamicRecordBuffers

Specifies the maximum number of overflow buffers that the board automatically allocates for recording, when recording is initiated in asynchronous board-to-host data transfer mode (using the **adiRecordAsync** function).

Syntax

DynamicRecordBuffers = ***buffercount***

Access

Read/Write

Type

Integer

Default

0

Allowed values

0 - 256

Example

```
DynamicRecordBuffers = 6
```

Details

This mode is often used to transfer data from the board to the host for near-real-time processing (for example, during voice recognition).

By default, when the application invokes **adiRecordAsync**, the board allocates a single buffer and begins filling it with recorded data. The application immediately invokes **adiSubmitRecordBuffer** to cause the board to allocate another buffer to fill when the first buffer is full. Whenever the ADI service indicates that a record buffer is full (by returning ADIEVN_RECORD_BUFFER_FULL), the application immediately invokes **adiSubmitRecordBuffer** again to cause a second buffer to be allocated. Thus at any given time there are two buffers allocated on the board: one being filled (or full, waiting to be sent), and a second one waiting to be filled (or filling).

However, at certain times both buffers can fill before the application has a chance to invoke **adiSubmitRecordBuffer** again. In this case, data can be lost.

To mitigate this problem, set DynamicRecordBuffers to the number of additional buffers that are automatically allocated by the board when **adiRecordAsync** is invoked. If the two initial buffers fill up, the additional buffers are filled one at a time. If the host falls behind, data is preserved in the additional buffers until the application can catch up.

Regardless of how a buffer is allocated, it will not be sent to the host until solicited by the host (by invoking **adiSubmitRecordBuffer**). Each buffer requires a separate request.

The size of the additional buffers is the size of the initial record buffer, requested by invoking **adiRecordAsync**. DynamicRecordBuffers does nothing unless recording is started with a buffer no larger than Buffers[1].Size. Consequently, additional buffers are allocated from the Buffers[1] buffer pool. All record buffers must be the same size. The final buffer can be smaller.

For example, suppose you set the buffer size to 200 ms (Buffers[1].Size = 1600 for mu-law encoding), and DynamicRecordBuffers = 6. These settings mean that once the first buffer is filled and sent to the host, the host can delay up to 1.4 seconds before requesting more data:

200 ms x (1 initial buffer + 6 additional buffers)

For more information about asynchronous board-to-host recorded data transfer, refer to the *ADI Service Developer's Reference Manual*.

See also

Buffers[x].Size

Echo.AutoSwitchingRefSource

Determines if the on-board switching manager performs automatic switching of the echo canceler reference stream.

Syntax

Echo.AutoSwitchingRefSource = ***setting***

Access

Read/Write

Type

String

Default

NO

Allowed values

NO | YES

Example

```
Echo.AutoSwitchingRefSource = NO
```

Details

Echo.EnableExternalPins must be set to YES to use the Echo.AutoSwitchingRefSource keyword.

Automatic switching occurs when a connection is made to a line from another line (or any other source) and when the destination line is also connected to a DSP that has echo cancellation enabled.

For example, using *swish*:

```
swish> openswitch b = agsw 0
swish> makeconnection b local:0:0 to local:65:0          # line 0 to DSP
swish> makeconnection b local:0:0 to local:1:1 duplex    # line 0 to/from line 1
```

The first connection connects DSP 0 to listen to line 0.

The second connection connects lines 0 and 1 together. The remote parties on line 0 and line 1 are able to talk to each other. DSP 0 is still monitoring line 0. This configuration is referred to as tromboning.

The switching manager automatically makes the following connection:

```
local:0:1 --> local:71:0
```

This connects line 1 to the echo canceler reference. It enables cancellation of echoes that occur on line 0 from energy originating on line 1.

Note: This keyword is not applicable for setting hardware echo cancellation values.

Echo.EnableExternalPins

Determines if the echo canceler reference and output can be switched.

Syntax

Echo.EnableExternalPins = ***setting***

Access

Read/Write

Type

String

Default

NO

Allowed values

NO | YES

Example

```
Echo.EnableExternalPins = NO
```

Details

Setting this keyword to YES enables the echo canceler reference input and the echo canceler output to be switched. They appear on output stream 70 and reference stream 71.

Note: This keyword is not applicable for setting hardware echo cancellation values.

See also

Echo.AutoSwitchingRefSource

EnableMonitor

Determines whether error messages are displayed.

Syntax

EnableMonitor = ***message_number***

Access

Read/Write

Type

Integer

Default

1

Allowed values

0 | 1

Example

```
EnableMonitor = 1
```

Details

The following entries are valid:

Value	Description
0	No error messages are displayed.
1	Error messages are displayed.

HardwareEcho.EchoChipEnabled

Enables or disables the echo cancellation hardware.

Syntax

HardwareEcho.EchoChipEnabled = ***setting***

Access

Read/Write

Type

String

Default

NO

Allowed values

NO | YES

Example

```
HardwareEcho.EchoChipEnabled = YES
```

Details

HardwareEcho.EchoChipEnabled must be set to YES to use the echo cancellation hardware.

Note: This keyword must be set to NO when you use software echo cancellation on the board.

For more information, refer to Configuring hardware echo cancellation.

See also

HardwareEcho.Trunk[x].OnOffTimeslots, HardwareEcho.Xlaw

HardwareEcho.Trunk[x].OnOffTimeslots

Enables or disables hardware echo cancellation by trunk timeslot.

Syntax

HardwareEcho.Trunk[***x***].OnOffTimeslots = ***bit mask***

x = Trunk number starting at 0.

Access

Read/Write

Type

Unsigned integer (bit mask)

Default

0xFFFFFFFF (enabled for all timeslots)

Allowed values

0 - 0xFFFFFFFF

Example

```
HardwareEcho.Trunk[ 0..3 ].OnOffTimeslots = 0xFFFFFFF0
```

In this example, hardware echo cancellation is enabled for timeslots 4..31 on trunks 0..3. Hardware echo cancellation is disabled for timeslots 0..3 on trunks 0..3.

Details

To use this option, the HardwareEcho.EchoChipEnabled keyword must be set to YES.

If HardwareEcho.EchoChipEnabled = YES and HardwareEcho.Trunk[x].OnOffTimeslots is not specified, then the hardware echo cancellation settings default to enabled for all timeslots on all trunks.

To enable or disable hardware echo cancellation for a specific timeslot, set the corresponding bit position to 1 for ON (enable) and 0 for OFF (disable).

The least significant bit (LSB) is timeslot 0 and the most significant bit (MSB) is timeslot 31. For a T1 setting, timeslots greater than 23 are ignored.

See also

HardwareEcho.EchoChipEnabled, HardwareEcho.XLaw

HardwareEcho.XLaw

Determines the echo hardware companding mode.

Syntax

HardwareEcho.XLaw = ***mode***

Access

Read/Write

Type

String

Default

A_LAW

Allowed values

A_LAW | MU_LAW

Example

```
HardwareEcho.XLaw = MU_LAW
```

Details

The hardware companding mode must match the DSP operating system service library used. Therefore, A_LAW must use *cg6kliba.r41*, and MU_LAW must use *cg6klibu.r41*.

For more information, refer to Configuring hardware echo cancellation.

Note: This keyword is not applicable when you use software echo cancellation on the board.

See also

HardwareEcho.EchoChipEnabled, HardwareEcho.Trunk[x].OnOffTimeslots

Hdlc[x].Boot

Enables or disables the HDLC channel associated with a particular trunk.

Syntax

Hdlc[***x***].Boot = ***boot***

x = 0-(***n***-1) (0-based trunk number or range of trunk numbers where ***n*** equals the number of trunks on the board.)

Access

Read/Write

Type

String

Default

NO

Allowed values

YES | NO

Example

```
Hdlc[0..3].Boot = YES
```

See also

Hdlc[x].RxTimeSlot, Hdlc[x].TxTimeSlot

Hdlc[x].RxTimeSlot

Specifies the TDM timeslot of the receiving HDLC channel for a specific trunk.

Syntax

Hdlc[***x***].RxTimeSlot = ***timeslot***

x = 0-(***n***-1) (0-based trunk number or range of trunk numbers where ***n*** equals the number of trunks on the board.)

Access

Read/Write

Type

Integer

Default

0

Allowed values

0..31 (timeslot or range of timeslots)

Example

```
Hdlc[0..3].RxTimeSlot = 16
```

See also

Hdlc[x].Boot, Hdlc[x].TxTimeSlot

Hdlc[x].TxTimeSlot

Specifies the TDM timeslot of the transmitting HDLC channel for a specific trunk.

Syntax

Hdlc[***x***].TxTimeSlot = ***timeslot***

x = 0-(***n***-1) (0-based trunk number or range of trunk numbers where ***n*** equals the number of trunks on the board.)

Access

Read/Write

Type

Integer

Default

0

Allowed values

0..31 (timeslot or range of timeslots)

Example

```
Hdlc[0..3].TxTimeSlot = 16
```

See also

Hdlc[x].Boot, Hdlc[x].RxTimeSlot

IPC.AddRoute[x].DestinationAddress

Specifies the IPv4 address of the Ethernet interface.

Syntax

IPC.AddRoute[***x***].DestinationAddress = ***IP_address***

x = index of the route entry

Access

Read/Write

Type

IP address

Default

255.255.255.255

Allowed values

Valid IP address.

Example

```
IPC.AddRoute[1].DestinationAddress = 198.62.139.32
```

Details

You can specify up to 32 destination addresses.

For more information, refer to Configuring IPv4 Ethernet connections.

See also

IPC.AddRoute[x].GatewayAddress, IPC.AddRoute[x].Interface, IPC.AddRoute[x].Mask

IPC.AddRoute[x].GatewayAddress

Specifies the IPv4 address of the router.

Syntax

IPC.AddRoute[***x***].GatewayAddress = ***IP_address***

x = index of the route entry

Access

Read/Write

Type

IP address

Default

255.255.255.255

Allowed values

Valid IP address.

Example

```
IPC.AddRoute[1].GatewayAddress = 198.62.139.1
```

Details

This keyword cannot be used in conjunction with the IPC.AddRoute[x].Interface keyword.

For more information, refer to Configuring IPv4 Ethernet connections.

See also

IPC.AddRoute[x].DestinationAddress, IPC.AddRoute[x].Mask

IPC.AddRoute[x].Interface

Specifies the Ethernet interface (1 or 2) associated with IPv4 connections.

Syntax

IPC.AddRoute[***x***].Interface = ***Ethernet_number***

x = index of the route entry

Access

Read/Write

Type

Integer

Default

2

Allowed values

1 | 2

Example

```
IPC.AddRoute[1].Interface = 1
```

Details

This keyword cannot be used in conjunction with the IPC.AddRoute[x].GatewayAddress keyword.

For more information, refer to Configuring IPv4 Ethernet connections.

See also

IPC.AddRoute[x].DestinationAddress, IPC.AddRoute[x].Mask

IPC.AddRoute[x].Mask

Specifies a subnet mask for the IPv4 address specified in IPC.AddRoute[x].DestinationAddress. For more information, refer to Configuring IPv4 Ethernet connections.

Syntax

IPC.AddRoute[***x***].Mask = ***subnet_mask***

x = index of the route entry

Access

Read/Write

Type

IP mask

Default

255.255.255.255

Allowed values

Valid subnet mask.

Example

```
IPC.AddRoute[1].Mask = 255.255.255.0
```

See also

IPC.AddRoute[x].GatewayAddress, IPC.AddRoute[x].Interface

IPC.AddRoute[x].VlanTag

Specifies a VLAN tag to be added to all packets sent to the IPv4 subnet specified by the address and mask in IPC.AddRoute[x].DestinationAddress and IPC.AddRoute[x].Mask.

Syntax

IPC.AddRoute[***x***].VlanTag = ***tag***

x = index of the route entry

Access

Read/Write

Type

Integer

Default

0

Allowed values

0 - 4094

Example

IPC.AddRoute[1].VlanTag = 5

Details

A VLAN tag is not valid with route definitions (GatewayAddress must not be set.)

For more information, refer to Configuring IPv4 Ethernet connections.

See also

IPC.AddRoute[x].Interface, IPC.AddRoute[x].Mask

IPv6.Link[x].Enable

Enables or disables IPv6 on the specified Ethernet interface. For more information, refer to Configuring IPv6 Ethernet connections.

Syntax

IPv6.Link[***x***].Enable = ***mode***

x = index of the link entry

Access

Read/Write

Type

String

Default

NO

Allowed values

YES | NO

Example

```
IPv6.Link[1].Enable = YES
```

See also

IPv6.Link[x].IPSec, IPv6.Link[x].EnablePing

IPv6.Link[x].EnablePing

Enables or disables IPv6 PING on the specified Ethernet interface.

Syntax

IPv6.Link[***x***].EnablePing = ***mode***

x = index of the link entry

Access

Read/Write

Type

String

Default

YES

Allowed values

YES | NO

Example

```
IPv6.Link[1].EnablePing = YES
```

Details

By default, PING is enabled for IPv6 Ethernet interfaces. For more information, refer to Configuring IPv6 Ethernet connections.

See also

IPv6.Link[x].Enable, IPv6.Link[x].IPSec

IPv6.Link[x].HopLimit

Specifies the default IPv6 hop limit value for the specified Ethernet interface.

Syntax

IPv6.Link[***x***].HopLimit = ***hoplimit***

x = index of the link entry

Access

Read/Write

Type

Integer

Default

128

Allowed values

1 - 255

Example

```
IPv6.Link[1].HopLimit = 255
```

Details

Typically, this value should be left at its default value of 128. For information about hop limits, refer to the *RFC 2460 - Internet Protocol, Version 6 (IPv6) Specification*. For information about using board keywords to configure IPv6 Ethernet interfaces, refer to Configuring IPv6 Ethernet connections.

See also

IPv6.Link[x].ICMPRateLimit, IPv6.Link[x].NDAttempts

IPv6.Link[x].ICMPRateLimit

Specifies the IPv6 ICMP rate limit for the specified Ethernet interface.

Syntax

IPv6.Link[***x***].ICMPRateLimit = ***icmplimit***

x = index of the link entry

Access

Read/Write

Type

String

Default

100

Allowed values

0 - 9999

Example

```
IPv6.Link[1].ICMPRateLimit = 5250
```

Details

For information about ICMP, refer to *RFC 2463 Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6).* For information about using board keywords to configure IPv6 Ethernet interfaces, refer to Configuring IPv6 Ethernet connections.

See also

IPv6.Link[x].HopLimit, IPv6.Link[x].NDAttempts

IPv6.Link[x].IPSec

Enables or disables IPSec for IPv6 on the specified Ethernet interface.

Syntax

IPv6.Link[***x***].IPSec = ***mode***

x = index of the link entry

Access

Read/Write

Type

String

Default

NO

Allowed values

YES | NO

Example

```
IPv6.Link[1].IPSec = YES
```

Details

For information about using IPsec with the CG board IPv6 stack, refer to cgsetkey - Configuring IPv6 security keys and policies and *RFC 2401 Security Architecture for the Internet Protocol (IPSec)*. For information about using board keywords to configure IPv6 Ethernet interfaces, refer to Configuring IPv6 Ethernet connections.

See also

IPv6.Link[x].ICMPRateLimit, IPv6.Link[x].HopLimit

IPv6.Link[x].MTU

Sets the IPv6 maximum transmission unit (MTU) for the specified Ethernet interface.

Syntax

IPv6.Link[***x***].MTU = ***mtuvalue***

x = index of the link entry

Access

Read/Write

Type

Integer

Default

1500

Allowed values

1280 - 1500

Example

```
IPv6.Link[1].MTU = 1280
```

Details

Typically, this value should be left at its default of 1500 bytes. For information about maximum transmission units, refer to the *RFC 2460 Internet Protocol, Version 6 (IPv6) Specification*. For information about using board keywords to configure IPv6 Ethernet interfaces, refer to Configuring IPv6 Ethernet connections.

See also

IPv6.Link[x].HopLimit

IPv6.Link[x].NDAttempts

Specifies the neighbor discovery attempt (NDA) limit for the specified Ethernet interface.

Syntax

IPv6.Link[***x***].NDAttempts = ***ndalimit***

x = index of the link entry

Access

Read/Write

Type

Integer

Default

3

Allowed values

0 - 99

Example

```
IPv6.Link[1].NDAttempts = 55
```

Details

This keyword configures the number of neighbor solicitations sent to a particular neighbor address prior to determining that the neighbor is unreachable. This value should typically be left at its default value of 3 attempts. For information about neighbor discovery, refer to *RFC 2461 - Neighbor Discovery for IP Version 6*. For information about using board keywords to configure IPv6 Ethernet interfaces, refer to Configuring IPv6 Ethernet connections.

See also

IPv6.Link[x].NDRetranTimer, IPv6.Link[x].ICMPRateLimit

IPv6.Link[x].NDReachabilityTimer

Specifies the neighbor discovery reachability timer (in milliseconds) for the specified Ethernet interface.

Syntax

IPv6.Link[***x***].NDReachabilityTimer = ***ndreach***

x = index of the link entry

Access

Read/Write

Type

Integer

Default

30000

Allowed values

1000 - 6000000

Example

```
IPv6.Link[1].NDReachabilityTimer = 60000
```

Details

This keyword configures how often the IPv6 stack re-verifies that a particular neighbor is reachable. For information about neighbor discovery, refer to *RFC 2461 - Neighbor Discovery for IP Version 6*. For information about using board keywords to configure IPv6 Ethernet interfaces, refer to Configuring IPv6 Ethernet connections.

See also

IPv6.Link[x].NDRetranTimer, IPv6.Link[x].NDAttempts

IPv6.Link[x].NDRetranTimer

Specifies the neighbor discovery retransmission timer (in milliseconds) for the specified Ethernet interface.

Syntax

IPv6.Link[***x***].NDRetranTimer = ***ndretran***

x = index of the link entry

Access

Read/Write

Type

Integer

Default

1000

Allowed values

100 - 60000

Example

```
IPv6.Link[1].NDRetranTimer = 2000
```

Details

This keyword configures how often the IPv6 stack retransmits neighbor solicitations when corresponding neighbor advertisements are not received. For information about neighbor discovery, refer to *RFC 2461 - Neighbor Discovery for IP Version 6*. For information about using board keywords to configure IPv6 Ethernet interfaces, refer to Configuring IPv6 Ethernet connections.

See also

IPv6.Link[x].NDAttempts, IPv6.Link[x].NDReachabilityTimer

Location.PCI.Bus

Specifies the PCI logical bus location of the board.

Syntax

Location.PCI.Bus = ***busnum***

Access

Read/Write

Type

Integer

Default

0

Example

```
Location.PCI.Bus = 2
```

Allowed values

0 - 256

Details

A PCI bus and slot number identifies every slot in the system. Identify a board by specifying its logical bus and slot number.

A PCI board's address and interrupt is automatically set by the system. This statement along with the Location.PCI.Slot keyword simply assigns the board number to the physical board.

pciscan can be used to determine the PCI logical bus and slot assigned for all NaturalAccess PCI boards in the system. Refer to the *Dialogic® NaturalAccess™ OAM System Developer's Manual* for information about *pciscan*.

Note: The Bus setting in the *oamsys* system configuration file overrides this keyword.

Location.PCI.Bus is mandatory for PCI boards.

Location.PCI.Slot

Defines the logical slot location of the board on the PCI bus.

Syntax

Location.PCI.Slot = ***slotnum***

Access

Read/Write

Type

Integer

Default

0

Example

```
Location.PCI.Slot = 5
```

Allowed values

0 - 9999

Details

A PCI bus and slot number identifies every slot in the system. Identify a board by specifying its bus and slot number.

A PCI board's address and interrupt is automatically set by the system. This statement along with Location.PCI.Bus assigns the board number to the physical board.

pciscan can be used to determine the PCI bus and slot assigned for all NaturalAccess PCI boards in the system. Refer to the *Dialogic® NaturalAccess™ OAM System Developer's Manual* for information about *pciscan*.

Note: The Slot setting in the *oamsys* system configuration file overrides this keyword.

Location.PCI.Slot is mandatory for PCI boards.

MaxChannels

Specifies the maximum number of channels to allocate on the board. A channel is needed for each instance of the ADI service that is opened by an application.

Syntax

MaxChannels = ***num_channels***

Access

Read/Write

Type

Integer

Default

900

Allowed values

0 - 1500

Example

```
MaxChannels = 256
```

Details

The number of channels affects memory requirements. If this statement is omitted, the OAM API assigns an appropriate value for the board type.

See also

Buffers[x].Size

Name

Specifies the name assigned to the CG 6060 board.

Syntax

Name = ***board_name***

Access

Read/Write

Type

String

Default

CG_6060

Allowed values

String up to 64 characters long.

Example

```
Name = Brd1
```

Details

This setting is assigned by the OAM Supervisor if the user does not specify a name in the system configuration file. This setting is guaranteed to be unique within the chassis.

See also

Number

NetworkInterface.Ethernet[x].MAC_Address

Specifies the MAC address.

Syntax

NetworkInterface.Ethernet[***x***].MAC_Address = ***MAC_address***

x varies depending on the number of Ethernet interfaces on the board. The CG 6060 board has two Ethernet interfaces. The first one is the primary Ethernet interface.

Access

Read-only

Type

String

Allowed values

Not applicable.

Details

There are two MAC addresses because the board has two Ethernet interfaces.

After you boot a CG board, you can obtain the MAC address information at the board level. For example, after booting Board 1, use *oaminfo* and enter the following command:

```
oaminfo -b1 -k NetworkInterface.Ethernet
```

NetworkInterface.T1E1[x].AlarmMode

For T1 interfaces, specifies if alarm conditions are declared immediately or after a specific period of time when an alarm event (for example, out of frame) occurs. This keyword is not applicable for E1 interfaces.

Syntax

NetworkInterface.T1E1[***x***].AlarmMode = ***mode***

x = 0-(***n***-1) (0-based trunk number or range of trunk numbers where ***n*** equals the number of trunks on the board.)

Access

Read/write

Type

String

Default

T1_US

Allowed values

The allowed values are:

Value	Alarm	Time to clear	Time to set
T1_US	Red	In frame condition for approximately 16.5 seconds	Out of frame condition for approximately 2.55 seconds
	Yellow	Far end alarm clear for approximately 0.5 seconds	Far end alarm active for approximately 0.5 seconds
T1_G706	Red	Immediate	Immediate
	Yellow	Immediate	Immediate

The T1_G706 value is based on the G.706 ITU recommendation that signals be debounced. It does not specify an exact amount of time to wait.

Example

```
NetworkInterface.T1E1[x].AlarmMode = T1_US
```

See also

NetworkInterface.T1E1[x].CRCMFMode, NetworkInterface.T1E1[x].FrameType, NetworkInterface.T1E1[x].Impedance, NetworkInterface.T1E1[x].Length, NetworkInterface.T1E1[x].LineCode, NetworkInterface.T1E1[x].SignalingType, NetworkInterface.T1E1[x].Type

NetworkInterface.T1E1[x].CRCMFMode

Specifies whether or not the board performs CRC signal checking.

Syntax

NetworkInterface.T1E1[***x***].CRCMFMode = ***mode***

x = 0-(***n***-1) (0-based trunk number or range of trunk numbers where ***n*** equals the number of trunks on the board.)

Access

Read/Write

Type

String

Default

C44ON

Allowed values

C44ON | C44OFF

Example

```
NetworkInterface.T1E1[0].CRCMFMode = C44OFF
```

See also

NetworkInterface.T1E1[x].D_Channel, NetworkInterface.T1E1[x].FrameType, NetworkInterface.T1E1[x].Impedance, NetworkInterface.T1E1[x].Length, NetworkInterface.T1E1[x].LineCode, NetworkInterface.T1E1[x].SignalingType, NetworkInterface.T1E1[x].Type

NetworkInterface.T1E1[x].D_Channel

Specifies whether the trunk has a primary D channel with ISDN running on it.

Syntax

NetworkInterface.T1E1[***x***].D_Channel = ***setting***

x = 0-(***n***-1) (0-based trunk number or range of trunk numbers where ***n*** equals the number of trunks on the board.)

Access

Read/Write

Type

String

Default

ISDN_NONE

Allowed values

ISDN | ISDN_NONE

Example

```
NetworkInterface.T1E1[0..3].D_Channel = ISDN
```

Details

If NetworkInterface.T1E1[x].D_Channel = ISDN for any of the trunks on a board, a configuration message is sent to the ISDN stack on that board to initialize that stack. You must initialize the ISDN stack for any trunk that has a D channel. You must also enable the HDLC controller for that trunk by setting the NetworkInterface.T1E1[x].SignalingType keyword to the value PRI.

In the case of an NFAS group with a backup D channel, specify this keyword for the primary D channel only. The backup D channel is specified using the NetworkInterface.T1E1[x].ISDN.D_Channel_Backup_Trunk keyword.

NetworkInterface.T1E1[x].D_Channel is required in any configuration where NFAS is used. For information about NFAS groups, refer to the *Dialogic® NaturalAccess™ ISDN API Installation Manual*.

See also

NetworkInterface.T1E1[x].CRCMFMode, NetworkInterface.T1E1[x].FrameType, NetworkInterface.T1E1[x].Impedance, NetworkInterface.T1E1[x].Length, NetworkInterface.T1E1[x].LineCode, NetworkInterface.T1E1[x].Type

NetworkInterface.T1E1[x].FrameType

Defines the T1 or E1 trunk framing format for current boards or current trunks.

Syntax

NetworkInterface.T1E1[***x***].FrameType = ***frame_format***

x = 0-(***n***-1) (0-based trunk number or range of trunk numbers where ***n*** equals the number of trunks on the board.)

Access

Read/Write

Type

String

Default

CEPT

Allowed values

D4 | ESF | CEPT

Example

```
NetworkInterface.T1E1[0..3].FrameType = D4
```

Details

The following formats are available for T1 trunks:

Format	Description
D4	Standard superframe formatting
ESF	Extended superframe formatting

The following format is available for E1 trunks:

Format	Description
CEPT	Framing format conforming to ITU recommendation G.703 for PCM 30 (30 channels with channel associated signaling).

For more information, refer to Framing.

See also

NetworkInterface.T1E1[x].CRCMFMode, NetworkInterface.T1E1[x].D_Channel, NetworkInterface.T1E1[x].Impedance, NetworkInterface.T1E1[x].Length, NetworkInterface.T1E1[x].LineCode, NetworkInterface.T1E1[x].SignalingType, NetworkInterface.T1E1[x].Type

NetworkInterface.T1E1[x].Impedance

Specifies the type of cable connecting a CG 6060 board to the T1 or E1 network.

Syntax

NetworkInterface.T1E1[***x***].Impedance = ***cable_type***

x = 0-(***n***-1) (0-based trunk number or range of trunk numbers where ***n*** equals the number of trunks on the board.)

Access

Read/Write

Type

String

Default

G703_120_OHM

Allowed values

DSX1 | G703_75_OHM | G703_120_OHM | HIGH_IMPEDANCE

Example

```
NetworkInterface.T1E1[0..3].Impedance = G703_120_OHM
```

Details

Trunk impedance on the CG 6060 is controlled through a board configuration file. Since trunks are configured in pairs, the configuration setting must be identical for both trunks in a given pair. Trunks are paired as follows:

- 0 and 2
- 1 and 3

Valid settings are:

Setting	Description
DSX1 (T1 only)	For DSX1 cables (up to 655 feet - short haul).
G703_75_OHM (E1 only)	For G.703-standard 75 ohm cables.
G703_120_OHM (E1 only)	For G.703-standard 120 ohm cables.
HIGH_IMPEDANCE (T1 or E1)	Reserved for T1 or E1 trunk monitoring applications.

Refer to Configuring the T1 or E1 interface for more information.

