Dialogic® AG 2000-BRI Media Board Installation and Developer's Manual

October 2009 64-0490-01

www.dialogic.com

Copyright and legal notices

Revision history

Revision	Release date	Notes
9000-60098-10	February 2001	NBS
9000-60098-11	May 2001	NBS
9000-60098-12	August 2001	NBS
9000-60098-13	November 2001	NBS, Natural Access 2002-1 Beta
9000-60098-14	May 2002	NBS, Natural Access 2002-1
9000-60098-15	November 2002	MVH, Natural Access 2003-1 Beta
9000-60098-16	April 2003	SRG, Natural Access 2003-1
9000-60098-17	April 2004	SRG, Natural Access 2004-1
64-0490-01	October 2009	LBG, NaturalAccess R9.0
Last modified: September 12, 2009		

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® AG 2000-BRI Media Board Installation and Developer's Manual* explains how to configure and install an AG 2000-BRI board, and how to verify that it has been installed correctly and is operating correctly. It also provides general information about developing an application that uses an AG 2000-BRI board.

This manual targets developers of telephony and voice applications who are using the AG 2000-BRI board with NaturalAccess. This manual defines terms where applicable, but assumes that readers are familiar with telephony concepts, switching, and the C programming language.

2 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
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

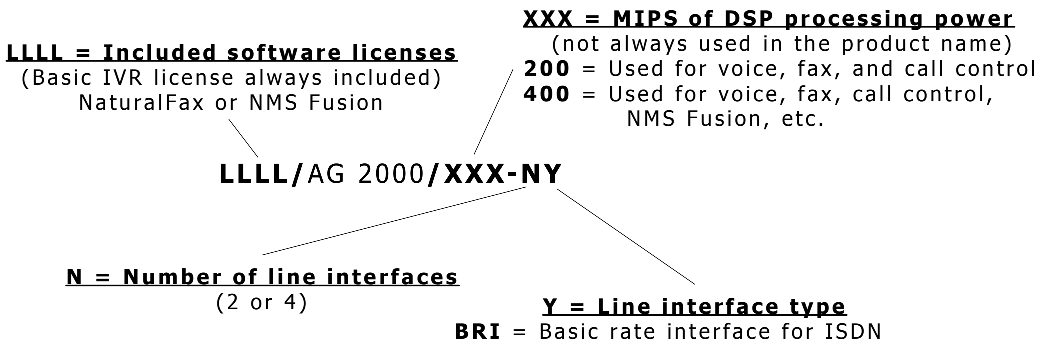
Former terminology	**Dialogic terminology**
Point-to-Point Switching Service	Dialogic® NaturalAccess™ Point-to-Point Switching API
Switching Service	Dialogic® NaturalAccess™ Switching Interface API
Voice Message Service	Dialogic® NaturalAccess™ Voice Control Element API
NMS CAS for Natural Call Control	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
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

3 Overview of the AG 2000-BRI board

AG 2000-BRI board features

The AG 2000-BRI board is part of the Alliance Generation family of telephony boards. It provides up to four digital line interfaces with up to eight ports of call processing and programmable voice processing.

The following illustration shows the available AG 2000-BRI board types:


LLLL = Included software licenses
(Basic IVR license always included)
NaturalFax or NMS Fusion
XXX = MIPS of DSP processing power
(not always used in the product name)
200 = Used for voice, fax, and call control
400 = Used for voice, fax, call control, NMS Fusion, etc.
LLLL/AG 2000/XXX-NY
N = Number of line interfaces
(2 or 4)
Y = Line interface type
BRI = Basic rate interface for ISDN

AG 2000-BRI board types

Refer to the NMS web site (www.nmscommunications.com) for a list of available AG 2000-BRI board configurations, for a list of countries where NMS has obtained approval for the AG 2000-BRI board, and for product updates.

An AG 2000-BRI board contains the following main features:

- DSP resources

  Each board has two or four high-performance digital signal processors (DSPs) that provide resources for eight ports of call processing and programmable voice processing. Each DSP supports one or more tasks. These tasks include voice recording and playback, DTMF detection and generation, and call progress analysis. Fax is supported on an AG 2000-BRI board with two DSPs.

- PCI bus connectivity

  Each AG 2000-BRI board is designed to reside in a single PCI bus slot. Each board contains a 5 volt PCI bus interface compliant with the PCI specification, version 2.1. The PCI interface is a 33 MHz, 32-bit target device.

- Trunk connectivity

  Each board contains BRI network interfaces for digital ISDN trunk connectivity with the following characteristics:

  - S0/T0 interfaces support
  - TE and NT modes
  - Point-to-point and point-to-multipoint wiring support:
    - Short and extended passive bus (in-building wiring)
    - Up to eight terminals

- H.100 bus connectivity

  The AG 2000-BRI board fully supports the H.100 bus specification. The H.100 bus allows boards to share data and signaling information with other boards on the H.100 bus. For example, you can connect two or more AG 2000-BRI boards for applications that perform trunk-to-trunk switching. You can add additional DSP resources, analog station interfaces, or loop start line interfaces using other AG boards. You can also use H.100 compatible products from other manufacturers with the AG 2000-BRI board.

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

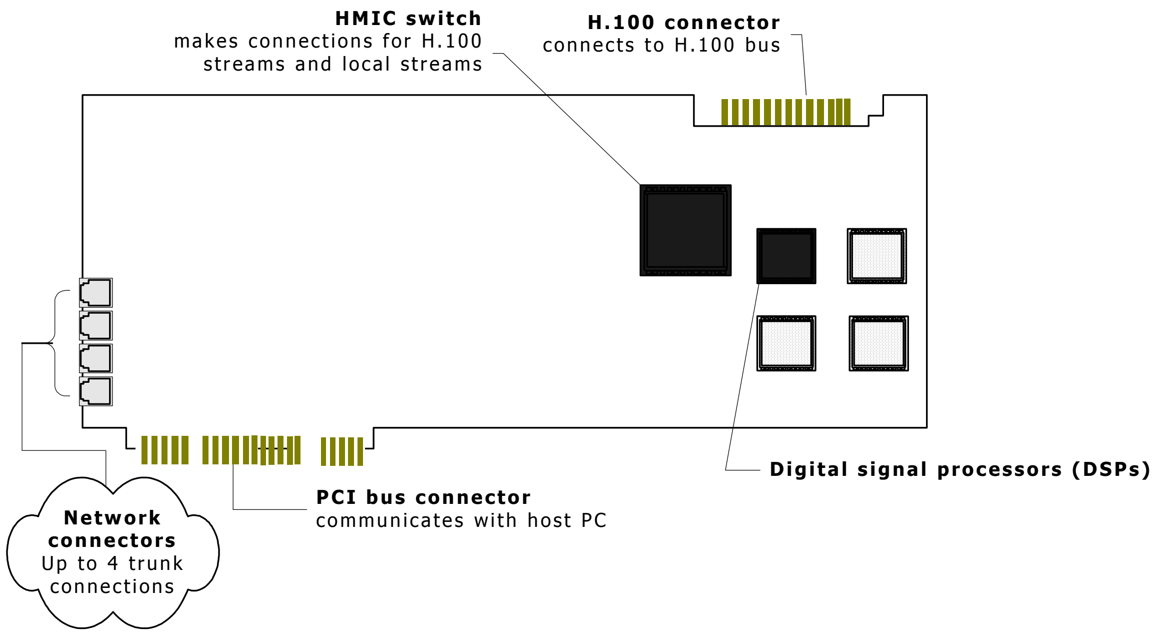
  - Full mode: 32 streams at 8 MHz each that provide 128 timeslots each for a total of 4096 timeslots.
  - Backward compatibility mode: 16 8MHz streams, 16 2MHz streams (total of 2560 timeslots). The H.100 interface operates with MVIP-90 boards on the same bus. In these configurations, use an H.100 board as the system bus master.

- Telephony bus switching

  Switching for the AG 2000-BRI board is implemented with the HMIC (H.100/MVIP integrated circuit). The HMIC is a single chip that offers full support for the H.100 bus within the MVIP architecture providing access to all 4096 slots.

  On the AG 2000-BRI board, switch connections are allowed for up to 128 full duplex connections between local devices and the H.100 bus. Non-blocking switch connections are allowed between local devices. On the AG 2000-BRI board, switching is controlled by the MVIP-95 model; the MVIP-90 model is not supported.

- The following illustration shows where various components are located on an AG 2000-BRI board:


HMIC switch
makes connections for H.100 streams and local streams
H.100 connector
connects to H.100 bus
Digital signal processors (DSPs)
PCI bus connector
communicates with host PC
Network connectors
Up to 4 trunk connections

AG 2000-BRI board

ISDN

Integrated services digital network (ISDN) is a continually evolving international standard for networking a wide range of services, including voice and non-voice services. The network is completely digital, from one end to the other: voice information is digitized and sent in digital form. Signaling information is sent separately from voice information, using a method called common channel signaling (CCS).

This topic presents the following information:

- ISDN protocols and protocol layering
- ISDN carriers
- Specificity of BRI boards

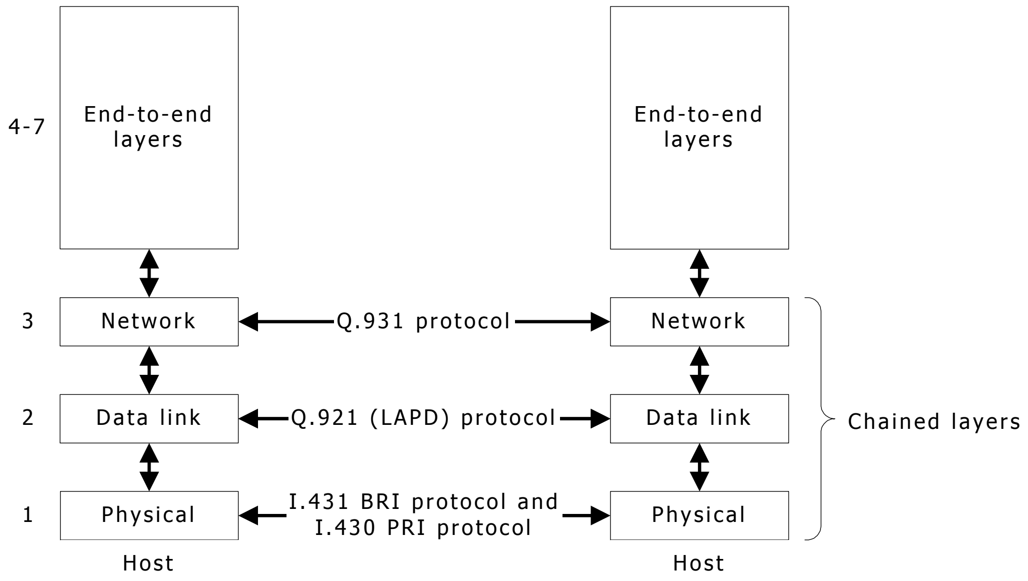
ISDN protocols and protocol layering

ISDN communications can be described at many levels, from the way bits are transferred from machine to machine to the sets of messages computers pass to one another. A scheme for communication at a certain level is called a protocol.

In the late 1970's, the International Standards Organization (ISO) established the open systems interconnect (OSI) model for communication. ISDN is based on this model. In OSI, seven separate levels, or layers, of communication are defined (refer to the following illustration). The first three layers, called the chained layers, are the lowest levels. The chained layers are:

Layer	Description
1 - Physical	The electrical and mechanical layer. Protocols for this layer describe, on an electrical and mechanical basis, the methods used to transfer bits from one device to another. One protocol used at this layer is CCITT recommendation I.430/I.431.
2 - Data link	The layer above the physical layer. Protocols for this layer describe methods for error-free communication between devices across the physical link. One protocol used at this layer is CCITT recommendation Q.921, also known as Link Access Procedures on the D Channel (LAPD).
3 - Network	The layer above the data link layer. Protocols for this layer describe methods for transferring information between computers. They also describe how data is routed within and between networks. One protocol used at this layer is CCITT recommendation Q.931.

Layers higher than these are end-to-end layers. They describe how information is exchanged and delivered end-to-end. They also define process-to-process communication, and describe application-independent user services, user interfaces and applications.



OSI protocol layering model

The functionality provided by a layer includes the services and functions of all of the layers below it. A service access point (SAP) is the point at which a layer provides services to the layer directly above it. A unique service access point identifier (SAPI) is associated with each SAP.

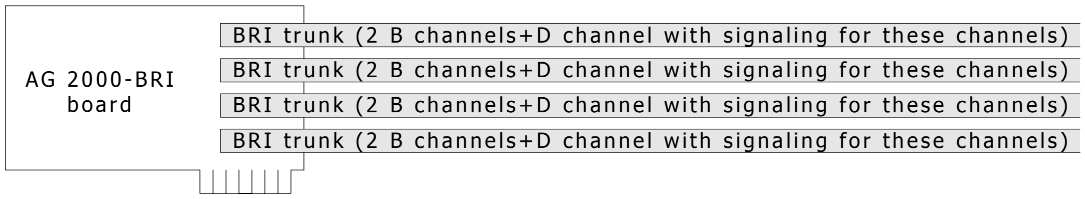
The layer 3 protocol is not a symmetric protocol. There is a network side and a terminal side. Typically, applications that act as a network node must use ISDN network side stacks. Applications that act as a terminal (for example, IVR, or any application which is connected on the PSTN) must use ISDN terminal side stacks.

ISDN carriers

ISDN is transmitted over T1 carriers, E1 carriers, and BRI (basic rate interface) carriers. These are typically four-wire digital transmission links.

With basic-rate ISDN, the channels are usually used as follows:

- Two channels carry data digitized at 64 Kbps: voice, audio, data and/or video signals. These channels are called bearer channels (B channels).
- One channel carries signaling information for the two B channels. This is called the D channel.

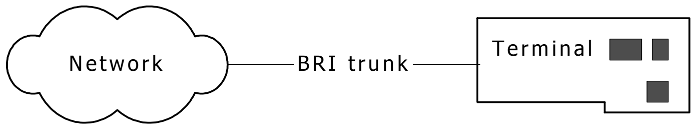

AG 2000-BRI board
BRI trunk (2 B channels+D channel with signaling for these channels)
BRI trunk (2 B channels+D channel with signaling for these channels)
BRI trunk (2 B channels+D channel with signaling for these channels)
BRI trunk (2 B channels+D channel with signaling for these channels)

AG 2000-BRI configuration

Specificity of BRI boards

The AG 2000-BRI board is designed to work with two types of CCITT I.430 BRI configurations, generally known as S-bus.

The following illustration shows the first type of S-bus configuration which is called point-to-point configuration. The AG 2000-BRI board can be connected either at the terminal or at the network side.

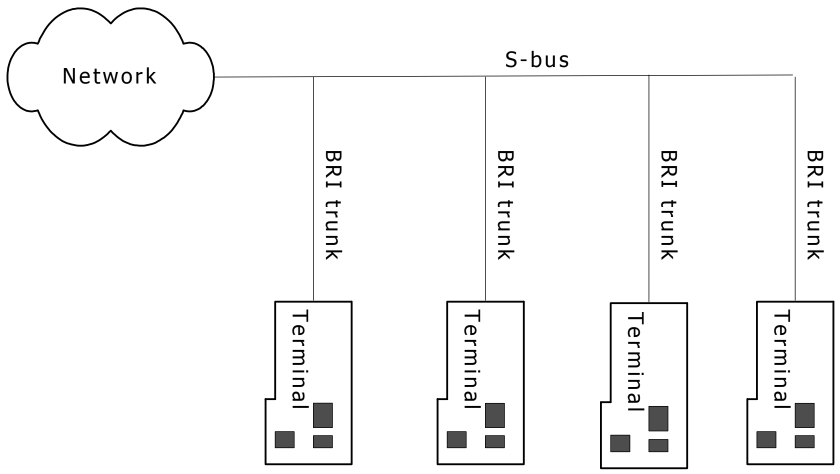


Point-to-point configuration

The following illustration shows the second type of S-bus configuration which is called point-to-multipoint configuration. One of the four BRI trunks of the AG 2000-BRI board shares the S-bus with other terminal equipment devices (such as other AG 2000-BRI board trunks or ISDN phones, fax machines, PC for data transfer). The S-bus supports up to eight terminal equipment devices.

When several terminals share the S-bus, they share the two B channels available on the S-bus. Only two calls can be handled simultaneously. When calls arrive from the network, the terminals are notified but only one establishes the incoming call. In point-to-multipoint configurations, bus contention issues must be addressed.

Note: In this example, each BRI trunk of an AG 2000-BRI board is terminal equipment. It provides a standardized connector called the S-interface as a connection to the network. Since the S-bus is standard, all terminal equipment providing an S-interface (a standardized connector to connect to the network), can be connected to a PBX or network.