See also

NetworkInterface.T1E1[x].CRCMFMode, NetworkInterface.T1E1[x].D_Channel, NetworkInterface.T1E1[x].FrameType, NetworkInterface.T1E1[x].Length, NetworkInterface.T1E1[x].LineCode, NetworkInterface.T1E1[x].SignalingType, NetworkInterface.T1E1[x].Type

NetworkInterface.T1E1[x].ISDN.D_Channel_Backup_Trunk

Specifies the 0-based trunk to carry the backup D channel for this NFAS group.

Syntax

NetworkInterface.T1E1[***x***].ISDN.D_Channel_Backup_Trunk = ***trunk***

x = 0-(***n***-1) (0-based trunk number or range of trunk numbers where ***n*** equals the number of trunks on the board.)

Access

Read/Write

Type

Integer

Default

-1 (no backup D channel)

Allowed values

-1 to 3

Example

```
NetworkInterface.T1E1[0].ISDN.D_Channel_Backup_Trunk = 2
```

This example specifies that the backup trunk for T1E1[0] is trunk 2.

Details

The specified trunk must be a different trunk on the same board as the primary D channel interface and must be part of the same NFAS group.

When using this keyword, NetworkInterface.T1E1[x].D_Channel must be set to ISDN for the trunk (***x***) bearing the primary D channel.

See also

NetworkInterface.T1E1[x].ISDN.NFAS_Member[y].Board,
NetworkInterface.T1E1[x].ISDN.NFAS_Member[y].NAI,
NetworkInterface.T1E1[x].ISDN.NFAS_Member[y].Trunk,
NetworkInterface.T1E1[x].ISDN.NFASGroup

NetworkInterface.T1E1[x].ISDN.NFAS_Member[y].Board

Specifies the board number (as defined in *oamsys.cfg*) on which this NFAS member resides.

Syntax

NetworkInterface.T1E1[***x***].ISDN.NFAS_Member[***y***].Board = ***setting***

x = 0-(***n***-1) (0-based trunk number or range of trunk numbers where ***n*** equals the number of trunks on the board.)

y = index of the NFAS group member

Access

Read/Write

Type

Integer

Default

For every member of an NFAS group, this keyword must be set in the board keyword file of the board where the D channel resides.

Allowed values

Any board number, as defined in the OAM system configuration file *oamsys.cfg*.

Example

```
NetworkInterface.T1E1[0].ISDN.NFAS_Member[1].Board = 0
```

Details

This board number must match the board number specified in the OAM system configuration file.

When using this keyword, NetworkInterface.T1E1[x].D_Channel must be set to ISDN for the trunk (***x***) bearing the primary D channel.

For information about the system configuration file, refer to the *Dialogic® NaturalAccess™ OAM System Developer's Manual*. For information about configuring NaturalAccess ISDN software, refer to the *Dialogic® NaturalAccess™ ISDN API Installation Manual*.

See also

NetworkInterface.T1E1[x].ISDN.D_Channel_Backup_Trunk, NetworkInterface.T1E1[x].ISDN.NFAS_Member[y].NAI, NetworkInterface.T1E1[x].ISDN.NFAS_Member[y].Trunk, NetworkInterface.T1E1[x].ISDN.NFASGroup

NetworkInterface.T1E1[x].ISDN.NFAS_Member[y].NAI

Identifies the Network Access Identifier (NAI) for this NFAS member.

Syntax

NetworkInterface.T1E1[***x***].ISDN.NFAS_Member[***y***].NAI = ***nai***

x = 0-(***n***-1) (0-based trunk number or range of trunk numbers where ***n*** equals the number of trunks on the board.)

y = index of the NFAS group member

Access

Read/Write

Type

Integer

Default

For every member of an NFAS group, this keyword must be set in the board keyword file of the board where the D channel resides.

Allowed values

0 - 127

Example

```
NetworkInterface.T1E1[0].ISDN.NFAS_Member[1].NAI = 4
```

Details

ISDN applications use this number to refer to the trunk within an NFAS group. The NAI of each trunk in an NFAS group must be unique.

If an NFAS group is not defined, every D channel controls only one trunk (the trunk where the D channel resides). In this case, the ISDN stack sets the NAI to be equal to the trunk number. If you want the NAI for an interface to be different from the trunk number, define an NFAS group consisting of one member and explicitly set the NAI trunk and board numbers for this member.

When using this keyword, NetworkInterface.T1E1[x].D_Channel must be set to ISDN for the trunk (***x***) bearing the primary D channel.

For information about NFAS groups, refer to the *Dialogic® NaturalAccess™ ISDN Software Installation Manual*.

See also

NetworkInterface.T1E1[x].ISDN.D_Channel_Backup_Trunk, NetworkInterface.T1E1[x].ISDN.NFAS_Member[y].Board, NetworkInterface.T1E1[x].ISDN.NFAS_Member[y].Trunk, NetworkInterface.T1E1[x].ISDN.NFASGroup

NetworkInterface.T1E1[x].ISDN.NFAS_Member[y].Trunk

Specifies the zero-based trunk number of a member trunk of an NFAS group.

Syntax

NetworkInterface.T1E1[***x***].ISDN.NFAS_Member[***y***].Trunk = ***trunk***

x = 0-(***n***-1) (0-based trunk number or range of trunk numbers where ***n*** equals the number of trunks on the board.)

y = index of the NFAS group member

Access

Read/Write

Type

Integer

Default

For every member of an NFAS group, this keyword must be set in the board keyword file of the board where the D channel resides.

Allowed values

0 - 3

Example

```
NetworkInterface.T1E1[0].ISDN.NFAS_Member[1].Trunk = 0
```

Details

When using this keyword, NetworkInterface.T1E1[x].D_Channel must be set to ISDN for the trunk (***x***) bearing the primary D channel.

For information about setting up NFAS groups, refer to the *Dialogic® NaturalAccess™ ISDN Software Installation Manual*.

See also

NetworkInterface.T1E1[x].ISDN.D_Channel_Backup_Trunk, NetworkInterface.T1E1[x].ISDN.NFAS_Member[y].Board, NetworkInterface.T1E1[x].ISDN.NFAS_Member[y].NAI, NetworkInterface.T1E1[x].ISDN.NFASGroup

NetworkInterface.T1E1[x].ISDN.NFASGroup

Specifies the NFAS group number.

Syntax

NetworkIInterface.T1E1[***x***].ISDN.NFASGroup = ***groupnum***

x = 0-(***n***-1) (0-based trunk number or range of trunk numbers where ***n*** equals the number of trunks on the board.)

Access

Read/Write

Type

Integer

Default

For every NFAS group, this keyword must be set in the board keyword file of the board where the D channel resides.

Allowed values

0 - 255

Example

```
NetworkInterface.T1E1[3].ISDN.NFASGroup = 0
```

Details

If NetworkInterface.T1E1[x].D_Channel is set to ISDN and NetworkInterface.T1E1[x].ISDN.NFASGroup is not specified, this trunk runs ISDN, but is not part of an NFAS group.

This keyword is valid only on a trunk where NetworkInterface.T1E1[x].D_Channel = ISDN.

For information about NFAS groups, refer to the *Dialogic® NaturalAccess™ ISDN Software Installation Manual*.

See also

NetworkInterface.T1E1[x].ISDN.D_Channel_Backup_Trunk, NetworkInterface.T1E1[x].ISDN.NFAS_Member[y].Board, NetworkInterface.T1E1[x].ISDN.NFAS_Member[y].NAI, NetworkInterface.T1E1[x].ISDN.NFAS_Member[y].Trunk

NetworkInterface.T1E1[x].Length

Specifies the length (in feet) of the cable connecting the board to the network so that the framer can adjust the pulse shape accordingly. Only applicable in T1 mode.

Syntax

NetworkInterface.T1E1[***x***].Length = ***length***

x = 0-(***n***-1) (0-based trunk number or range of trunk numbers where ***n*** equals the number of trunks on the board.)

Access

Read/Write

Type

Integer

Default

0

Allowed values

0 - 655 (feet)

Example

```
NetworkInterface.T1E1[0..3].Length = 0
```

See also

NetworkInterface.T1E1[x].CRCMFMode, NetworkInterface.T1E1[x].D_Channel, NetworkInterface.T1E1[x].FrameType, NetworkInterface.T1E1[x].Impedance, NetworkInterface.T1E1[x].LineCode, NetworkInterface.T1E1[x].SignalingType, NetworkInterface.T1E1[x].Type

NetworkInterface.T1E1[x].LineCode

Specifies the ones density maintenance method used on the trunk line to maintain a clear channel transmission.

Syntax

NetworkInterface.T1E1[***x***].LineCode = ***code***

x = 0-(***n***-1) (0-based trunk number or range of trunk numbers where ***n*** equals the number of trunks on the board.)

Access

Read/Write

Type

String

Default

HDB3

Note: For T1 trunks you must specify a value, typically B8ZS.

Allowed values

AMI | B8ZS | HDB3 | AMI_ZCS | AMI_BELL | AMI_DDS | AMI_GTE

Example

```
NetworkInterface.T1E1[0..3].LineCodeCode = AMI
```

Details

For more information, refer to AMI, ones density, and zero code suppression.

The valid T1 trunk formats are:

Format	Definition
AMI	Alternate mark inversion - standard line coding with no zero code suppression.
B8ZS	Binary 8-zero code suppression that uses patterns of bipolar violations to replace zero data bytes; especially useful for clear channel transmission.
AMI_ZCS	AMI with jammed bit 7 zero code suppression.
AMI_BELL	Same as AMI_ZCS.
AMI_DDS	AMI with zero data byte replaced with 1001 1000.
AMI_GTE	AMI with jammed bit 8 zero code suppression, except in signaling frames when jammed bit 7 is used if the signaling bit is zero.

The valid E1 trunk formats are:

Format	Definition
AMI	Alternate mark inversion - standard line coding with no zero code suppression.
HDB3	High density bipolar 3 code that uses patterns of bipolar violations to replace sequences of 4 zero data bits to maintain ones density on clear channel transmission.

NetworkInterface.T1E1[***x***].LineCode is optional.

See also

NetworkInterface.T1E1[x].CRCMFMode, NetworkInterface.T1E1[x].D_Channel, NetworkInterface.T1E1[x].FrameType, NetworkInterface.T1E1[x].Impedance, NetworkInterface.T1E1[x].Length, NetworkInterface.T1E1[x].Type

NetworkInterface.T1E1[x].SignalingType

Determines how voice and signaling information is routed to and from the T1 or E1 trunk and DSP resources.

Syntax

NetworkInterface.T1E1[***x***].SignalingType = ***type***

x = 0-(***n***-1) (0-based trunk number or range of trunk numbers where ***n*** equals the number of trunks on the board.)

Access

Read/Write

Type

String

Default

CAS

Allowed values

CAS | PRI | RAW

Example

```
NetworkInterface.T1E1[0..3].SignalingType = CAS
```

Details

The switch model for the board changes based on the NetworkInterface.T1E1[***x***].SignalingType setting.

NetworkInterface.T1E1[***x***].SignalingType can be set to any of the following values:

This value...	Makes settings appropriate for...
CAS	Channel associated signaling. This is the default value.
PRI	Primary-rate ISDN. There are 30 bearer channels for E1 and 23 bearer channels for T1.
RAW	Primary-rate ISDN with no signaling information (that is, no D channel). Connects all channels as voice channels (B channels) and turns off robbed bit signaling. There are 31 bearer channels for E1 and 24 bearer channels for T1.

NetworkInterface.T1E1[***x***].SignalingType is required for ISDN configurations. If no NetworkInterface.T1E1[***x***].SignalingType statement is provided in ISDN configurations, an ISDN_BAD_NAI error can be returned, even if the NAI statement is correct.

See also

NetworkInterface.T1E1[x].CRCMFMode, NetworkInterface.T1E1[x].D_Channel, NetworkInterface.T1E1[x].FrameType, NetworkInterface.T1E1[x].Impedance, NetworkInterface.T1E1[x].Length, NetworkInterface.T1E1[x].LineCode, NetworkInterface.T1E1[x].Type

NetworkInterface.T1E1[x].Type

Specifies the trunk type for each trunk on the board. This setting must be the same for all active trunks.

Syntax

NetworkInterface.T1E1[***x***].Type = ***trunk_type***

x = 0-(***n***-1) (0-based trunk number or range of trunk numbers where ***n*** equals the number of trunks on the board.)

Access

Read/Write

Type

String

Default

NONE

Allowed values

T1 | E1 | NONE

Example

```
NetworkInterface.T1E1[0..3].Type = E1
```

Details

If NetworkInterface.T1E1[***x***].Type is not specified, no trunk type is associated with the board.

See also

NetworkInterface.T1E1[x].CRCMFMode, NetworkInterface.T1E1[x].D_Channel, NetworkInterface.T1E1[x].FrameType, NetworkInterface.T1E1[x].Impedance, NetworkInterface.T1E1[x].Length, NetworkInterface.T1E1[x].LineCode, NetworkInterface.T1E1[x].SignalingType

Number

Specifies a logical board number for this board.

Syntax

Number = ***board_number***

Access

Read/Write

Type

Integer

Default

0

Allowed values

Non-zero integer.

Example

```
Number = 1
```

Details

This setting is assigned by the OAM Supervisor if the user does not specify a number in the system configuration file. This setting is guaranteed to be unique within the chassis.

See also

Name

Products[x]

At the plug-in level, indicates the product supported by the plug-in.

Syntax

Products[***x***] = ***product_type***

Access

Read-only (CG plug-in level)

Type

String

Allowed values

Not applicable.

See also

Name

Resource[x].Definitions

Provides a relational string of data processing functions (DPFs) that describes the functionality that can occur on a single port and how the DSP functions execute in relation to each other.

Syntax

Resource[***x***].Definitions = ***definition***

x = 0..9 (index of the associated resource pool)

Access

Read/Write

Type

String

Default

None.

Allowed values

Any valid DPF name or identifier.

Example

```
Resource[0].Definitions = ( echo.ln20_apt100 | dtmf.det_all )
```

or

```
Resource[0].Definitions = ( echo.ln20_apt100 & dtmf.det_all )
```

Details

The DPFs in this string specify the functions that execute on the DSPs and whether they execute simultaneously.

The notation used to associate functions that run simultaneously is the AND operator (&). The notation used to associate functions that do not run simultaneously is the OR operator (|).

These operators are used with parentheses to determine the relationship between the functions and the calculation of DSP resources. The AND-OR-parentheses notation is used to optimize the allocation of resources by specifying to the board the worst-case resource usage over the duration of the call.

The Resource[***x***].Definitions keyword specifies the processing functions that are available to applications during the life of a call or channel. For example, if you expect to run echo cancellation at any time on the board, you must specify an echo DPF using this keyword. Since echo runs at the same time as the decoder and encoder in the universal ports full duplex implementation, the Resource string must combine echo (using the AND operator) with the decoder and the encoder.

Note: Use no more than one occurrence of echo cancellation in the Resource[***x***].Definitions string.

It is not necessary for you to specify the DPFs for the trunk control programs (TCPs) with the Resource[***x***].Definitions keyword. To use a TCP, specify the name of the TCP(s) to use with the Resource[x].TCPs keyword. The on-board resource manager uses the OR operation to compare the TCPs with the DPFs specified in the Resource[***x***].Definitions string.

The structure of a Resource[***x***].Definitions keyword must start with an open parenthesis and end with a matching close parenthesis. For example:

```
(( dtmf.det_all | echo.ln20_apt25) & \
( oki.rec_24 | tone.gen ))
```

The following example is not correct:

```
( dtmf.det_all | echo.ln20_apt25) & \
( oki.rec_24 | tone.gen )
```

Caution:	If you have not specified a DPF in the Resource[***x***].Definitions keyword and you attempt to create or start the DPF, it fails. All DPFs that you plan to use must be specified in Resource[***x***].Definitions.

For more information, refer to Managing board DSP resources.

See also

Resource[x].DSPs, Resource[x].Name, Resource[x].Size

Resource[x].DSPs

Specifies the DSPs associated with a resource pool (identified by ***x***).

Syntax

Resource[***x***].DSPs = ***dspIDnumber***

x = 0..9 (index of the associated resource pool)

Access

Read/Write

Type

Integer

Default

None.

Allowed values

A list of DSP numbers.

Example

```
Resource[0].DSPs = 1 2 3 4 5 6 7 8 9 10 11 12
```

If you use the API rather than the *oamcfg* utility to set the DSPs associated with a resource pool, you must set the values individually. For example:

```
oamSetKeyword(hObj, "Resource[0].DSPs[0]", "0");
oamSetKeyword(hObj, "Resource[0].DSPs[1]", "1");
```

Details

The CG board plug-in determines the image to load to the DSPs. To determine the image, the CG board plug-in uses the data processing modules (DPMs) specified by the Resource[x].Definitions keyword and the definitions associated with the TCPs found in the Resource[x].TCPs keyword.

For more information, refer to Managing board DSP resources.

See also

Resource[x].Name, Resource[x].Size

Resource[x].Name

Specifies a name to associate with a resource pool (identified by ***x***). For more information, refer to Managing board DSP resources.

Syntax

Resource[***x***].Name = ***label***

x = 0..9 (index of the associated resource pool)

Access

Read/Write

Type

String

Default

None.

Allowed values

A character string up to ten characters long.

Example

```
Resource[0].Name = RSC1
```

See also

Resource[x].Definitions, Resource[x].DSPs, Resource[x].Size, Resource[x].TCPs

Resource[x].Size

Specifies the number of channels or ports for the resource pool reserved by the on-board DSP resource manager.

Syntax

Resource[***x***].Size = ***number***

x = 0..9 (index of the associated resource pool)

Access

Read/Write

Type

Integer

Default

None.

Allowed values

0 - 999

Example

```
Resource[0].Size = 120
```

Details

If this value is 0, a resource is defined, but no resources are pre-allocated at boot time.

For more information, refer to Managing board DSP resources.

See also

Resource[x].Definitions, Resource[x].DSPs, Resource[x].Name, Resource[x].TCPs

Resource[x].StartTimeSlot

Specifies the starting timeslot for a pool.

Syntax

Resource[***x***].StartTimeSlot = ***y***

x = 0..9 (index of the associated resource pool)

y = the first timeslot of a range to associate with this pool

Access

Read/Write

Type

Integer

Default

0

Allowed values

0..max timeslot - 1

Example

```
Resource[1].StartTimeSlot = 0
```

Details

The number of timeslots is based on the Resource[x].Size keyword.

For more information, refer to Managing board DSP resources.

See also

Resource[x].Definitions, Resource[x].DSPs, Resource[x].Name, Resource[x].TCPs

Resource[x].TCPs

Specifies the names of the TCPs used on the board to set up and tear down calls. For a list of available TCPs, refer to the *Dialogic® NaturalAccess™ CAS API Developer's Manual*.

Syntax

Resource[***x***].TCPs = ***tcpname tcpname***

x = 0..9 (index of the associated resource pool)

Access

Read/Write

Type

String

Default

Null

Allowed values

One or more supported TCP names separated by spaces.

Example

```
Resource[0].TCPs = WNK0 NOCC
```

If you use the API rather than the *oamcfg* utility to set the TCP names, you must set the values individually. For example:

```
oamSetKeyword(hObj, "Resource[0].TCPs[0]", "WNKO");
oamSetKeyword(hObj, "Resource[0].TCPs[1]", "NOCC");
```

See also

Resource[x].Definitions, Resource[x].DSPs, Resource[x].Name, Resource[x].Size

SwitchConnections

Specifies whether the board nails up default switch connections when initialized.

Syntax

SwitchConnections = ***mode***

Access

Read/Write

Type

String

Default

AUTO

Allowed values

YES | NO | AUTO

Example

```
SwitchConnections = No
```

Details

Valid entries include:

Value	Description
YES	Nails up switch connections regardless of the Clocking.HBus.ClockMode keyword setting.
NO	Does not nail up switch connections.
AUTO	Nails up connections automatically if the Clocking.HBus.ClockMode keyword is set to STANDALONE.

When running the Point-to-Point Switching service, set SwitchConnections = NO. Use the *ppx.cfg* file to define default connections. For more information, refer to the *Point-to-Point Switching Service Developer's Reference Manual*.

See also

SwitchConnectMode

SwitchConnectMode

Specifies how switch connections are made on the board.

Syntax

SwitchConnectMode = ***setting***

Access

Read/Write

Type

String

Default

ByChannel

Allowed values

AllConstantDelay | AllDirect | ByChannel

Example

```
SwitchConnectMode = AllDirect
```

Details

Valid options are:

Option	Description
AllConstantDelay	Data is delayed so that the destination timeslot is always in the next frame, regardless of whether it is a forward connection.
AllDirect	For all board connections, data is transferred directly from the source timeslot to the destination timeslot. For forward connections (from lower-numbered timeslots to higher-numbered timeslots), data is transferred in the same frame. For backward connections (from higher-numbered timeslots to lower-numbered timeslots) data is transferred in the next frame.
ByChannel	The mode for each board connection depends on whether the connection is made using **swiMakeConnection** or **swiMakeFramedConnection**.

This keyword is used for configurations that transfer non-voice data in multiple timeslots (for example, HDLC in TDM).

For more information, refer to **swiMakeConnection** and **swiMakeFramedConnection** in the *Switching Service Developer's Reference Manual*.

See also

SwitchConnections

TPKT.ComplexForward.Count

Specifies the number of condition sets for the system when sending ThroughPacket packets.

Syntax

TPKT.ComplexForward.Count = ***numconditions***

Access

Read/Write

Type

Integer

Default

None.

Allowed values

8

Example

```
TPKT.ComplexForward.Count = 8
```

Details

These condition sets are defined by TPKT.ComplexForward[x].LifeTimeTicks, TPKT.ComplexForward[x].DestinationPacketSize keyword pairs.

This value must always be set to 8, so you must always specify eight TPKT.ComplexForward[x].LifeTimeTicks, TPKT.ComplexForward[x].DestinationPacketSize keyword pairs. However, if you do not want to define eight conditions, you can define NULL conditions by setting the TPKT.ComplexForward[x].DestinationPacketSize keywords to 0 as shown in the following example:

```
TPKT.NumberOfComplexForwardConditions = 4
TPKT.ComplexForward.Count = 8
TPKT.ComplexForward[0].LifeTimeTicks = 0
TPKT.ComplexForward[0].DestinationPacketSize = 1440
TPKT.ComplexForward[1].LifeTimeTicks = 1
TPKT.ComplexForward[1].DestinationPacketSize = 980
TPKT.ComplexForward[2].LifeTimeTicks = 2
TPKT.ComplexForward[2].DestinationPacketSize = 700
TPKT.ComplexForward[3].LifeTimeTicks = 3
TPKT.ComplexForward[3].DestinationPacketSize = 1
TPKT.ComplexForward[4].LifeTimeTicks = 0
TPKT.ComplexForward[4].DestinationPacketSize = 0
TPKT.ComplexForward[5].LifeTimeTicks = 0
TPKT.ComplexForward[5].DestinationPacketSize = 0
TPKT.ComplexForward[6].LifeTimeTicks = 0
TPKT.ComplexForward[6].DestinationPacketSize = 0
TPKT.ComplexForward[7].LifeTimeTicks = 0
TPKT.ComplexForward[7].DestinationPacketSize = 0
```

For information about the Clarent ThroughPacket multiplexing algorithm and its implementation in Fusion software, refer to the *Fusion Developer's Manual*.

See also

TPKT.ComplexRxPort, TPKT.ComplexTxPort, TPKT.NumberOfComplexForwardConditions

TPKT.ComplexForward[x].DestinationPacketSize

Specifies the amount of packet data (in bytes) that must accumulate before a ThroughPacket packet can be sent out.

Syntax

TPKT.ComplexForward[***x***].DestinationPacketSize = ***packetsize***

x = index of a particular ThroughPacket transmission condition

Access

Read/Write

Type

Integer

Default

None.

Allowed values

0..1500 (bytes)

Example

```
TPKT.ComplexForward[0].DestinationPacketSize = 1440
```

Details

You can combine TPKT.ComplexForward[***x***].DestinationPacketSize keywords and TPKT.ComplexForward[x].LifeTimeTicks keywords to define condition sets that specify when packets are transferred by the system. Packets are transferred only when the amount of data specified by the TPKT.ComplexForward[***x***].DestinationPacketSize keyword has accumulated within the time period specified by the associated TPKT.ComplexForward[x].LifeTimeTicks keyword. For example:

```
TPKT.ComplexRxPort = 4046
TPKT.ComplexTxPort = 4046
TPKT.NumberOfComplexForwardConditions = 4
TPKT.ComplexForward.Count = 8
TPKT.ComplexForward[0].LifeTimeTicks = 0
TPKT.ComplexForward[0].DestinationPacketSize = 1440
TPKT.ComplexForward[1].LifeTimeTicks = 1
TPKT.ComplexForward[1].DestinationPacketSize = 980
TPKT.ComplexForward[2].LifeTimeTicks = 2
TPKT.ComplexForward[2].DestinationPacketSize = 700
TPKT.ComplexForward[3].LifeTimeTicks = 3
TPKT.ComplexForward[3].DestinationPacketSize = 1
TPKT.ComplexForward[4].LifeTimeTicks = 0
TPKT.ComplexForward[4].DestinationPacketSize = 0
TPKT.ComplexForward[5].LifeTimeTicks = 0
TPKT.ComplexForward[5].DestinationPacketSize = 0
TPKT.ComplexForward[6].LifeTimeTicks = 0
TPKT.ComplexForward[6].DestinationPacketSize = 0
TPKT.ComplexForward[7].LifeTimeTicks = 0
TPKT.ComplexForward[7].DestinationPacketSize = 0
```

In the previous example, the system sends out ThroughPacket packets only when the following conditions are met:

Within this period of time...	At least this much data must accumulate...
0 ms	1440 bytes
10 ms	980 bytes
20 ms	700 bytes
30 ms	1 byte

The example sets less demanding packet payload size restrictions as time elapses. The system uses these varying restrictions to minimize the latency it introduces when it holds on to packets until a specific amount of data accumulates. After the third tick (tick number 3 in the example), the system sends the packet out with whatever data it has accumulated up to that point, so long as a single byte of data has accumulated. Therefore, the maximum amount of latency that ThroughPacket introduces in this example is 30 ms.

You can set NULL ThroughPacket conditions by setting the associated TPKT.ComplexForward[***x***].DestinationPacketSize keyword to 0.

For information about the Clarent ThroughPacket multiplexing algorithm and its implementation in Fusion software, refer to the *Fusion Developer's Manual*.

See also

TPKT.ComplexForward.Count, TPKT.ComplexRxPort, TPKT.ComplexTxPort, TPKT.NumberOfComplexForwardConditions

TPKT.ComplexForward[x].LifeTimeTicks

Specifies the number of 10 millisecond timer ticks to wait before sending out a ThroughPacket packet.

Syntax

TPKT.ComplexForward[***x***].LifeTimeTicks = ***numticks***

x = index of a particular ThroughPacket transmission condition

Access

Read/Write

Type

Integer

Default

None.

Allowed values

0 - 99 (number of 10 millisecond increments)

Example

```
TPKT.ComplexForward[0].LifeTimeTicks = 1
```

Details

Specifying a TPKT.ComplexForward[***x***].LifeTimeTicks value of 0 marks the moment when data is first received (that is, at 0 milliseconds).

Combine TPKT.ComplexForward[***x***].LifeTimeTicks keywords and TPKT.ComplexForward[x].DestinationPacketSize keywords to define condition sets that specify when packets are transferred by the system. Packets are transferred only when the amount of data specified by the TPKT.ComplexForward[x].DestinationPacketSize keyword has accumulated within the time period specified by the associated TPKT.ComplexForward[***x***].LifeTimeTicks keyword.

For example:

```
TPKT.ComplexRxPort = 4046
TPKT.ComplexTxPort = 4046
TPKT.NumberOfComplexForwardConditions = 4
TPKT.ComplexForward.Count = 8
TPKT.ComplexForward[0].LifeTimeTicks = 0
TPKT.ComplexForward[0].DestinationPacketSize = 1440
TPKT.ComplexForward[1].LifeTimeTicks = 1
TPKT.ComplexForward[1].DestinationPacketSize = 980
TPKT.ComplexForward[2].LifeTimeTicks = 2
TPKT.ComplexForward[2].DestinationPacketSize = 700
TPKT.ComplexForward[3].LifeTimeTicks = 3
TPKT.ComplexForward[3].DestinationPacketSize = 1
TPKT.ComplexForward[4].LifeTimeTicks = 0
TPKT.ComplexForward[4].DestinationPacketSize = 0
TPKT.ComplexForward[5].LifeTimeTicks = 0
TPKT.ComplexForward[5].DestinationPacketSize = 0
```

```
TPKT.ComplexForward[6].LifeTimeTicks = 0
TPKT.ComplexForward[6].DestinationPacketSize = 0
TPKT.ComplexForward[7].LifeTimeTicks = 0
TPKT.ComplexForward[7].DestinationPacketSize = 0
```

In the previous example, the system sends out ThroughPacket packets only if the following conditions are met:

Within this period of time...	At least this much data must accumulate...
0 ms	1440 bytes
10 ms	980 bytes
20 ms	700 bytes
30 ms	1 byte

The example sets less demanding packet payload size restrictions as time elapses. The system uses these varying restrictions to minimize the latency it introduces when it holds on to packets until a specific amount of data accumulates. After the third tick (tick number 3 in the example), the system sends the packet out with whatever data it has accumulated up to that point, so long as a single byte of data has accumulated. Therefore, the maximum amount of latency that ThroughPacket introduces in this example is 30 ms.

For information about the Clarent ThroughPacket multiplexing algorithm and its implementation in Fusion software, refer to the *Fusion Developer's Manual*.

See also

TPKT.ComplexForward.Count, TPKT.ComplexRxPort, TPKT.ComplexTxPort, TPKT.NumberOfComplexForwardConditions

TPKT.ComplexRxPort

Specifies a UDP port number on which to receive complex ThroughPacket packets.

Syntax

TPKT.ComplexRxPort = ***portnumber***

Access

Read/Write

Type

Integer

Default

None.

Allowed values

A valid UDP port number.

Example

```
TPKT.ComplexRxPort = 49152
```

Details

For information about the Clarent ThroughPacket multiplexing algorithm and its implementation in Fusion software, refer to the *Fusion Developer's Manual*.

See also

TPKT.ComplexForward.Count, TPKT.ComplexForward[x].DestinationPacketSize, TPKT.ComplexForward[x].LifeTimeTicks, TPKT.ComplexTxPort, TPKT.NumberOfComplexForwardConditions

TPKT.ComplexTxPort

Specifies a UDP port number on which to transmit complex ThroughPacket packets.

Syntax

TPKT.ComplexTxPort = ***portnumber***

Access

Read/Write

Type

Integer

Default

None.

Allowed values

A valid UDP port number.

Example

```
TPKT.ComplexTxPort = 49152
```

Details

For information about the Clarent ThroughPacket multiplexing algorithm and its implementation in Fusion software, refer to the *Fusion Developer's Manual*.

See also

TPKT.ComplexForward.Count, TPKT.ComplexForward[x].DestinationPacketSize, TPKT.ComplexForward[x].LifeTimeTicks, TPKT.ComplexRxPort, TPKT.NumberOfComplexForwardConditions

TPKT.Enable

Enables or disables ThroughPacket packet multiplexing functionality on the board.

Syntax

TPKT.Enable = ***value***

Access

Read/Write

Type

Integer

Default

0

Allowed values

0 | 1

Example

```
TPKT.Enable = 1
```

Details

Set TPKT.Enable to 1 to enable ThroughPacket multiplexing. Set TPKT.Enable to 0 to disable ThroughPacket multiplexing.

For information about the Clarent ThroughPacket multiplexing algorithm and its implementation in Fusion software, refer to the *Fusion Developer's Manual*.

See also

TPKT.ComplexRxPort, TPKT.ComplexTxPort

TPKT.NumberOfComplexForwardConditions

Specifies the number of conditions specified for ThroughPacket data transmission.

Syntax

TPKT.NumberOfComplexForwardConditions = ***numconditions***

Access

Read/Write

Type

Integer

Default

None.

Allowed values

Number of conditions set for transmitting complex packets.

Example

```
TPKT.NumberOfComplexForwardConditions = 4
```

Details

This keyword specifies the number of TPKT.ComplexForward[x].DestinationPacketSize keyword strings in which the value is not set to 0.

In the following example, the number of TPKT.NumberOfComplexForwardConditions is 3:

```
TPKT.ComplexForward[0].LifeTimeTicks = 0
TPKT.ComplexForward[0].DestinationPacketSize = 1440
TPKT.ComplexForward[1].LifeTimeTicks = 1
TPKT.ComplexForward[1].DestinationPacketSize = 650
TPKT.ComplexForward[2].LifeTimeTicks = 2
TPKT.ComplexForward[2].DestinationPacketSize = 1
TPKT.ComplexForward[3].LifeTimeTicks = 3
TPKT.ComplexForward[3].DestinationPacketSize = 0
TPKT.ComplexForward[4].LifeTimeTicks = 0
TPKT.ComplexForward[4].DestinationPacketSize = 0
TPKT.ComplexForward[5].LifeTimeTicks = 0
TPKT.ComplexForward[5].DestinationPacketSize = 0
TPKT.ComplexForward[6].LifeTimeTicks = 0
TPKT.ComplexForward[6].DestinationPacketSize = 0
TPKT.ComplexForward[7].LifeTimeTicks = 0
TPKT.ComplexForward[7].DestinationPacketSize = 0
```

For information about the Clarent ThroughPacket multiplexing algorithm and its implementation in Fusion software, refer to the *Fusion Developer's Manual*.

See also

TPKT.ComplexForward.Count, TPKT.ComplexForward[x].LifeTimeTicks, TPKT.ComplexRxPort, TPKT.ComplexTxPort, TPKT.Enable

TPKT.SimpleRxPort

Specifies a valid UDP port number on which to receive simple ThroughPacket packets.

Syntax

TPKT.SimpleRxPort = ***portnumber***

Access

Read/Write

Type

Integer

Default

None.

Allowed values

A valid UDP port number.

Example

```
TPKT.SimpleRxPort = 49152
```

Details

For information about the Clarent ThroughPacket multiplexing algorithm and its implementation in Fusion software, refer to the *Fusion Developer's Manual*.

See also

TPKT.SimpleTxPort

TPKT.SimpleTxPort

Specifies a UDP port number on which to transmit simple ThroughPacket packets.

Syntax

TPKT.SimpleTxPort = ***portnumber***

Access

Read/Write

Type

Integer

Default

None.

Allowed values

A valid UDP port number.

Example

```
TPKT.SimpleTxPort = 49152
```

Details

For information about the Clarent ThroughPacket multiplexing algorithm and its implementation in Fusion software, refer to the *Fusion Developer's Manual*.

See also

TPKT.SimpleRxPort

Version.Major

Indicates the major version number of the CG plug-in.

Syntax

Version.Major = ***number***

Access

Read-only (CG plug-in level)

Type

Integer

Default

1

Allowed values

Not applicable.

Details

The keyword value is incremented when a change is made to the plug-in.

See also

Version.Minor

Version.Minor

Indicates the minor version number of the CG plug-in.

Syntax

Version.Minor = ***number***

Access

Read-only (CG plug-in level)

Type

Integer

Default

0

Allowed values

Not applicable.

Details

The keyword value is incremented when a change is made to the plug-in.

See also

Version.Major

12. Hardware specifications

General hardware specifications

This topic describes the following types of hardware specifications:

- Physical
- Media stream DSP processing
- IP network connectivity
- PSTN network connectivity
- Software environment
- Host interface
- H.100 compliant interface
- Environment
- Power requirements
- Cooling requirements

Physical

PCI bus	32-bit 66, 33, 66 MHz PCI bus
PCI universal expansion board	Compatible with 5.0 V or 3.3 V signaling environments
Digital trunk interfaces	Four trunks: Two Dialogic® MD1 RJ-45 interfaces, each with two trunks
Board weight	Main board: 0.45 lb to 0.47 lb (0.20 kg)

Media stream DSP processing

Universal port capacity	• Comprehensive IVR support • Echo cancellation • Vocoding: G.711, G.723.1, G.729a/b, G.726 • Fax: T.38 real-time, T.37 store-and-forward
Processors	T1 TMS320C5441, each with four 133 MHz cores
Capacity	Up to 120 universal ports

IP network connectivity

Physical	Dual 10/100Base-T Ethernet interfaces
Protocols	ARP ICMP IPSec IPv4 IPv6 RTP/RTCP UDP

PSTN network connectivity

Physical	Up to four Dialogic® MD1 RJ-45 T1/E1 digital trunk interfaces
Protocols	ISDN CAS
Capacity	Up to 120 ports
Approvals	Refer to www.dialogic.com/declarations/default.htm for a list of countries where Dialogic has obtained approval for the board.

Software environment

Development environment	NaturalAccess
Operating systems	Windows UNIX Red Hat Linux

Host interface

Feature	Specification
Electrical	PCI bus designed to the PCI Local Bus Specification, Revision 2.2.
Mechanical	PCI bus designed to the PCI Local Bus Specification, Revision 2.2.
Bus speed	DC to 66 MHz
Bus type	32-bit master or slave

Feature	Specification
I/O mapped memory	Memory-mapped interface for efficient block data transfers
Addresses/interrupts	Address and interrupts automatically configured by PCI BIOS (no jumpers or switches)

H.100 compliant interface

- Flexible connectivity between DS0 streams and H.100 bus
- 2048 full-duplex connections to bus
- 1024 local connections
- Switchable access to any of 4096 H.100 timeslots
- H.100 clock master or clock slave (software-selectable)
- H.100 bus termination (switch enabled)

Environment

Feature	Description
Operating temperature	0 to 50 ° C
Storage temperature	-20 to 70 ° C
Humidity	5 to 80%, non-condensing

Power requirements

State	3.3 V requirement	5.0 V requirement
CG 6060 board in reset	0.574 A maximum	0.557 A maximum
CG 6060 board active	1.446 A maximum	1.102 A maximum

Note: Voltage tolerances are +/- 5% of nominal.

Cooling requirements

Ambient temperature	CFM (per board)	Altitude
35 ° C	0.8	Sea level
45 ° C	1.4	1000 m (3281 ft)

CEPT E1 G.703 telephony interface

Specification	G.703 2048 kbit/s trunk interface
Framing	CEPT G.703/G.704 channel associated signaling
Signaling capabilities	ABCD bits for channel associated signaling and HDLC/LAPD for generating/terminating data link
Line code	HDB3 (in zero code suppression) or AMI
Alarm signal capabilities	Loss of frame alignment (LOF), loss of signaling multiframe alignment and loss of CRC multiframe alignment (red), remote alarm and remote multiframe alarm (yellow), alarm indication signal (AIS)
Counts	Bit error rate, CRC errors, slips, line code violations, far-end block errors
Loopback	Per channel and across channels under software control
Interfaces	Four trunks: Two Dialogic® MD1 RJ-45 interfaces, each with two trunks

DSX-1 telephony interface

Specification	ANSI T1.102, T1.403
Framing	D4, ESF
Insertion and extraction	ABCD bits
Line code	AMI
Zero bits	Selectable B8ZS, jammed bit (ZCS) or no zero code suppression
Alarm signal capabilities	Yellow and red
Counts	Bipolar violation, F(t) error, and CRC error
Robbed bit	Selectable on a per-trunk basis
Loopback	Per channel and overall under software control. Automatic remote loopback with CSU option.
Interfaces	Four trunks: Two Dialogic® MD1 RJ-45 interfaces, each with two trunks

13. DSP resource management

Managed DSP resources

CG boards are based on a flexible software and hardware architecture. The architecture uses an array of digital signal processors (DSPs) under the control of a specially designed operating system to execute algorithms that detect, encode, decode, and generate voice and call status signals. These DSP resources are managed by the DSP resource manager executing on the board.

CG board DSP resource management is configured to operate on a per-port basis. A port is associated with a circuit-switched call (for PSTN-based applications) or another type of media stream. DSP resource management determines the DSPs on which particular DSP functions run. Resource management can ensure that the DSP resources required to support a call are available when needed.

CG board DSP resource management reserves all of the resources required for a port before the port is used. The DSP resources that are reserved for a port are specified in the Resource[x].Definitions keyword and the Resource[x].TCPs keyword in the board keyword file.

The standard set of board keyword files provided with CG software (and other NaturalAccess software, if applicable) contains DSP resource management settings suitable for most applications. Therefore, in most cases you do not need to modify these resource definitions, and you can skip this section. However, if your application requires resources not specified in the sample board keyword files, you may need to customize the CG board's DSP resource management settings. You should understand how the CG DSP resource manager allocates resources before modifying the standard DSP resource definitions.

DSPs, DPMs, and DPFs

DSP programs are distributed in files called data processing modules (DPMs). These files use the extension *.f41*, and contain executable code for a family of algorithms. Algorithms in the family are called data processing functions (DPFs). They can be referenced by a unique string generated by combining the family string ID that corresponds to the DPM with the function ID string that corresponds to the DPF. The string is formatted as ***dpm.dpt***. Both the DPM and DPF have associated hexadecimal IDs that can be combined to uniquely identify the DSP function.

For example, the file *ptf.f41* is the DPM for precise tone filters. All of the DPFs in *ptf.f41* provide precise tone filtering functionality. The precise tone filter DPM ID is ptf and the associated hexadecimal ID is 0x1C. For a precise tone filter that has a pair of filters, the DPF ID is det_2f and the associated hexadecimal ID is 0x0700. Therefore, the precise tone DPF is identified by the string ID ptf.det_2f or the hexadecimal ID 0x1C0700.

To list all function IDs in a family and their associated hexadecimal IDs, run the *f41info* program and specify the DPM file name for the family. For example, for the echo cancellation family of functions, type the following command at the prompt:

```
f41info ptf
```

For more information about *f41info* and about DPM and DPF hexadecimal IDs, refer to f41info - Displaying DPF file resource usage.

DSP resources

CG board DSP resource usage is calculated based on the following categories:

- MIPS
- Memory
- Timeslots
- Packet queues
- Buffer size

Each DPF that executes on a DSP core consumes resources from each of these categories. The overall resource usage dictates how many DPFs can run on a DSP core, and consequently how many ports can run per board. The following table describes these resource categories:

Resource category	Description
MIPS	Measures the computational capacity of a DSP core and the fraction of a DSP core's processing power consumed by a given DPF. Each DSP core on a CG board runs at 133 MIPS. The DSP operating system (DSPOS) consumes approximately 5 MIPS, leaving approximately 128 MIPS per DSP core available for the loadable DPFs.
Memory	Each DSP has a total of four DSP cores arranged as two core pairs. Each core pair has 128 K of shared program memory and 64 K of data memory.
Timeslots (input and output)	Each DSP core is connected to an H.100 interface chip through shared local streams consisting of 128 input and 128 output timeslots. Groups of four DSP cores are connected to a single local stream. Therefore, each DSP core in the group is physically connected to 128 input and output timeslots. However, each DSP uses only a subset of the available timeslots. The DSP operating system supports 16 or 32 full duplex timeslots per DSP core. Input timeslots on CG boards can be shared between DPFs. In addition, there are 32 input and 32 output timeslots internal to each DSP core. These internal timeslots are used for DPF-to-DPF communication.
Packet queue (input and output)	Carries data between the DSP cores and the CG board processor. The number of queues is limited in the DSP operating system to conserve system memory and optimize performance.
Buffer size	Packets and buffers take up resources in the DSP-to-coprocessor buffer or coprocessor-to-DSP buffer.

If no other limitations apply (for example MIPS or memory) the maximum number of DPFs that can run simultaneously on a DSP is 63.

The following table summarizes resources available on each CG board DSP core:

Resource category	Available resource
MIPS	Approximately 128 MIPS per DSP
External timeslots (input and output)	16 or 32 full duplex
Internal timeslots (input and output)	32 in and 32 out
Memory	Approximately 64 K words
Input packet queues	64
Output packet queues	64
Buffer size (coprocessor-to-DSP and DSP-to-coprocessor)	768 words each

Worst-case resource management calculation

One way to find out if the available board resources can support all the processes is to calculate the resources required by the application in its most demanding scenario. This is called a worst-case scenario calculation.

When pool sizes are specified in the board keyword file, the CG boards calculate the requested resources at boot time and determine if the requested DSP resources are available.

This topic describes a worst-case resource usage example (a telephony application using an on-board CAS or ISDN protocol) to illustrate the methods used to calculate resources under particular conditions.

Example

For calls in the connected state (also called the conversation state), media processing functions (that is, vocoders or fax functions) consume the greatest amount of DSP processing power. TCPs used during the call also require DSP resources, but they use the greatest amount of resources during the set up phase of the call. In general, the resources required during call setup are less than the resources required in the connected state. For this reason, the examples that follow calculate resources used during calls in the connected state.

The following example shows resource requirements for a call that uses the following functions in the connected state:

- Software echo cancellation (echo.ln20_apt100 DPF)
- DTMF detection with energy detector and two tone detectors (dtmf.det_all DPF)
- Precise tone detection (ptf.det_2f DPF)
- NMS ADPCM record vocoder (voice.rec_32 DPF)

This example assumes that the worst-case usage (that is, the point where DSP resources experience their highest MIPS usage in every category) occurs during the connected state of the call.

ISDN signaling and out-of-band signaling for CAS use signaling resources executing on the control processor. The DSP resource manager does not manage these resources.

The following table shows approximate resources consumed by the DSP functions in this scenario:

DPF	MIPS	Input slots	Output slots	Data memory	Input Pkt queue	Output Pkt queue	DSP-to-coprocessor buffer size	Comments
echo	4.0	2	1	800	0	0	-	20 ms length, 100% adapt rate
dtmf	2.5	1	0	200	0	0	-	
ptf	1.3	1	0	100	0	0	-	
voice	3.3	1	0	300	0	1	40	NMS Record 32 kbit/s
Port resource consumption	**11.1**	**5**	**1**	**1400**	**0**	**1**	**42**	

To find the most up-to-date resource requirements for specific DPFs, run *f41info* as described in f41info - Displaying DPF file resource usage.

Resource calculations

Based on the previous example, you can calculate the following resource limitations for the sample application:

Resource category	Worst case usage
MIPS	MIPS used by this worst case example are the sum of the MIPS requirements for all of the functions, or 11.1 MIPS. Divide the 128 MIPS (approximately) available per DSP core by the sum, and truncate the result (128 / 11). This example yields 11 ports of conversation state per DSP core.
Memory	Each DSP core has 64 K of data memory that is used for both data and context. This example assumes 20 K words of data/context memory. As a result, the limiting factor for memory is the amount of context memory needed per port. In this example, the number of ports that can be allocated from memory are (44 x 1024) / 1400 = 32.

Resource category	Worst case usage
Timeslots (input and output)	Each connected port consumes one full duplex timeslot because input slots are shared. Furthermore, two of the echo canceler slots, one input slot and one output slot, are internal. The port limit per DSP core due to timeslots is 32 ports for both external and internal slots. If the total input or output timeslots calculation is greater than 1, the resource manager sets it to 1.
Packet queues	The number of packet queues required in the example is 1. The packet queue limit is therefore 64 (64 / 1).
Buffer size	The coprocessor packet size limit is applicable only for DSP functions sending or receiving data from the coprocessor. Typically, this limit is imposed by a vocoder in the conversation/connected state. Since this example uses a voice recording DPF, the packets take up resources in the DSP-to-coprocessor buffer. An NMS 32 kbit/s vocoder uses 42 words (40 words of data, 2 header words) per coprocessor packet, resulting in an 18 port per DSP core limit (obtained by dividing DSP-to-coprocessor buffer size by packet size or 768 / 42). An NMS 64 K vocoder uses 82 word buffers in a connected state, and therefore limits each DSP core to nine ports. In the case of play functions, the packets consume resources in the coprocessor-to-DSP buffer plus an additional four words in the DSP-to-coprocessor buffer. This is because the DSP operating system sends data acknowledgment events through DSP-to-coprocessor buffers.

Overall resource requirements

In this example, the number of ports per DSP core is limited to a total of 11 by the MIPS requirements.

Determining the maximum number of ports available

The following steps describe how to determine the maximum number of ports available when using a specific resource definition:

Step	Action
1	Set the Resource[x].Size keyword to an impossibly high number (for example, 1000).
2	Attempt to boot the board.
3	When the board fails to boot, monitor *oammon*. An error message specifying the maximum number of ports possible with that resource configuration appears. For example: `Board Error 0xe40: Resource Manager: Insufficient resources. 120 ports allocated`

Step	Action
4	Reset the Resource[x].Size keyword to the number indicated in the *oammon* error message.

DSP resource management keywords

The following keywords configure CG board DSP resource management:

Keyword	Description
Resource[x].TCPs	Specifies the TCPs that the resource manager uses with the resource definition.
Resource[x].Name	Associates a name (character string) with a particular resource definition.
Resource[x].Definitions	Specifies a relational string of data processing functions (DPFs), describes the functionality that can occur on a single port, and describes how and when DSP functions execute in relation to each other.
Resource[x].Size	Specifies the number of channels or ports managed by the on-board resource manager.
Resource[x].DSPs	Specifies the DSP numbers on which to allocate the resources.
Resource[x].StartTimeSlot	Specifies the starting timeslot on which to associate the pool.

Resource definition string syntax

When specifying resource definitions, you can use a set of logical operators in board keyword files to combine DPFs and define the relationships between them. The following logical operators are supported in board keyword files:

Operator	Description
&	And
|	Or
()	Open and close parentheses
\	Line break

Note: Resource[x].Definitions strings always start with an open-parenthesis and end with a close-parenthesis.

DSP image and resource definitions

The DPMs specified in Resource[x].Definitions and the TCPs listed in Resource[x].TCPs are used to create the image. The image is loaded to the DSPs specified in the Resource[x].DSPs keyword.

Resource definitions

This topic provides examples of Resource[x].Definitions strings that use the DPFs echo.ln20_apt100 (software echo), dtmf.det_all, ptf.det_2f, and voice.rec_32. These examples specify which DPFs run on a specific DSP as well as which DPFs can run simultaneously and which cannot.

Note: The resource requirements for the DPFs specified in this topic are subject to change. To verify resource usage for specific DPFs, run *f41info* as described in f41info - Displaying DPF file resource usage.

In the DSP resource management examples that follow, the input slots requirements are calculated differently from other resource categories in a worst-case scenario.

Input and output slots

In the following examples, the input and output slots columns show the number of timeslots required to move data to and from a DSP. Because all of the DPFs specified for a call have been allocated on the same DSP, the resource manager allocates only one input timeslot and one output timeslot to a call.

For example, the resource manager determines if all of the resources specified in the Resource[x].Definitions string can be allocated on a single DSP. It also determines if the DPFs specified in the Resource[x].Definitions string use DPF-to-DPF communication. Because all the DPFs specified for each port are allocated on a single DSP, the DPFs can use internal timeslots to move data between the DPFs. In this case, the on-board resource manager can efficiently allocate sufficient resources by assigning only one input timeslot per port.

If the resource manager is not able to allocate all DPFs for a port on one DSP, you receive an error message stating that there are not enough resources.

Resource definition examples

The following examples describe the resource strings used to set up DPFs to run under the following DPF conditions:

- All DPFs running exclusively
- All DPFs running simultaneously
- Some DPFs running simultaneously

Example: All DPFs running exclusively

In the following example, all of the DPFs specified in the resource definition string run exclusively (in other words, only one DPF can run at a time). The Resource[x].Definitions keyword string is set as follows:

```
Resource[1].Definitions = ( dtmf.det_all | ptf.det_2f | voice.rec_32)
```

The resources consumed by the DPFs in this resource string are shown in the following table, which lists the resources consumed by each function:

DPF	MIPS	Input slots	Output slots	Data memory (words)	Input Pkt queue	Output Pkt queue	DSP-to-coprocessor buffer size	Comments
dtmf	2.5	1	0	200	0	0	-	
ptf	1.3	1	0	100	0	0	-	
voice	3.3	1	1	300	0	1	40	NMS Record 32 kbit/s
Port resource consumption	**3.3**	**1**	**1**	**300**	**0**	**1**	**40**	

Since the example specifies that only one DPF function can execute at a time, the worst case MIPS requirement is 3.3 MIPS, the worst case data memory requirement is 300 words, and the worst case output slots requirement is 1 (the highest resource usage in each category).

Example: All DPFs running simultaneously

In the following example, all of the DPFs specified in the resource definition string can run at the same time. The Resource[x].Definitions keyword string is set as follows:

```
Resource[1].Definitions = ( echo.ln20_apt100 & dtmf.det_all & \
ptf.det_2f & voice.rec_32 )
```

DPF	MIPS	Input slots	Output slots	Data memory (words)	Input Pkt queue	Output Pkt queue	DSP-to-coprocessor buffer size	Comments
echo	4.0	2	1	800	0	0	-	20 ms length, 100% adapt rate
dtmf	2.5	1	0	200	0	0	-	
ptf	1.3	1	0	100	0	0	-	
voice	3.3	1	0	300	0	1	40	NMS Record 32 kbit/s
Port resource consumption	**11.1**	**1**	**1**	**1400**	**0**	**1**	**40**	

Since all of the MIPS are added together, the MIPS calculation for this string is 11.1 MIPS, while the data memory requirement is 1400 words (the cumulative total of adding up the requirements for each DPF).

However, when all of the DPFs reside on the same DSP core, the slots internal to the DSP core are used for DPF to DPF communication. Since they use internal timeslots, the input slot requirements are equal to 1.

Example: Some DPFs running simultaneously

In the following example, only the echo and voice DPFs can execute at the same time. The Resource[x].Definitions keyword string is set as follows:

```
Resource[1].Definitions = ( ptf.det_2f | dtmf.det_all | \
( echo.ln20_apt100 & voice.rec_32 ) )
```

DPF	MIPS	Input slots	Output slots	Data memory (words)	Input Pkt queue	Output Pkt queue	DSP-to-coprocessor buffer Size	Comments
echo	4.0	2	1	800	0	0	-	20 ms length, 100% adapt rate
dtmf	2.5	1	0	200	0	0	-	
ptf	1.3	1	0	100	0	0	-	
voice	3.3	1	0	300	0	1	40	NMS Record 32 kbit/s
Port resource consumption	**7.3**	**1**	**1**	**1100**	**0**	**1**	**40**	

Parentheses are used to indicate groupings and order of operation. In the example, software echo and voice resources are added together with the AND operator before the comparisons with the OR operator occur.

The MIPS calculation for this example adds the MIPS requirements for the software echo and voice DPFs and compares the result to dtmf and then to ptf. The largest of the three is allocated to the MIPS resource.

In this case echo + voice is 7.3 MIPS, which is greater than the MIPS requirement for the dtmf and greater than the MIPS requirement for the ptf (2.5 and 1.3 MIPS respectively). Therefore, the maximum MIPS requirement is 7.3 MIPS. The sum of voice and software echo requirements for data memory is also larger than the dtmf and ptf requirements, so the maximum data memory requirement is 1100 words.

The output slots requirement is 1 and the input slots requirement is 1 because all DPFs are executing on one DSP. The resource manager uses these calculations to determine how many ports the board supports.

Conditional relationships between DPFs

The following examples define complex conditional relationships between DPFs using the AND operators, OR operators, and parentheses to combine DPF string IDs.