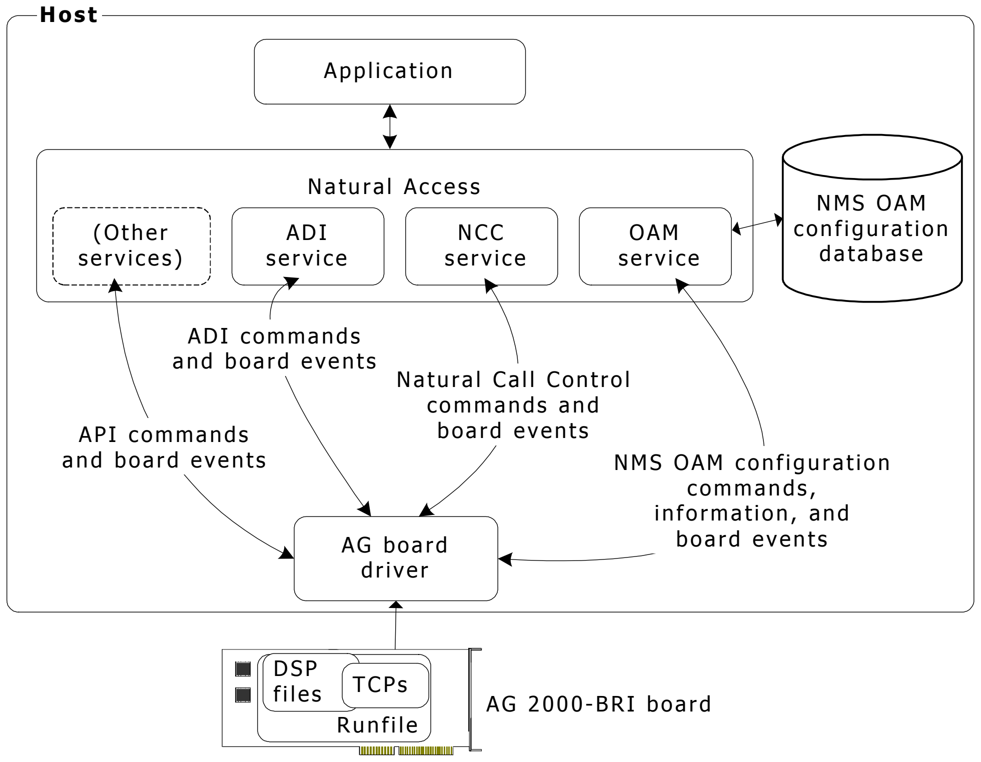
Point-to-multipoint configuration

Software components

AG 2000-BRI boards require the following software components:

- Natural Access development environment that provides services for call control, voice store and forward, switching, and other functions.
- NMS OAM (Operations, Administration, and Maintenance) software and related utilities.
- Configuration files that describe how the board is set up and initialized.
- NMS ISDN software components that provide the software and tools for ISDN interfaces. NMS ISDN also provides configuration files based on country-specific settings.
- Runtime software that controls the AG 2000-BRI board.
- One or more trunk control programs (TCPs) that enable applications to communicate with the telephone network using the signaling schemes (protocols) used on the trunk.

The following illustration shows how these software components relate to one another:


Host
Application
Natural Access
(Other services)
ADI service
NCC service
OAM service
NMS OAM configuration database
ADI commands and board events
Natural Call Control commands and board events
API commands and board events
NMS OAM configuration commands, information, and board events
AG board driver
DSP files
TCPs
Runfile
AG 2000-BRI board

Software components

Natural Access

Natural Access 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 Natural Access services, refer to the *Natural Access Developer's Reference Manual*.

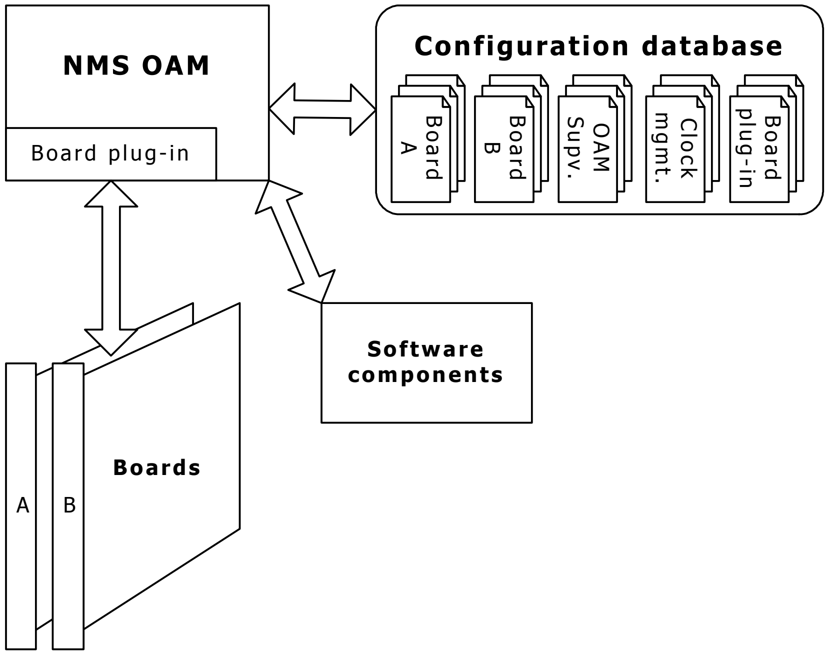
NMS OAM

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

Using NMS OAM, you can:

- Create, delete, and query the configuration of a component
- Start, stop, and test a component
- Receive notifications from components

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



NMS OAM components

Each parameter and value is expressed as a keyword name and value pair (for example, AutoStart = NO). You can query the NMS OAM database for keyword values for any component. Keywords and values can be added, modified, or deleted.

To use NMS OAM or any related utility, ensure that the Natural Access Server (*ctdaemon*) is running. For more information about *ctdaemon*, refer to the *Natural Access Developer's Reference Manual*. For more information about NMS OAM, refer to the *NMS OAM System User's Manual*.

AG board plug-in

NMS OAM uses the AG board plug-in software module to communicate with AG boards. The name of the AG plug-in is *agplugin.bpi*. This file must reside in the *\nms\bin* directory (or */opt/nms/lib* directory for UNIX) for NMS OAM to load it when it starts up.

NMS ISDN

NMS ISDN is a set of APIs and tools used with ISDN protocols. NMS ISDN protocol software enables you to write Natural Access applications that communicate with T1, E1, or BRI trunks to perform voice processing functions and call control using ISDN common channel signaling (CCS) protocols.

NMS ISDN is designed to be used with one or more AG or CG boards to provide the physical interface to trunk lines. In addition to trunk interfaces, most of these boards also feature on-board digital signal processing (DSP) resources that can perform call control and voice processing functions.

NMS ISDN software can be configured to access ISDN services in any of the following ways:

- In the channelized configuration, the application performs call control and other operations using the standard Natural Call Control (NCC) API. For more information about this configuration, refer to the *NMS ISDN for Natural Call Control Developer's Manual*.
- In the ACU configuration, the application accesses ISDN services at the ACU SAP using the NMS ISDN Messaging API. This enables the application to perform a wide range of Q.931 ISDN D channel functions. For more information about this configuration, refer to the *NMS ISDN Messaging API Developer's Reference Manual*.
- In the LAPD configuration, the application accesses ISDN services at the data link layer (Layer 2) using the NMS ISDN Messaging API. This setup enables the application to send and receive I-frame data in LAPD messages. This data typically consists of Q.931 messages. For more information about this configuration, rerfer to the *NMS ISDN Messaging API Developer's Reference Manual*.

You are required to choose the method for accessing ISDN services when you initialize the NMS ISDN procotol stack, as described in the *NMS ISDN Messaging API Developer's Reference Manual* and the *NMS ISDN for Natural Call Control Developer's Manual*.

Configuration files

NMS OAM uses two types of configuration files:

File type	Description
System configuration	Contains a list of boards in the system and the name 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. Each of these files configures the board to use a different protocol. You can reference these files in your system configuration file or modify them.

When you run the NMS OAM *oamsys* utility, it creates NMS OAM database records based on the contents of the specified system configuration file and board keyword files. *oamsys* directs NMS OAM to start the boards and configure them according to the specified parameters. For more information, refer to *Configuring and starting the system with oamsys* on page 32.

Runtime software

The runtime software consists of runfiles and DSP files. The runfile is the basic low-level software that an AG board requires to operate. DSP files enable the AG on-board digital signal processors to perform certain tasks, such as DTMF signaling, voice recording, and playback.

Several runfiles and DSP program files are installed with Natural Access. Specify the files to use for your configuration in the board keyword file. Refer to *Using board keyword files* on page 33 for more information. When NMS OAM boots a board, the runfiles and DSP program files are transferred from the host into on-board memory.

For more information about the DSP files shipped with Natural Access, refer to the *ADI Service Developer's Reference Manual*.

Trunk control programs (TCPs)

AG 2000-BRI boards are compatible with the ISDN protocol. To program an AG board for a specific protocol, a trunk control program (TCP) is loaded onto the board. The TCP performs all of the signaling tasks to interface with the protocol used on the line.

Several different protocol standards are used throughout the world. These standards differ considerably from country to country. For these reasons, different TCPs are supplied with Natural Access for various protocols and country-specific variations.

You can load more than one TCP at a time for applications that support multiple protocols simultaneously. TCPs are specified in the configuration file and are downloaded to the board by *oamsys*. TCPs run on the board, relieving the host computer from the task of processing the protocol directly. For more information about TCPs, refer to the *NMS CAS for Natural Call Control Developer's Manual*.

4 Installing the hardware

Installation summary

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

Step	Description
1	Ensure that your PC system meets the *system requirements* on page 24.
2	Configure the AG 2000-BRI board.
3	Install the AG 2000-BRI board into one of the computer's PCI bus slots.
4	If there are multiple H.100 boards, connect the H.100 bus to your H.100 boards.
5	Install Natural Access, which also installs the AG 2000-BRI board driver and runtime software.
6	Add configuration information for each board to the NMS OAM database. For more information, refer to the *NMS OAM System User's Manual*.
7	Direct the OAM service to start the boards. For more information, refer to *Configuring and starting the system with oamsys* on page 32 and to the *NMS OAM System User's Manual*.
8	Verify that the installation is operational.

Note: If your system is powered down, you can install the board before you install the software. It does not matter if you install the board or the software first.

AG driver software

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

Operating system	Driver names
Windows	*aghw* *ag* *agsw* *ag95sw*
UNIX	*aghw* *agsw* *ag95sw* *agmx*

System requirements

To install and use AG 2000-BRI boards, your system must have:

- Natural Access installed.
- An available PCI bus slot.
- An H.100 bus connector cable if you are connecting to any other H.100 boards.
- Cables to connect to an S0 or T0 trunk.
- A grounded chassis (with three-prong power cord).

NMS recommends an uninterruptable power supply (UPS) for increased system reliability.

Configuring the hardware

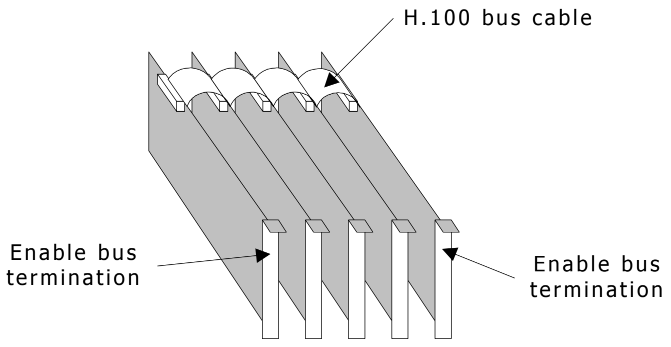
Caution:	The AG 2000-BRI board is shipped in a protective anti-static container. Leave the board in its container until you are ready to install it. Handle the board carefully and hold it only by its edges. We recommend that you wear an anti-static wrist strap connected to a good earth ground whenever you handle the board. Take care not to touch the gold fingers that plug into the PCI bus connectors.

This topic describes:

- Terminating the H.100 bus
- Configuring the DIP switch block
- Configuring the network termination
- TE/NT mode conversion jumpers
- NT power jumpers

Terminating the H.100 bus

In your system, the H.100 boards are connected to one another with an H.100 bus cable. The two boards located at the 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.



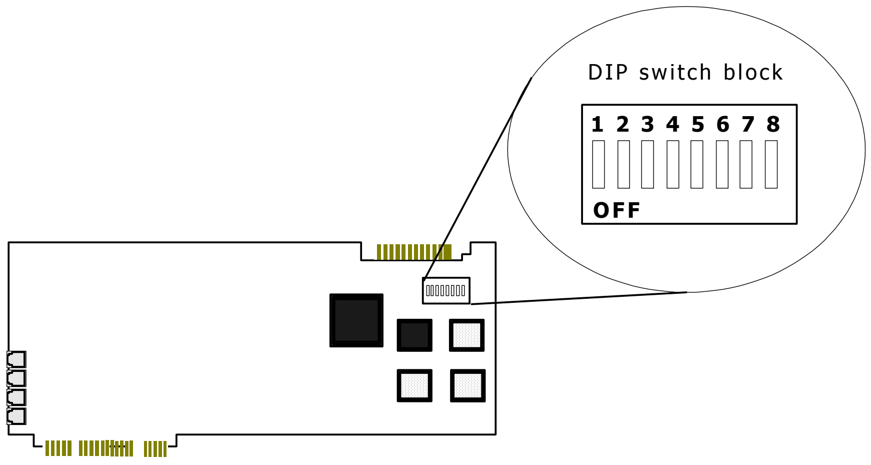
H.100 bus configuration

Configuring the DIP switch block

The AG 2000-BRI DIP switch block is located on the component side of the board.

DIP switch block S1 (shown in the following illustration) controls the H.100 bus termination. By default, all S1 switches are set to OFF (H.100 bus termination disabled). Setting all switches to ON enables H.100 bus termination. Set these switches to ON for the boards that are on the ends of the H.100 bus.

DIP switch S1 should be set to either all ON or all OFF.



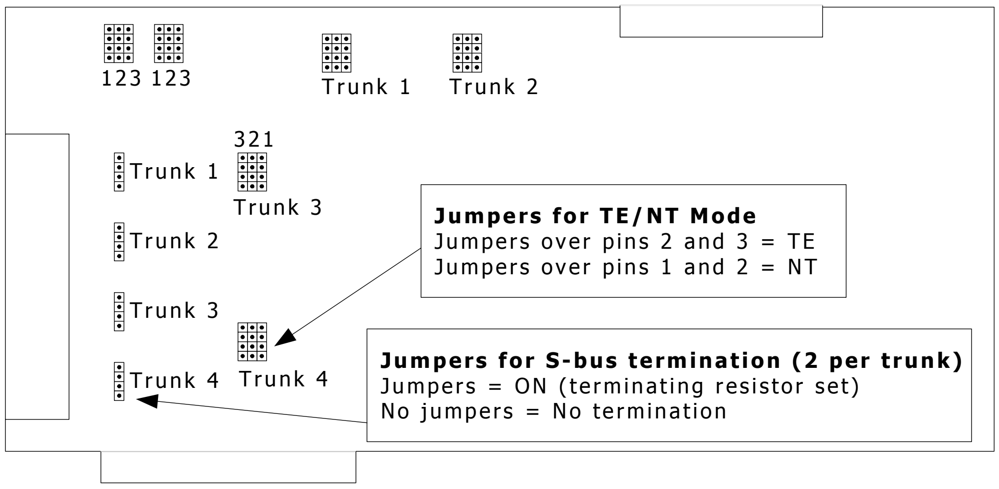
DIP switch block on the AG 2000-BRI board

Configuring the network termination

If your installation does not provide bus termination, you can set up bus termination on the AG 2000-BRI board. Generally, if several terminals are connected to a BRI trunk, only one network termination must be set on at the end of the bus.

Note: Contact your telephone company to get information about your wiring configuration.

The AG 2000-BRI board allows you to set up one network termination per trunk. In the case of point-to-multipoint configuration, only the trunk connected at the far end of the S-bus must be configured with network termination ON. Otherwise, if each trunk of the AG 2000-BRI board is connected in a point-to-point configuration, you must configure all the trunks with network termination ON. If you configure one of the AG 2000-BRI trunks as the last device on the S-bus, you must configure the corresponding trunk with the network termination, as shown in the following illustration:


123 123
Trunk 1
Trunk 2
321
Trunk 1
Trunk 3
Trunk 2
Trunk 3
Trunk 4
Trunk 4
Jumpers for TE/NT Mode
Jumpers over pins 2 and 3 = TE
Jumpers over pins 1 and 2 = NT
Jumpers for S-bus termination (2 per trunk)
Jumpers = ON (terminating resistor set)
No jumpers = No termination

S-bus termination

Termination jumpers are arranged on the board in groups of two sets of two pins per trunk. To terminate the end trunk, set jumpers over both sets of two pins for that trunk.

For more information, refer to *Connecting to the telephone network* on page 28.

TE/NT mode conversion jumpers

The AG 2000-BRI board is shipped configured for TE mode for all trunks. For each trunk, the gang jumper is set over the four pins in positions 2 and 3 to set TE mode. The gang jumper placed over positions 1 and 2 sets the board for NT mode. There are four groups of conversion jumpers, one group per trunk.

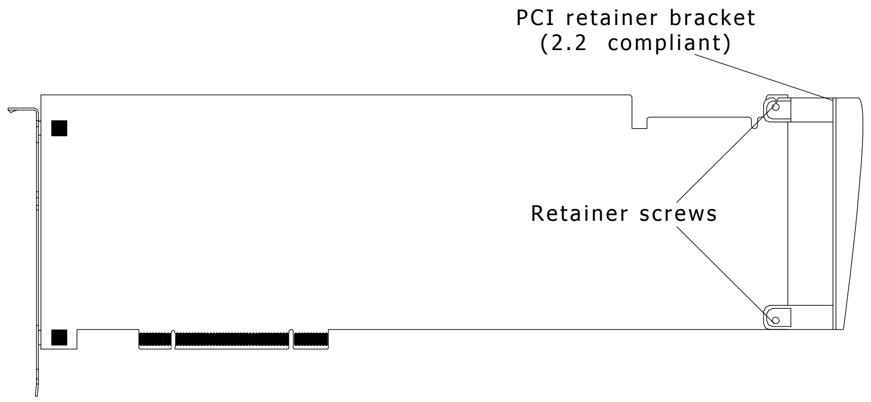
NT power jumpers

The AG 2000-BRI board does not provide a power supply to the terminals. When the terminals are not self-powered, supply power by using an external power source directly on the S-bus. NT power jumpers on the board are reserved for future use.

Installing the board

After you configure the DIP switch block and any necessary jumpers on the board, install the board and connect it to the trunk.

Complete the following steps to initially install an AG 2000-BRI board:

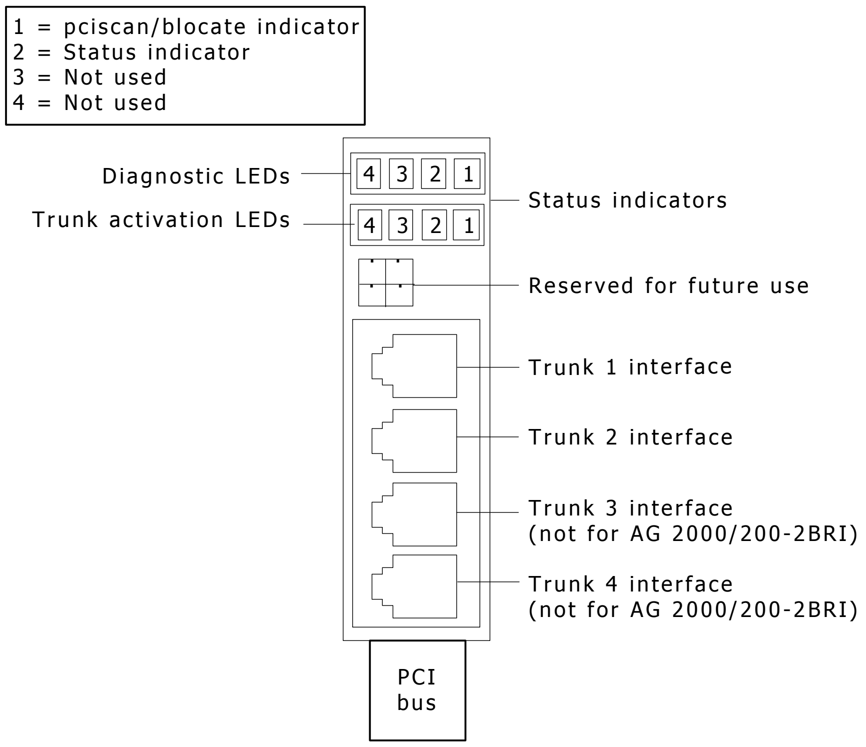
Step	Action
1	If necessary, configure the board as described in *Configuring the hardware* on page 24.
2	Turn off the computer and disconnect it from the power source. Remove the cover and set it aside.
3	If you are placing the board into: • A PCI chassis, remove the PCI retainer bracket by unscrewing it from the board. The bracket is not needed for the board to properly fit into the chassis. The 2.2 PCI retainer bracket is shown in the following illustration. • An ISA chassis, leave the PCI retainer bracket attached to the board. The bracket is needed for the board to properly fit into the chassis. 
4	Arrange the AG 2000-BRI 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	Connect the AG 2000-BRI board to the H.100 bus cable.
6	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 telephone network

BRI access provides the means to connect one or more devices to the network. Depending on your requirements, you can connect the AG 2000-BRI interfaces to a T0 trunk for a point-to-point configuration or to an S0 trunk for a point-to-multipoint configuration.

Warning:	**Important safety notes for telephony connections**
	• Allow only qualified technical personnel to install this board and associated telephone wiring. • Make sure the PC chassis is grounded through the power cord or by other means before connecting the telephone line. • 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.

As shown in the following illustration, AG 2000-BRI boards have four RJ-45C connectors. Shielded cables are also available with AG 2000-BRI boards.

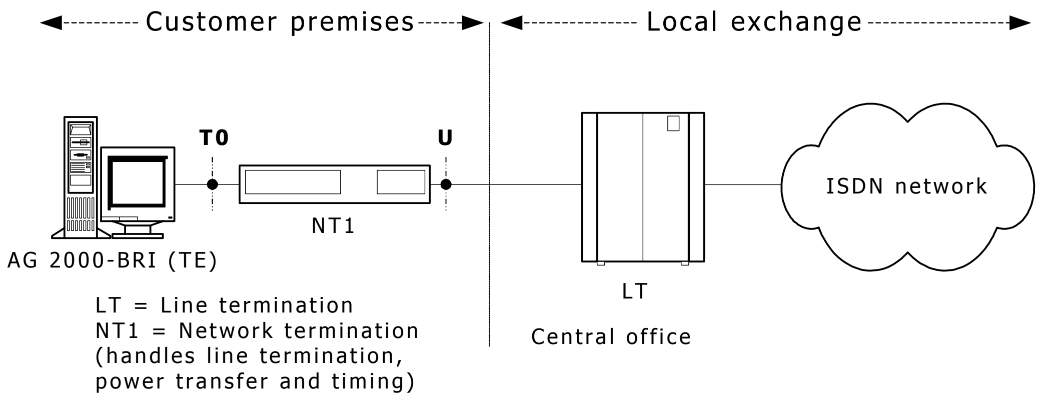


AG 2000-BRI end bracket with four RJ-45C connectors

The current version of the AG 2000-BRI board supports the terminal (TE) and the network (NT) modes. All wiring configurations refer to the ITU-T Recommendation I.430.

Connecting to a T0 trunk

A typical point-to-point configuration is shown in the following illustration. In this case, the maximum distance between the network equipment (NT1) and the terminal (TE) is 1 kilometer.



Connection at the T0 interface

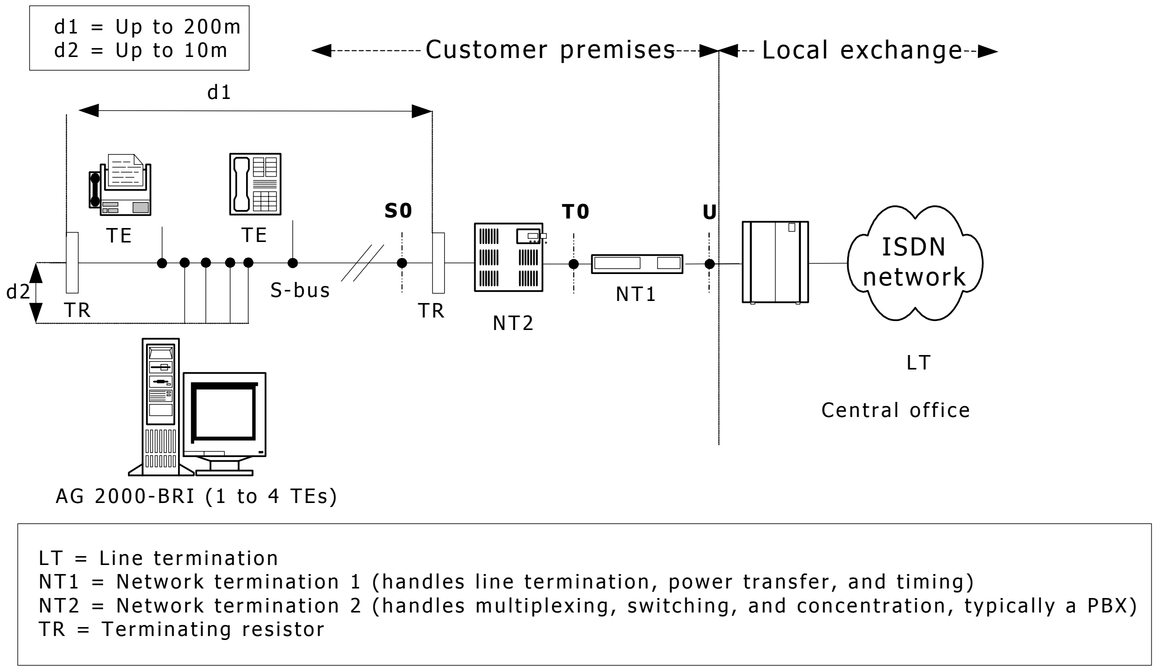
In Europe, the NT1 equipment is usually handled by the network provider.

Connecting to an S0 trunk

The following illustration shows a typical point-to-multipoint configuration (up to eight terminals) where two main topologies are supported (also called S-bus).

Short passive bus

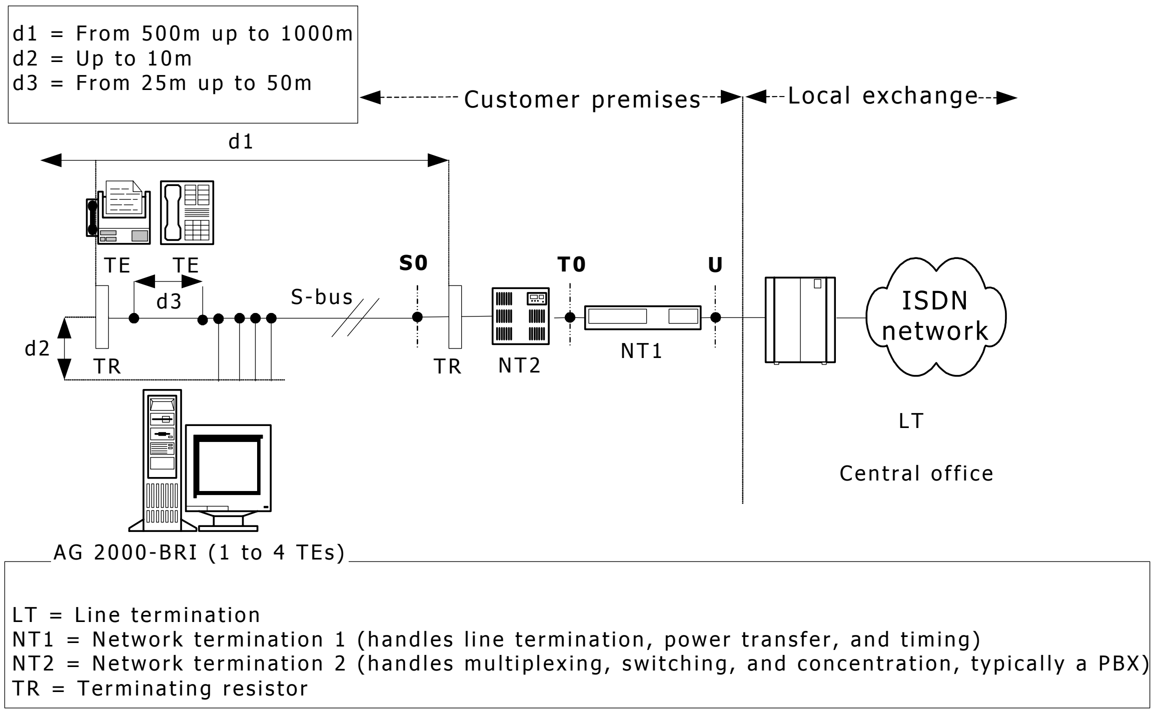
With this configuration, up to eight TEs can be connected anywhere on the bus. The bus length is limited to 200 meters (refer to the following illustration):


d1 = Up to 200m
d2 = Up to 10m
Customer premises
Local exchange
d1
TE
TE
S0
T0
U
ISDN network
d2
TR
S-bus
TR
NT2
NT1
LT
Central office
AG 2000-BRI (1 to 4 TEs)
LT = Line termination
NT1 = Network termination 1 (handles line termination, power transfer, and timing)
NT2 = Network termination 2 (handles multiplexing, switching, and concentration, typically a PBX)
TR = Terminating resistor

Connection at the S0 interface-short passive bus configuration

Extended passive bus

With this configuration, up to eight TEs can be connected, but only at the far end of the bus from the NT side. In this case, the length of the bus can be from 500 meters up to 1 kilometer (refer to the following illustration).


d1 = From 500m up to 1000m
d2 = Up to 10m
d3 = From 25m up to 50m
Customer premises
Local exchange
d1
TE
TE
d3
S-bus
S0
T0
U
d2
TR
TR
NT2
NT1
ISDN network
LT
Central office
AG 2000-BRI (1 to 4 TEs)
LT = Line termination
NT1 = Network termination 1 (handles line termination, power transfer, and timing)
NT2 = Network termination 2 (handles multiplexing, switching, and concentration, typically a PBX)
TR = Terminating resistor

Connection at the S0 interface-extended passive bus configuration

5 Configuring the board

Adding board configurations to the NMS OAM database

Each board that NMS OAM configures and starts must have a separate set of configuration parameters. Each parameter value is expressed as a keyword name and value pair (for example, AutoStart = NO). You can use NMS OAM to retrieve parameters for any component. These parameters (set through board keywords) can be added, modified, or deleted.

Before using NMS OAM, make sure that the Natural Access Server (*ctdaemon*) is running. For more information about the Natural Access Server (*ctdaemon*), refer to the *Natural Access Developer's Reference Manual*.

The following utilities are shipped with NMS OAM:

Utility	Description
oamsys	Configures and starts up boards on a system-wide basis. Attempts to start all specified boards based on system configuration files you supply.
oamcfg	Provides greater access to individual NMS OAM service configuration functions.
oaminfo	Displays keywords and settings for one or more components. Can also set individual keywords.

Refer to the *NMS OAM System User's Manual* for more information about *oamsys* and *oamcfg*.

An application can control NMS OAM using OAM service functions. For more information about the OAM service functions and about *oaminfo*, refer to the *NMS OAM Service Developer's Reference Manual*.

Configuring and starting the system with oamsys

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

Step	Action
1	Install the boards and software as described in the *installation summary* on page 23.
2	Determine which board keyword file you will use, or edit one of the sample AG 2000-BRI board keyword files, to specify appropriate configuration information for each board. For more information, refer to *Using board keyword files* on page 33.
3	Determine the PCI bus and slot locations of the boards using the *pciscan* utility. *pciscan* identifies the NMS PCI boards installed in the system and returns each board's bus, slot, interrupt, and board type.
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.
5	Start *oammon* to monitor the NMS OAM system and all NMS boards. For more information about *oammon*, refer to the *NMS OAM System User's Manual*. Start *oammon* before running *oamsys*. Keep *oammon* running to 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 (*ctdaemon* must be running when you use *oamsys*) according to the configuration information specified in the system configuration file and any associated board keyword files. For more information, refer to *Running oamsys* on page 40.

To determine the physical slot location of a specific board:

Operating system	Procedure
Windows	Use *pciscan* 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 *pciscan* with the PCI bus and PCI slot locations.
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.

For information about *pciscan* and *blocate*, refer to the *NMS OAM System User's Manual*.

Using board keyword files

A sample board keyword file is installed with Natural Access. This board keyword file is used to select the ISDN protocol variant:

File	Description
agpi2bri.cfg	AG 2000-BRI default configuration.

Board keyword files have many keywords in common. The differences in these files are related to the protocols, whose names appear as part of the name of the file. For more information about board keyword files, refer to the *NMS OAM System User's Manual*.