Example 1

In the following example, OKI play and OKI record DPFs run simultaneously with:

- DTMF detection
- Precise tone detection with two single frequencies

Simultaneous 24 kbit/s OKI ADPCM play and record functions are specified with the following Resource[x].Definitions string:

```
Resource[1].Definitions = ( dtmf.det_all &  \
ptf.det_2f & ( oki.rec_24 & (oki.play_24_100 | oki.play_24_150 | \
oki.play_24_200 ) ) )
```

This resource definition string reserves DSP resources so that the worst-case resource usage of the play functions (oki.play_24_100, oki.play_24_150, oki.play_24_200) run simultaneously with the record function (oki.rec_24).

Example 2

In this example, OKI play, OKI record, or tone generator functions run in the connected state, but not at the same time. Functions that execute simultaneously with OKI play or OKI record functions include:

- DTMF detection
- Precise tone detection with two single frequencies

In this example, the tone generator does not run if an OKI ADPCM play or OKI ADPCM record function is running.

You can run a 24 kbit/s ADPCM OKI play function or a 24 kbit/s ADPCM OKI record function by specifying the following Resource[x].Definitions string:

```
Resource[1].Definitions = ( dtmf.det_all & \
pf.det_2f & ( oki.rec_24 | oki.play_24_100 | oki.play_24_150 |
oki.play_24_200 | tone.gen ))
```

This resource definition string allows either the record functions, one of the play functions, or the tone generator to run at the same time as the DTMF detection, and PTF functions. A 24 kbit/s ADPCM OKI play function never runs at the same time as a 24 kbit/s ADPCM OKI record function.

Natural Access media masks and call progress masks

Natural Access uses a media mask and a call progress mask that affect DSP resource management. These NCC Service masks are:

- NCC.X.ADI_START.mediamask
- NCC.X.ADI_PLACECALL.CALLPROG.precmask

These masks indicate DPFs that will execute on the board. If any of the bits in either of these two masks are set, the DPFs relating to the set bits must be specified in the Resource[x].Definitions keyword. Since the bits in the media mask are set by default, the DPFs that correspond to these set bits are in the default Resource[x].Definitions keyword in the templates. For more information about these masks, refer to the *Natural Access Developer's Reference Manual*.

Trunk control program resource usage

The Resource[x].TCPs keyword specifies which TCPs are used to perform call control for a resource definition. A number of TCPs are supplied with CG boards. The following table shows the digital TCPs provided with CG software and their MIPS and memory requirements:

Protocol	MIPS	Memory
MFC-R2, E&M	2.762	488
EUC, AP2	2.762	378
SS5	4.227	670
R15	2.762	528
MFS	3.715	636
WNK, FGD	2.762	378
OPS, GDS	3.102	378
NEC PBX	2.762	488
ISDN	2.762	378

This information is useful for predicting the number of ports per DSP core. One input timeslot and one output timeslot are allocated as part of the resources (assuming that all of the resources for a TCP are allocated on one DSP core).

Note: TCP MIPS usage numbers are subject to change.

The resource calculations displayed when the board is booted include only resources for TCPs specified with Resource[x].TCPs.

Debugging resource management

There are several ways to debug the resource management strings defined in CG boards. This topic describes:

- Using CG board debug masks
- Resource management board errors
- Debugging resource management with cgtrace

Using CG board debug masks

Each CG board keyword file includes a DebugMask keyword. When this keyword value is set, it turns on global flags in the system. These flags display status information about different board components whenever the board is booted. After the board has been booted, run *cgtrace* to obtain a list of global and local debug masks and a brief description of the types of information they provide. For more information, refer to cgtrace - Performing CG board debugging.

Note: To view debug information, you must be running the OAM API utility *oammon*. For more information about using *oammon*, refer to the *Dialogic® NaturalAccess™ OAM System Developer's Manual.*

If you encounter DSP resource management problems when booting a CG board, set the DebugMask keyword value to 0x08. This setting configures the board so that it displays all of the resources available for each DSP. In some circumstances, it also displays the calculated resource usage based on the resource management keywords specified in the board keyword file.

Resource management board errors

If an application tries to use the allocated DSP resources in a way not specified in the board's DSP resource management configuration, the board returns an error. Use the OAM utility *oammon* to monitor these errors.

Some possible debug errors include the following error messages:

Error message	Problem	Solution
Board Error 0xe40: Resource Manager: Insufficient resources. In this case, the total resource usage for the board is displayed.	The board can allocate resources for only a portion of the ports you require, because the resource consumption exceeds available DSP resources.	Adjust the number (or type) of DPFs that can execute simultaneously on each port so that the board supports the required number of ports, or decrease the number of ports used.
Board Error 0xa0e: Function 0x001A0000 not found on any engine. In this case, the calculated resource is not displayed.	A DSP has not been loaded with all the DPMs specified in the Resource[x].Definitions or Resource[x].TCPs keyword strings.	The error provides the DPF's family and function ID in hexadecimal form (in the example error, this is 0x001A0000). Make sure a DPM has been loaded for each DPF specified in the resource definition string.

For more information about CG board errors, refer to the *Dialogic® NaturalAccess™ Board and Driver Error Reference*. For more information about the hexadecimal IDs associated with DPMs and DPFs, refer to f41info - Displaying DPF file resource usage.

Debugging resource management with cgtrace

cgtrace is an interactive debugging tool that enables you to debug CG board output. Once the CG board is booted, use *cgtrace* to evaluate CG board DSP resource management on a per-port and per-DSP basis.

Refer to cgtrace - Performing CG board debugging for more information.

DSP files and MIPS requirements

The following table shows the MIPS usage for all the available DPMs supported by Natural Access:

DSP file / ASCII ID string	Function	MIPS	Related API function	Related arguments
adsir.f41 adsir.rcv	ADSI receiver	2.67	**adiStartReceivingFSK**	
adsix.f41 adsir.xmt	ADSI transmitter	0.88	**adiStartSendingFSK**	
amr.f41 amr.play amr.play_edtx	AMR play All rates	3.00	**adiStartPlaying**	**encoding** = ADI_ENCODE_AMR_475 ADI_ENCODE_AMR_515 ADI_ENCODE_AMR_59 ADI_ENCODE_AMR_67 ADI_ENCODE_AMR_74 ADI_ENCODE_AMR_795 ADI_ENCODE_AMR_102 ADI_ENCODE_AMR_122
amr.f41 amr.rec_475 amr.rec_515 amr.rec_590 amr.rec_670 amr.rec_740 amr.rec_795 amr.rec_102 amr.rec_122	AMR record 4.75 kbit/s 5.15 kbit/s 5.90 kbit/s 6.70 kbit/s 7.40 kbit/s 7.95 kbit/s 10.20 kbit/s 12.20 kbit/s	17.70	**adiStartRecording**	**encoding** = ADI_ENCODE_AMR_475 ADI_ENCODE_AMR_515 ADI_ENCODE_AMR_59 ADI_ENCODE_AMR_67 ADI_ENCODE_AMR_74 ADI_ENCODE_AMR_795 ADI_ENCODE_AMR_102 ADI_ENCODE_AMR_122
callp.f41 callp.gnc	Call progress	0.96	**adiStartCallProgress**	
dtmf.f41/dtmfe.f41 dtmf.det_dtmf	DTMF only	1.81	**adiStartDTMFDetector**	

DSP file / ASCII ID string	Function	MIPS	Related API function	Related arguments
dtmf.f41/dtmfe.f41 dtmf.det_sil	Post- and pre- tone silence	0.69	**adiStartEnergyDetector**	
dtmf.f41/dtmfe.f41 dtmf.dtmf_sil_clrdwn	DTMF, post- and pre-tone silence, and one tone pair	2.46	**adiStartProtocol**	
dtmf.f41/dtmfe.f41 dtmf.det_clrdwn	1 tone pair	1.28	**adiStartToneDetector**	
dtmf.f41/dtmfe.f41 dtmf.det_all	DTMF, post- and pre-tone silence, one tone pair, and one frequency	2.72	**adiStartToneDetector**	
dtmf.f41/dtmfe.f41 dtmf.det_sil_clrdwn_ced	Post- and pre-tone silence, one tone pair, and one frequency	1.57	**adiStartToneDetector**	
f_amr.f41 f_amr.cod	AMR encode	18.40	**mspCreateChannel**	**channelType** = AMREncodeSimplex
f_amr.f41 f_amr.dec	AMR decode	3.80	**mspCreateChannel**	**channelType** = AMRDecodeSimplex
f_amr.f41 f_amr.cod_rfc2833	AMR encode with RFC 2833	19.60	**mspCreateChannel**	**channelType** = AMREncodeSimplex **FilterAttribs** = MSP_FCN_ATTRIB_RFC2833
f_amr.f41 f_amr.dec_rfc2833	AMR decode with RFC 2833	3.80	**mspCreateChannel**	**channelType** = AMRDecodeSimplex **FilterAttribs** = MSP_FCN_ATTRIB_RFC2833
f_evrc.f41 f_evrc.cod	EVRC encode	27.50	**mspCreateChannel**	**channelType** = EVRCEncodeSimplex

DSP file / ASCII ID string	Function	MIPS	Related API function	Related arguments
f_evrc.f41 f_evrc.dec	EVRC decode	3.50	**mspCreateChannel**	**channelType**= EVRCDecodeSimplex
f_evrc.f41 f_evrc.cod_rfc2833	EVRC encode with RFC 2833	29.00	**mspCreateChannel**	**channelType**= EVRCEncodeSimplex **FilterAttribs**= MSP_FCN_ATTRIB_RFC2833
f_evrc.f41 f_evrc.dec_rfc2833	EVRC decode with RFC 2833	3.80	**mspCreateChannel**	**channelType**= EVRCDecodeSimplex **FilterAttribs**= MSP_FCN_ATTRIB_RFC2833
f_faxt38.f41 f_faxt38.relay	T.38 fax relay	14.00	**mspCreateChannel**	**channelType**= FaxRelayFullDuplex
f_g711.f41 f_g711.cod	G.711 encode	1.50	**mspCreateChannel**	**channelType**= G711EncodeSimplex
f_g711.f41 f_g711.dec	G.711 decode	0.50	**mspCreateChannel**	**channelType**= G711DecodeSimplex
f_g711.f41 f_g711.cod_rfc2833	G.711 encode with RFC 2833	2.77	**mspCreateChannel**	**channelType**= G711EncodeSimplex **FilterAttribs**= MSP_FCN_ATTRIB_RFC2833
f_g711.f41 f_g711.dec_rfc2833	G.711 decode with RFC 2833	0.75	**mspCreateChannel**	**channelType**= G711DecodeSimplex **FilterAttribs**= MSP_FCN_ATTRIB_RFC2833
f_g711vad.f41 f_g711vad.cod	G.711 encode	1.50	**mspCreateChannel**	**channelType**= G711EncodeSimplex
f_g711vad.f41 f_g711vad.dec	G.711 decode	0.50	**mspCreateChannel**	**channelType**= G711DecodeSimplex
f_g711vad.f41 f_g711vad.cod_rfc2833	G.711 encode with RFC 2833	2.77	**mspCreateChannel**	**channelType**= G711EncodeSimplex **FilterAttribs**= MSP_FCN_ATTRIB_RFC2833
f_g711vad.f41 f_g711vad.dec_rfc2833	G.711 decode with RFC 2833	0.75	**mspCreateChannel**	**channelType**= G711DecodeSimplex **FilterAttribs**= MSP_FCN_ATTRIB_RFC2833

DSP file / ASCII ID string	Function	MIPS	Related API function	Related arguments
f_g723.f41 f_g723.cod	G.723 encode	16.2	**mspCreateChannel**	**channelType**= G723EncodeSimplex
f_g723.f41 f_g723.dec	G.723 decode	1.9	**mspCreateChannel**	**channelType**= G723DecodeSimplex
f_g723.f41 f_g723.cod_rfc2833	G.723 encode with RFC 2833	17.4	**mspCreateChannel**	**channelType**= G723EncodeSimplex **FilterAttribs**= MSP_FCN_ATTRIB_RFC2833
f_g723.f41 f_g723.dec_rfc2833	G.723 decode with RFC 2833	1.9	**mspCreateChannel**	**channelType**= G723DecodeSimplex **FilterAttribs**= MSP_FCN_ATTRIB_RFC2833
f_g726.f41 f_g726.cod	G.726 encode	8.05	**mspCreateChannel**	**channelType**= G726EncodeSimplex
f_g726.f41 f_g726.dec	G.726 decode	7.64	**mspCreateChannel**	**channelType**= G726DecodeSimplex
f_g726.f41 f_g726.cod_rfc2833	G.726 encode with RFC 2833	9.32	**mspCreateChannel**	**channelType**= G726EncodeSimplex **FilterAttribs**= MSP_FCN_ATTRIB_RFC2833
f_g726.f41 f_g726.dec_rfc2833	G.726 decode with RFC 2833	7.64	**mspCreateChannel**	**channelType**= G726DecodeSimplex **FilterAttribs**= MSP_FCN_ATTRIB_RFC2833
f_g729a.f41 f_g729a.cod	G.729 encode	13.1	**mspCreateChannel**	**channelType**= G729EncodeSimplex
f_g729a.f41 f_g729a.dec	G.729 decode	3.2	**mspCreateChannel**	**channelType**= G729DecodeSimplex
f_g729a.f41 f_g729a.cod_rfc2833	G.729 encode with RFC 2833	14.4	**mspCreateChannel**	**channelType**= G729EncodeSimplex **FilterAttribs**= MSP_FCN_ATTRIB_RFC2833
f_g729a.f41 f_g729a.dec_rfc2833	G.729 decode with RFC 2833	3.2	**mspCreateChannel**	**channelType**= G729DecodeSimplex **FilterAttribs**= MSP_FCN_ATTRIB_RFC2833

DSP file / ASCII ID string	**Function**	**MIPS**	**Related API function**	**Related arguments**
f_gsm_fr.f41 f_gsm_fr.cod	GSM-FR encode	5.10	**mspCreateChannel**	**channelType** = GSMFREncodeSimplex
f_gsm_fr.f41 f_gsm_fr.dec	GSM-FR decode	3.60	**mspCreateChannel**	**channelType** = GSMFRDecodeSimplex
f_gsm_fr.f41 f_gsm_fr.cod_rfc2833	GSM-FR encode with RFC 2833	5.30	**mspCreateChannel**	**channelType** = GSMFREncodeSimplex **FilterAttribs** = MSP_FCN_ATTRIB_RFC2833
f_gsm_fr.f41 f_gsm_fr.dec_rfc2833	GSM-FR decode with RFC 2833	2.50	**mspCreateChannel**	**channelType** = GSMFRDecodeSimplex **FilterAttribs** = MSP_FCN_ATTRIB_RFC2833
f_ilbc_20.f41 f_ilbc_20.cod	iLBC encode 20 ms	15.10	**mspCreateChannel**	**channelType** = ILBC20EncodeSimplex
f_ilbc_20.f41 f_ilbc_20.dec	iLBC decode 20 ms	7.10	**mspCreateChannel**	**channelType** = ILBC20DecodeSimplex
f_ilbc_20.f41 f_ilbc_20.cod_rfc2833	iLBC encode 20 ms with RFC 2833	16.60	**mspCreateChannel**	**channelType** = ILBC20EncodeSimplex **FilterAttribs** = MSP_FCN_ATTRIB_RFC2833
f_ilbc_20.f41 f_ilbc_20.dec_rfc2833	iLBC decode 20 ms with RFC 2833	8.60	**mspCreateChannel**	**channelType** = ILBC20DecodeSimplex **FilterAttribs** = MSP_FCN_ATTRIB_RFC2833
f_ilbc_30.f41 f_ilbc_30.cod	iLBC encode 30 ms	17.10	**mspCreateChannel**	**channelType** = ILBC30EncodeSimplex
f_ilbc_30.f41 f_ilbc_30.dec	iLBC decode 30 ms	7.50	**mspCreateChannel**	**channelType** = ILBC30DecodeSimplex

DSP file / ASCII ID string	**Function**	**MIPS**	**Related API function**	**Related arguments**
f_ilbc_30.f41 f_ilbc_30.cod_rfc2833	iLBC encode 30 ms with RFC 2833	18.60	**mspCreateChannel**	**channelType**= ILBC30EncodeSimplex **FilterAttribs**= MSP_FCN_ATTRIB_RFC2833
f_ilbc_30.f41 f_ilbc_30.dec_rfc2833	iLBC decode 30 ms with RFC 2833	9.00	**mspCreateChannel**	**channelType=** ILBC30DecodeSimplex **FilterAttribs=** MSP_FCN_ATTRIB_RFC2833
g723.f41 g723.rec_64 g723.rec_53	G.723 record 6.4 kbit/s 5.3 kbit/s	 15.5 14.5	**adiStartRecording**	***encoding*** = ADI_ENCODE_G723_6 ADI_ENCODE_G723_5
g723.f41 g723.play g723.play g723.play_edtx	G.723 play 6.4 kbit/s 5.3 kbit/s Variable	 1.8 1.4 1.8	**adiStartPlaying**	***encoding*** = ADI_ENCODE_G723_6 ADI_ENCODE_G723_5 ADI_ENCODE_G723_EDTX_G_723_6 ADI_ENCODE_G723_EDTX_G_723_5
g726.f41 g726.play_32	G.726 play	7.33	**adiStartPlaying**	***encoding*** = ADI_ENCODE_G726
g726.f41 g726.rec_32	G.726 record	6.72	**adiStartRecording**	***encoding*** = ADI_ENCODE_G726
g729a.f41 g729a.rec_64	G.729 record 8 kbit/s	12.5	**adiStartRecording**	***encoding*** = ADI_ENCODE_G729A
g729a.f41 g729a.play g729a.play_edtx	G.729 play 8 kbit/s Variable	 2.8 2.8	**adiStartPlaying**	***encoding*** = ADI_ENCODE_G729A ADI_ENCODE_EDTX_G729A
gsm_ms.f41 gsm_ms.frgsm_play gsm_ms.play_100	MS-GSM play 8 kHz	1.60	**adiStartPlaying**	***encoding*** = ADI_ENCODE_GSM maxspeed = 100

DSP file / ASCII ID string	Function	MIPS	Related API function	Related arguments
gsm_ms.f41 gsm_ms.play_150	MS-GSM play 8 kHz	3.60	**adiStartPlaying**	***encoding*** = ADI_ENCODE_GSM maxspeed = 150
gsm_ms.f41 gsm_ms.play_200	MS-GSM play 8 kHz	4.20	**adiStartPlaying**	***encoding*** = ADI_ENCODE_GSM maxspeed = 200
gsm_mspl.f41 gsm_mspl.frgsm_play	MS-GSM play (Limited) 8 kHz	2.30	**adiStartPlaying**	***encoding*** = ADI_ENCODE_GSM maxspeed = 100
gsm_ms.f41 *gsm_mspl.f41* gsm_ms.frgsm_rec gsm_mspl.frgsm_rec	MS-GSM record 8 kHz	3.60	**adiStartRecording**	***encoding*** = ADI_ENCODE_GSM
ima.f41 ima.play_24	IMA/DVI ADPCM play 6 kHz	1.91	**adiStartPlaying**	***encoding*** = ADI_ENCODE_IMA_24
ima.f41 ima.play_32	IMA/DVI ADPCM play 8 kHz	1.62	**adiStartPlaying**	***encoding*** = ADI_ENCODE_IMA_32
ima.f41 ima.rec_24	IMA/DVI ADPCM record 6 kHz	1.91	**adiStartRecording**	***encoding*** = ADI_ENCODE_IMA_24
ima.f41 ima.rec_32	IMA/DVI ADPCM record 8 kHz	2.00	**adiStartRecording**	***encoding*** = ADI_ENCODE_IMA_32
mf.f41 mf.fdet_bcmpl	Forward detect, backward compelling	2.56	**adiStartMFDetector**	

DSP file / ASCII ID string	Function	MIPS	Related API function	Related arguments
mf.f41 mf.bdet_fcmpl	Backward detect, forward compelling	2.56	**adiStartMFDetector**	
mf.f41 mf.fdet_USA	MF detection	1.81	**adiStartMFDetector**	
mf.f41 mf.fdet	MF forward detection	1.81	**adiStartMFDetector**	
mf.f41 mf.bdet	MF backward detection	1.81	**adiStartMFDetector**	
nmsfax.f41 nmsfax	NaturalFax	11.25	See NaturalFax functions	
oki.f41 oki.play_24_100	OKI Play 6 kHz	2.10	**adiStartPlaying**	***encoding*** = ADI_ENCODE_OKI_24 maxspeed = 100
oki.f41 oki.play_32_100	OKI play 8 kHz	1.80	**adiStartPlaying**	***encoding*** = ADI_ENCODE_OKI_32 maxspeed = 100
oki.f41 oki.play_24_150	OKI play 6 kHz 1.5X	4.11	**adiStartPlaying**	***encoding*** = ADI_ENCODE_OKI_24 maxspeed = 150
oki.f41 oki.play_32_150	OKI play 8 kHz 1.5X	3.78	**adiStartPlaying**	***encoding*** = ADI_ENCODE_OKI_32 maxspeed = 150
oki.f41 oki.play_24_200	OKI play 6 kHz 2.0X	5.43	**adiStartPlaying**	***encoding*** = ADI_ENCODE_OKI_24 maxspeed = 200
oki.f41 oki.play_32_200	OKI play 8 kHz 2.0X	5.00	**adiStartPlaying**	***encoding*** = ADI_ENCODE_OKI_32 maxspeed = 200
oki.f41 oki.rec_24	OKI record 6 kHz	2.21	**adiStartRecording**	***encoding*** = ADI_ENCODE_OKI_24

DSP file / ASCII ID string	Function	MIPS	Related API function	Related arguments
oki.f41 oki.rec_32	OKI record 8 kHz	2.12	**adiStartRecording**	***encoding*** = ADI_ENCODE_OKI_32
ptf.f41 ptf.det_2f	2 single frequency or 1 tone pair	1.29	**adiStartToneDetector**	
ptf.f41 ptf.det_4f	4 single frequency or 2 tone pair	1.81	**adiStartCallProgress**	precmask!=0
rvoice.f41 rvoice.play_mulaw	mu-law play	0.63	**adiStartPlaying**	***encoding*** = ADI_ENCODE_MULAW
rvoice.f41 rvoice.play_alaw	A-law play	0.63	**adiStartPlaying**	***encoding*** = ADI_ENCODE_ALAW
rvoice.f41 rvoice.play_lin	WAVE play 8 kHz 16-bit	0.63	**adiStartPlaying**	***encoding*** = ADI_ENCODE_PCM8M16
rvoice.f41 rvoice.rec_mulaw	mu-law record	0.63	**adiStartRecording**	***encoding*** = ADI_ENCODE_MULAW
rvoice.f41 rvoice.rec_alaw	A-law record	0.63	**adiStartRecording**	***encoding*** = ADI_ENCODE_ALAW
rvoice.f41 rvoice.rec_lin	WAVE record 8 kHz 16-bit	0.63	**adiStartRecording**	***encoding*** = ADI_ENCODE_PCM8M16
rvoice_vad.f41 rvoice_vad.play_mulaw	mu-law play	0.63	**adiStartPlaying**	***encoding*** = ADI_ENCODE_MULAW
rvoice_vad.f41 rvoice_vad.play_alaw	A-law play	0.63	**adiStartPlaying**	***encoding*** = ADI_ENCODE_ALAW

DSP file / ASCII ID string	Function	MIPS	Related API function	Related arguments
rvoice_vad.f41 rvoice_vad.play_lin	WAVE play 8 kHz 16-bit	0.63	**adiStartPlaying**	***encoding*** = ADI_ENCODE_PCM8M16
rvoice_vad.f41 rvoice_vad.rec_mulaw	mu-law record	0.85	**adiCommandRecord** **adiStartRecording**	***encoding*** = ADI_ENCODE_MULAW
rvoice_vad.f41 rvoice_vad.rec_alaw	A-law record	0.85	**adiCommandRecord** **adiStartRecording**	***encoding*** = ADI_ENCODE_ALAW
rvoice_vad.f41 rvoice_vad.rec_lin	WAVE record 8 kHz 16-bit	0.85	**adiCommandRecord** **adiStartRecording**	***encoding*** = ADI_ENCODE_PCM8M16
tone.f41 tone.gen	Tone generator	0.61	**adiStartDial** **adiStartDTMF** **adiStartTones**	
voice.f41 voice.play_16_100	NMS play 16 kbit/s	2.95	**adiStartPlaying**	***encoding*** = ADI_ENCODE_NMS_16 maxspeed = 100
voice.f41 voice.play_24_100	NMS play 24 kbit/s	2.96	**adiStartPlaying**	***encoding*** = ADI_ENCODE_NMS_24 maxspeed = 100
voice.f41 voice.play_32_100	NMS play 32 kbit/s	2.95	**adiStartPlaying**	***encoding*** = ADI_ENCODE_NMS_32 maxspeed = 100
voice.f41 voice.play_64_100	NMS play 64 kbit/s	0.51	**adiStartPlaying**	***encoding*** = ADI_ENCODE_NMS_64 maxspeed = 100
voice.f41 voice.play_16_150	NMS play 16 kbit/s 1.5X speedup	5.86	**adiStartPlaying**	***encoding*** = ADI_ENCODE_NMS_16 maxspeed = 150

DSP file / ASCII ID string	Function	MIPS	Related API function	Related arguments
voice.f41 voice.play_24_150	NMS play 24 kbit/s 1.5X speedup	5.88	**adiStartPlaying**	***encoding*** = ADI_ENCODE_NMS_24 maxspeed = 150
voice.f41 voice.play_32_150	NMS play 32 kbit/s 1.5X speedup	5.95	**adiStartPlaying**	***encoding*** = ADI_ENCODE_NMS_32 maxspeed = 150
voice.f41 voice.play_64_150	NMS play 64 kbit/s 1.5X speedup	2.44	**adiStartPlaying**	***encoding*** = ADI_ENCODE_NMS_64 maxspeed = 150
voice.f41 voice.play_16_200	NMS play 16 kbit/s 2.0X speedup	7.41	**adiStartPlaying**	***encoding*** = ADI_ENCODE_NMS_16 maxspeed = 200
voice.f41 voice.play_24_200	NMS play 24 kbit/s 2.0X speedup	7.47	**adiStartPlaying**	***encoding*** = ADI_ENCODE_NMS_24 maxspeed = 200
voice.f41 voice.play_32_200	NMS play 32 kbit/s 2.0X speedup	7.53	**adiStartPlaying**	***encoding*** = ADI_ENCODE_NMS_32 maxspeed = 200
voice.f41 voice.play_64_200	NMS play 64 kbit/s 2.0X speedup	2.85	**adiStartPlaying**	***encoding*** = ADI_ENCODE_NMS_64 maxspeed = 200
voice.f41 voice.rec_16	NMS record 16 kbit/s	3.33	**adiStartRecording**	***encoding*** = ADI_ENCODE_NMS_16
voice.f41 voice.rec_24	NMS record 24 kbit/s	3.36	**adiStartRecording**	***encoding*** = ADI_ENCODE_NMS_24

DSP file / ASCII ID string	Function	MIPS	Related API function	Related arguments
voice.f41 voice.rec_32	NMS record 32 kbit/s	3.35	**adiStartRecording**	***encoding*** = ADI_ENCODE_NMS_32
voice.f41 voice.rec_64	NMS record 64 kbit/s	0.58	**adiStartRecording**	***encoding*** = ADI_ENCODE_NMS_64
wave.f41 wave.play_11_8b	WAVE play 11 kHz 8-bit	1.58	**adiStartPlaying**	***encoding*** = ADI_ENCODE_PCM11M8
wave.f41 wave.play_11_16b	WAVE play 11 kHz 16-bit	1.36	**adiStartPlaying**	***encoding*** = ADI_ENCODE_PCM11M16
wave.f41 wave.rec.11_8b	WAVE record 11 kHz 8-bit	1.59	**adiStartRecording**	***encoding*** = ADI_ENCODE_PCM11M8
wave.f41 wave.rec_11_16b	WAVE record 11 kHz 16-bit	1.20	**adiStartRecording**	***encoding*** = ADI_ENCODE_PCM11M16

Software echo cancellation MIPS

The CG 6060 includes an echo chip for providing hardware echo cancellation capabilities and to free up DSP resources. However, if an application requires flexibility and you must modify echo cancellation parameters, use DSP resources to provide software echo cancellation capabilities. For more information, refer to the *ADI Service Developer's Reference Manual*.

The following table provides the filter length, adaptation times, and MIPS consumption for software echo cancellation DPFs. Filter length represents the maximum echo delay that can be handled by the software echo canceler.

Note: MIPS in parentheses are best case scenarios. These numbers are guaranteed for the first four instances of echo per DSP core.

DSP file	Filter length (ms)	Adapt time (ms)	MIPS
echo.f41 echo.ln2_apt100	2	100	2.85
echo.f41 echo.ln2_apt50	2	200	2.78
echo.f41 echo.ln2_apt25	2	400	2.78
echo.f41 echo.ln2_apt12	2	800	2.78
echo.f41 echo.ln4_apt100	4	100	3.13
echo.f41 echo.ln4_apt50	4	200	2.98
echo.f41 echo.ln4_apt25	4	400	2.91
echo.f41 echo.ln4_apt12	4	800	2.91
echo.f41 echo.ln6_apt100	6	100	3.41
echo.f41 echo.ln6_apt50	6	200	3.19

DSP file	**Filter length (ms)**	**Adapt time (ms)**	**MIPS**
echo.f41 echo.ln6_apt25	6	400	3.04
echo.f41 echo.ln6_apt12	6	800	3.04
echo.f41 echo.ln8_apt100	8	100	3.69
echo.f41 echo.ln8_apt50	8	200	3.39
echo.f41 echo.ln8_apt25	8	400	3.24
echo.f41 echo.ln8_apt12	8	800	3.17
echo.f41 echo.ln10_apt100	10	100	3.97
echo.f41 echo.ln10_apt50	10	200	3.60
echo.f41 echo.ln10_apt25	10	400	3.37
echo.f41 echo.ln10_apt12	10	800	3.30
echo.f41 echo.ln16_apt100	16	100	4.80
echo.f41 echo.ln16_apt50	16	200	4.21
echo.f41 echo.ln16_apt25	16	400	3.91
echo.f41 echo.ln16_apt12	16	800	3.76

DSP file	Filter length (ms)	Adapt time (ms)	MIPS
echo.f41 echo.ln20_apt100	20	100	5.36
echo.f41 echo.ln20_apt50	20	200	4.62
echo.f41 echo.ln20_apt25	20	400	4.25
echo.f41 echo.ln20_apt12	20	800	4.03
echo_v3.f41 echo_v3.ln2_apt100	2	100	1.90 (1.68)
echo_v3.f41 echo_v3.ln2_apt50	2	200	1.70 (1.54)
echo_v3.f41 *echo_v3.ln2_apt25*	2	400	1.59 (1.46)
echo_v3.f41 echo_v3.ln2_apt12	2	800	1.55 (1.43)
echo_v3.f41 echo_v3.ln4_apt100	4	100	2.43 (1.95)
echo_v3.f41 echo_v3.ln4_apt50	4	200	2.11 (1.74)
echo_v3.f41 echo_v3.ln4_apt25	4	400	1.94 (1.63)
echo_v3.f41 echo_v3.ln4_apt12	4	800	1.85 (1.57)
echo_v3.f41 echo_v3.ln6_apt100	6	100	2.97 (2.22)
echo_v3.f41 echo_v3.ln6_apt50	6	200	2.51 (1.95)

DSP file	Filter length (ms)	Adapt time (ms)	MIPS
echo_v3.f41 echo_v3.ln6_apt25	6	400	2.27 (1.81)
echo_v3.f41 echo_v3.ln6_apt12	6	800	2.15 (1.73)
echo_v3.f41 echo_v3.ln8_apt100	8	100	3.51 (2.49)
echo_v3.f41 echo_v3.ln8_apt50	8	200	2.91 (2.15)
echo_v3.f41 echo_v3.ln8_apt25	8	400	2.62 (1.98)
echo_v3.f41 echo_v3.ln8_apt12	8	800	2.47 (1.89)
echo_v3.f41 echo_v3.ln10_apt100	10	100	4.04 (2.77)
echo_v3.f41 echo_v3.ln10_apt50	10	200	3.32 (2.36)
echo_v3.f41 echo_v3.ln10_apt25	10	400	2.97 (2.15)
echo_v3.f41 echo_v3.ln10_apt12	10	800	2.78 (2.04)
echo_v3.f41 echo_v3.ln16_apt100	16	100	5.65 (3.57)
echo_v3.f41 echo_v3.ln16_apt50	16	200	4.54 (2.97)
echo_v3.f41 echo_v3.ln16_apt25	16	400	3.98 (2.66)
echo_v3.f41 echo_v3.ln16_apt12	16	800	3.71 (2.51)

DSP file	Filter length (ms)	Adapt time (ms)	MIPS
echo_v3.f41 echo_v3.ln20_apt100	20	100	6.72 (4.11)
echo_v3.f41 echo_v3.ln20_apt50	20	200	5.36 (3.38)
echo_v3.f41 echo_v3.ln20_apt25	20	400	4.67 (3.01)
echo_v3.f41 echo_v3.ln20_apt12	20	800	4.32 (2.82)
echo_v3.f41 echo_v3.ln24_apt100	24	100	7.80 (4.65)
echo_v3.f41 echo_v3.ln24_apt50	24	200	6.18 (3.79)
echo_v3.f41 echo_v3.ln24_apt25	24	400	5.36 (3.36)
echo_v3.f41 echo_v3.ln24_apt12	24	800	4.95 (3.13)
echo_v3.f41 echo_v3.ln32_apt100	32	100	9.94 (5.74)
echo_v3.f41 echo_v3.ln32_apt50	32	200	7.80 (4.61)
echo_v3.f41 echo_v3.ln32_apt25	32	400	6.73 (4.04)
echo_v3.f41 echo_v3.ln32_apt12	32	800	6.19 (3.75)
echo_v3.f41 echo_v3.ln40_apt100	40	100	12.09 (6.82)
echo_v3.f41 echo_v3.ln40_apt50	40	200	9.43 (5.43)

DSP file	Filter length (ms)	Adapt time (ms)	MIPS
echo_v3.f41 echo_v3.ln40_apt25	40	400	8.10 (4.73)
echo_v3.f41 echo_v3.ln40_apt12	40	800	7.43 (4.37)
echo_v3.f41 echo_v3.ln48_apt100	48	100	14.23 (7.92)
echo_v3.f41 echo_v3.ln48_apt50	48	200	11.06 (6.27)
echo_v3.f41 echo_v3.ln48_apt25	48	400	9.47 (5.44)
echo_v3.f41 echo_v3.ln48_apt12	48	800	8.77 (5.02)
echo_v3.f41 echo_v3.ln64_apt100	64	100	18.52 (10.07)
echo_v3.f41 echo_v3.ln64_apt50	64	200	14.31 (7.89)
echo_v3.f41 echo_v3.ln64_apt25	64	400	12.20 (6.80)
echo_v3.f41 echo_v3.ln64_apt12	64	800	11.15 (6.25)
echo_v4.f41 echo_v4.ln2_apt100	2	100	3.742 (3.531)
echo_v4.f41 echo_v4.ln2_apt50	2	200	3.547 (3.387)
echo_v4.f41 echo_v4.ln2_apt25	2	400	3.441 (3.313)
echo_v4.f41 echo_v4.ln2_apt12	2	800	3.398 (3.273)

DSP file	Filter length (ms)	Adapt time (ms)	MIPS
echo_v4.f41 echo_v4.ln4_apt100	4	100	4.277 (3.805)
echo_v4.f41 echo_v4.ln4_apt50	4	200	3.949 (3.594)
echo_v4.f41 echo_v4.ln4_apt25	4	400	3.781 (3.480)
echo_v4.f41 echo_v4.ln4_apt12	4	800	3.695 (3.430)
echo_v4.f41 echo_v4.ln6_apt100	6	100	4.816 (4.066)
echo_v4.f41 echo_v4.ln6_apt50	6	200	4.359 (3.797)
echo_v4.f41 echo_v4.ln6_apt25	6	400	4.129 (3.652)
echo_v4.f41 echo_v4.ln6_apt12	6	800	4.008 (3.578)
echo_v4.f41 echo_v4.ln8_apt100	8	100	5.355 (4.344)
echo_v4.f41 echo_v4.ln8_apt50	8	200	4.770 (3.996)
echo_v4.f41 echo_v4.ln8_apt25	8	400	4.473 (3.824)
echo_v4.f41 echo_v4.ln8_apt12	8	800	4.316 (3.734)
echo_v4.f41 echo_v4.ln10_apt100	10	100	5.891 (4.609)
echo_v4.f41 echo_v4.ln10_apt50	10	200	5.180 (4.203)

DSP file	Filter length (ms)	Adapt time (ms)	MIPS
echo_v4.f41 echo_v4.ln10_apt25	10	400	4.816 (3.996)
echo_v4.f41 echo_v4.ln10_apt12	10	800	4.633 (3.895)
echo_v4.f41 echo_v4.ln16_apt100	16	100	7.496 (5.430)
echo_v4.f41 echo_v4.ln16_apt50	16	200	6.395 (4.816)
echo_v4.f41 echo_v4.ln16_apt25	16	400	5.832 (4.516)
echo_v4.f41 echo_v4.ln16_apt12	16	800	5.559 (4.355)
echo_v4.f41 echo_v4.ln20_apt100	20	100	8.570 (5.965)
echo_v4.f41 echo_v4.ln20_apt50	20	200	7.203 (5.230)
echo_v4.f41 echo_v4.ln20_apt25	20	400	6.523 (4.859)
echo_v4.f41 echo_v4.ln20_apt12	20	800	6.180 (4.680)
echo_v4.f41 echo_v4.ln24_apt100	24	100	9.648 (6.504)
echo_v4.f41 echo_v4.ln24_apt50	24	200	8.023 (5.637)
echo_v4.f41 echo_v4.ln24_apt25	24	400	7.203 (5.199)
echo_v4.f41 echo_v4.ln24_apt12	24	800	6.797 (4.980)

DSP file	Filter length (ms)	Adapt time (ms)	MIPS
echo_v4.f41 echo_v4.ln32_apt100	32	100	11.789 (7.598)
echo_v4.f41 echo_v4.ln32_apt50	32	200	9.648 (6.453)
echo_v4.f41 echo_v4.ln32_apt25	32	400	8.574 (5.891)
echo_v4.f41 echo_v4.ln32_apt12	32	800	8.035 (5.602)
echo_v4.f41 echo_v4.ln40_apt100	40	100	13.941 (8.680)
echo_v4.f41 echo_v4.ln40_apt50	40	200	11.277 (7.281)
echo_v4.f41 echo_v4.ln40_apt25	40	400	9.941 (6.574)
echo_v4.f41 echo_v4.ln40_apt12	40	800	9.277 (6.223)
echo_v4.f41 echo_v4.ln48_apt100	48	100	16.082 (9.773)
echo_v4.f41 echo_v4.ln48_apt50	48	200	12.902 (8.113)
echo_v4.f41 echo_v4.ln48_apt25	48	400	11.316 (7.289)
echo_v4.f41 echo_v4.ln48_apt12	48	800	10.523 (6.871)
echo_v4.f41 echo_v4.ln64_apt100	64	100	20.375 (11.914)
echo_v4.f41 echo_v4.ln64_apt50	64	200	16.156 (9.734)

DSP file	Filter length (ms)	Adapt time (ms)	MIPS
echo_v4.f41 echo_v4.ln64_apt25	64	400	14.055 (8.648)
echo_v4.f41 echo_v4.ln64_apt12	64	800	12.988 (8.102)

DSP files

The following files are included with Natural Access:

DSP file	Description
adsir(_j).f41	Contains the caller ID function that decodes the modem burst that occurs between the first and second rings on a loop start line. In addition, it contains the FSK data receiver. (_*j*) is the V.23 variant.
adsix(_j).f41	Contains the FSK data transmitter. (_*j*) is the V.23 variant.
amr.f41	Contains AMR play and record functions.
callp.f41	Contains voice and tone detectors used for call progress detection. Use for any outgoing or two-way trunk protocol and for call progress analysis.
cg6conf.f41	Contains functions for conferences that use only a single DSP. See *readme_cnf.txt* for more information.
cgcnfm.f41	Contains functions for conferences across multiple DSPs. This is for the master DSP. See *readme_cnf.txt* for more information.
cgcnfs.f41	Contains functions for conferences across multiple DSPs. This is for a slave DSP. See *readme_cnf.txt* for more information.
dtmf.f41	Contains the DTMF receiver, energy detector, silence detector, and precise tone filter typically used for cleardown.
dtmfe.f41	A variant of *dtmf.f41*, optimized for use with the echo canceler (*echo.f41*). It yields better talk-off resistance, but requires the echo canceler to achieve the best cut-through performance. **Note:** You must use echo cancellation with this function.

DSP file	Description
echo.f41	Contains the echo cancellation function. The echo canceler removes reflected transmit channel energy from the incoming signal, which improves DTMF detection and voice recognition while playing. CG board DSP echo functions are characterized by two parameters: tail length and adaptation rate. Tail length represents the maximum duration of the echo that can be cancelled, in ms. The adaptation rate specifies the percentage of the echo canceler filter coefficients that are adapted every period. The CG 6060 echo function has an adapt period of 8 ms. Therefore, an echo function with a 20 ms tail length and 100% rate will adapt all the coefficients in 8 ms while the same function with a 25% rate will adapt in 32 ms. **Note:** Substitute *dtmfe.f41* for *dtmf.f41* when using the echo canceler.
echo_v3.f41	Contains an improved echo cancellation function. This echo canceler presents a higher performance than the one in *echo.f41*. It also has a maximum tail length of 64 ms. **Note:** Substitute *dtmfe.f41* for *dtmf.f41* when using this echo canceler.
echo_v4.f41	Combines *echo_v3.f41* functionality with comfort noise generation and tone disabling enhancements.
f_amr.f41	Contains AMR encoder and decoder for voice over IP transmissions. See *readme_amr.txt* for more information.
f_evrc.f41	Contains EVRC encoder and decoder for voice over IP transmissions. See *readme_evrc.txt* for more information.
f_faxt38.f41	Contains T.38 encoder and decoder for voice over IP transmissions. See *readme_faxt38.txt* for more information.
f_g711.f41	Contains G.711 encoder and decoder for voice over IP transmissions. See *readme_g711.txt* for more information.
f_g711vad.f41	Contains G.711 VAD encoder and decoder for voice over IP transmissions. *See readme_g711.txt* for more information.
f_g723.f41	Contains G.723 encoder and decoder for voice over IP transmissions. See *readme_g723.txt* for more information.
f_g726.f41	Contains G.726 encoder and decoder for voice over IP transmissions. See *readme_g726.txt* for more information.
f_g729a.f41	Contains G.729a encoder and decoder for voice over IP transmissions. See *readme_g729a.txt* for more information.

DSP file	Description
f_gsm_fr.f41	Contains GSM-FR encoder and decoder for voice over IP transmissions. See *readme_gsm_fr.txt* for more information.
f_ilbc_20.f41	Contains iLBC 20 (20 ms frames) encoder and decoder for voice over IP transmissions. See *readme_ilbc.txt* and RFC 3951 for more information.
f_ilbc_30.f41	Contains iLBC 30 (30 ms frames) encoder and decoder for voice over IP transmissions. See *readme_ilbc.txt* and RFC 3951 for more information.
g723.f41	Contains ITU G.723.1 play and record functions for both 5.3 kbit/s and 6.4 kbit/s rates. The codec data is output as raw bytes of the encoded 30 ms frames.
g726.f41	Contains ITU G.726 ADPCM play and record functions. G.726 is a standard for 32 kbit/s speech coding. These functions require considerably more DSP processing time than the functions in *voice.f41*. *g726.f41* is required if you start play and record with an encoding type of ADI_ENCODE_G726.
g729a.f41	Contains ITU G.729A play and record functions. The 8 kbit/s codec data is output as raw bytes of encoded 10 ms frames.
gsm_ms.f41	Contains MS-GSM play and record functions. The 13 kbit/s full rate GSM speech codec outputs data in Microsoft formatted frames.
gsm_mspl.f41	Contains identical play and record functions as the *gsm_ms.f41* except that the maximum output power of the play function is maintained.
ima.f41	Contains IMA ADPCM play and record functions. IMA is a standard for 32 kbit/s speech encoding.
mf.f41	Contains the multi-frequency receiver required for any trunk control protocol (TCP) that uses MF signaling, and is required by the MF detector.
nmsfax.f41	Contains NaturalFax send and receive functions. See *readme_nfx.tx*t for more information.
oki.f41	Contains play and record functions for OKI ADPCM speech encoding, at 24 kbit/s or 32 kbit/s (used to play and record compatible voice files).
ptf.f41	Contains precise tone filters. Typically used for CNG, CED, or custom tone detection.

DSP file	Description
rvoice.f41	Contains PCM play and record functions. *rvoice.f41* is required to play or record with an encoding of ADI_ENCODE_MULAW, ADI_ENCODE_ALAW, or ADI_ENCODE_PCM8M16.
rvoice_vad.f41	Contains PCM play and record functions. Record functions can enable the voice activity detection (VAD) capability. *rvoice_vad.f41* is required to play or record with an encoding of ADI_ENCODE_MULAW, ADI_ENCODE_ALAW, or ADI_ENCODE_PCM8M16.
tone.f41	Contains the tone generation function. This file is required for any trunk protocol except NOCC. It is also required for generating tones, generating DTMF tones, MF tones, initiating dialing, and for generating a beep tone with any record function.
voice.f41	Contains NMS ADPCM play and record functions. The compressed speech is in a framed format with 20 ms of data per frame. Speech is compressed to 16, 24, or 32 kbit/s or stored as uncompressed mu-law or A-law (64 kbit/s). This file is required to play or record with encoding values of ADI_ENCODE_NMS_16, ADI_ENCODE_NMS_24, ADI_ENCODE_NMS_32, or ADI_ENCODE_NMS_64.
wave.f41	Contains play and record functions for PCM speech in formats commonly used in WAVE files, including 8 and 16 bit 11 kHz sampling.

For all NMS ISDN installations, load the following files:

- *dtmf.f41*
- *callp.f41*
- *tone.f41*

Additional *.f41* files are available for Fusion and fax configurations. For more information, refer to the *Dialogic® NaturalAccess™ Fusion™ VoIP API Developer's Manual* and *Dialogic® NaturalAccess™ NaturalFax™ API Developer's Manual*.

The *f41info* utility can be used to list DSP file resources. For more information, refer to f41info - Displaying DPF file resource usage.

14. T1 and E1 trunk channels

Channels and transmission rates

T1 and E1 are four-wire digital transmission links. T1 is used mainly in the United States, Canada, Hong Kong, and Japan. E1 is used in Europe.

Data on a T1 or E1 trunk is transmitted in channels. Each channel carries information digitized at 64,000 bits per second (b/s). This transmission rate is called the digital signal level 0 (DS-0) rate.

T1 carries 24 channels. E1 carries 32 channels. The total throughput rate (called digital signal level 1 or DS-1) is:

- For T1, 24 channels, each carrying 64,000 b/s, yield a throughput rate of 1,536,000 b/s. An extra 8000 b/s are used to carry framing and other information (as described in Framing). DS-1 for T1 is 1,544,000 b/s.
- For E1, 32 channels, each carrying 64,000 b/s, yield rate of 2,048,000 b/s.

Signaling

Two types of information are carried on a trunk:

- Voice information
- Signaling information indicating, for example, if a channel is on-hook or off-hook

Signaling information can be conveyed using either channel associated signaling (CAS) or common channel signaling (CCS).

Channel associated signaling (CAS)

With CAS, signaling information is sent for all channels at regular intervals, regardless of whether each channel's state changes. The information for each channel consists of a set of bits (called the ABCD bits). Whenever a channel's state changes, the ABCD bit pattern for that channel changes to convey the signaling bits.

On T1 trunks using the CAS protocol, such as Wink Start, the signaling information for each channel is transmitted using a method called robbed-bit signaling. With this method, one of the bits in the voice information in each channel is changed at regular intervals to indicate the state of the channel. Since the intervals are widely spaced, sound quality in the channel is not compromised.

On E1 trunks using a CAS protocol, such as Wink Start, channel 16 carries the ABCD bits for all of the other channels. No robbed-bit signaling is used.

Different CAS protocols use the ABCD bits in different ways. For example, MFC-R2 protocols use only two bits to signal four separate states; the other bits are not used. Pulsed E&M protocols convey signaling using one bit only, by setting and resetting the bit at specific intervals to signal different states. The specific patterns of bits used to indicate signaling states differ from country to country. Refer to the appropriate protocol reference manual for more information.

To interpret the signaling bits properly in a given country, your board must run a trunk control program (TCP) compatible with that country's protocol.

Common channel signaling (CCS)

With CCS, packets of signaling information for a channel are sent instead of signaling bits when the channel's state changes. CCS information is sent in a dedicated channel, the data channel or D channel. Voice information is carried in bearer channels (B channels).

On T1 trunks using a CCS protocol (such as ISDN), channel 24 is used as the D channel. It transmits packets of signaling information whenever the status of any of the other channels changes. No robbed-bit signaling is used. On E1 trunks using ISDN, the packets are sent in channel 16.

Framing

On T1 and E1 trunks, the data in the channels is combined into a single continuous stream of data using time-division multiplexing (TDM). With TDM, the channels take turns sharing the trunk over and over again. Each channel broadcasts 8 bits at a time. The time given a channel during a given round is called a timeslot. One cycle of timeslots is called a frame.

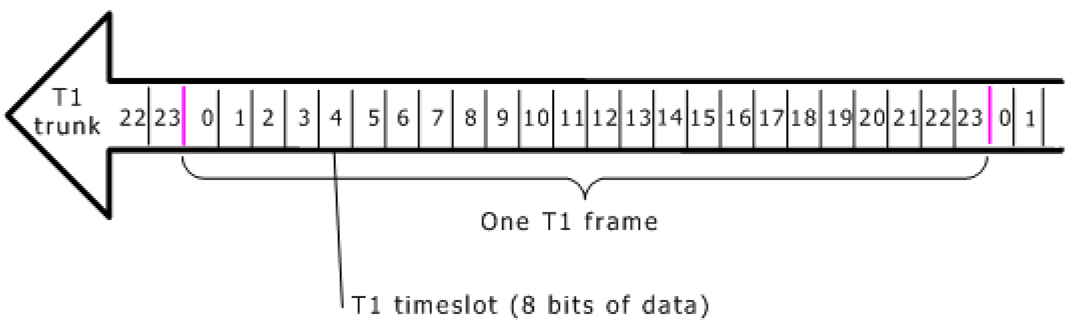
T1 and E1 delineate frames differently. This topic describes:

- T1 framing
- E1 framing

When configuring the CG 6060 board, you specify which framing format to use with the NetworkInterface.T1E1[x].FrameType keyword.

T1 framing

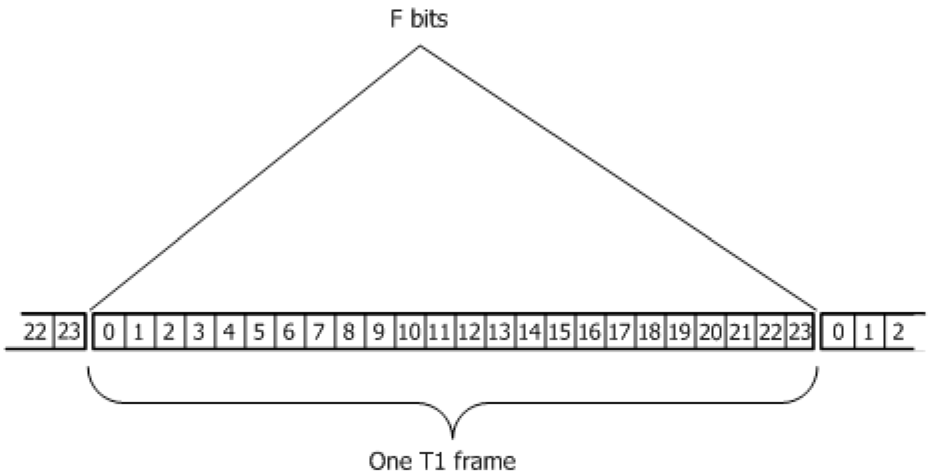
On T1 trunks, a frame consists of 24 timeslots, sent every 125 µsec (1/8000 sec). The following illustration shows a T1 frame:



The CG 6060 board supports two T1 framing formats: D4 framing and extended superframe (ESF).

With D4 framing, a single framing bit (F bit) is sent after each frame, to mark the end of the frame and the beginning of the next one. Each frame consists of (24x8)+1 = 193 bits. The framing bits (8000 per second) take up the extra bandwidth.

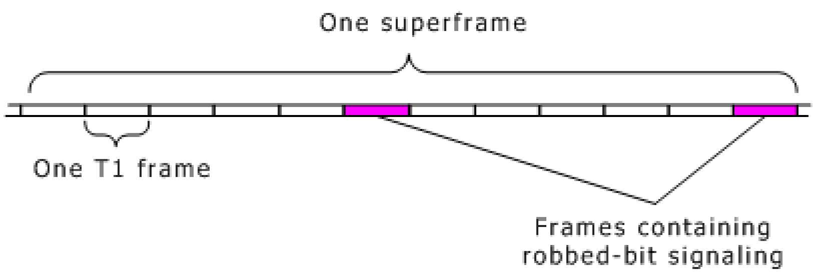
The following illustration shows the framing bits on a T1 trunk:



After each frame, the F bit is set or reset according to a pattern that repeats once every 12 frames: 100011011100. This makes the F bit recognizable even in the high-speed T1 bit stream. The 12 frames in this cycle constitute one superframe.

With CAS protocols, the least significant bit in each timeslot is robbed for signaling in the 6th and 12th frames in each superframe. Since each bit has only two possible states (0 or 1), only four separate signaling conditions can be transmitted with CAS protocols.

The following illustration shows robbed-bit signaling (D4 framing format):



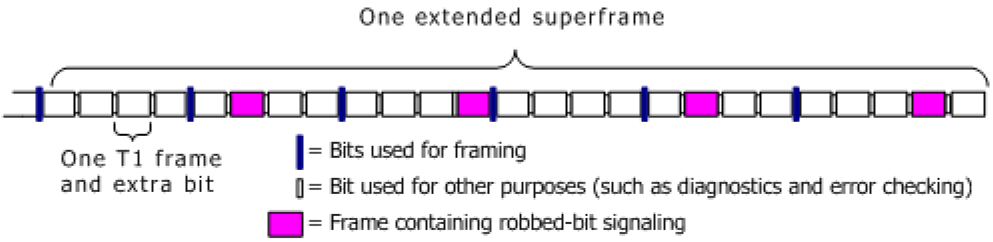
With ESF framing, an extra bit appears after every frame, as in D4 framing. However, only every fourth extra bit is used for framing. This bit is set or reset in a pattern that repeats once every 24 frames, instead of the 12-frame repetition in D4 framing. The 24 frames in the cycle constitute one extended superframe.

All of the other extra bits (18 in all) are used alternately:

- Six of the bits are used for a cyclic redundancy check (CRC), to detect errors.
- The other 12 carry diagnostic data. This bandwidth is called the facilities data link (FDL).