This topic presents:

- The sample board keyword file *agpi2bri.cfg*
- A multiple board system example

Sample board keyword files are located in the *ag\cfg* subdirectory under the Natural Access installation directory. *agpi2bri.cfg* shows the set of board keywords necessary to configure and start an AG 2000-BRI board.

AG 2000-BRI board keyword file

The following sample board keyword file (*agpi2bri.cfg*) describes one AG 2000-BRI board:

```
#
#       AG Plug-in Config File for AG 2000 BRI
#

# TCP files are shipped with the NMS CAS sub-package of Natural Access.
# Be sure that you installed the protocols that are specified below before
# trying to start a board with this configuration file.

 TCPFiles[0] = nocc.tcp           # "no trunk control" protocol
 TCPFiles[1] = isd0.tcp           # ISDN protocol

# DSP (.m54) files to link in

 DSP.C5x[0..3].Files = callp.m54 dtmf.m54 mf.m54 ptf.m54 signal.m54 tone.m54 voice.m54

 DLMFiles[0] = gtp.leo
 DLMFiles[1] = voice.leo
 DLMFiles[2] = svc.leo
 DLMFiles[3] = brietsi.leo   # ISDN protocol with ETSI variant

 XLaw = A-LAW    # A-Law silence, idle signaling bit code

# Embedded system file

 RunFile = ag2bri.cor

# Clocking configuration for one board in a system

 Clocking.HBus.ClockSource = NETWORK
 Clocking.HBus.ClockMode = STANDALONE
```

Multiple board system example

The following example assumes a system configuration in which four AG 2000-BRI boards reside in a single chassis. The boards are configured in the following way using keywords:

Board	Configuration
Board 0	System primary bus master (driving the A clock)
Board 1	Clock slave
Board 2	Clock slave
Board 3	Clock slave

The following table shows keywords used to configure the boards as described in the preceeding table:

Board	Role	Clocking keyword settings

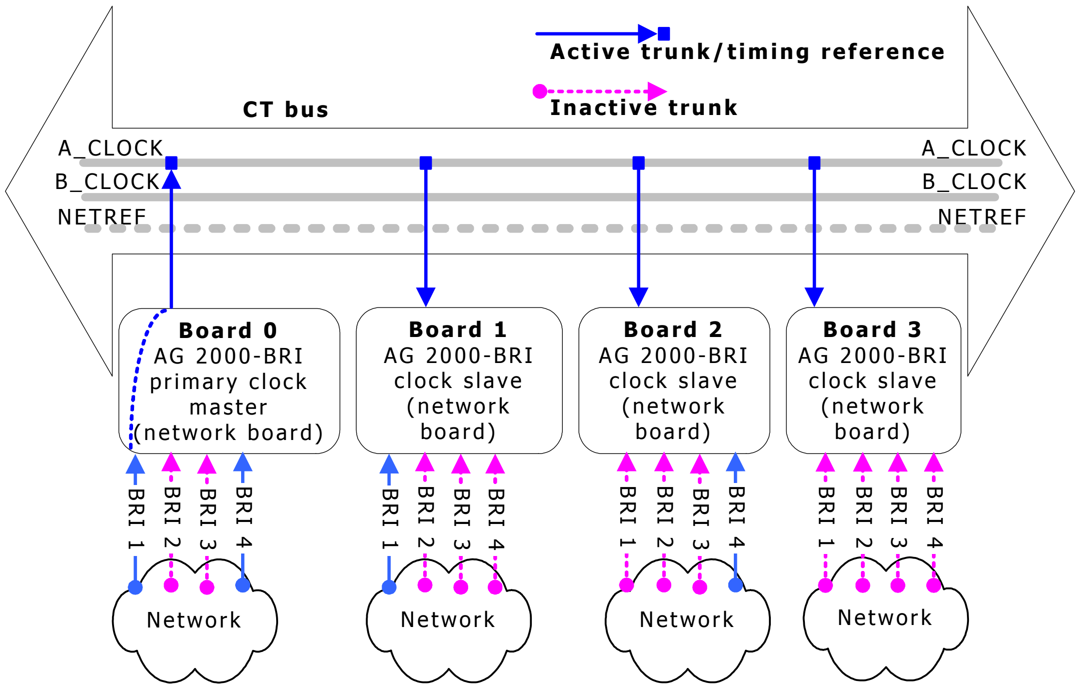
0	Primary clock master	Clocking.HBus.ClockMode = MASTER_A
		Clocking.HBus.ClockSource = NETWORK
		Clocking.HBus.BriRef = YES

1	Clock slave	Clocking.HBus.ClockMode = SLAVE
		Clocking.HBus.ClockSource = NETWORK
		Clocking.HBus.BriRef = YES

2	Clock slave	Clocking.HBus.ClockMode = SLAVE
		Clocking.HBus.ClockSource = NETWORK
		Clocking.HBus.BriRef = YES

3	Slave driving NETREF	Clocking.HBus.ClockMode = SLAVE
		Clocking.HBus.ClockSource = NETWORK
		Clocking.HBus.BriRef = YES

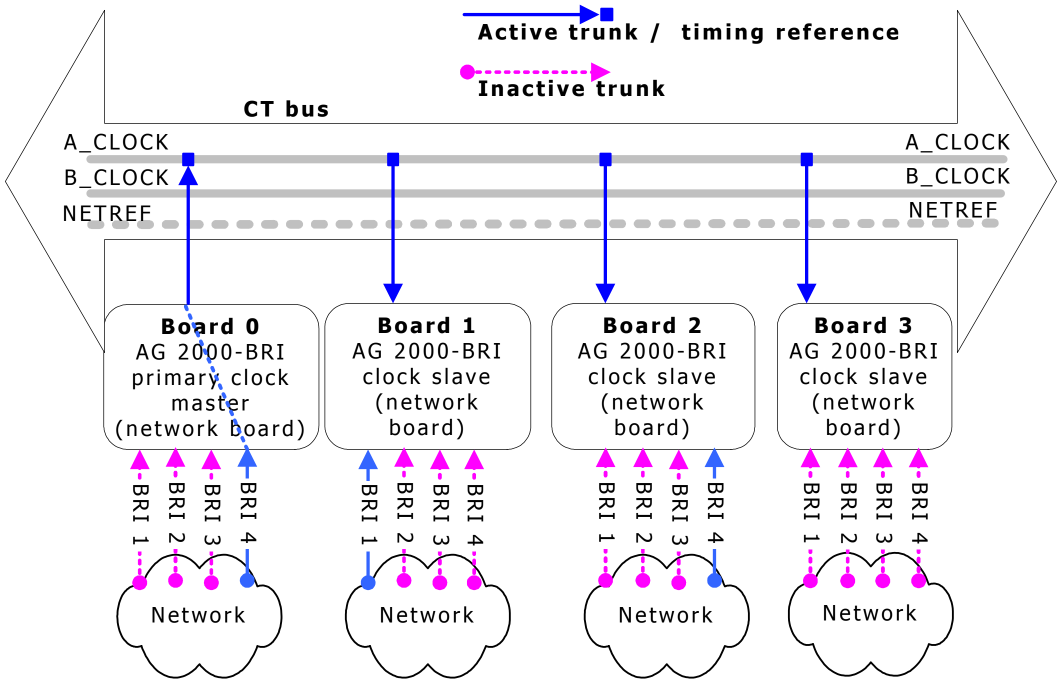
In this configuration, Board 0 is the primary clock master and drives A_CLOCK. All slave boards on the system use A_CLOCK as their timing reference. The Clocking.HBus.BriRef keyword is set to YES for Board 0, so Board 0 can use any of its local trunks interchangeably as a timing reference. In the following illustration, Board 0 is using local trunk 1:



Board 0 drives A_CLOCK using local BRI trunk 1 as a timing reference.
All other boards slave to A_CLOCK.

Sample board clocking configuration

When trunk 1 no longer provides a timing signal (for example, if no calls exist on the trunk) Board 0 switches to another currently active local trunk. In the following illustration, this trunk is local trunk 4:

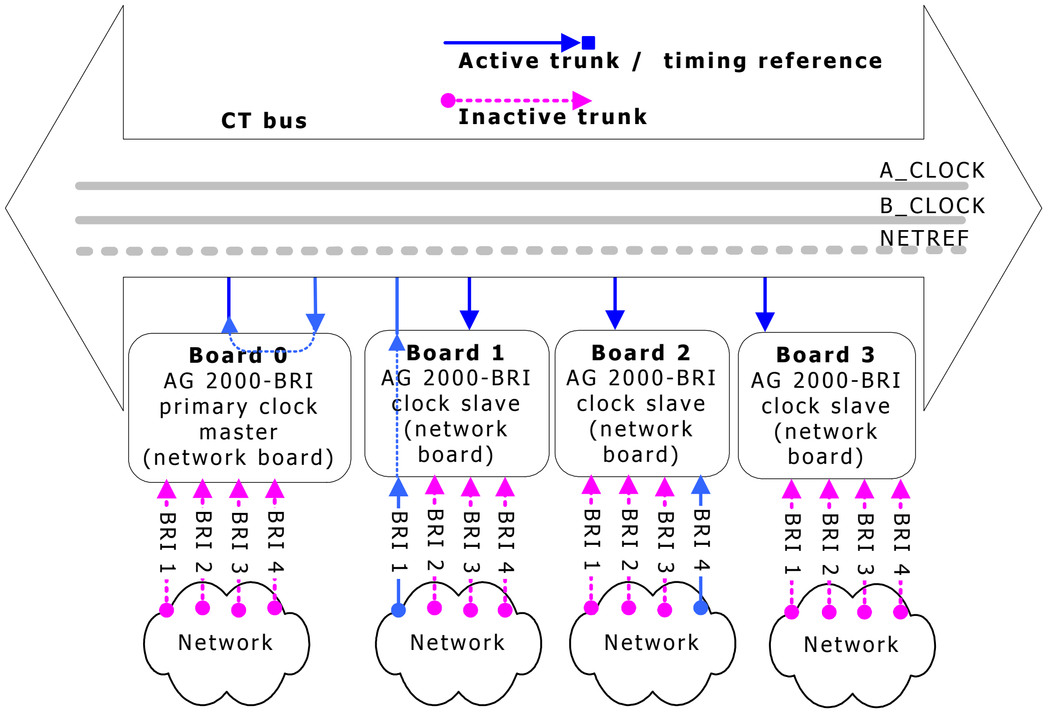


BRI trunk 1 goes down because no calls exist on the trunk. Board 0 now drives A_CLOCK using active local BRI trunk 4 as a timing reference. All other boards slave to A_CLOCK.

New local trunk timing reference

If no local trunk provides a timing signal, the clock fallback manager finds another AG 2000-BRI board with the Clocking.HBus.BriRef keyword set to YES that has an active BRI trunk. It causes this board to drive NETREF based upon the timing signal on the active trunk. The primary master now drives the clock using NETREF as the timing reference.

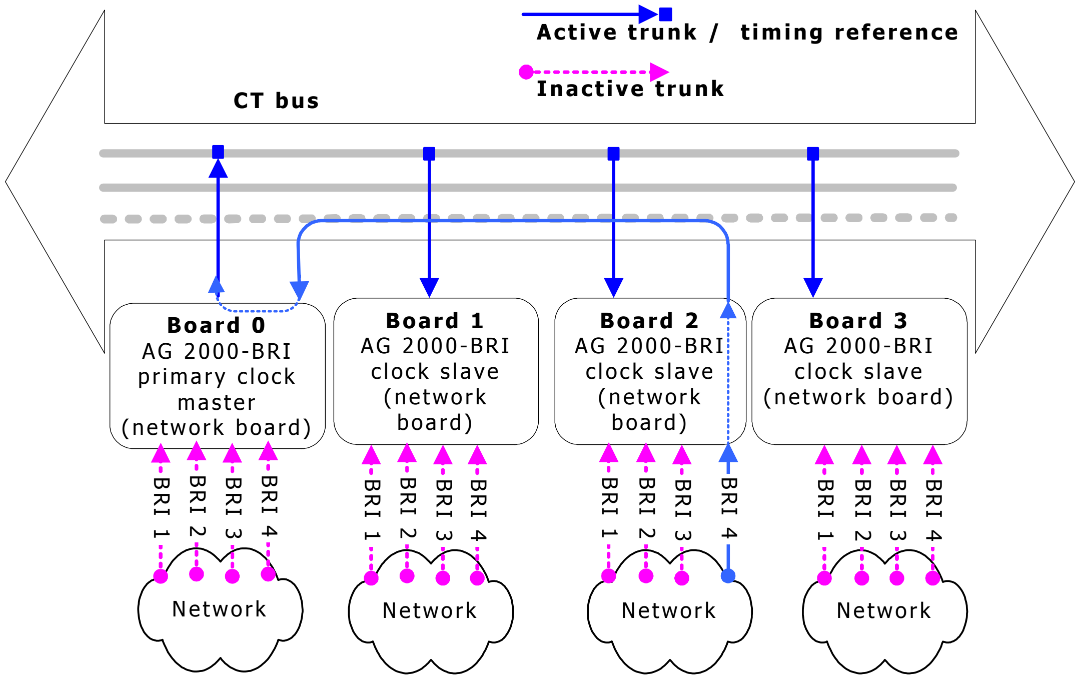
In the following illustration, Board 0, trunk 4 has gone down. Since there are no local trunks available, the clock fallback manager directs Board 1 to drive NETREF using its local trunk 1 as a timing reference. Board 0 continues to drive A_CLOCK using NETREF as the timing reference:



All Board 0 BRI trunks go down because no calls exist on the trunks. The clock fallback manager directs Board 1 to drive NETREF based on its local BRI trunk 1. Board 0 now drives A_CLOCK using NETREF as a timing reference. All other boards (including Board 1) continue to slave to A_CLOCK.

Primary master drives clock based on NETREF

The clock fallback manager continues to poll each board to determine the status of its trunks. If the trunk driving NETREF goes down, the clock fallback manager picks another trunk on the same board or on another board. For example, in the following illustration, the trunk driving NETREF (Board 1, local trunk 1) has gone down. NETREF is now being driven by Board 2, trunk 4:



All Board 1 BRI trunks go down because no calls exist on the trunks. The clock fallback manager directs Board 2 to drive NETREF based on its local BRI trunk 4. Board 0 now drives A_CLOCK using NETREF as a timing reference. All other boards (including Board 1) continue to slave to A_CLOCK.

New NETREF configuration

Creating a system configuration file for oamsys

Create a system configuration file describing all of the boards in your system. *oamsys* creates the records, and then directs NMS OAM to start the boards, configured as specified. The system configuration file is typically named *oamsys.cfg*. By default, *oamsys* looks for a file with this name when it starts up. Refer to the *NMS OAM System User's Manual* for specific information on the syntax and structure of this file.

Note: You can use the *oamgen* utility (included with the NMS OAM 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 more information about *oamgen*, refer to the *NMS OAM System User's Manual*.

The following table describes the AG board-specific settings to include in the system configuration file for each AG board:

Keyword	Description	Allowed values for AG boards
[name]	Name of the board to be used to refer to the board in the software. The board name must be unique.	Any string, in square brackets [].
Product	Name of the board product.	AG_2000_BRI
Number	Board number you use in the Natural Access application to refer to the board.	Each board's number must be unique.
Bus	PCI bus number. The bus:slot location for each board must be unique.	Values returned by *pciscan*.
Slot	PCI slot number. The bus:slot location for each board must be unique.	Values returned by *pciscan*.

File	Name of the board keyword file containing settings for the board.
	
	Several board keyword files are installed with the AG software, one for each country or region.
		For information about creating your own custom board keyword file, refer to *Changing configuration parameter settings* on page 41.
		
		You can specify more than one file after the File keyword:
		`File=mya.cfg myb.cfg myc.cfg`
		
		Alternatively, you can specify the File keyword more than once:
		`File = mya.cfg`
		`File = myb.cfg`
		`File = myc.cfg`
		
		Board keyword files are applied in the order in which they are listed. The value for a given keyword in each file overrides any value specified for the keyword in earlier files.

Sample system configuration file

The following system configuration file describes two AG 2000-BRI boards:

```
[First AG 2000-BRI]
Product = AG_2000_BRI
Number  = 0
Bus     = 0
Slot    = 15
File    = agpi2bri.cfg

[Second AG 2000-BRI]
Product = AG_2000_BRI
Number  = 1
Bus     = 0
Slot    = 16
File    = agpi2bri.cfg
```

Running oamsys

To run *oamsys*, enter there following command:

```
oamsys -f filename
```

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

Note: If you invoke *oamsys* without command line options, NMS 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 system configuration files but not of board keyword files.
- Checks for uniqueness of board names, board numbers, and board bus and slot numbers.
- Shuts down all boards recognized by NMS OAM (if any).
- Deletes all board configuration information currently maintained for the recognized boards (if any).
- Sets up the NMS OAM database and creates all records as described in the system configuration file.
- Attempts to start all boards as specified in the system configuration file and the board keyword files it references.