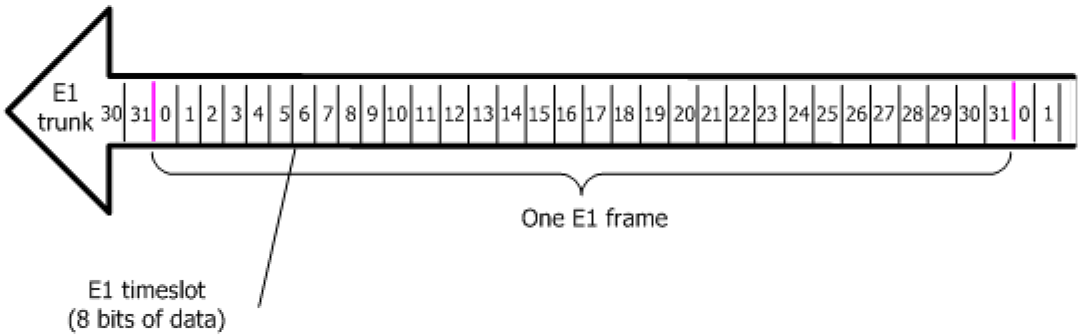
With CAS protocols, bits are robbed from each timeslot in the 6th, 12th, 18th, and 24th frame in the extended superframe (as shown in the following illustration). Instead of two signaling bits per superframe, ESF has 4 bits, allowing up to 16 separate signaling conditions to be transmitted.

The following illustration shows an extended superframe:



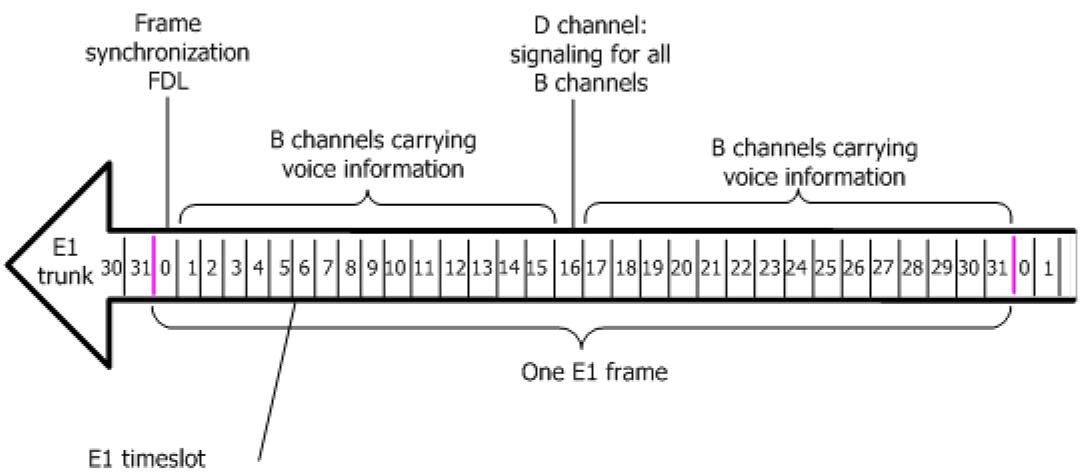
E1 framing

On E1 trunks, a frame consists of 32 timeslots. A frame is sent every 125 µsec (1/8000 sec). The following illustration shows an E1 frame:



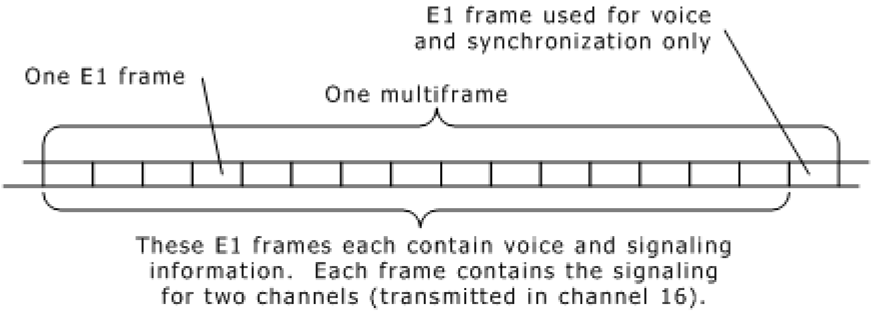
In each frame, channels are numbered 0 through 31. Half of the first channel (channel 0) is used for frame synchronization. The other half can be used as a facilities data link (FDL).

With CAS protocols, signaling information for each channel is carried in channel 16. This eliminates the need for robbed-bit signaling. Channels 1 through 15 and 17 through 31 (30 channels in all) carry voice information. The following illustration shows CEPT E1 timeslots:



With CAS protocols, four ABCD bits are sent for each channel at a time. Since timeslot 16 can carry only 8 bits of information per frame, it is not possible to send the signaling for all 30 channels in each frame. Therefore, channels take turns using channel 16, two at a time. It takes 15 frames to cycle through the signaling for all channels.

After every 15 frames, an extra frame is sent to synchronize the receiver to the signaling channel. Thus the full cycle contains 16 frames. A group of 16 such frames is called a multiframe. The following illustration shows an E1 multiframe:



Voice encoding

For the CG 6060 board, the information received is already pulse code modulation (PCM) encoded.

Companding

Only 256 possible amplitude measurements can be represented with 8 bits. 256 digital values are not enough to represent the entire amplitude range of the human voice at a usable quality level. However, most of the characteristics of a voice signal that make it understandable to the human ear exist at the lower end of the amplitude range. Therefore, the values are assigned to amplitude values non-linearly, with many values available to represent various amplitudes in the low end of the range, and few values to measure the high end. This compression method is called companding.

Different companding algorithms are used in different geographic regions. A companding method called mu-law is used in the US, Canada, and Japan. Another method, called A-law, is used in the rest of the world.

AMI, ones density, and zero code suppression

To reduce crosstalk on T1 and E1 trunks, and to eliminate DC bias, each 1 bit on the trunk is sent with the opposite electrical polarity of the preceding 1 bit. This transmission method is called alternate mark inversion (AMI).

Zero bits are sent as intervals of zero voltage. Multiple zeros in a row appear at the receiving end as one long interval of no voltage. If these gaps are too long, it is difficult for the receiving end to maintain framing synchronization with the transmitting end. There are various algorithms used in E1 transmissions to get around this problem, by ensuring that there are sufficient ones (enough ones density) to keep the transmitting and receiving ends synchronized. These algorithms are called zero code suppression algorithms.

CG 6060 boards configured as T1 boards support the following zero code suppression algorithms:

Algorithm	Description
B8ZS - binary 8-zero suppression	This is the algorithm used with ISDN protocols. To send an interval of successive zeroes, the sending end replaces the zeroes with a pattern of ones and zeroes in which bipolar violations occur; that is, one or more successive ones are sent with the same polarity, disrupting the AMI pattern. The pattern of bipolar violations is recognized at the receiving end and turned back into zeroes.
HDB3	High density bipolar 3 code uses patterns of bipolar violations to replace sequences of 4 zero data bits in order to maintain ones density on clear channel transmission.
Jammed bit 7 zero code suppression	In an interval of zeroes, the sending end jams every bit 7 high so the receiving end can recognize it. This method sacrifices data integrity, but quality is sufficient for voice transmissions.

CG 6060 boards configured as E1 boards can be set up to transmit without zero code suppression, or to use the high density bipolar 3 code (HDB3) algorithm. In HDB3, sequences of 4 zero data bits are replaced by patterns of bipolar violations.

When configuring the CG 6060 board, use the NetworkInterface.T1E1[x].LineCode keyword to specify which algorithm to use.

15. Utilities

Utility summary

This section provides detailed information about the following CG 6060 board utilities:

Utility	Description
f41info	Parses the contents of an *.f41* file and displays resources used by the DPFs associated with the specified DPM.
cg6kcon	Gathers information and statistics about an active CG board.
cg6ktool	Displays EEPROM and RAM contents on a CG board.
cgroute	Configures the IPv4 routing table for a CG board.
cgsetkey	Adds, updates, dumps, or flushes security association database (SAD) entries and security policy database (SPD) entries on the board.
cgtrace	Enables debugging and tracing of CG boards.
cgv6if	Adds, prints, and deletes IPv6 addresses for a CG board.

f41info - Displaying DPF file resource usage

Parses the contents of a *.f41* file and displays resources used by the DPFs associated with the specified DPM.

Usage

```
f41info f41name [options]
```

where ***f41name*** is the name of a supported CG board DPM file.

The following table lists the available ***options***:

Options	Use this option to...
-t	Display information about DPFs associated with the specified DPM in a table format.
-d	Display information about DPFs associated with the specified DPM.
-a	Display information about all DPFs in a table format. Process all *.f41* files found in the current working directory. Search the path specified by the AGLOAD environment variable.

Description

Run *f41info* to display information about specific DPMs (*.f41* files) that can run on CG board DSP resources. *f41info* displays the following information:

- DPM revision and creation date
- DPM size in bytes
- DPF MIPs requirements

Note: DPM MIPS requirements are listed according to the DPF hexadecimal identifier (not its string identifier). For a list of the hexadecimal IDs associated with DPF strings IDs, refer to the Hexadecimal and ASCII ID strings table.

For more information about managing on-board resources, refer to Managed DSP resources.

Hexadecimal and ASCII ID strings

The following table shows the ASCII string IDs and hexadecimal IDs of supported CG board DPFs. For echo cancellation, the ASCII string IDs associated with the hexadecimal corresponds to the version of the software echo cancellation DPF, for example, *echo.f41*, *echo_v3.f41*, or *echo_v4.f41*, specified as ***f41name***.

Hexadecimal ID string	ASCII ID string
0x020B00	voice.rec_64
0x020A00	voice.rec_32
0x020900	voice.rec_24
0x020800	voice.rec_16
0x020300	voice.play_64_100
0x021300	voice.play_64_150
0x022300	voice.play_64_200
0x020200	voice.play_32_100
0x021200	voice.play_32_150
0x022200	voice.play_32_200
0x020100	voice.play_24_100
0x021100	voice.play_24_150
0x022100	voice.play_24_200
0x020000	voice.play_16_100

Hexadecimal ID string	ASCII ID string
0x021000	voice.play_16_150
0x022000	voice.play_16_200
0x050100	signal.xmt
0x050A00	signal.rcv
0x050E00	signal.rcv_QA
0x0A0000	callp.gnc
0x0C0000	tone.gen
0x0D0800	rvoice.play_mulaw
0x0D1000	rvoice.play_alaw
0x0D0900	rvoice.play_mulaw_edtx
0x0D1100	rvoice.play_alaw_edtx
0x0D2000	rvoice.play_lin
0x0D4100	rvoice.rec_mulaw
0x0D4200	rvoice.rec_alaw
0x0D4400	rvoice.rec_lin
0x0D8900	rvoice.mu2mu
0x0D8A00	rvoice.mu2a
0x0D8C00	rvoice.mu2lin
0x0D9100	rvoice.a2mu
0x0D9200	rvoice.a2a
0x0D9400	rvoice.a2lin
0x0DA100	rvoice.lin2mu
0x0DA200	rvoice.lin2a

Hexadecimal ID string	ASCII ID string
0x0DA400	rvoice.lin2lin
0x0D8000	rvoice.passthru
0x0D0000	rvoice.passthru_play
0x0D4000	rvoice.passthru_rec
0x0D0800	rvoice_vad.play_mulaw
0x0D1000	rvoice_vad.play_alaw
0x0D2000	rvoice_vad.play_lin
0x0D4100	rvoice_vad.rec_mulaw
0x0D4200	rvoice_vad.rec_alaw
0x0D4400	rvoice_vad.rec_lin
0x080100	dtmf.det_dtmf
0x080200	dtmf.det_sil
0x080400	dtmf.det_clrdwn
0x080700	dtmf.dtmf_sil_clrdwn
0x080F00	dtmf.det_all
0x080100	dtmfe.det_dtmf
0x080200	dtmfe.det_sil
0x080400	dtmfe.det_clrdwn
0x080700	dtmfe.dtmf_sil_clrdwn
0x080E00	dtmf.det_sil_clrdwn_ced
0x080F00	dtmfe.det_all
0x090A00	mf.fdet_bcmpl
0x090C00	mf.bdet_fcmpl

Hexadecimal ID string	ASCII ID string
0x090100	mf.fdet_USA
0x090200	mf.fdet
0x090400	mf.bdet
0x160A00	echo.ln20_apt100 echo_v3.ln20_apt100 echo_v4.ln20_apt100
0x161A00	echo.ln20_apt50 echo_v3.ln20_apt50 echo_v4.ln20_apt50
0x162A00	echo.ln20_apt25 echo_v3.ln20_apt25 echo_v4.ln20_apt25
0x163A00	echo.ln20_apt12 echo_v3.ln20_apt12 echo_v4.ln20_apt12
0x160800	echo.ln16_apt100 echo_v3.ln16_apt100 echo_v4.ln16_apt100
0x161800	echo.ln16_apt50 echo_v3.ln16_apt50 echo_v4.ln16_apt50
0x162800	echo.ln16_apt25 echo_v3.ln16_apt25 echo_v4.ln16_apt25
0x163800	echo.ln16_apt12 echo_v3.ln16_apt12 echo_v4.ln16_apt12

Hexadecimal ID string	**ASCII ID string**
0x160500	echo.ln10_apt100 echo_v3.ln10_apt100 echo_v4.ln10_apt100
0x161500	echo.ln10_apt50 echo_v3.ln10_apt50 echo_v4.ln10_apt50
0x162500	echo.ln10_apt25 echo_v3.ln10_apt25 echo_v4.ln10_apt25
0x163500	echo.ln10_apt12 echo_v3.ln10_apt12 echo_v4.ln10_apt12
0x160400	echo.ln8_apt100 echo_v3.ln8_apt100 echo_v4.ln8_apt100
0x161400	echo.ln8_apt50 echo_v3.ln8_apt50 echo_v4.ln8_apt50
0x162400	echo.ln8_apt25 echo_v3.ln8_apt25 echo_v4.ln8_apt25
0x163400	echo.ln8_apt12 echo_v3.ln8_apt12 echo_v4.ln8_apt12
0x160300	echo.ln6_apt100 echo_v3.ln6_apt100 echo_v4.ln6_apt100
0x161300	echo.ln6_apt50 echo_v3.ln6_apt50 echo_v4.ln6_apt50

Hexadecimal ID string	ASCII ID string
0x162300	echo.ln6_apt25 echo_v3.ln6_apt25 echo_v4.ln6_apt25
0x163300	echo.ln6_apt12 echo_v3.ln6_apt12 echo_v4.ln6_apt12
0x160200	echo.ln4_apt100 echo_ v3.ln4_apt100 echo_v4.ln4_apt100
0x161200	echo.ln4_apt50 echo_v3.ln4_apt50 echo_v4.ln4_apt50
0x162200	echo.ln4_apt25 echo_v3.ln4_apt25 echo_v4.ln4_apt25
0x163200	echo.ln4_apt12 echo_v3.ln4_apt12 echo_v4.ln4_apt12
0x160100	echo.ln2_apt100 echo_v3.ln2_apt100 echo_v4.ln2_apt100
0x161100	echo.ln2_apt50 echo_v3.ln2_apt50 echo_v4.ln2_apt50
0x162100	echo.ln2_apt25 echo_v3.ln2_apt25 echo_v4.ln2_apt25
0x163100	echo.ln2_apt12 echo_v3.ln2_apt12 echo_v4.ln2_apt12

Hexadecimal ID string	ASCII ID string
0x164300	echo_v3.ln24_apt100 echo_v4.ln24_apt100
0x165300	echo_v3.ln24_apt50 echo_v4.ln24_apt50
0x166300	echo_v3.ln24_apt25 echo_v4.ln24_apt25
0x167300	echo_v3.ln24_apt12 echo_v4.ln24_apt12
0x164400	echo_v3.ln32_apt100 echo_v4.ln32_apt100
0x165400	echo_v3.ln32_apt50 echo_v4.ln32_apt50
0x166400	echo_v3.ln32_apt25 echo_v4.ln32_apt25
0x167400	echo_v3.ln32_apt12 echo_v4.ln32_apt12
0x164500	echo_v3.ln40_apt100 echo_v4.ln40_apt100
0x165500	echo_v3.ln40_apt50 echo_v4.ln40_apt50
0x166500	echo_v3.ln40_apt25 echo_v4.ln40_apt25
0x167500	echo_v3.ln40_apt12 echo_v4.ln40_apt12
0x164600	echo_v3.ln48_apt100 echo_v4.ln48_apt100
0x165600	echo_v3.ln48_apt50 echo_v4.ln48_apt50

Hexadecimal ID string	ASCII ID string
0x166600	echo_v3.ln48_apt25 echo_v4.ln48_apt25
0x167600	echo_v3.ln48_apt12 echo_v4.ln48_apt12
0x164800	echo_v3.ln64_apt100 echo_v4.ln64_apt100
0x165800	echo_v3.ln64_apt50 echo_v4.ln64_apt50
0x166800	echo_v3.ln64_apt25 echo_v4.ln64_apt25
0x167800	echo_v3.ln64_apt12 echo_v4.ln64_apt12
0x1B0100	oki.play_24_100
0x1B1100	oki.play_24_150
0x1B2100	oki.play_24_200
0x1B0200	oki.play_32_100
0x1B1200	oki.play_32_150
0x1B2200	oki.play_32_200
0x1B0900	oki.rec_24
0x1B0A00	oki.rec_32
0x1C0700	ptf.det_2f
0x1C7700	ptf.det_4f
0x1D0000	wave.play_11_16b
0x1D0100	wave.play_11_8b
0x1D0800	wave.rec_11_16b

Hexadecimal ID string	ASCII ID string
0x1D0900	wave.rec_11_8b
0x190000	adsir.rcv
0x190000	adsir_j.rcv
0x1A0000	adsix.xmt
0x1A0000	adsix_j.xmt
0x1E0000	nmsfax
0x2C0000	amr.rec_475
0x2C0100	amr.rec_515
0x2C0200	amr.rec_590
0x2C0300	amr.rec_670
0x2C0400	amr.rec_740
0x2C0500	amr.rec_795
0x2C0600	amr.rec_102
0x2C0700	amr.rec_122
0x2C0800	amr.play
0x2C0900	amr.play_edtx
0x2D0000	cmvt_sbc.record
0x2D0100	cmvt_sbc.play
0x260100	ima.play_24
0x260200	ima.play_32
0x260900	ima.rec_24
0x260A00	ima.rec_32

Hexadecimal ID string	**ASCII ID string**
0x270000	gsm_ms.frgsm_play gsm_mspl.frgsm_play gsm_ms.play_100
0x271000	gsm_ms.play_150
0x272000	gsm_ms.play_200
0x270100	gsm_ms.frgsm_rec gsm_mspl.frgsm_rec
0x410000	f_gsm_fr.cod
0x410100	f_gsm_fr.dec
0x410200	f_gsm_fr.cod_rfc2833
0x410300	f_gsm_fr.dec_rfc2833
0x2A0000	g723.play_53
0x2A0100	g723.rec_53
0x2A0200	g723.play_64
0x2A0300	g723.rec_64
0x2A0400	g723.play_edtx
0x2B0000	g729a.play
0x2B0100	g729a.record
0x2B0300	g729a.play_edtx
0x0F0200	g726.play_32
0x0F0a00	g726.rec_32
0x0F0300	g726.play_32_edtx
0x400000	f_g711.cod
0x400100	f_g711.dec

Hexadecimal ID string	ASCII ID string
0x400000	f_g711vad.cod
0x400100	f_g711vad.dec
0x400200	f_g711.cod_rfc2833
0x400300	f_g711.dec_rfc2833
0x400200	f_g711vad.cod_rfc2833
0x400300	f_g711vad.dec_rfc2833
0x430000	f_g723.cod
0x430100	f_g723.dec
0x430000	f_g723r.cod
0x430100	f_g723r.dec
0x430200	f_g723.cod_rfc2833
0x430300	f_g723.dec_rfc2833
0x430200	f_g723r.cod_rfc2833
0x430300	f_g723r.dec_rfc2833
0x440000	f_g729a.cod
0x440100	f_g729a.dec
0x440200	f_g729a.cod_rfc2833
0x440300	f_g729a.dec_rfc2833
0x450000	f_faxt38.relay
0x460000	f_g726.cod
0x460100	f_g726.dec
0x460200	f_g726.cod_rfc2833
0x460300	f_g726.dec_rfc2833

Hexadecimal ID string	ASCII ID string
0x470000	f_g728.cod
0x470100	f_g728.dec
0x470200	f_g728.cod_rfc2833
0x470300	f_g728.dec_rfc2833
0x4B0000	f_amr.cod
0x4B0100	f_amr.dec
0x4B0200	f_amr.cod_rfc2833
0x4B0300	f_amr.dec_rfc2833
0x4C0000	f_evrc.cod
0x4C0100	f_evrc.dec
0x4C0200	f_evrc.cod_rfc2833
0x4C0300	f_evrc.dec_rfc2833
0x500000	f_ilbc_20.cod
0x500100	f_ilbc_20.dec
0x500200	f_ilbc_20.cod_rfc2833
0x500300	f_ilbc_20.dec_rfc2833
0x510000	f_ilbc_30.cod
0x510100	f_ilbc_30.dec
0x510200	f_ilbc_30.cod_rfc2833
0x510300	f_ilbc_30.dec_rfc2833

Example 1

If you run this command:

```
f41info dtmf -d
```

Information similar to the following example appears:

```
Dialogic Corporation Show F41 File Info            Version 1.00

File name: C:\NMS\CG\LOAD\dtmf.f41
Revision : 0.2
Size     : 12964 bytes
Created  : Fri Jan 21 16:43:28 2000

FUNCTION STATE MIPS Msec In  Out Cmd Context Description
---------------------------------------------------------------------------
   807h   0   2.523  2   8v      16     113  DTMF, Silence, Cleardown
   801h   0   1.996  2   8v      16     113  DTMF only
   802h   0   0.773  2   8v      16     113  post-/pre- Silence only
   804h   0   1.359  2   8v      16     113  Cleardown only

POOL NAME                           SIZE  ADDRESS TYPE
------------------------------------------------------
.DTMF_TABLE                         1h       0h%    DATA_GLOBAL_FAST

SECTION NAME                        SIZE
----------------------------------------
.text                               e9h
```

The following table describes the output that appears:

Column	Description
FUNCTION	The lower byte is the DPF ID. The upper byte is the DPM family code.
STATE	There is one row of resource information per state. STATE indicates the state number. A P in the STATE column indicates that the resources used are in the DPF persist mode.
MIPS	MIPS (millions of instructions per second) used by the DPF.
Msec	DPF period in milliseconds.
In	Input frame size, in words. V indicates that the voice bit is set, meaning that this queue is typically circuit switched.
Out	Output frame size, in words. V indicates that the voice bit is set, meaning that this queue is typically circuit switched.
Cmd	Command packet size, in words.
Context	User context size, in words. The user context holds DPF static data.
Description	DPF functional description.

Column	Description
POOL NAME	Displays data pool definitions used by the DSP linker to relocate existing sections or to create new sections as scratch pad areas.
SECTION NAME	Displays only when the -d option is used. The .text section is program code while other sections represent a user-defined table.

Example 2

If you run this command:

```
f41info crc -t
```

Information similar to the following example appears:

```
Dialogic Corporation Show F41 File Info            Version 1.00

File name: C:\NMS\CG\LOAD\crc.f41
Revision : 0.2
Size     : 8154 bytes
Created  : Fri Jan 21 16:43:18 2000
                                              Slots     Memory       Packet
FUNCTION    Description               MIPS    In  Out   Data       Up  Down
---------------------------------------------------------------------------
  d08h      CRC Play mu-law           0.621   0   1     282         0    40
  d10h      CRC Play A-law            0.621   0   1     282         0    40
  d41h      CRC Record mu-law         0.621   1   0     245        40     0
  d42h      CRC Record A-law          0.621   1   0     245        40     0

Program Memory = 426 words

POOL NAME                           SIZE  ADDRESS TYPE
------------------------------------------------------
crcdebug                            100h    0h%   DATA
```

The output is similar to the preceding table with the following changes:

Column	Description
MIPS	Prefaced with state number. A single state is not displayed.
Slots	Circuit switched queues, as determined by the V (voice) bit. Consumes timeslots. V-bit set indicates a slot is used. Otherwise, the queue is assumed to be a packet (DSP to/from Host) queue.
Memory Data	DPM instance context size in words. Obtained by summing data requirements of queues, command and context, plus instance overhead.
Packet	Up - If the V-bit is clear, the queue is assumed to be a packet queue. Packet Up displays DSP to Host frame size in words. Down - If the V-bit is clear, the queue is assumed to be a packet queue. Packet Down displays Host to DSP frame size in words. Packet up and down sizes are cumulative. If there is more than one up or down queue, the sizes are summed.

cg6kcon - Displaying statistics about CG board activity

Gathers information and statistics about an active CG board.

Usage

```
cg6kcon options
```

Valid ***options*** include:

Option	Function
-b ***boardnumb***	CG board number. Default = 0
-i ***ipaddress***	IP address of a remote system
-p ***portnumber***	Port number. Default = 759

You can start a board by using any of the following options:

- -b
- -i and -p

To run *cg6kcon* across the host PCI interface, enter the following command:

```
cg6kcon -b boardnumber
```

To run *cg6kcon* across the Ethernet interface, enter the following command:

```
cg6kcon -i ipaddress
```

You can now enter a valid *cg6kcon* command.

Valid commands include:

Command	Description
help	Shows the full command set or details for a specified command name.
arptable	Displays current ARP table contents.
conptdet	Displays details on a specific connect point.
dump	Displays a hex or ASCII dump of the memory address specified (256 bytes display).
eeprom	Displays CG board-specific EEPROM contents.
eth	Displays detailed information about the Ethernet interfaces. For more information, see eth command.
ethcfg	Sets Ethernet interface configuration parameters. For more information, see ethcfg command.
ipallow	Allows the host computer with the specified IP address to query the board.
ipdisable	Disables the IP interface.

Command	Description
ipdisallow	Prevents the host computer with the specified IP address from querying the board.
ipenable	Enables the IP interface.
ipshow	Displays the IP addresses that can query the board.
ipv6	Displays the IPv6 configuration information. For more information, see ipv6 command.
ipv6if	Displays the IPv6 interface table. For more information, see ipv6if command.
ipv6nd	Displays the IPv6 neighbor discovery table. For more information, see ipv6nd command.
ipv6dest	Displays the IPv6 destination table. For more information, see ipv6dest command.
ipv6rtr	Displays the IPv6 default routers table. For more information, see ipv6rtr command.
ping	Sends an ICMP ping packet. This command contains its own syntax and commands. For more information, see ping command.
ppe	Displays current packet processing entities. For more information, see ppe command.
ppedet	Displays details on a specific packet processing entity.
resettutil	Resets the task utilization monitor snapshot or statistics.
rtpstat	Displays current RTP statistics for a session.
routetable	Displays current routing table contents.
starttutil	Starts the task utilization monitor snapshot or statistics.
stoptutil	Stops the task utilization monitor snapshot or statistics.

<table>
<tr><th>Command</th><th>Description</th></tr>
<tr><td>tasks</td><td>Displays the following status information for all tasks created on the CG board:
<table>
<tr><th>Status information</th><th>Description</th></tr>
<tr><td>Name</td><td>Name of the given task.</td></tr>
<tr><td>Prior</td><td>Priority of the task (32 = highest priority; 0 = lowest).</td></tr>
<tr><td>Context</td><td>Task context address.</td></tr>
<tr><td>State</td><td>Shows the current state of a task. Valid states include:
• Waiting - Task is asleep and waiting for work.
• Run/sch - Task is running or scheduled to run. Typically, the highest priority task is the task that is currently running, and all lower priority tasks in this state have been preempted by the higher priority task or tasks.
• Idle - Task is waiting on a trigger.
• Suspend - Task has been suspended.
• Halted - Task is at breakpoint.</td></tr>
</table></td></tr>
<tr><td>util</td><td>Shows the current and average system CPU utilization. For more information, see util command.</td></tr>
</table>

Procedure

Complete the following steps to run *cg6kcon*:

Step	Action
1	Open a command line window.
2	Enter the following command: `cg6kcon -b` ***boardnumber***
3	Enter one of the *cg6kcon* commands. *cg6kcon* performs the commands.
4	Exit the program by entering **q** or quit.

Description

Use *cg6kcon* as a diagnostic tool to monitor the flow of data to and from CG board communication processors. Use *cg6kcon* to:

- Verify that connections are set up appropriately between the CG board DSPs on the same host system.
- Verify that network connections are set up appropriately between different host systems.
- View the operating characteristics of an active CG board on a local system by specifying the local board number, for example:

```
cg6kcon -b 2
```

- View the operating characteristics of an active CG board on a remote system by specifying the configured IP address of the CG board, for example:

```
cg6kcon -i 197.23.57.212
```

Running cg6kcon from a remote host

By default, remote access is always disabled. You can authorize remote access only from the local system. Each remote system must be authorized to access a specific CG board.

The following commands allow you to enable and disable remote access:

Command	Description
ipenable	Authorizes remote access to *cg6kcon* and sets global access restrictions for any remote IP system that uses *cg6kcon*.
ipallow	Enables access by a specific remote system.
ipdisallow	Removes specific IP addresses from the remote access list.
ipdisable	Completely disables remote IP access to *cg6kcon*. ipdisable does not affect the rest of the on-board IP stack.

The following example shows enabling and disabling remote access for *cg6kcon*:

```
C:\>cg6kcon
Console program V1.0 : ['quit' to Exit]
                       [For multi-screen reply, 'more' to scroll]
>
>ipenable
Socket interface enabled
>
>ipallow 198.62.139.32
IP Address 198.62.139.32 added successfully
>
>ipshow
IP ADDRESSES
============
198.62.139.32
>
>ipdisallow 198.62.139.32
IP Address 198.62.139.32 deleted successfully
>
>ipshow
No allowable IP entries
>
>ipdisable
Socket interface disabled
>
```

util command

Use the util command to view the current and average CPU utilization of the CG board co-processor. This command does not provide any utilization information about the DSPs on the board.

CPU utilization	Description
Current	The value and graph of CPU utilization during the last second of operation.
Average	Average CPU utilization over the last 16 seconds of operation.
Idle Peak	Peak value reached by idle loop.

The current and average CPU values display as a percentage of available CPU. For example, a value of 25 means that 25 percent of the CPU is being utilized and 75 percent is available.

Note: The current CPU utilization number can vary considerably from moment to moment. Because of this variation, *cg6kcon* also displays the average CPU value.

eth command

Use the eth command to display detailed information about the operational status and statistical information for each Ethernet connection on the CG board. The operational status is provided in the following fields:

Field	Indicates if the...
Mode	Ethernet is running at a speed of 10 Mb or 100 Mb.
Duplex	Connection is running full duplex or half duplex.
State	Physical interface for the Ethernet is active (UP) or inactive (DOWN).

Note: If the State indicates that the physical interface is DOWN, usually the 10/100Base-T cable is not plugged in at the CG board or at the associated hub or router. Other possibilities include a hub or router that is not turned on or an incorrect cable. T1 cables and Ethernet cables are not interchangeable even though the connectors are the same.

If the State of the physical interface is DOWN, the other two fields (Duplex and Mode) are meaningless.

The Ethernet interfaces on the CG board use a standard known as NWAY Autonegotiation. This standard allows each link partner in an Ethernet connection to inform the other link partner of its speed and capabilities. The CG board supports all combinations of 10 Mb or 100 Mb with full or half duplex. The CG board uses the corresponding information from the other link partner and runs at the highest capability level the link partner can support.

The following capabilities are supported:

- 100 Mb full duplex
- 100 Mb half duplex
- 10 Mb full duplex
- 10 Mb half duplex

The following sample shows the eth command output:

```
>nopause
>eth
Context     Adp#     Ethernet Addr     Mode    Duplex   State
=========   ====   =================   =====   ======   =====
$1B9A98     0001   00:21:22:31:23:CA   100BT   FULL     UP
Statistics
==========
TX Collisions     : 0     Multi Collisions  : 0
TX Underruns      : 0     TX Overruns       : 0     TX CRC Error      : 0
TX CXR Lost       : 0     TX Excessive defer: 0     TX Excessive coll.: 0
TX Late Collisions: 0     TX SQE            : 0     TX channel error  : 0
RX Overruns       : 0     RX Bad Packets    : 0     RX Runt packets   : 0
RX Short events   : 0     RX Alignment errs : 0     RX Bad FCS        : 0
RX Packet Too long: 0     RX Out of range   : 0     RX In range errors: 0
RX pause packets  : 0     RX Symbol errors  : 0     RX Out of memory  : 0
Link Failures     : 0
Context     Adp#     Ethernet Addr     Mode    Duplex   State
=========   ====   =================   =====   ======   =====
$1BAC58     0002   00:21:22:31:23:CA     -       -      DOWN
Statistics
==========
TX Collisions     : 0     Multi Collisions  : 0
TX Underruns      : 0     TX Overruns       : 0     TX CRC Error      : 0
TX CXR Lost       : 0     TX Excessive defer: 0     TX Excessive coll.: 0
TX Late Collisions: 0     TX SQE            : 0     TX channel error  : 0
RX Overruns       : 0     RX Bad Packets    : 0     RX Runt packets   : 0
RX Short events   : 0     RX Alignment errs : 0     RX Bad FCS        : 0
RX Packet Too long: 0     RX Out of range   : 0     RX In range errors: 0
RX pause packets  : 0     RX Symbol errors  : 0     RX Out of memory  : 0
Link Failures     : 0
Current and Average Ethernet Interrupt Rates
============================================
Tx: 0/Sec  Rx: 0/Sec  Avg Tx: 0/Sec  Avg Rx: 0/Sec
Current and Average Packet Processing Rates
===========================================
Tx: 0/Sec  Rx: 0/Sec  Avg Tx: 0/Sec  Avg Rx: 0/Sec
```

The following table provides a description of the fields in the eth command output:

Field	Description
Context	Memory location for this Ethernet interface control block.
ADP#	Adapter number that indicates which Ethernet interface is displayed.
Ethernet Addr	Ethernet hardware address.
Mode and Duplex	Information about the results of the negotiation. The capability limitations of the link partner can impose lower settings. Mode indicates Ethernet mode (10Base-T or 100Base-T) and duplex indicates full or half duplex connection.
State	Indicates whether the Ethernet is UP or DOWN.

Field	Description
Statistics: TX Collisions TX Multi Collisions	Statistical information that is valid only when the Ethernet connection is half duplex. These statistics show the number of times a transmit signal was deferred or a transmit collision occurred due to the Carrier Sense Multiple Access with Collision Detect (CSMA/CD) algorithm defined by the Ethernet standard. Informational use only.
Statistics: TX Underruns, TX Overruns, TX CRC Error, and so on	Statistical information about errors that occurred on this Ethernet link. The Ethernet logic on the CG board detects and recovers from any errors on the Ethernet link. These errors are therefore informational, and display the general quality of the local Ethernet segment.
Current and Average Ethernet Interrupt Rates	Current and average interrupt rates for the Ethernet. CG boards use various forms of interrupt mitigation logic to minimize the effect of interrupts on the system. These statistics in combination with the packet statistics are used to verify the efficiency of the interrupt mitigation logic.
Current and Average Packet Processing Rates	Current and average packet rates for the Ethernet. Because current packet rates can vary significantly from moment to moment, the average packet rates are also displayed. Displays the current number of packets transmitted and received in the previous second and the average number of packets transmitted and received over the last 16 seconds.

ethcfg command

The ethcfg command takes the following arguments:

```
ethcfg [int=interface#] [auto | [[half|full] [10|100]]] [prom]
```

where...

Argument	Description
int=***interface#***	Configures only the interface indicated by the ***interface#***.
auto	Determines speed selection and duplex mode through auto-negotiation. If ***interface#*** is omitted, both interfaces 1 and 2 are configured.
half|full	Half-duplex or full-duplex operation. Default = auto.
10|100	Ethernet interface transfer speed. Default = auto.
prom	Ethernet interface runs in promiscuous mode (in which a network device can read arriving packets).

ppe command

A packet processing entity (PPE) is an entity that performs some form of packet processing on a CG board. After booting the CG board, use the ppe command to display information similar to the following example:

```
>ppe

 Name                State   Type      Reg CPTs Ena CPTs   Context
 ============        ======= ========= ======== ========   ========
 Ethernet-1          Active  Ethernet         2        2   $CD6D30
 IP_Over_Eth-1       Active  IP/Ether         2        2   $C7EE10
 IP_Router           Active  IP Router        2        2   $C7E6A4
 UDP                 Active  UDP              1        2   $C7DBC4
>
```

This example shows the following PPEs:

PPE	Description
Ethernet-1	Specifies the Ethernet driver for interface 1.
IP_Over_Eth-1	Processes IP packets for Ethernet 1.
IP_Router	Manages the routing of packets to the correct Ethernet interface based on the configured IP Routing table and the destination IP address of each packet.
UDP	Specifies the UDP number associated with the IP stack. This interface provides a socket-based interface to the rest of the CG board software.

Note: Non-IP packets are forwarded to the host Ethernet task. If you install the CG board Ethernet driver on the host, these non-IP packets are forwarded to the host protocol stack.

When you create an RTP Endpoint, *cg6kcon* displays three additional PPEs associated with each RTP session:

PPE	Description
RTP In (simplex)	Connects to the UDP layer using a socket, receives all RTP packets from the IP network with the matching UDP port number and the local IP address (if specified).
RTP Out (simplex)	Connects directly to the Ethernet PPE, and transmits all outbound RTP packets to the IP network (half duplex).
RTP Full Duplex (duplex)	Manages a typical RTP/voice session operating in full duplex.

The displayed Context address is used to identify a particular PPE when using ppedit.

Use the ppedit command to display detailed information about a specific PPE. The following example shows details about the IP router PPE:

```
>ppedit $C7e6a4
 me                     State   Type       Reg CPTs  Ena CPTs       Context
 ==================     ======= ========   ========   ========      ========
 IP_Router              Active  IP Router         2          2      $C7E6A4

 Active Connect Points  State  Addr 1   Addr 2   Addr 3   Addr 4    Context
 ====================   ====== ======== ======== ======== ========  =======
 ICMP                   Active   1             0       0        0   $C78D34
 UDP                    Active  11             0       0        0   $C7DB14

 Statistics
 ==========

 TX Pkts: 0x1E         TX Bytes: 0x9DB        TX Fails: 0x0
 RX Pkts: 0x6          RX Bytes: 0x2AA        RX Drops: 0x0
>
```

The first line in the example repeats the information provided in the ppe command.

The Active Connect Points section provides information about the active connect points using this PPE. A connect point is a socket-like connection to the PPE. Two connect points, the UDP and ICMP protocol layers, are currently registered with the IP Router.

The Statistics section provides information about the number of packets transmitted or received by this protocol layer. RX Drop displays the number of packets discarded because no socket is registered for the address contained in the received packet. In the example, ICMP is registered for IP packets with an IP Protocol field. This configuration indicates that:

- The IP packet is an ICMP packet (1).
- UDP is registered for IP packets with an IP Protocol field indicating that the IP packet is a UDP packet (11).

For example, if another type of IP packet is received for TCP, the packet is discarded except when the CG board host Ethernet driver is installed on the host.

IPv6 command examples

The following examples show how to use the IPv6 commands supported by *cg6kcon*. They include:

- ipv6 command
- ipv6if command
- ipv6nd command
- ipv6dest command
- ipv6rtr command
- ping command

ipv6 command

The ipv6 command displays the current IPv6 configuration information based on the board keyword file. For example:

```
>ipv6
Link #1
Link MTU : 1500
Hop Limit : 64
PING Enabled : 1
ICMPv6 Rate Limit : 100 pkts/sec
Neighbor Discovery Retransmission Attempts : 3
Neighbor Discovery Retransmission Timeout : 1000 Milliseconds
Neighbor Discovery Reachability Timeout : 30000 Milliseconds
Link #2
Link MTU : 1500
Hop Limit : 128
PING Enabled : 1
ICMPv6 Rate Limit : 100 pkts/sec
Neighbor Discovery Retransmission Attempts : 3
Neighbor Discovery Retransmission Timeout : 1000 Milliseconds
Neighbor Discovery Reachability Timeout : 30000 Milliseconds
```

ipv6if command

The ipv6if command displays all IPv6 addresses associated with the board. Each IPv6 address is derived from the MAC address of the Ethernet device. Refer to *RFC 2373 IP Version 6 Addressing Architecture* for more information.

For example:

```
>ipv6if
IPv6 Address                                 Eth  State     MTU
===========================================  =======
FE80:0000:0000:0000:0220:22FF:FE40:2E20/ 64    1    Active  1500
FF02:0000:0000:0000:0000:0001:FF40:2E20/128    1    Active  1500
FF02:0000:0000:0000:0000:0000:0000:0001/128    1    Active  1500
FE80:0000:0000:0000:0220:22FF:FE40:2E21/ 64    2  InActive  1500
FF02:0000:0000:0000:0000:0001:FF40:2E21/128    2  InActive  1500
FF02:0000:0000:0000:0000:0000:0000:0001/128    2  InActive  1500
0000:0000:0000:0000:0000:0000:0000:0001/128    3    Active  1500
0001:0000:0000:0000:0220:22FF:FE40:2E20/ 64    1    Active  1500
FEC0:0000:0000:0004:0220:22FF:FE40:2E20/ 64    1    Active  1500
FEC0:0000:0000:0003:0220:22FF:FE40:2E20/ 64    1    Active  1500
FEC0:0000:0000:0002:0220:22FF:FE40:2E20/ 64    1    Active  1500
FEC0:0000:0000:0001:0220:22FF:FE40:2E20/ 64    1    Active  1500
```

ipv6nd command

The ipv6nd command displays the board's IPv6 neighbor discovery table. For example:

```
>ipv6nd
Statistics
==========
TX Neighbor Solicit: 54    TX Neighbor Advert : 54    TX Router Solicit: 1
RX Neighbor Advert : 54    RX Neighbor Solicit: 54    RX Router Advert : 2555
RX Redirects       : 0     TX Dup Addr Detect : 18    DSP Signal       : 0
ERRORS
======
RX Inv Hop Limit   : 0     RX Inv Options Len : 0     RX Inv Packet Len: 0
RX Inv Target Addr : 0     RX Inv Dest IP Addr: 0     RX Inv SrcIP Addr: 0
RX Inv Solicit Flag: 0     No Matching NDEntry: 0     RX NS during DAD : 0
IPv6 Address                             Eth  State       MAC Address
=======================================  ===  ==========
FE80:0000:0000:0000:02A0:24FF:FE23:5A0E    1  Stale       00A0:2423:5A0E
FE80:0000:0000:0000:0260:08FF:FE96:5E31    1  Stale       0060:0896:5E31
FE80:0000:0000:0000:0220:22FF:FE40:42BA    1  Stale       0020:2240:42BA
FE80:0000:0000:0000:0202:FDFF:FEBA:5CE1    1  Stale       0002:FFFFFDBA:5CE1
```

ipv6dest command

The ipv6dest command displays the board's IPv6 destination table. For example:

```
>ipv6dest
Eth Num : 1   Path MTU : 1500
Destination Addr :FE80:0000:0000:0000:0220:22FF:FE40:42BA
Next Hop Addr    :FE80:0000:0000:0000:0220:22FF:FE40:42BA
Default Src Addr :FE80:0000:0000:0000:0220:22FF:FE40:2E20
```

ipv6rtr command

The ipv6rtr command displays the board's IPv6 default routers table. For example:

```
>ipv6rtr
Router Address                          Eth  State     LifeTime Pref
======================================= ===  ========
FE80:0000:0000:0000:0202:FDFF:FEBA:5CE1   1    Active      1695 0
FE80:0000:0000:0000:02A0:24FF:FE23:5A0E   1    Active      1278 0
FE80:0000:0000:0000:0260:08FF:FE96:5E31   1    Active      1323 0
```

ping command

The ping command sends an IPv4 or IPv6 ICMP ping packet. To use this command, enter the command with the following arguments:

```
ping -i ifnumb -s size -c count -t interval ipversion ipaddress
```

Valid arguments include:

Arguments	Description
-i ***ifnumb***	Network interface number (1 or 2).
-s ***size***	Size of the packet (optional). Default = 4 bytes + IP header.
-c ***count***	Packet count (optional). Default = 1.
-t ***interval***	Time between each packet (optional). Default = 1.
ipversion	Protocol version. Valid options include: v4 = IPv4 v6 = IPv6
ipaddress	IP address for this interface.

cg6ktool - Displaying EEPROM and RAM

Displays EEPROM contents and dumps the on-board error log.

Usage

```
cg6ktool options
```

where ***options*** are:

<table>
<tr><th>Options</th><th>Description</th></tr>
<tr><td>-A</td><td>Lists the board's bus and slot information.</td></tr>
<tr><td>-B -lbus:slot</td><td>Blinks the board's LED.</td></tr>
<tr><td>-R -lbus:slot</td><td>Resets a specified board.</td></tr>
<tr><td>-M -lbus:slot
-aaddress
-ssize</td><td>Dumps memory in binary format to the file memdump_at_bus_slot.txt. If the starting byte (address) and number of bytes (size) are not specified, the defaults are 0 and 1 MB respectively. To dump the file in ASCII format, add the -t1 option.</td></tr>
<tr><td>-tfiletype</td><td>Specifies the file type (text = 1, binary = 2).</td></tr>
<tr><td>-S -lbus:slot</td><td>Dumps error and stack information to the file errdump_at_bus_slot.txt.</td></tr>
<tr><td>-eboarddeviceID</td><td>Applies the specified command to all CG boards of the specified type in the chassis. Can be used instead of -lbus:slot.
<table>
<tr><th>CG board type</th><th>Device ID</th></tr>
<tr><td>CG 6060</td><td>6060</td></tr>
<tr><td>CG 6060C</td><td>6061</td></tr>
<tr><td>CG 6565</td><td>6565</td></tr>
<tr><td>CG 6565C</td><td>6566</td></tr>
</table>
</td></tr>
<tr><td>-h</td><td>Display help information.</td></tr>
</table>

An option applies to all boards if you do not specify either -l or -e.

cg6ktool supports multiple boards. Some of these commands are shown for reference only and may not pertain to the CG 6060 board.

cgroute - Setting up CG board IPv4 routing tables

Configures the IPv4 routing table for a CG board.

Usage

```
cgroute command destination MASK netmask gateway -i interface -b boardnumber -p persistence
```

Valid commands include:

Command	Description
print	Prints the routing table.
add	Adds a route.
delete	Deletes a route.
deleteall	Deletes all the persistent entries in the routing table.

And valid arguments include:

Argument	Description
destination	IP address for this route entry.
netmask	Subnet mask value for this route entry. The default value is 255.255.255.255. This argument is always preceded with the string MASK.
gateway	Gateway address for this route.
-i ***interface***	CG board Ethernet adapter (1 or 2) associated with the specified IP address.
-b ***boardnumber***	Number of the CG board. The default value is 0.

Argument	Description
-p ***persistence***	Specifies whether or not a specified IP route is saved in non-volatile RAM and is automatically reloaded when the board is rebooted. When you enter -p1, the specified routing information is saved in non-volatile RAM. The routing information is automatically retrieved and reloaded into the board's IP routing table when the CG board is rebooted as in the following example: `cgroute 1.1.1.1 mask 255.255.255.0 -p 1` A value of 0 (the default) specifies that the address is not persistent across reboots. The number of persistent routing table entries you can specify is limited to: • Two Ethernet interface addresses (per Ethernet adapter on the board) • Five route entries • Two default gateway addresses

Description

Use *cgroute* to add or delete routes from the routing table and print current routing table contents.

Caution:	*cgroute* adds and deletes routing table entries from the CG board routing table but does not update the host operating system's routing table.

When booting a CG board, use *cgroute* to set the board's IP address. Use the add command to enter the interface address for each Ethernet adapter (that is, the board's IP address) in the routing table.

The following example routes all packets directed to the IP address subnet 198.62.139.***x*** to the gateway with IP address 198.62.139.1:

```
cgroute add 198.62.139.12 mask 255.255.255.0 198.62.139.1
```

cgsetkey - Configuring IPv6 security keys and policies

Adds, updates, dumps, or flushes security association database (SAD) entries and security policy database (SPD) entries on the board.

Usage

```
cgsetkey [option] - f filename
```

or

```
cgsetkey [option] command operation arguments
```

Where valid commands are:

Command	Description
- c	Carries out a series of operations from standard input. For information about valid arguments to include within the file, refer to the options table that follows.
-D	Displays all SAD entries.
- F	Flushes all SAD entries.
- f ***filename***	Carries out a series of operations from a specified file (***filename***). For information about valid arguments to include within the file, refer to the options table that follows.

Options

Valid options include:

Option	Description
-b ***boardnumb***	Indicates the OAM board number of the board to configure.
-v	Runs *cgsetkey* in verbose mode. The program dumps messages exchanged on PF_KEY socket.
-P	Displays all SPD and SAD entries when used with the -D command. Flushes all SPD and SAD entries when used with the -F command.
-d	Prints debugging messages for the command parser without communicating with the kernel.

Operations specified through standard input or through a file must use the following syntax:

operation arguments;

Operations

The following table provides a description of the valid operation. For more information on ***arguments***, refer to Arguments.

Operation	Description
add ***arguments***;	Adds an SAD entry. The add operation takes the following form: `add` ***`src dst protocol spi`*** `-t` ***`tag algorithm`***`... ;`
get ***arguments***;	Retrieves a particular SAD entry. The get operation takes the following form: `get` ***`src dst protocol spi`*** `;`
delete ***arguments***;	Deletes an SAD entry. The delete operation takes the following form: `delete` ***`src dst protocol spi`*** `;`
deleteall ***arguments***;	Deletes all SAD entries specified by ***arguments***. The deleteall operation takes the following form: `deleteall` ***`src dst protocol`*** `;`
flush;	Removes all SAD entries specified by ***arguments***. The flush operation takes the following form: `flush;`
dump ***arguments***;	Displays all SAD entries specified by ***arguments***. The dump operation takes the following form: `dump` ***`protocol`***`;`
spdadd ***arguments***;	Adds an SPD entry. The spdadd operation takes the following form: `spdadd` ***`src_range dst_range upperspec policy`*** `;`
spddelete ***arguments***;	Deletes an SPD entry. The spddelete operation takes the following form: `spddelete` ***`src_range dst_range upperspec`*** `-P` ***`direction`*** `;`
spdflush;	Clears all SPD entries (as well as all linked SA entries).
spddump;	Displays all SPD entries.

Note: In text files, lines beginning with a number sign (#) are regarded as comments. All lines containing an operation must end with a semicolon (;). Spaces within statements are ignored.

Arguments

Valid arguments include:

Argument	Description
src	Source of the secure communication specified in numeric form. DNS lookups are not performed.
dst	Destination of the secure communication specified in numeric form. DNS lookups are not performed.
protocol	Security protocol to implement. Valid protocols include: esp - encapsulating security payload header ah - authentication header When you specify ESP as the protocol, you can associate both an encryption and an authentication algorithm with the entry.
spi	Security Parameter Index (SPI) for the SAD and the SPD. This value must be a decimal number or hexadecimal number. You cannot use SPI values in the range 0 through 255 (hexadecimal numbers are permitted when prefixed with 0x).
tag	Used in the following form: `-t id` where ***id*** specifies the identifier of the policy entry in the SPD.

<table>
<tr><th>Argument</th><th>Description</th></tr>
<tr><td>algorithm</td><td>
Specifies an encryption or authentication algorithm to use. Allowed values vary depending on the specified algorithm. Valid arguments include the following:
<table>
<tr><th>Argument</th><th>Description</th></tr>
<tr><td>-E ealgo key</td><td>Encryption algorithm.</td></tr>
<tr><td>-A aalgo key</td><td>Authentication algorithm. When -A is used in conjunction with the esp protocol, it is treated as the ESP payload authentication algorithm.</td></tr>
</table>
In either case, the key must be a double-quoted character string or series of hexadecimal digits (prefixed by 0x).

The following encryption algorithm is supported when -E ealgo is specified as the algorithm argument:
<table>
<tr><th>Algorithm</th><th>Key length in bits</th><th>Description</th></tr>
<tr><td>des-cbc</td><td>64</td><td>ESP-old: RFC 1829, ESP: RFC 2405</td></tr>
</table>
The following authentication algorithms are supported when -A aalgo is specified as the algorithm argument:
<table>
<tr><th>Algorithm</th><th>Key length in bits</th><th>Description</th></tr>
<tr><td>hmac-md5</td><td>128</td><td>AH: RFC 403</td></tr>
<tr><td>hmac-sha1</td><td>160</td><td>AH: RFC 2404</td></tr>
</table>
When you specify ah as the protocol argument, you can use only -A to specify an authentication algorithm. When you specify esp as the protocol argument, you can use both -E to specify an encryption algorithm and -A to specify an authentication algorithm.
</td></tr>
<tr><td>src_range</td><td>
IPv6 source address or range of IPv6 source addresses to add or delete. This argument can be accompanied by a TCP/UDP port specification. Addresses and address ranges take the following form:

address
address/prefixlen
address[UDPport]
address/prefixlen[UDPport]

prefixlen and UDPport must be specified as decimal numbers. address and UDPport must be expressed in numeric form.
</td></tr>
</table>

Argument	Description
dst_range	IPv6 destination address or range of IPv6 destination addresses to add or delete. This argument can be accompanied by a TCP/UDP port specification. Addresses and address ranges can take the following form: • ***address*** • ***address/prefixlen*** • ***address[UDPport]*** • ***address/prefixlen[UDPport]*** ***prefixlen*** and ***UDPport*** must be specified as decimal numbers. ***address*** and ***UDPport*** must be expressed in numeric form.
upperspec	Upper-layer protocol to use. The following protocols are supported: • udp • icmp6 • any • ***protocolnumb*** where ***protocolnumb*** represents the protocol number in decimal notation.

<table>
<tr><th>Argument</th><th>Description</th></tr>
<tr><td>policy</td><td>IPSec policy argument that takes one of the following forms:

-P direction discard
-P direction bypass
-P direction ipsec protocol/transport/tag:index

Where:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>direction</td><td>Direction of the policy can be set to either out or in. Values include:

discard
Drop packets matching the specified policy.
bypass
No IP security is required for packets associated with the specified policy.
ipsec
IPSec is required for packets associated with the specified policy.
</td></tr>
<tr><td>protocol</td><td>Protocol to use that can be set to either esp (encapsulating security payload) or ah (authentication header). A value of transport establishes that packets are transferred using the transport mode.</td></tr>
<tr><td>tag:index</td><td>Number (index) between 1 and 32767 with which to bind the policy and create a unique identifier for the policy. This field associates manually configured SAs with policy entries.
The decimal number (index) you enter as the policy identifier must be separated from the tag statement by a colon, as in the following example:
<code>tag:number</code></td></tr>
</table></td></tr>
</table>

Details

Use *cgsetkey* to add, change, or delete IPv6 security keys and policies. The IPSec authentication header and the encapsulating security payload are supported.

The security policy database (SPD) consists of a list of policies that describe a set of packets to match and an action to be taken for those packets. If the action is ipsec, then the policy must contain links to one or more security associations (SAs) that contain keying material for a simplex packet flow between two hosts. These links are made using the tag argument for the spdadd and add commands.

The spdadd command is used to add entries to the database of policies that is scanned when packets are transmitted and received. Policies are scanned in the order in which they are added to the database. Therefore, more general policies follow more specific policies in the database. The first policy found matching a packet is used for that packet.