The Natural Access Server (*ctdaemon*) must be running for *oamsys* to operate. For more information about the Natural Access Server, refer to the *Natural Access Developer's Reference Manual*.

Changing configuration parameter settings

When you run *oamsys*, the utility starts all boards according to the configuration parameters specified in their associated board keyword files.

To change a parameter:

- Use or modify one of the sample board keyword files corresponding to your country and board type. Specify the name of this new file in the File statement in *oamsys.cfg* and run *oamsys* again. Refer to the *NMS OAM System User's Manual* for information about the syntax of board keyword files.
- Specify parameter settings with *oamcfg*. Refer to the *NMS OAM System User's Manual* for information about *oamcfg*.
- Create a new board keyword file, either with additional keywords or with keywords whose values override earlier settings.
- Specify the settings using the OAM service functions. Refer to the *NMS OAM Service Developer's Reference Manual* for more information.

You can use *oamsys* to:

- Change which software module files are downloaded to the board at startup. Refer to *Specifying configuration file locations* on page 42 for more information.
- Specify board switching.
- Configure CT bus clocking.

.leo files

A *.leo* (loadable extensible object) file is a run module, a modular extension to the core file. The core file and the run modules make up the software that runs on the board's coprocessor.

The following *.leo* files are included with AG 2000-BRI:

File	**Description**
svc.leo	DSP function manager.
gtp.leo	Trunk protocol engine.
voice.leo	Play and record manager.
brietsi.leo	ISDN protocol (ETSI variant).
brivn6.leo	ISDN protocol (France Telecom VN6 variant).
brintt.leo	ISDN protocol (Nippon Telegraph Telephone INS-1500 variant).

Trunk control programs

Trunk control programs (TCPs) perform all the signaling tasks necessary to interface with the telephony protocol used on the line or trunk. TCPs are loaded onto an AG 2000-BRI board at board initialization. After a TCP has been loaded to the AG 2000-BRI board, the application must start up its protocol before it can use the TCP to perform call control on a specific port.

The following table lists the TCPs that are applicable to AG 2000-BRI boards:

File	**Description**
nocc.tcp	No call control protocol.
isd0.tcp	ISDN protocol.

Specifying configuration file locations

Files to be downloaded to the AG boards are specified with keywords in the AG board's keyword file. For example:

```
DLMFiles[0] = filename
```

If ***filename*** contains a path specification, NMS OAM searches for the file in the specified directory. Otherwise, NMS OAM searches for the file in the current working directory of *ctdaemon*. If the file does not exist in the current working directory, NMS OAM searches for the file in the search path defined by the AGLOAD environment variable.

Configuring board clocking

When multiple boards are connected to the CT bus, you must set up a bus clock 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.

Caution:	NMS strongly recommends that you do not mix H.100/H.110 systems with MVIP systems when doing clock fallback.

Refer to the NMS OAM *readme* for more information about clocking capabilities.

This topic describes:

- AG 2000-BRI clocking capabilities
- AG 2000-BRI clock fallback
- Clock configuration methods
- Configuring CT bus clocks with keywords

To create a robust clocking configuration, you must understand basic clocking concepts such as clock mastering and clock fallback. This topic assumes that you have a basic understanding of CT bus clocking. For a complete overview of CT bus clocking, refer to the *NMS OAM System User's Manual*.

AG 2000-BRI clocking capabilities

This topic describes the rules and limitations that apply to setting up CT bus clocking for AG 2000-BRI boards.

When an AG 2000-BRI board is configured as the system primary clock master, the AG 2000-BRI board uses its own clock fallback mechanism. For more information, refer to *AG 2000-BRI clock fallback* on page 45.

Note: Use the AG 2000-BRI board as primary clock master only if no other boards with digital trunks (except other AG 2000-BRI boards) exist in the system.

An AG 2000-BRI board cannot drive one CT bus clock (A_CLOCK or B_CLOCK) based upon the other. For this reason, an AG 2000-BRI board cannot be used as a secondary clock master.

When an AG 2000-BRI board is configured as a clock slave:

- The board's first timing reference must be the system's primary clock.
- The board cannot switch to a secondary timing reference if the primary reference fails. For this reason, it cannot take advantage of a secondary clock master if the primary clock master stops driving the clock.

The following tables summarize the CT bus clocking capabilities of AG 2000-BRI boards.

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 board can use any of its trunks interchangeably as a timing reference.
NETREF1	Yes	The application must reconfigure the board as soon as possible if NETREF1 fails.
NETREF2	No	NETREF2 is available for H.110 boards only.
OSC	Yes	
Fallback to secondary timing reference	No	
Available secondary timing references:		
Local trunk	No	
NETREF1	No	
NETREF2	No	NETREF2 is available for H.110 boards only.
OSC	No	
Slave to secondary master if both references fail	No*	*The AG 2000-BRI uses a special clock fallback mechanism in this case. For more information, refer to *AG 2000-BRI clock fallback* on page 45.

Clocking capabilities as secondary master

Capability	**Yes/No**	**Comments**
Serve as secondary master	No	
Drive A_CLOCK	No	
Drive B_CLOCK	No	
Available secondary timing references:		
Local trunk	No	
NETREF1	No	
NETREF2	No	NETREF2 is available for H.110 boards only.
OSC	No	

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	No	
B_CLOCK	No	
OSC	No	

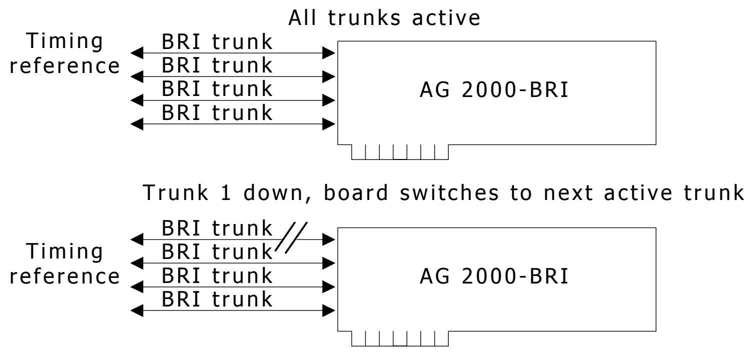
Other clocking capabilities

Capability	**Yes/No**	**Comments**
Drive NETREF1	Yes	A slave may be directed by the clock fallback manager to drive NETREF to provide the clock master with a timing signal. For more information, refer to *AG 2000-BRI clock fallback* on page 45.
Drive NETREF2	No	NETREF2 is available for H.110 boards only.
Operate in standalone mode	Yes	

AG 2000-BRI clock fallback

When an AG 2000-BRI is used in standalone mode or as a clock master, it implements a clock fallback scheme that is different from the standard fallback scheme implemented on other NMS boards. A BRI trunk does not provide timing signals unless there is a call on the trunk. If no calls exist on the trunk, the trunk shuts down.

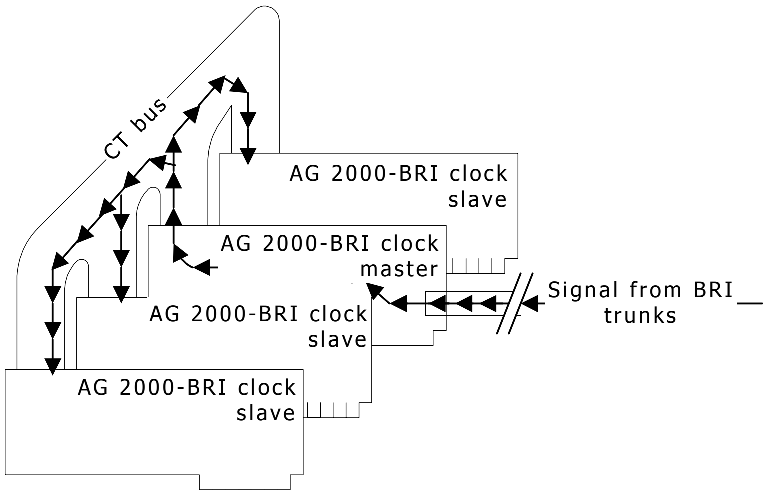
To get around this problem, the AG 2000-BRI is designed to switch between any of its trunks as a timing reference: if one trunk shuts down, the board immediately checks its other trunks, and switches to the first live trunk it finds. In this way, as long as at least one BRI trunk is active, the board has a valid timing reference.



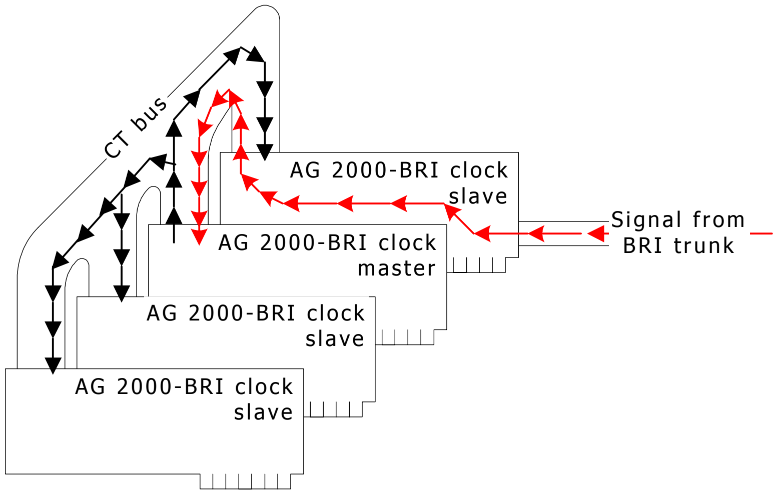
AG 2000-BRI standalone board

If multiple AG 2000-BRI boards are connected over the CT bus and an AG 2000-BRI board is clock master, the clock master can obtain a valid timing reference even if all of its trunks shut down. In this case, the clock fallback manager locates another board on the CT bus that has an active BRI trunk, and directs that board to drive the NETREF signal based on the trunk. The clock master then drives the system clock using NETREF as a timing reference.

When all of the clock master's BRI trunks go down...



Another board provides a timing reference over NETREF.



AG 2000-BRI clock fallback

If all of the trunks on the board that is driving NETREF shut down, the clock fallback manager chooses another board to drive NETREF. Thus the clock master has a timing reference as long as at least one BRI trunk on one board is active.

Clocking.HBus.BriRef activates the special AG 2000-BRI clock fallback mechanism. Refer to Configuring CT bus clock with keywords for more information.

Clock configuration methods

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 a 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 *NMS OAM System User's Manual*. **Note:** The *clockdemo* program does not support the special AG 2000-BRI clock fallback mechanism described in *AG 2000-BRI clock fallback* on page 45. If your system contains only AG 2000-BRI boards, configure board clocking using keywords, as described in *Configuring CT bus clocks with keywords* on page 47.
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 described in *Configuring CT bus clocks with keywords* on page 47.

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

Configuring CT bus clocks with keywords

The AG 2000-BRI board keywords allow you to configure the board in the following ways:

- System primary clock master
- Clock slave
- Standalone board

Note: Use the AG 2000-BRI as primary clock master only if no other boards with digital trunks (except other AG 2000-BRI boards) exist in the system.

The following sections describe how to use board keywords to specify clocking configurations on multiple-board or multiple-chassis systems.

Configuring the AG 2000-BRI as primary clock master

Use the following board keywords to configure an AG 2000-BRI as primary clock master:

Keyword	Description
Clocking.HBus.ClockSource	Specifies the source from which this board derives its timing. Set this keyword to NETWORK.
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.BriRef	Set to YES to enable the special AG 2000-BRI clock fallback mechanism.

Configuring the AG 2000-BRI as a clock slave

Use the following board keywords to configure an AG 2000-BRI 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.BriRef	Enables the special AG 2000-BRI clock fallback mechanism. Set this keyword to YES to allow the clock fallback manager to use any of the trunks on this board to drive NETREF if the primary master's timing reference fails. If it is set to NO, the clock fallback manager will not cause this board to drive NETREF.

Configuring the AG 2000-BRI as a standalone board

Use the following board keywords to configure an AG 2000-BRI in standalone mode:

Keyword	**Description**
Clocking.HBus.ClockMode	Specifies the CT bus clock from which the board derives its timing. Set this keyword to STANDALONE to specify standalone mode.
Clocking.HBus.ClockSource	Specifies the source from which this clock derives its timing. Set this keyword to NETWORK to allow the board to use any of its BRI trunks interchangeably as a timing reference.

When the AG 2000-BRI board is in standalone mode, the board cannot make connections to the CT bus.

Enabling echo cancellation

Echo cancellation is generally not required on digital trunks. It is disabled by default on the AG 2000-BRI board.

To enable echo cancellation:

Step	**Action**
1	Include the following statement in the board keyword file: `DSP.C5x[x].Files = echo.m54` where ***x*** = the next available index
2	Set the appropriate ADI service parameters in your application and in your system.

Refer to the *ADI Service Developer's Reference Manual* for information on configuring echo cancellation about the AG 2000-BRI board.

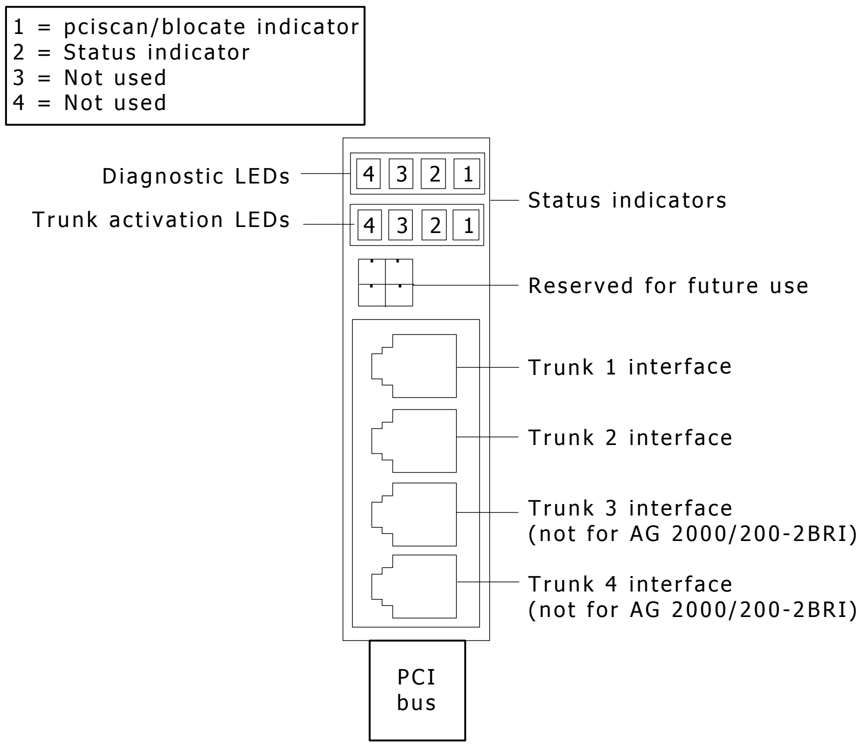
6 Verifying the installation

Status indicator LEDs

The AG 2000-BRI board has two banks of indicators (LEDs) on the end bracket of the board and 30 indicators (LEDs) on the primary side of the board.