The security association database (SAD) contains the set of all active security associations. Each SA must be linked explicitly to a policy in the SPD when it is created. When a policy is deleted, the SA is deleted as well.

Command examples

The following examples show possible entries within *cgsetkey* configuration files for setting, retrieving, printing, or deleting IPSec keys or policies.

Example 1: Adding an entry to the SAD

The following operation adds an entry to the security association database:

```
add 3ffe:501:4819::1 3ffe:501:481d::1 esp 1234567
-E des-cbc "secret key" ;
```

where:

- Source address is 3ffe:501:4819::1
- Destination address is 3ffe:501:481d::1
- esp protocol is selected for the entry
- Security parameter index (SPI) for the entry is set to 123456
- des-cbc encryption algorithm is selected for the entry

Example 2: Adding an entry to the SAD

The following operation adds an entry to the security association database:

```
add 3ffe:501:4819::1 3ffe:501:481d::1 ah 123456
-A hmac-sha1 "AH SA configuration!" ;
```

where:

- Source address is 3ffe:501:4819::1
- Destination address is 3ffe:501:481d::1
- ah protocol is implemented for the entry
- SAD and SPD security parameter index for the entry is set to 123456
- hmac-sha1 authentication algorithm is enabled for the entry

Example 3: Adding an entry to the SAD

The following operation adds an entry to the security association database:

```
add 3ffe:501:4819::1 3ffe:501:481d::1 esp 0x10001
-E des-cbc "ESP with"
-A hmac-md5 "authentication!!" ;
```

where:

- Source address is 3ffe:501:4819::1
- Destination address is 3ffe:501:481d::1
- esp protocol is specified
- The security parameter index for the entry is 0x10001
- des-cbc is the selected encryption algorithm
- hmac-md5 is the selected authentication algorithm

When using the esp protocol, you can specify both an encryption algorithm and authentication algorithm for the SAD entry.

Example 4: Retrieving an entry from the SAD

The following operation retrieves an entry from the security association database:

```
get 3ffe:501:4819::1 3ffe:501:481d::1 ah 123456 ;
```

where the SAD database entry is the same one added in Example 2.

Example 5: Flushing all SAD entries

The following operation removes all entries from the security association database:

```
flush ;
```

Example 6: Dumping all SAD entries

The following operation displays all entries that use the ESP protocol from the security association database:

```
dump esp ;
```

Example 7: Adding an entry to the SPD

The following operation adds an entry to the security policy database:

```
spdadd 3ffe:501:4819::1/32[21] 3ffe:501:481d::1/32[any] any
-P out ipsec esp/transport/654321;
```

where:

- Source address is 3ffe:501:4819::1 on UDP port 21
- Destination address is 3ffe:501:481d::1 on any UDP port
- The entry can use any upper layer protocol
- The following IPSec policy is established for the entry:
    - The policy is implemented on outbound packets.
    - The policy uses the esp protocol.
    - The policy associates the entry in the SPD with the string 654321, creating an out-bound security association for the entry.

Example cgsetkey command file

The following sample *cgsetkey* command file shows how to use *cgsetkey* commands in a text file to set up SAD and SPD entries for a particular board:

```
######################################################################
# This file assumes that one of the CG board's IPv6 interfaces has the
# link-local address FE80::220:22FF:FE31:4C46.
##############################################
# Clear out the SPD
spdflush;
# Clear out the SADB
flush;

#######################################################################
# Policy section
# Policies are added in the order they will be searched.
# If more than one policy matches a packet, the first match will be
# used.
# Add a policy requiring IPSEC for all outbound UDP packets
spdadd 0::0/0 0::0/0 udp -P out ipsec
ah/transport//tag:1;
# Add a policy requiring IPSEC for all inbound UDP packets.
spdadd 0::0/0 0::0/0 udp -P in ipsec
ah/transport//tag:2;

######################################################################
# Key section
# All SAs must contain a tag parameter which specifies the policy
# entry the SA will be linked to.
# If unspecified, the tag will default to zero.
# Add an SA.  Since ...4C46 is a local address, this is an outbound SA.
# The destination is ...0C37 and the SPI is 1234.  This SA will be
# linked to the outbound policy (tag 1).
# The key is specifyed as an ascii string of 160 bits.
add FE80::220:22FF:FE31:4C46 FE80:0000:0000:0000:206:4cff:Fe25:0C37 ah 1234 -t 1
-A hmac-sha1 "abcdefghijklmnopqrst";
# Add another SA.  This is an inbound SA because the destination address is
# local.  This one will be linked to the inbound policy (tag 2).
# The key is specified as a 160 bit hexadecimal number.
add FE80::206:4cff:Fe25:0c37 FE80:0000:0000:0000:0220:22FF:FE31:4C46 ah 4321 -t 2
-A hmac-sha1 0x1234567812345678123456781234567812345678;
```

cgtrace - Performing CG board debugging

Enables debugging and tracing of CG boards.

Usage

```
cgtrace -bboardnum options
```

where ***boardnum*** is the CG board number (0 by default) and ***options*** are:

Option	Description
-g***globalmask***	Sends the specified global trace mask ***globalmask*** in hexadecimal format to the board.
-q	Displays the global trace masks and manager IDs of the board.
-r	Displays the current global or manager trace masks.
-m***managernum tracemask params***	Sends a new tracing configuration to the specified manager, where: • ***managernum*** is the manager ID in hexadecimal format. • ***tracemask*** is the manager trace mask in hexadecimal format. • ***params*** are the manager-specific tracing parameters. The parameters can be decimal or hexadecimal numbers or strings in quotes. Prefix hexadecimal numbers with 0x. Specify tracing parameters depending on the needs of the particular manager to be traced.

Description

cgtrace enables debugging output for various on-board software components (managers). The resulting debug output appears in the *oammon* display.

Procedure

Complete the following procedure to run *cgtrace*:

Step	Action

1	Enter the following command after a board has been booted:
	`cgtrace -bboardnumber options`
	If ***options*** are omitted, the following menu of commands appears:
	`L - List All Managers`
	`D Manager_ID - Display Trace Info`
	`S Manager_ID [Trace_Data] - Send New Tracing Configuration`
	`R Manager_ID / Global_ID - Request Current Tracing Configuration`
	`G Trace_Mask - Send New Global Trace Mask`
	`Q - Quit`

<table>
<tr><th>Step</th><th>Action</th></tr>
<tr><td>2</td><td>Enter one of the following commands followed by the required parameters (if any):
<table>
<tr><th>Command</th><th>Description</th></tr>
<tr><td>L</td><td>Lists the manager IDs of all the on-board managers that support tracing.
<pre>
ID: 0 - Global tracing
ID: 1 - Filter manager
ID: 2 - Executive
ID: 3 - Host interface
ID: 4 - Switching manager
ID: 5 - Legacy manager
ID: 7 - Image manager
ID: 8 - Framer manager
ID: 9 - Resource manager
ID: A - HDLC manager
ID: D - DSP manager
ID: 10 - CLK manager
ID: 13 - Diagnostic manager
</pre></td></tr>
<tr><td>D</td><td>Displays tracing information for the given manager ID in hexadecimal format or displays global tracing information (Manager ID = 0).</td></tr>
<tr><td>S</td><td>Sends a new tracing configuration to the given manager ID. This command requires a manager ID in hexadecimal format, a trace mask in hexadecimal format, and optionally integers in decimal or hexadecimal format or strings in quotes. Prefix hexadecimal numbers with 0x.

Use the D command to determine valid trace masks, as well as integer and string parameters for a given manager. Output resulting from this command appears in the oammon display.</td></tr>
<tr><td>R</td><td>Displays the current trace mask for the given manager ID or displays the current global trace masks (Manager ID = 0).</td></tr>
<tr><td>G</td><td>Sends the given global trace mask (hexadecimal number) to the board. Output resulting from this command appears in the oammon display.</td></tr>
<tr><td>Q</td><td>Quits the application.</td></tr>
</table></td></tr>
</table>

Details

Use the D command to determine valid trace masks, as well as valid integer and string parameters, for a given manager.

For example, selecting the filter manager (Manager ID = 1) displays the following trace options:

```
Trace Masks:
.. 00000001: Trace Commands
.. 00000002: Object Creates and Destroys
.. 00000004: Object Starts and Stops
.. 00000010: Extra Pin Connect and Disconnect Errors
```

Each of these lines describes a tracing option that can be enabled for the filter manager. Combine the options to get a 32-bit tracing mask. Use the S command to send the tracing mask to the manager. For example, to enable command tracing and object starts and stops for the filter manager, use the tracing mask 00000005.

The DSP manager is an example of a manager with an optional integer parameter. Enter the D command and select the DSP manager (Manager ID = D). The following trace options display:

```
Trace Masks:
.. 00000001: Trace Commands
.. 00000002: DSP - HPI Cmd Queue Sent
.. 00000004: DSP - HPI Cmd Queue Buffered
.. 00000010: DSP - HPI DSP Out Queue Reads
.. 00000020: DSP - HPI DSP Out Queue Parse: OS Acks
.. 00000040: DSP - HPI DSP out queue parse:OS other events
.. 00000080: DSP - HPI DSP Out Queue Parse: DPF Events
.. 00000100: DSP - HPI DSP Out Queue Parse: DPF Data Events
.. 00000200: DSP - HPI DSP Out Queue Parse: Data
.. 00000400: DSP - HPI DSP Out Queue Parse: Data Requests
.. 00001000: DSP - HPI DSP Data In Queue Sent
.. 00002000: DSP - HPI DSP Data In Queue Sent With Data
.. 00010000: DSP - DPF Starts and Stops
.. 00020000: DSP - DPF Events and Command Acks
.. 00040000: DSP - DPF Pauses and Resumes
.. 00080000: DSP - DPF Modifies
.. 00100000: DSP - High Speed Memory available
.. 00200000: DSP - Managed Memory checking during execution
.. 01000000: DSP - Extended Pin Information
.. 02000000: DSP - DPF Proxy Creates and Destroys
.. 10000000: DSP - # DSP Resource Groups and DSPs in each group
.. 20000000: DSP - Display and reset DSP packet statistics
.. 40000000: DSP - Dump the amount of resources available
.. 80000000: DSP - Dump DPF and Pin Data
Integer 0:
.. Min -1, Max 96, Default 0 (Optional): DSP to Trace
```

The DSP manager has a large number of different tracing configurations. Most of the configurations involve the DSPs and not the manager. Therefore, most trace commands to the DSP manager make use of the optional integer parameter. For example, a common trace flag for the DSPs is to trace all DPF (DSP function) starts and stops. To set this flag for DSP 5, use the S command. Enter the following information at the command line:

```
S D 00010000 5
```

where 5 is the optional integer parameter (in this example, the DSP to trace). The DSP manager also contains a special case: if -1 is specified as the DSP to trace, all DSPs obtain that trace mask.

cgtrace and resource management

To display a list of resource management trace values available with a brief description of what each value traces, enter 9.

cgtrace displays the following list:

```
Tracing data for Resource Manager...

Trace Masks:

.. 00000001: RM CMD Enable Trace Commands
.. 00000002: RM EVT Enable Trace Events
.. 00000010: Resource Objects creation and destruction
.. 00200000: List Pool names indexed by timeslots in global table
.. 00400000: Display all DPFs in resource definition (Pool name required)
.. 00000040: Allocate and Destroy of Resource Objects while running
.. 00100000: Print Prestart list
.. 01000000: Prints Host ResDef (Pool name required)
.. 02000000: Prints all TCP ResDefs (TCP name required)
.. 04000000: Resource Calculation of Definition in use( Pool name required)
.. 10000000: Number of pools and their names
.. 20000000: Single pool: number of objects, number objects in use (Pool name required)
.. 40000000: Single pool details: List of resource objects and their engine (pool name required)
String 0:
.. MaxLen 11, Default "" Optional): Res Label or TCP label

Ex: S Manager_ID Trace_Mask ["String"]
```

To send a trace configuration to the on-board resource manager, enter the S command.

When the menu indicates the pool name required for a particular trace mask, you must enter, in quotes, the name (set with the Resource[x].Name keyword) associated with the resources you want to trace.

To set the global trace mask on the board after the board has been booted, enter the G command.

cgv6if - Adding, printing, and deleting IPv6 addresses

Adds, prints, and deletes IPv6 addresses for a CG board.

Usage

`cgv6if` ***`command v6address/prefixlength`*** `-i` ***`interface`*** `-b` ***`boardnumber`***

Valid commands include:

Command	Description
print	Prints all IPv6 addresses.
add	Adds a new IPv6 address.
delete	Deletes a manually created IPv6 address.
deleteall	Deletes all manually created IPv6 addresses.
router	Adds a static router.
showrouter	Shows static routers.
delrouter	Deletes a static router.

Valid arguments include:

Argument	Description
v6address	IPv6 address for this interface.
prefixlength	IPv6 prefix length for the corresponding address.
-i ***interface***	Network interface number (1 or 2).
-b ***boardnumber***	Number of the CG board. The default value is 0.

Description

Use *cgv6if* to add, print, and delete IPv6 addresses without editing individual board keyword files. *cgv6if* is similar to the standard *ifconfig* utility found on most systems with IP processing capabilities.

Example

```
cgv6if print
cgv6if delete fe80::1245:5678:9abc:def0/64 -i 1
cgv6if deleteall -i 1
cgv6if add fe80::1245:5678:9abc:def0/64 -i 1 -b 3
cgv6if router 2001:DB8::1234:5678:9abc:def0 -i 1
cgv6if delrouter 2001:DB8::1234:5678:9abc:def0 -i 1
```

16. Migrating from the AG 4040 board

Hardware differences: AG 4040 to CG 6060

The following table describes the hardware differences between AG 4040 and CG 6060 boards:

Item	AG 4040 board	CG 6060 board
Board status LEDs	No board status LEDs.	Red, yellow, and green board status LEDs.
Bus support	Supports 32-bit, 33 MHz PCI bus.	Supports 32-bit, 33 MHz or 66 MHz PCI bus.
CAS signaling	Processed in a DSP.	Processed in the framer.
Connections	Limited to 256 simple connections on the H.100 bus.	Can simultaneously connect (simplex) to all 4096 timeslots on the H.100 bus.
CT bus timeslot connections	A connection to or from a CT bus timeslot will fail if the timeslot is already connected in the other direction.	If a connection is made to or from a CT bus timeslot, any existing connection in the other direction on that timeslot is disconnected.
DSP connections	Each DSP core has its own 64-slot TSI stream.	Four DSP cores per TSI stream. Each local TSI stream has 128 timeslots.
DSPs	• C549 DSP. Each chip has two cores. • 100 MIPS per DSP core. • Maximum 40 cores, 4000 MIPS.	• C5441 DSP. Each chip has four cores. • 133 MIPS per DSP core. • Maximum 32 cores, 4256 MIPS.
E1 impedance	E1 impedance default is 120 ohms but switch selectable to 75 ohm.	E1 impedance is software configurable to 75 ohm, 100 ohm, 120 ohm, or high impedance.
Echo cancellation	Software echo cancellation.	Proprietary echo cancellation chip. Software echo cancellation is also available.

Item	AG 4040 board	CG 6060 board
Ethernet	No Ethernet connectivity.	Dual 10/100Base-T Ethernet with LED speed and activity indicators on the sockets.
Signaling	Signaling is switchable from trunk-to-trunk and between trunks and the CT bus.	Does not support switch signaling from trunk-to-trunk or between trunks and the bus, only between trunks and DSPs.
Stream speed	The default stream speed for CT bus streams 0 through 15 is 2 Mbit/s.	All CT bus streams operate at 8 Mbit/s only.
Timeslot Interchangers (TSI)	Dual Agere T8100A.	Proprietary.
Trunk interface jacks	Four RJ-48C trunk interface jacks.	Two Dialogic® MD1 RJ-45 trunk interface jacks. A splitter cable is required to access the second trunk on each jack.
Trunk LEDs	Red, yellow, and green LED per trunk.	Red, yellow, or OFF LED per trunk.

Software differences: AG 4040 to CG 6060

The software differences between the AG 4040 and CG 6060 boards include:

- ADI board type
- Switch model
- Stream speed
- Echo API

ADI board type

AG 4040 board	CG 6060 board
ADI boardtype = ADI_BOARDTYPE_AG4000_***XX*** where ***XX*** represents 4T, 4E, 2T, 2E, 1T, or 1E **Note:** At the ADI and NCC levels, the AG 4040 board is identified as an AG 4000 board.	ADI boardtype = ADI_BOARDTYPE_CG6060

Switch model

Item	AG 4040 board	CG 6060 board
Conferencing streams	32 and 33	68 and 69
DSP signaling streams	18 and 19	66 and 67 and cannot be connected to the CT bus.
DSP voice streams	16 and 17	64 and 65
Echo canceling streams	34 and 35	70 and 71
HDLC pseudo streams (for AG 4000 compatibility)	20-27	No HDLC pseudo streams.
HDLC switching connections	To support existing AG 4000 applications on an AG 4040 board, legacy HDLC switching connections are accepted without error, but they have no functional impact on the board.	Not supported.

Note: The CG boards use the same switch device name as AG boards, agsw.

Stream speed

AG 4040 board	CG 6060 board
The first 16 H.100 streams are initially configured as 2 Mbit/s. The speed can be changed after the board is booted using the **swiConfigStreamSpeed** function or the *swish* utility swi.ConfigStreamSpeed command.	Does not support the **swiConfigStreamSpeed** function. All 32 H.100 streams are automatically configured for maximum capacity (8 Mbit/s or 128 timeslots).

Echo API

AG 4040 board	CG 6060 board
Supports **adiCommandEchoCanceller** and **adiModifyEchoCanceller.**	Supports **adiCommandEchoCanceller** and **adiModifyEchoCanceller** only if DSP-based echo canceling is used. If hardware echo canceling is used, individual channels can be enabled and disabled using **swiConfigLocalTimeslot**.

Configuration differences: AG 4040 to CG 6060

The configuration differences between the AG 4040 and CG 6060 boards include:

- Product name in system configuration file
- DSP configuration and resource allocation
- Configuration keywords
- Data input and output queue constraints
- File locations
- Runtime files

Product name in system configuration file

AG 4040 board	CG 6060 board
Product name in the system configuration file: AG_4040 AG_4040_1TE AG_4040_2TE AG_4040_4TE	Product name in the system configuration file: CG_6060 = Generic name for CG 6060 board CG_6060_4 = One, two, or four trunks

DSP configuration and resource allocation

Item	AG 4040 board	CG 6060 board
Available functions	When a DSP file is configured on a DSP, all of its functions are available to applications.	Only specific functions in a DSP file are available to applications. Functions are spelled out in Resource[***x***].Definitions.
DSP files	If DSP.C5x[***x***].Files[***y***] is not used, DSP files are assigned to DSPs according to the default rules specified by DSP.C5***x***.DSPFiles. Defaults are: DSP.C5***x***.DSPFiles = mf callp tone dtmf ptf	If DSP.C5x[***x***].Files[***y***] is not used then DSP files are assigned to DSPs according to Resource[***x***].Definitions and Resource[***x***].DSPs.
Fax	Fax is in a DSP image file that requires dedicated DSPs.	Fax shares a DSP with other functions.

Item	AG 4040 board	CG 6060 board
Number of ports	The number of ports supported depends on the DSP load.	DSP resources are preallocated for a specific number of ports. All DSP resources except conferencing must be managed by the CG resource management scheme.

Configuration keywords

The following table lists the AG keywords that may need to be changed when porting a configuration from an AG4040 board to a CG6060 board:

AG Keyword	AG	CG
BootDiagnosticLevel	Valid values are 0-3.	Valid values are 0 and 1.
Buffers[***x***].Num		Ignored. Buffer pools grow dynamically.
DebugMask	Equivalent to setting a trace mask with agtrace <***mask***>.	Equivalent to setting a trace mask with cgTrace -g <***mask***>.

AG Keyword	AG	CG
DLMFiles[***x***]	Required. The file names have optional *.leo* extensions: Trunk protocol engine: *gtp[.leo]* DSP function manager: *svc[.leo]* Play and record manager: *voice[.leo]* Fax manager: *ag2fax[.leo]* ISDN: • *isdn4ess[.leo]* • isdn5ess[.leo] • isdndms[.leo] • isdngen[.leo] • isdnni2[.leo] • isdnqsig[.leo] ISDN management: *imgt[.leo]* DPNSS: *dpnss[.leo]*	Optional. The file names have optional *.dlm* extensions: Fax manager: *cg6060fax[.dlm]* ISDN: *c6060igen[.dlm]* ISDN management: *c6060imgt[.dlm]* DPNSS: *c6060dpnss[.dlm]* Fusion: *cg6060fusion[.dlm]* ThroughPacket: *cg6060tpkt[.dlm]* **Note:** Trunk protocol engine, DSP function manager and play and record manager are included in runtime.
DSP.C5x.Lib	Default: ag2liba.r54 if XLaw = A-LAW ag2libu.r54 if XLaw = MU-LAW	Replaced by DSP.C5x[***x***].Libs[***y***]. Default is cg6kliba, set to cg6klibu for mu-law. This keyword must be specified when using mu-law, conferencing, Fusion or HDLC.
DSP.C5x.Loader	AG only (rarely used).	
DSP.C5x.DSPFiles	AG only.	Use Resource[***x***].Definitions to specify DSP modules.

AG Keyword	AG	CG
DSP.C5x[***x***].Files[***y***]	If this keyword is absent, default rules are used to assign DSP files to DSPs using the files specified by DSP.C5x.DSPFiles.	If this keyword is absent, DSP files are assigned according to Resource[***x***].Definitions and Resource[***x***].DSPs. Required for conferencing.
DSP.C5x[***x***].Image	Required for fax and conferencing.	Not normally used. Fax DSP is specified with Resource[***x***].Definitions. Conference DSP is specified with DSP.C5x[***x***].Files[***y***].
DSP.C5x[***x***].Limits[***y***]	AG only.	DSP load balancing is automatic due to CG resource management.
DSP.C5x[x].Os	Not normally set on AG.	The sample configuration files set this to the default, dspos6u.
Echo.AutoSwitchingRefSource		Can only be used with DSP echo canceling.
Echo.EnableExternalPins	Echo canceler streams are 34 and 35.	Can only be used with DSP echo canceling. Echo canceler streams are 70 and 71.
Eeprom.xxx*	AG only (read only).	
LoadFile	AG only (rarely used).	
LoadSize	AG only (rarely used).	
MaxChannels	Default is 124.	Default is 900.
NetworkInterface.T1E1[***x***].FrameType	Default is ESF for T1, CEPT for E1.	Default is CEPT.

AG Keyword	**AG**	**CG**
NetworkInterface.T1E1[***x***].Impedance	Default is G703_120_OHM for E1, DSX1 for T1.	Default is G703_120_OHM.
NetworkInterface.T1E1[***x***].LineCode	Default is B8ZS for T1, HDB3 for E1.	Default is HDB3.
RunFile	Specifies a *.cor* file; the default is *ag4040.cor* (rarely used).	The CG equivalent is ULMFile; its default is *cg6060core.ulm.*
SignalIdleCode	If XLaw = MU-LAW, default = 0. If XLaw = A-LAW, default = 0x09.	Replaced by DSPStream.SignalIdleCode[***x***], where x is a trunk number. Default is 0x0D. All trunks must be configured with the same setting.
TCPFiles[x]		Deprecated. Use Resource[x].TCPs instead. **Note:** AG and CG use the same TCP files.
VoiceIdleCode	If XLaw = MU-LAW, default = 0x7F. If XLaw = A-LAW, default = 0xD5.	Replaced by DSPStream.VoiceIdleCode[***x***], where ***x*** is a trunk number. The default is 0xD5. For T1, set to 0x7F.
XLaw		Replaced by DSP.C5x[***x***].XLaw. The value must match the DSP operating system service library specified with DSP.C5x[***x***].Libs.

Data input and output queue constraints

AG 4040 board	CG 6060 board
Size is fixed at 703 words of data output queue memory and 703 words of data input queue memory per DSP.	Sizes are configurable. The default size of the input and output data queues is 768 (0x300) words per DSP. The Command, Data In, and Data Out queues are placed in DSP memory. By default they are at addresses 0x2000, 0x2280 and 0x2580. To make the data queues larger, change DataInQSize, DspOutQStart and DspOutQSize, keeping the queues contiguous. For example, to double the size of the data queues to 0x600, make the following changes: DSP.C5x[0..47].CmdQStart = 0x2000 #(default) DSP.C5x[0..47].CmdQSize = 0x100 DSP.C5x[0..47].DataInQStart = 0x2100 DSP.C5x[0..47].DataInQSize = 0x600 DSP.C5x[0..47].DspOutQStart = 0x2700 DSP.C5x[0..47].DspOutQSize = 0x600 **Note:** Increasing the queue sizes reduces the memory available to DSP functions.

File locations

Operating system	Files	AG 4040 boards	CG 6060 boards
Windows	Sample configuration files	*\nms\ag\cfg*	*\nms\cg\cfg*
	Runtime files	*\nms\ag\load*	*\nms\cg\load* (except TCP files)
	TCP (trunk control protocol) files	*\nms\ag\load*	*\nms\ag\load*
UNIX	Sample configuration files	*/opt/nms/ag/cfg/*	*/opt/nms/cg/cfg/*
	Runtime files	*/opt/nms/ag/load/*	*/opt/nms/cg/load/* (except TCP files)
	TCP (trunk control protocol) files	*/opt/nms/ag/load/*	*/opt/nms/ag/load/*

Runtime files

File type	AG 4040 board	CG 6060 board
Co-processor files	The default boot diagnostics and loader file is *ag4040.lod*. The default run file is *ag4040.cor*.	Diagnostic and loader software is installed in flash memory at the factory. The default run file is *cg6060core.ulm*. The OAM API downloads the run file directly into SDRAM.
DSP function files	AG boards use C549 DSPs and the DSP function files have *.m54* extensions. For example, *dtmf.m54*. Use *m54info* to display a file's resource usage.	CG boards use C5441 DSPs and the DSP function files have *.f41* extensions. For example, *dtmf.f41*. Use *f41info* to display a file's resource usage.
DSP image files	*.c54* extensions. Fax and conferencing are provided as DSP image files.	DSP image files are not normally used on CG 6060 boards. Fax and conferencing use *.f41* files instead.
DSP libraries	*.r54* extensions.	*.r41* extensions.
DSP OS files	*.k54* extensions	*.k41* extensions
Runtime extensions	In downloadable modules with *.leo* extensions.	In downloadable modules with *.dlm* extensions. Each *.dlm* file has a matching *.sym* file.
TCP files	*.tcp* extensions.	CG boards use the AG TCP files.

Other differences: AG 4040 to CG 6060

Other software differences between the AG 4040 and the CG 6060 boards include:

Software component	AG 4040 board	CG 6060 board
Board location utility (UNIX)	Use *blocate* to associate the PCI bus assignment to a physical board by flashing an LED on the board. To flash the LED on a board, call *blocate* with the PCI bus and PCI slot locations.	Use *cg6ktool* to associate the PCI bus assignment to the physical board by flashing an LED on the board. Usage: `cg6ktool -B -lbus:slot`

Software component	**AG 4040 board**	**CG 6060 board**
Board location utility (Windows)	Use *pciscan* to associate the PCI bus assignment to the physical board by flashing an LED on the board. To flash the LED on a board, call *pciscan* with the PCI bus and PCI slot locations.	Same as the AG board.
Boot log	OAM appends booting information and tracing to *agpierror.log*.	OAM logs boot-specific information to *CG_6060boot_<**bus**>_<**slot**>.txt*. The file is overwritten each time the board is booted. Use *oammon* to capture operational error and tracing messages.
OAM plug-in	The OAM plug-in for AG boards is *agplugin.bpi*. Keyword default values are built-in.	The OAM plug-in for CG boards is *cg6kpi.bpi*. Keyword default values are in *cg6k.dft*.

17. Index