LEDs on the end bracket

The following illustration shows the two rows of LED indicators on the end bracket of the AG 2000-BRI board:

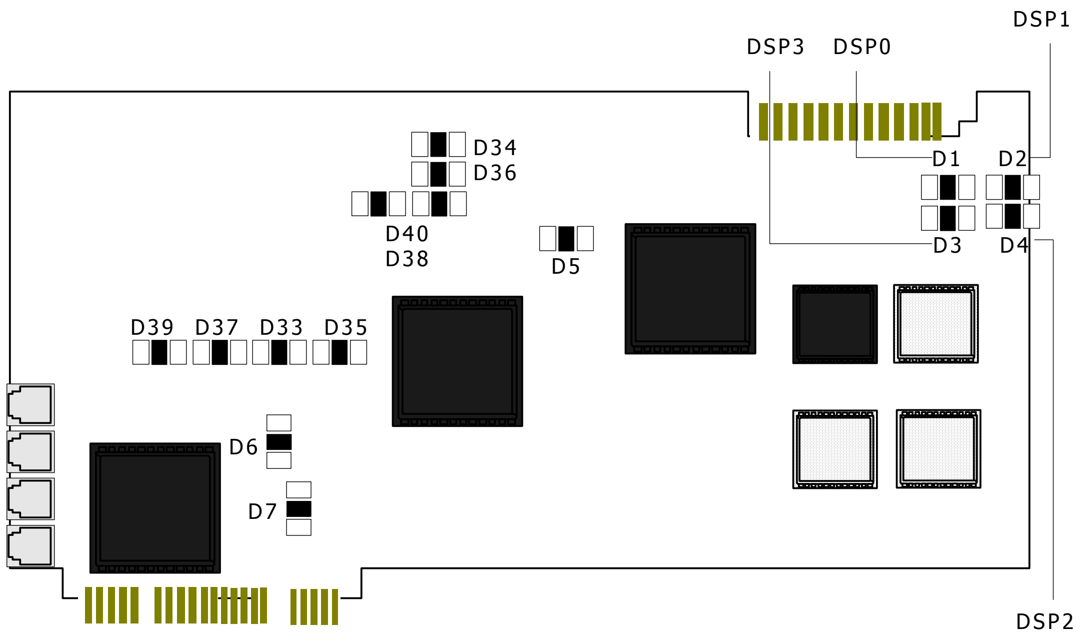


LEDs on the end bracket

The board locate indicator identifies the board using the software. The status indicator remains on after the board is booted.

LEDs on the primary side of the board

The location of the LEDs on the primary side of the AG 2000-BRI board is shown in the following illustration:



LEDs on the primary side of the AG 2000-BRI board

The following table describes the LEDs on the primary side of the AG 2000-BRI board:

LED	Purpose	Description

D1 - D4	DSPx_RST (4 Red LEDs)
	DSPx_XF (4 Green LEDs)
		The corresponding DSP is communicating with the NS486.
		Red = Problem
		Green = No problem

D5	HMIC (green)	Lit = No problem with the HMIC. Not lit = Problem with the HMIC.
D5	CLK (red)	Lit = No problem with the clock section of the HMIC. Not lit = Problem with the clock section of the HMIC.
D6	OWN (green)	Owner of SRAM. Lit = Idle of host. Not lit = Coprocessor.
D6	TD (red)	Transfer direction bit. Lit = Host access to SRAM. Not lit = 486 access to SRAM.
D7	BUSY (green)	Lit = SRAM is idle. Not lit = SRAM is in use.
D7	76GO (red)	NS486 out of reset.
D33-35-37-39	IPAC aux_6 (red)	User controlled.
D33-35-37-39	IPAC aux_7 (green)	User controlled.

LED	Purpose	Description
D34-36-38-40	IPAC res (red)	ISDN framer set. Lit = Out of reset. Not lit = No problem.

Verifying the board installation

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

Step	Action
1	Create a board keyword file to boot an AG 2000-BRI board by copying or editing one of the sample board keyword files to match your specific configuration. Refer to *Using board keyword files* on page 33 for more information. For example, use the *agpi2bri.cfg* file to configure the board for the Euro ISDN protocol.
2	Run *oammon* to monitor the status of all boards.
3	Use the *pciscan* utility to determine the bus and slot number. For more information about *pciscan*, refer to the *NMS OAM System User's Manual*.
4	Edit the *oamsys.cfg* file to reflect the board locations in your system.
5	Boot the board using the command: `oamsys`

Retrieving AG board configuration information: boardinf

boardinf is a program that reports the board number, address, type, number of ports, memory, and DSP timeslot assignments for each board in a system.

boardinf opens the AG driver and retrieves the configuration information for up to 16 AG 2000-BRI boards. If an AG 2000-BRI board exists and is properly initialized, its configuration is displayed and its DSP port addresses are displayed as one or more timeslot ranges.

To run *boardinf*:

Step	Action
1	Ensure that the AG 2000-BRI boards were initialized.
2	Open a command window.
3	Enter the following command: `boardinf` *boardinf* displays the configuration information for each AG 2000-BRI board in the system that has been loaded and initialized.
4	If no boards are detected, verify that the AG 2000-BRI board(s) has been loaded and initialized and repeat the command. If the AG 2000-BRI board configuration information is not as expected, review the board keyword file.

Verifying board operation

To verify that the board is working:

Step	Action
1	Using *agpi2bri.cfg*, initialize the board following the procedure in Retrieving AG board configuration information: boardinf.
2	Run the digital trunk monitor utility *isdndemo*.

Using swish for a standalone board

No default connections are made for a standalone board if H.100 connectivity is enabled in the board keyword file. Use *swish* to connect the local network interface to the local DSP resource. You can use *swish* interactively, or create a script in a flat text file.

The following example of *swish* commands nails up the voice and signaling streams for all 8 line interfaces of an AG 2000-BRI board that has been configured as board 0. The *swish* commands are expressed in MVIP-95 terms.

```
openswitch ag2000 = agsw 0

resetswitch ag2000


# Make voice and signaling connections
makeconnection ag2000 local:0:0..7 to local:5:0..7 QUAD

closeswitch ag2000

exit
```

Demonstration programs

The following demonstration programs are provided with Natural Access and can be used to verify that the AG 2000-BRI board is operating correctly:

Program	Demonstrates...
isdndemo	Starting the ISDN stack, receiving and placing calls in ACU configuration using the NMS ISDN Messaging API.
isdncta	Starting and stopping the ISDN stack.
isdnncc	Receiving and placing calls in channelized configuration using the Natural Call Control (NCC) API.
ctatest	Receiving and placing calls in channelized configuration, media functions using NCC and ADI APIs.
vceplay	Natural Access functions.
vcerec	Recording one or more messages to a voice file.

Note: Executables for *isdndemo* and *isdnncc* are in the respective sub-directories under *nms\ctaccess\demos*.

To run these demonstration programs on the AG 2000-BRI board, specify the board number, the ISDN variant, and the trunk configuration (TE or NT mode).

For example, enter the following command to run *isdndemo* for board 2, NTT variant, and NT mode:

```
isdndemo -b2 -p9 -n NT
```

Refer to the *Natural Access Developer's Reference Manual* for information on Natural Access demonstration programs and to the NMS ISDN manuals for ISDN demonstration programs.

7 Using CT bus switching

AG 2000-BRI switch model

This topic describes:

- The specific use of each stream, as shown for H.100 streams and local streams
- An illustration of the AG 2000-BRI switch model
- HMIC switch blocking

H.100 streams

H.100 streams	
H.100 bus	Streams 0..31, timeslots vary based on clock rate: Streams clocked at 8 MHz: timeslots 0..127 Streams clocked at 4 MHz: timeslots 0..63 Streams clocked at 2 MHz: timeslots 0..31

Local streams

Local streams	
Line interface voice in and out	Streams 0 and 1, timeslot 0..7
DSP voice in and out	Streams 4 and 5, timeslot 0..7

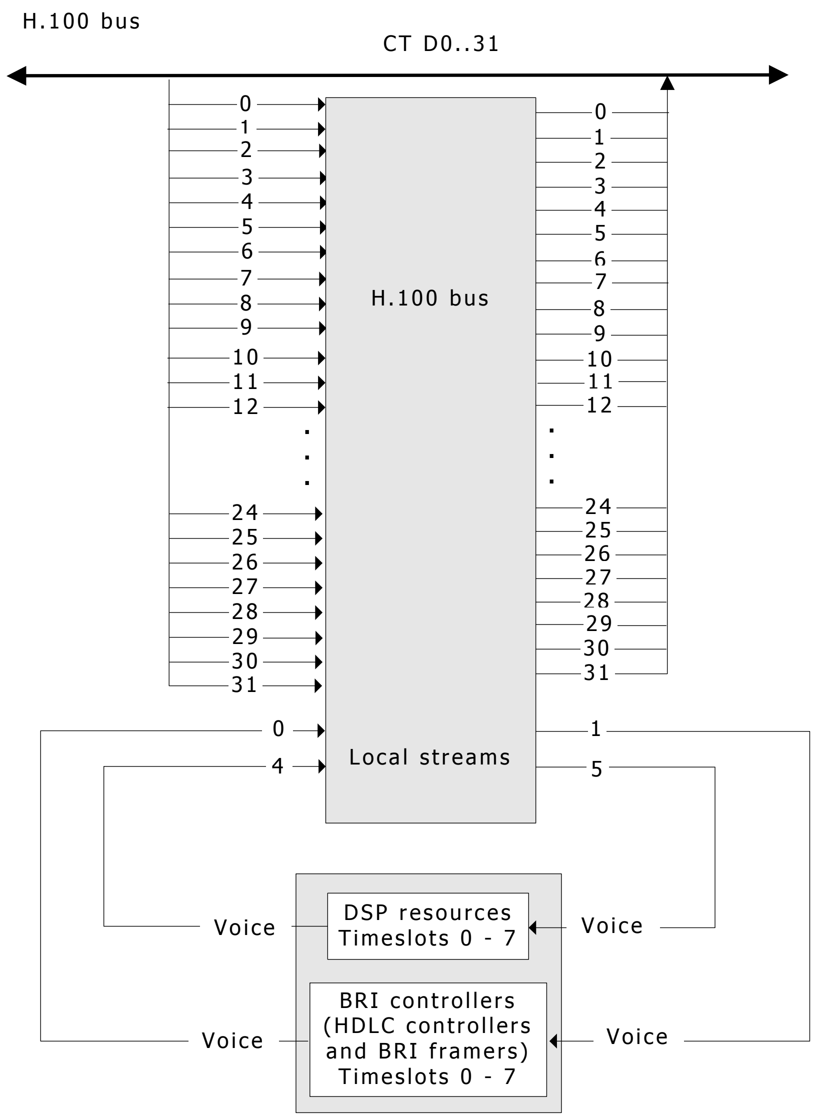
If you have an AG 2000-BRI board in the same chassis as a CG 6000 board and you want to pass data through all streams on the board, you must set the H.100 stream speed to 8 Mbit/s for all 32 streams on the AG 2000-BRI board.

The default stream speed for all streams on a CG 6000 board is 8 Mbit/s. On AG 2000-BRI boards, the default speed for streams 0 through 15 is set at 2 Mbit/s, while the default stream speed for streams 16 through 31 is set to 8 Mbit/s. Therefore, before using streams 0 through 15 to pass data, you must reset the speed for these streams to 8 Mbit/s.

To reset the stream speed, use the **swiConfigStreamSpeed** function in the Switching service or the **swi.ConfigStreamSpeed** command in the *swish* utility. For more information, refer to the *Switching Service Developer's Reference Manual*.

Switch model

The following illustration shows the AG 2000-BRI switch model:



AG 2000-BRI switch model

HMIC switch blocking

Switching on the AG 2000-BRI board is implemented by the HMIC chip. The HMIC chip can perform local bus to local bus switching in full non-blocking fashion.

The number of H.100 connections is limited to a maximum of 128 full duplex or 256 simplex (or half-duplex) connections, in any combination, from either:

- H.100 bus to the local bus, or
- H.100 bus to H.100 bus

There are no restrictions on local switching. Any local device can be connected to any other local device.

BRI trunk channels and H.100 timeslots

AG 2000-BRI boards place the voice information from the BRI trunk in timeslots and local streams as follows:

Trunk	Stream	Timeslot
1	stream 0 and stream 1	timeslots 0 and 1
2	stream 0 and stream 1	timeslots 2 and 3
3	stream 0 and stream 1	timeslots 4 and 5
4	stream 0 and stream 1	timeslots 6 and 7

Default connections

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

Switch connection	MVIP-95
Full duplex connection between line interface voice information and DSP resources.	local:0:0..7 => local:5:0..7 local:4:0..7 => local:1:0..7

To change this default routing so the board can interoperate with other boards connected to it over the H.100 bus, disable the automatic routing by setting SwitchConnections = NO. When the bus is enabled (Clocking.HBus.ClockMode is not equal to STANDALONE), there is no default routing unless you set SwitchConnections = YES.

8 Keyword summary

Using keywords

The keywords for an AG 2000-BRI 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.

This topic describes:

- Setting keyword values
- Retrieving keyword values

Note: To learn how to use NMS OAM utilities such as *oamsys* and *oamcfg*, refer to the *NMS OAM System User's Manual*. To learn about setting and retrieving keywords using OAM service functions, refer to the *NMS OAM Service Developer's Reference Manual.*

AG plug-in keywords exist in a separate record in the NMS OAM database. They indicate certain board family-level information.

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. An integer keyword can have a fixed numeric range of legal values. A string keyword can support a fixed set of legal values, or can accept any string.

Setting keyword values

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

- Use or modify one of the sample board keyword files corresponding to your country and board type. Specify the name of this new file in the File statement in *oamsys.cfg*, and run *oamsys* again. Refer to the *NMS OAM System User's Manual* for information about the syntax of board keyword files.

  Note: Using *oamsys* reboots all boards in the system.

- Create a new board keyword file, either with additional keywords or keywords whose values override earlier settings.
- Specify parameter settings using the *oamcfg* utility. Refer to the *NMS OAM System User's Manual* for information about *oamcfg*.
- Specify the settings using OAM service functions. Refer to the *NMS OAM Service Developer's Reference Manual* for more information.

To set board keywords, specify the board name in the system configuration file or on the *oamcfg* command line. To set AG plug-in level keywords, specify the AG plug-in name (*agplugin.bpi*).

Note: Keyword values take effect after the board is rebooted.

Retrieving keyword values

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

- Run the *oaminfo* sample program. On the command line, specify the board using either its name (with the `-n` option) or number (with the `-b` option):

  ```
  oaminfo -n boardname
  oaminfo -b boardnum
  ```

  To access AG plug-in level keywords, specify the AG plug-in name on the command line:

  ```
  oaminfo -n agplugin.bpi
  ```

  oaminfo returns a complete list of keywords and values. For more information about *oaminfo*, refer to the *NMS OAM Service Developer's Reference Manual*.

- Use the OAM service. Refer to the *NMS OAM Service Developer's Reference Manual* for more information.

Editable keywords

The following table summarizes the keywords that you can change:

If you want to...	Use these keywords...
Specify whether the board is started or stopped automatically	AutoStart AutoStop
Specify the board location	Location.PCI.Bus (set in the *oamsys.cfg* file) Location.PCI.Slot (set in the *oamsys.cfg* file)
Specify information about the board	LoadFile LoadSize Name (set in the *oamsys.cfg* file) Number (set in the *oamsys.cfg* file) DLMFiles[x] RunFile TCPFiles[x]
Set up debug level information	BootDiagnosticLevel
Modify memory allocation	Buffers[x].Num Buffers[x].Size MaxChannels
Set up clocking information	Clocking.HBus.BriRef Clocking.HBus.ClockMode Clocking.HBus.ClockSource Clocking.HBus.Segment
Set up information specific to NETREF1	Clocking.HBus.NetRefSource Clocking.HBus.NetRefSpeed
Set up switching information	SwitchConnections SwitchConnectMode
Control switching on the echo canceller reference stream	Echo.AutoSwitchingRefSource Echo.EnableExternalPins
Configure DSPs	DSP.C5x[x].Image DSP.C5x.Lib DSP.C5x.Loader DSP.C5x[x].Os DSP.C5x[x].Files[y] SignalIdleCode VoiceIdleCode Xlaw

Informational keywords

You cannot edit the keywords listed in this topic. Use these keywords for retrieving information about the:

- Board
- EEPROM
- Board driver

Retrieving board information

Keyword	Type	Description
Location.Type	String	Host system's bus type.
Product	String	At the board level, the product type of the board.
State	String	State of the physical board. Expected values are IDLE, BOOTED, or TESTING.

Retrieving EEPROM information

Keyword	Type	Description
Eeprom.AssemblyRevision	Integer	Hardware assembly level.
Eeprom.BoardSpecific	Integer	Board-specific data.
Eeprom.CheckSum	Integer	EEPROM checksum.
Eeprom.CPUSpeed	Integer	Coprocessor speed in MHz.
Eeprom.DRAMSize	Integer	DRAM size in kilobytes.
EEprom.DSPSpeed	Integer	DSP processor speed in MHz.
EEprom.Family	Integer	Board family.
Eeprom.MFGWeek	Integer	Week of the last full test.
Eeprom.MFGYear	Integer	Year of the last full test.
Eeprom.MSBusType	Integer	Media stream bus type. H.100 = 0
Eeprom.NumDSPCores	Integer	Total number of DSP cores on the motherboard.
Eeprom.SerialNum	Integer	Serial number unique to each board. This number is factory configured.
Eeprom.SoftwareCompatibility	Integer	Minimum software revision level.
Eeprom.SRAMSize	Integer	SRAM size in kilobytes.
Eeprom.SubType	Integer	AG family variant information.

Retrieving board driver information

Keyword	**Type**	**Description**
Driver.BoardID	String	Board driver ID for the current board. Each board accessed by a driver has a unique ID. However, two boards accessed by different drivers can have the same driver ID number.
Driver.Name	String	Operating system independent root name of the driver, for example, ag.
SwitchDriver.Name	String	Operating system independent root name of the switching driver. Expected value is AGSW.

Plug-in keywords

The AG plug-in keywords are:

- Boards[x]
- LoadSize
- Products[x]
- Version.Major
- Version.Minor

9 Keyword reference

Using the keyword reference

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

Syntax	The syntax of the keyword
Access	Read/write or read-only
Type	The data type of the value: string, integer, or file name
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 = ***setting***

Access

Read/Write

Type

String

Default

NO

Allowed values

YES | NO

Example

```
AutoStart = NO
```

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.

For more information, refer to the *NMS OAM System User's Manual*.

See also

AutoStop

AutoStop

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

Syntax

AutoStop = ***setting***

Access

Read/Write

Type

String

Default

NO

Allowed values

YES | NO

Example

```
AutoStop = NO
```

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.

For more information, refer to the *NMS OAM System User's Manual*.

See also

AutoStart

Boards[x]

Specifies the name of the board object that is managed by the AG plug-in.

Syntax

Boards[***x***] = ***boardname***

x = the index of the Board array keyword.

Access

Read-only (AG plug-in level)

Type

String

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

Type

Integer

Default

2

Allowed values

0 | 1 | 2 | 3

Example

```
BootDiagnosticLevel = 2
```

Details

This value takes precedence over the corresponding value of the BootDiagnosticLevel keyword set in the system configuration file.

The valid values for level are 0, 1, 2, and 3. Zero (0) indicates that no diagnostics are performed, and 3 is the maximum level. The trade-off for higher levels of diagnostics is the increased time needed to initialize each AG board at load time.

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, the mode of failure, or both. Some tests report additional information.

The following tests are performed during the boot diagnostics:

Test number	**Description**	**Error code**	**# WDS**	**Error number**
1	Coprocessor booted by writing 0x11 to SRAM base address.			
	• Coprocessor never booted at all.	1		
	• Coprocessor booted but crashed after writing to SRAM base address.	0x11		
	• 0xAAAA option switch selected and coprocessor crashed after updating SRAM base address.	0xAAAA		
2	Verifies the board type.	2	1	
3	Checks the DRAM size and BUSCLK programmed in the EEPROM, and sets up the part accordingly if valid EEPROM choice.	3	1	
4	Tests DSP control and status registers.	4	2	
6	Tests DRAM.	6	4	
7	Tests DSPs.	7	5	
8	Serial port test.			
	• Failed internal loopback test. Wrote a 49h and received something else back.	8	2	
9	HMIC tests			Refer to the following tables for an explanation of the error number.
	• Failed I/O test.	9	5	1
	• Failed register test.	9	5	1
	• Failed CAM test.	9	5	2
	• Failed local connections test.	9	5	3
12	DSP HPI tests.	12	4	

The following information is reported back to the host when there is a diagnostic failure:

Error code		**WORD1**	**WORD2**	**WORD3**	**WORD4**	**WORD5**
	# WDS	**Additional data**				
1	None					
2	1	EEPROM board type				
3	1	EEPROM DRAM size word				
4	2	Written	Read (masked by 0xF)			
6	4	Address low	Address high	Written	Read	
7	5	# DSPs booted	Number expected	Test ID	Memory failed address	Contents of failed address
8	2	Written	Read			
9	5	See the following table for more information.				
12	4	00 = HPIA test 01 = HPI memory test	DSP number	Written	Read	

The following information is reported back to the host for error code 9 when there is a diagnostic failure:

# WDS	**HMIC ID**	**Error number**	**Address**	**Write**	**Read**
5	0	1	5AA5	Write	Read
5	0	1	Register number	Write	Read
5	0	2	CAM address	Write	Read
5	0	3	Local connections address	Write	Read

Buffers[x].Num

Specifies the number of buffers available for play and record.

Syntax

Buffers[***x***].Num = ***buffercount***

x = 0 - 2

Access

Read/Write

Type

Integer

Default

Index 0 large	Index 1 medium	Index 2 small
16	16	32

Allowed values

Based on the available board memory.

Example

```
Buffers[0].Num = 16
```

Details

By default, two buffers are allocated per channel. For simultaneous play and record, you must configure four buffers per channel.

Buffers[1].Num is used for ISDN.

Buffers[2].Num is required for NMS Fusion systems.

See also

Buffers[x].Size, MaxChannels

Buffers[x].Size

Specifies the size, in bytes, of buffers used for play and record.

Syntax

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

Access

Read/Write

Type

Integer

Default

Index	Default value
0	16400
1	1024
2	92

Allowed values

0 - 1000000

Example

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

Details

The default buffer size is 16400.

Buffers[1].Size affects ISDN and some NMS Fusion systems. The default is 1024.

Small buffers (index[2]) cannot be configured.

See also

Buffers[x].Num

Clocking.HBus.AutoFallBack

Not applicable for AG 2000-BRI boards because the availability of network clocking reference is not guaranteed apart from a call session.

For information about setting up CT bus clocking, and rules and restrictions for configuring CT bus clocking, refer to *Configuring board clocking* on page 42.

Syntax

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

Access

Read only

Type

String

Default

NO

Allowed values

Not applicable.

See also

Clocking.HBus.ClockMode, Clocking.HBus.ClockSource

Clocking.HBus.BriRef

Activates or deactivates the special AG 2000-BRI clock fallback mechanism.

For information about setting up CT bus clocking, and rules and restrictions for configuring CT bus clocking, refer to *Configuring board clocking* on page 42.

Syntax

Clocking.HBus.BriRef = ***setting***

Access

Read/Write

Type

String

Default

NO

Allowed values

YES | NO

Details

If enabled, when the clock master's references both fail, the AG plug-in selects another board to provide a timing reference for the clock master. The signal is passed to the clock master using NETREF.

See also

Clocking.HBus.ClockMode, Clocking.HBus.ClockSource

Clocking.HBus.ClockMode

Specifies the board's control of the H.100 clock.

For information about setting up CT bus clocking, and rules and restrictions for configuring CT bus clocking, refer to *Configuring board clocking* on page 42.

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 for the keyword include:

Value	Description
MASTER_A	The board is used to drive the CT bus A clock based on the timing information derived from a clocking source.
MASTER_B	The board is used to drive the CT bus B clock based on the timing information derived from a clocking source.
SLAVE	The board acts as a clock slave, deriving its timing from the primary bus master. **Note**: Connections are allowed to the board's CT bus timeslots.
STANDALONE	The board references its timing signal from its own oscillator and does not drive any CT bus timing signal clocks. **Note**: Connections are not allowed to the board's CT bus timeslots in standalone mode.

For more information about standalone mode, refer to *Default connections* on page 57.

See also

Clocking.HBus.AutoFallBack, Clocking.HBus.ClockSource

Clocking.HBus.ClockSource

Specifies the clock reference origin.

For information about setting up CT bus clocking, and rules and restrictions for configuring CT bus clocking, refer to *Configuring board clocking* on page 42.

Syntax

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

Access

Read/Write

Type

String

Default

OSC

Allowed values

OSC | A_CLOCK | B_CLOCK | NETWORK | NETREF

Example

```
Clocking.HBus.ClockSource = OSC
```

Details

Valid entries for the keyword include:

Value	Description
OSC	Uses the on-board oscillator as a reference.
A_CLOCK	Causes the board to act as a clock slave to the H.100 bus A clock by deriving the local clock from the bus. Another H.100 board (or H.110 board) must drive the clock on the bus.
B_CLOCK	Causes the board to act as a clock slave to the H.100 bus B clock by deriving the local clock from the bus. Another H.100 board (or H.110 board) must drive the clock on the bus.
NETWORK	Causes the board to derive the local clock, telephony bus clock, and line transmit clock using the clock extracted from one of the BRI trunks.
NETREF	H.100 network reference.

Clocking.HBus.NetRefSource

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

For information about setting up CT bus clocking, and rules and restrictions for configuring CT bus clocking, refer to *Configuring board clocking* on page 42.

Syntax

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

Access

Read/Write

Type

String

Default

STANDALONE

Allowed values

OSC | STANDALONE | NETWORK

Example

```
Clocking.HBus.NetRefSource = NETWORK
```

Details

Valid entries for the keyword include:

Value	Description
OSC	The oscillator uses the board's local clock (for diagnostics only).
STANDALONE	The NETREF clock is not driven.
NETWORK	The timing signal is derived from a device source (digital trunk).

See also

Clocking.HBus.NetRefSpeed

Clocking.HBus.NetRefSpeed

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

For information about setting up CT bus clocking, and rules and restrictions for configuring CT bus clocking, refer to *Configuring board clocking* on page 42.

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.Segment

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

Syntax

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

Access

Read/Write

Type

Integer

Default

1

Allowed values

Non-zero integer

Example

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

DLMFiles[x]

Specifies a 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

Access

Read/Write

Type

String

Default

None.

Allowed values

A valid file name.

Example

```
DLMFiles[0] = ag2fax.leo
```

Details

A *.leo* (loadable extensible object) file is a type of run module. For AG boards, the software that runs on the board coprocessor consists of the core file and any run modules.

The following *.leo* files are included with and need to be configured with AG 2000-BRI boards:

File name	Description
svc.leo	DSP function manager.
gtp.leo	Trunk protocol engine.
voice.leo	Play and record manager.
brietsi.leo	ISDN protocol with ETSI variant.

To use NaturalFax, you must specify the NaturalFax run module to be downloaded to the board.

DLMFiles[***x***] is required for AG 2000-BRI boards.

See also

RunFile

DSP.C5x.Lib

Specifies the DSP library file.

Syntax

DSP.C5x.Lib = ***filename***

Access

Read/Write

Type

Filename

Default

ag2liba.r54 if Xlaw = A-LAW

ag2libu.r54 if Xlaw = MU-LAW

Allowed values

A valid file name.

Example

```
DSP.C5x.Lib = ag2liba.r54
```

DSP.C5x.Loader

Specifies the module to load DSP functions for boards.

Syntax

DSP.C5x.Loader = ***filename***

Access

Read/Write

Type

Filename

Default

ag2boot.b54

Allowed values

A valid file name.

Example

```
DSP.C5x.Loader = special.b54
```

Details

The naming convention for DSP loader files is ***filename****.b54*.

See also

DSP.C5x.Lib

DSP.C5x[x].Files[y]

Specifies the name or the ID of a DSP file that targets a specific DSP.

Syntax

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

x = 0..31

y = file number

Access

Read/Write

Type

File name

Default

None.

Allowed values

A valid file name.

Example

```
DSP.C5x[0..7].Files[0] = callp.m54
```

Details

These files are automatically distributed among the various DSPs by the AG plug-in according to internal rules. The naming convention for files is ***filename***.*m54*.

The following DSP files are available:

DSP file	Description
adsir(_j).m54	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).m54	Contains the FSK data transmitter. (*_j*) is the V.23 variant.
callp.m54	Contains voice and tone detectors used for call progress detection. Use for any outgoing or two-way trunk protocol and for call progress analysis.
dtmf.m54	Contains the DTMF receiver, energy and silence detector, and precise tone filter typically used for cleardown.
dtmfe.m54	A variant of *dtmf.m54*, optimized for use with the echo canceller (*echo.m54*). It yields better talk-off resistance but requires the echo canceller to achieve the best cut-through performance. **Note**: You must use the echo canceller with this function.

DSP file	**Description**
echo.m54	Contains the echo cancellation function. The echo canceller removes reflected transmit channel energy from the incoming signal, which improves DTMF detection and voice recognition while playing. NMS 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 canceller filter coefficients that are adapted every period. The echo function has an adapt period of 2 ms. Therefore, an echo function with a 20 ms tail length and 100% rate adapts all the coefficients in 2 ms while the same function with a 25% rate adapts in 8 ms.
g726.m54	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.m54*. *g6726.m54* is required if you start play/record with an encoding type of ADI_ENCODE_G726.
gsm_ms.m54	Contains MS-GSM play and record functions. The 13 kbit/s full rate GSM speech codec is in Microsoft formatted frames.
gsm_mspl.m54	Contains identical play and record functions as *gsm_ms.m54* except that the maximum output power of the play function is limited.
ima.m54	Contains IMA ADPCM play and record functions. IMA is a standard for 32 kbit/s speech encoding.
mf.m54	Contains the multi-frequency receiver which is required for any trunk protocol (TCP) that uses MF signaling, and required by the MF detector.
oki.m54	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.m54	Contains precise tone filters. Typically used for CNG, CED, or custom tone detection.
rvoice.m54	Contains PCM play and record functions. *rvoice.m54* is required to play or record with an encoding of ADI_ENCODE_MULAW, ADI_ENCODE_ALAW, or ADI_ENCODE_PCM8M16.
rvoice_vad.m54	Contains PCM play and record functions. Record functions can enable the voice activity detection (VAD) capability. *rvoice_vad.m54* is required to play or record with an encoding of ADI_ENCODE_MULAW, ADI_ENCODE_ALAW, or ADI_ENCODE_PCM8M16.
signal.m54	Contains signaling, ring detector, and pulse functions. These are out of band functions which typically operate on the MVIP signaling stream. This file is required for: • Any trunk protocol except NOCC • The signal detector • Sending a pulse
tone.m54	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 second record function.

DSP file	**Description**
voice.m54	Contains NMS ADPCM play and record functions. The compressed speech is in a framed format with 20 milliseconds 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.m54	Contains play and record functions for PCM speech in formats commonly used in WAVE files, including 8 and 16 bit 11 kHz sampling.

Refer to *Functions for managing resources* on page 113 for information about the DSP resources available on each board and the DSP requirements for each ADI service function.

Refer to *DSP/task processor files and processing power* on page 116 to estimate the DSP requirements for your application and for instructions for re-configuring DSP resources if necessary.

DSP.C5x[x].Image

Specifies the DSP image file for the processor.

Syntax

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

x = 0..31

Access

Read/Write

Type

File name

Default

None.

Allowed values

A valid file name.

Example

```
DSP.C5x[1].Image = ag2fax.c54
```

Details

Specifies a pre-linked DSP image file for AG boards used by developers to develop their own DSP images.

The naming convention for DSP image files is ***filename****.c54.*

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

DSP.C5x[x].Os

Defines the operating systems per DSP.

Syntax

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

x = 0..31

Access

Read/Write

Type

File name

Default

dspos2f.k54 on all DSPs

Allowed values

A valid file name.

Example

```
DSP.C5x[1].Os = dspos2f.k54
```

Echo.AutoSwitchingRefSource

Determines if the on-board switching manager performs automatic switching of the echo canceller 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:17:0          # 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:35:0
```

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

Echo.EnableExternalPins

Determines if the echo canceller 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 canceller reference input and the echo canceller output to be switched. They appear on output stream 34 and reference stream 35.

See also

Echo.AutoSwitchingRefSource

LoadFile

Specifies the boot loader for the board.

Syntax

LoadFile = ***filename***

Access

Read/Write

Type

Filename

Default

ag2000.lod

Allowed values

A valid file name.

Example

Windows:

```
LoadFile = c:\nms\ag\load\AGxxxx.lod
```

See also

LoadSize

LoadSize

Indicates the coprocessor software download size specified in the system configuration file.

Syntax

LoadSize = ***size***

Access

Read/Write (AG plug-in level)

Type

Integer

Default

0x7500

Allowed values

0 - 0xFFFF

Example

```
LoadSize = 0x7500
```

See also

LoadFile

Location.PCI.Bus

Specifies the PCI logical bus location of the board.

Syntax

Location.PCI.Bus = ***busnum***

Access

Read/Write

Type

Integer

Default

0

Allowed values

0 - 255

Example

```
Location.PCI.Bus = 0
```

Details

Every PCI slot in the system is identified by a unique PCI logical bus and slot number. A PCI board is identified in the system configuration file by specifying its logical bus and slot number.

This statement along with the Location.PCI.Slot keyword assigns the board number to the physical board.

Use *pciscan* to determine the PCI logical bus and slot assigned for all NMS PCI boards in the system. For more information, refer to the *NMS OAM System User's Manual*.

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

Allowed values

0 - 255

Example

```
Location.PCI.Slot = 1
```

Details

Every PCI slot in the system is identified by a unique PCI bus and slot number. A PCI board is identified in the system configuration file by specifying its bus and slot number.

This statement along with Location.PCI.Bus assigns the board number to the physical board.

Use *pciscan* to determine the PCI bus and slot assigned for all NMS PCI boards in the system. For more information, refer to the *NMS OAM System User's Manual*.

MaxChannels

Specifies the maximum number of channels to allocate on the board.

Syntax

MaxChannels = ***numChannels***

Access

Read/Write

Type

Integer

Default

8

Allowed values

1 - 255

Example

```
MaxChannels = 4
```

Details

The number of channels affects memory requirements. If Buffers[0].Num is not configured, two buffers are allocated per channel.

See also

Buffers[x].Num

Name

Specifies the name of the board.

Syntax

Name = ***boardname***

Access

Read/Write

Type

String

Default

None.

Allowed values

The name can be up to 64 characters long.

Example

```
Name = AG_2000_BRI
```

See also

Number

Number

Specifies the logical board number for this board.

Syntax

Number = ***xxx***

Access

Read/Write

Type

Integer

Default

0

Allowed values

0 - 31

Example

```
Number = 0
```

Details

NMS OAM creates a board number that is guaranteed to be unique within a chassis. You can override this value.

See also

Name

Products[x]

At the AG plug-in level, indicates the product types supported by the plug-in.

Syntax

Products[***x***] = ***product_type***

Access

Read-only (AG plug-in level)

Type

String

Allowed values

AG_2000_BRI

Details

The contents of the Products[***x***] keyword in the AG plug-in (and all other installed plug-ins) are added to the Supervisor array keyword Products[***x***] at startup. You can retrieve the values in the Supervisor keyword Products[***x***] to determine all products supported by all installed plug-ins.

See also

Name

RunFile

Specifies the runtime software to be transferred to the board.

Syntax

RunFile = ***filename***

Access

Read/Write

Type

File name

Default

ag2bri.cor

Example

```
RunFile = ag2bri.cor
```

Details

The RunFile is the core file that is used with module extension files (specified by DLMFiles[x]).

RunFile is not mandatory.

See also

DLMFiles[x]

SignalIdleCode

Specifies the signal bit patterns transmitted by an idle DSP or to an unconnected line interface. In general, a DSP is considered to be idle when no application is using it.

Syntax

SignalIdleCode = ***signal_idlecode***

Access

Read/Write

Type

Integer

Default

If Xlaw = MU-LAW, default = 0.

If Xlaw = A-LAW, default = 9.

Allowed values

0x00 - 0xFF

Example

```
SignalIdleCode = 0xd
```

See also

VoiceIdleCode

SwitchConnections

Specifies whether or not to nail up default connections.

Syntax

SwitchConnections = ***setting***

Access

Read/Write

Type

String

Default

Auto

Allowed values

Yes | No | Auto

Example

```
SwitchConnections = Yes
```

Details

Valid entries include the following values:

Setting	Description
Yes	Nails up connections independent of the Clocking.HBus.ClockMode setting.
No	Does not nail up connections.
Auto	Nails up connections automatically if Clocking.HBus.ClockMode = 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 the HMIC switch connect mode.

Syntax

SwitchConnectMode = ***setting***

Access

Read/Write

Type

String

Default

ByChannel

Allowed values

ByChannel | AllDirect | AllConstantDelay

Example

```
SwitchConnectMode = AllConstantDelay
```

Details

Valid entries include the following values:

Option	Description
ByChannel	The mode for each board connection depends on whether the connection is made using **swiMakeConnection** or **swiMakeFramedConnection**.
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 time frame. For backward connections (from higher-numbered timeslots to lower-numbered timeslots), data is transferred in the next frame.
AllConstantDelay	Data is delayed so that the destination timeslot is always in the next frame regardless of whether it is a forward connection.

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

TCPFiles[x]

Specifies a trunk control program for the current board(s).

Syntax

TCPFiles[***x***] = ***filename***

x = the number of the TCP file.

Access

Read/Write

Type

String

Default

None.

Allowed values

A valid file name.

Example

```
TCPFiles[0] = nocc.tcp
TCPFiles[1] = isd0.tcp
```

Details

Trunk control programs perform all signaling tasks necessary to interface with the telephony protocol used on the line or trunk. TCPs are loaded onto an NMS board during initialization. After a TCP is loaded, applications must start the protocol before they can use the TCP to perform call control on specific ports.

For more information about starting protocols on NMS boards, refer to the *ADI Service Developer's Reference Manual*. For more information about loading and running TCP files, refer to the *NMS CAS for Natural Call Control Developer's Manual* or to the *NMS ISDN for Natural Call Control Developer's Manual*.

Note: The TCPFiles[***x***] keyword is required for configurations that run CAS signaling protocols.

Version.Major

Specifies the major version number of the AG plug-in. The Version.Major number is incremented if a change is made to the plug-in.

Syntax

Version.Major = ***number***

Access

Read-only (AG plug-in level)

Type

Integer

Allowed values

Not applicable.

See also

Version.Minor

Version.Minor

Specifies the minor version number of the AG plug-in. The Version.Minor value is changed when a change is made to the AG plug-in.

Syntax

Version.Minor = ***number***

Access

Read-only (AG plug-in level)

Type

Integer

Allowed values

Not applicable.

See also

Version.Major

VoiceIdleCode

Sets the voice bit pattern transmitted by an idle DSP or to an unconnected line interface.

Syntax

VoiceIdleCode = ***voice_idlecode***

Access

Read/Write

Type

Integer

Default

If Xlaw = MU-LAW, default = 0x7f.

If Xlaw = A-LAW, default = 0xd5.

Allowed values

0x00 - 0xFF

Example

```
VoiceIdleCode = 0xd5
```

Details

In general, a DSP is considered to be idle when no application is using it.

On digital trunks, the idle code is determined by local regulations and should not be altered.

See also

SignalIdleCode

Xlaw

Defines the switch idle codes.

Syntax

Xlaw = ***compandmode***

Access

Read/Write

Type

String

Default

A-LAW

Allowed values

A-LAW | MU-LAW

Example

```
XLaw = MU-LAW
```

See also

DSP.C5x[x].Files[y], SignalIdleCode, VoiceIdleCode

10 Hardware specifications

General hardware specifications

This topic describes:

- Mechanical specifications
- Host interface
- H.100 compliant interface
- Environment
- Power requirements

Mechanical specifications

This section presents the mechanical specifications for the AG 2000-BRI board including PCI and H.100 bus connectors.

The AG 2000-BRI board has:

- 64K x 16 of SRAM
- An HMIC, which provides enhanced-compliant MVIP switching
- 4MB of DRAM
- 100 MIPS C549 parts
- NS486SXL-25

TDM bus	Features one complete H.100 bus interface with MVIP-95 enhanced-compliant switching
DSP processing power	Up to four Texas Instruments TMS320VC549GGU-100 DSPs at 100 MIPS each
Microprocessor	One 25 MHz 80486 compatible embedded processor
Board weight	0.45 lb (.20 kg)
Software	Natural Access for Windows, UNIX, or Red Hat Linux

Host interface

Feature	**Specification**
Electrical	PCI bus designed to PCI local bus specification revision 2.1
Mechanical	Designed to the PCI local bus specification revision 2.2 for a long expansion card (physical dimensions 4.2 x 12.283 in)
Bus speed	33 MHz
Memory	32K on-board interface memory

H.100 compliant interface

- Flexible connectivity between line interfaces, DSPs, and H.100 bus.
- Switchable access to any of 4096 H.100 timeslots.
- H.100 clock master or clock slave (software-selectable).
- Compatible with any H.100 or H-MVIP compliant telephony interface.
- H.100 bus termination capability (switch-enabled).

AG 2000-BRI boards are not intended to operate with NMS MVIP-90 boards, since existing NMS MVIP-90 boards are not supported by NMS OAM.

Environment

Feature	**Description**
Operating temperature	0 to 50 degrees C
Storage temperature	-20 to 70 degrees C
Humidity	5% to 80%, non-condensing

Power requirements

AG 2000-BRI board	**Number of DSPs**	**+5 volt current**
200-2BRI	2	2.0A max 1.0A typical
400-4BRI	4	2.3A max 1.3A typical

Compliance and regulatory certification

In addition to the approval obtained by NMS for the board and its associated software, some countries require a system level approval before connecting the system to the public network. To learn what approvals you require, contact the appropriate regulatory authority in the target country.

Agency	Country	Standard
EMC	EU countries	EN 55022: (1998) Class B (with shielded cable) EN55024 (1998)
Safety	EU countries	EN 60950: (1992 + Amendments 1 to 4)
Telecom	EU countries	TBR3 (ISDN BRI)
	Other countries	Refer to the NMS web site.

EU R&TTE statement

The AG 2000-BRI board is intended to be connected to a Euro-ISDN basic rate access public telecommunication network in all EU countries.

The physical interface of the AG 2000-BRI board complies with the ITU-T I 430 recommendation and meets the European requirement for ISDN basic access (TBR3).

A copy of the R&TTE Declaration of Conformity is shipped with the board.

11 Managing resources

Functions for managing resources

Most Natural Access functions implicitly use processes that run on the DSP resources. For example, **adiStartToneDetector** starts the tone detector function running on a DSP. **adiStartRecording** starts one of many voice compression functions running on a DSP. AG boards are shipped with default configurations that make the most commonly used functions available.

Note: It is not feasible or practical to make every possible function simultaneously available to an application.

This topic lists default functions and custom functions available for AG 2000-BRI boards.

Default functions

The following functions are available in the default configuration files shipped with AG 2000-BRI boards:

- DTMF detection
- MF tone detection
- Tone detection
- Tone generation
- Cleardown detection
- Call progress detection
- NMS speech

Custom functions

The following functions can be loaded on AG 2000-BRI boards with NMS OAM:

- Caller ID
- ADSI
- NMS speech normal
- OKI speech normal
- IMA/DVI speech
- WAVE speech

The following functions can reduce the board's standard port count of 8:

- Echo cancellation
- NMS speech 1.5X
- NMS speech 2.0X
- OKI speech 1.5X
- OKI speech 2.0X
- G.726 speech
- MS-GSM speech

Customizing AG 2000-BRI board functions

Complete the following steps to configure the AG 2000-BRI boards in a system to use functions that are not in the default configuration:

Step	Action
1	List all of the functions that you want to make available to your application in the connected call state for the ports on a given AG board.
2	Determine which DSP files are required for the functions specified.

3	Add an entry to the DSP.C5x[x].Files[y] keyword for each new DSP file that is required. The syntax for the statement is:
	`DSP.C5x[x].Files = filename.m54`
	For example, to configure for echo cancellation, specify the following DSP file:
	`DSP.C5x[`***x***`].Files = echo.m54`
	where ***x*** = DSP file number

4	Check your MIPS usage. Take the worst-case MIPS usage for each port on a board. Add up the total MIPS usage for all ports. This must not exceed the available MIPS for any board in the system. If it does, reduce the number of ports used on that board by the application accordingly.
5	Check the list of configuration restrictions.
6	Initialize the boards by running *oamsys*.

Example 1: Configuring an AG 2000-BRI board

This example describes how to configure a standard AG 2000-BRI board to play and record OKI 6 kHz speech instead of NMS speech without using echo cancellation.

Step	Action
1	List all functions used in the connected state: • DTMF detector • Cleardown detector • Tone generator (for playing beeps). • OKI play 6 kHz • OKI record 6 kHz
2	The required DSP files are: • *tone.m54* • *dtmf.m54* • *ptf.m54* • *oki.m54* • *signal.m54*
3	Calculate maximum MIPS usage per port and for the board. The MIPS requirements for the selected functions are: • DTMF Detector = 1.94 MIPS • Tone Detector = 1.25 MIPS • Tone Generator = 0.75 MIPS • OKI Play 6kHz = 2.19 MIPS • OKI Record 6kHz = 2.25 MIPS Assume that the last three functions are mutually exclusive on each port. Only one of the three functions are active at any given time on a given port. Consequently, the per-port maximum MIPS usage is: 1.94 + 1.25 + 2.25 MIPS per port = 5.44 MIPS. The maximum board MIPS usage is: 8 ports * 5.44 MIPS per port = 43.52 MIPS. Since 90 MIPS are available, no restrictions apply.
4	Edit the board keyword file to contain the following: `DSP.C5x[x].Files = tone dtmf ptf oki` This loads the files on all DSPs.
5	Run *oamsys* with the edited board keyword file to load the DSP files.

DSP/task processor files and processing power

The binary code for running functions is contained in DSP files. One or more functions are contained in each file. NMS boards differ in the total number of DSPs they contain and the speed of the DSPs on the board.

DSP speed is measured in millions of instructions per second (MIPS). Each function that runs on a DSP consumes MIPS. If the total MIPS consumption for all the requested functions on all the ports of a given board exceeds the total MIPS available for that board, an error event occurs. If MIPS-intensive functions are required, you can reduce the total number of ports on a board, which makes more MIPS per port available.

The following table shows the MIPS usage for all the available functions shipped with Natural Access:

DSP file	Function	MIPS	Related API function	Related arguments
adsir.m54	ADSI receiver	3.13	**adiStartReceivingFSK**	
adsix.m54	ADSI transmitter	1.13	**adiStartSendingFSK**	
callp.m54	Call progress	1.06	**adiStartCallProgress**	
dtmf.m54	DTMF only	0.69	**adiStartDTMFDetector**	
dtmf.m54	Post- and pre-tone silence	0.69	**adiStartEnergyDetector**	
dtmf.m54	DTMF, post- and pre-tone silence	1.94	**adiStartToneProtocol**	
g726.m54	G.726 Play	7.63	**adiStartPlaying**	***encoding*** = ADI_ENCODE_G726
g726.m54	G.726 Record	7.25	**adiStartRecording**	***encoding*** = ADI_ENCODE_G726
gsm_ms.m54	MS-GSM Play 8 kHz	2.10	**adiStartPlaying**	***encoding*** = ADI_ENCODE_GSM
gsm_ms.m54	MS-GSM Record 8 kHz	4.40	**adiStartRecording**	***encoding*** = ADI_ENCODE_GSM
gsm_mspl.m54	MS-GSM Play limit 8 kHz	2.80	**adiStartPlaying**	***encoding*** = ADI_ENCODE_GSM
gsm_mspl.m54	MS-GSM Record 8 kHz	4.40	**adiStartRecording**	***encoding*** = ADI_ENCODE_GSM
ima.m54	IMA/DVI ADPCM Play 6 kHz	2.06	**adiStartPlaying**	***encoding*** = ADI_ENCODE_IMA_24
ima.m54	IMA/DVI ADPCM Play 8 kHz	1.81	**adiStartPlaying**	***encoding*** = ADI_ENCODE_IMA_32
ima.m54	IMA/DVI ADPCM Record 6 kHz	2.19	**adiStartRecording**	***encoding*** = ADI_ENCODE_IMA_24

DSP file	**Function**	**MIPS**	**Related API function**	**Related arguments**
imaply.m54	IMA/DVI ADPCM Record 8 kHz	2.00	**adiStartRecording**	***encoding*** = ADI_ENCODE_IMA_32
mf.m54	Forward detect, backward compelling	2.56	**adiStartMFDetector**	
mf.m54	Backward detect, forward compelling	2.56	**adiStartMFDetector**	
mf.m54	MF detection	1.81	**adiStartMFDetector**	
mf.m54	MF forward detection	1.81	**adiStartMFDetector**	
mf.m54	MF backward detection	1.81	**adiStartMFDetector**	
oki.m54	OKI Play 6 kHz	2.19	**adiStartPlaying**	***encoding*** = ADI_ENCODE_OKI_24, ***maxspeed*** = 100
oki.m54	OKI Play 8 kHz	2.06	**adiStartPlaying**	***encoding*** = ADI_ENCODE_OKI_32, ***maxspeed*** = 100
oki.m54	OKI Play 6 kHz 1.5X	4.13	**adiStartPlaying**	***encoding*** = ADI_ENCODE_OKI_24, ***maxspeed*** = 150
oki.m54	OKI Play 8 kHz 1.5X	3.63	**adiStartPlaying**	***encoding*** = ADI_ENCODE_OKI_32, ***maxspeed*** = 150
oki.m54	OKI Play 6 kHz 2.0X	5.44	**adiStartPlaying**	***encoding*** = ADI_ENCODE_OKI_24, ***maxspeed*** = 200
oki.m54	OKI Play 8 kHz 2.0X	4.81	**adiStartPlaying**	***encoding*** = ADI_ENCODE_OKI_32, ***maxspeed*** = 200
oki.m54	OKI Record 6 kHz	2.25	**adiStartRecording**	***encoding*** = ADI_ENCODE_OKI_24
oki.m54	OKI Record 8 kHz	2.00	**adiStartRecording**	***encoding*** = ADI_ENCODE_OKI_32
ptf.m54	2 single freq or 1 tone pair	1.25	**adiStartToneDetector**	
ptf.m54	4 single freq or 2 tone pair	1.81	**adiStartCallProgress**	***precmask***!=0
rvoice.m54	mu-law Play	0.63	**adiStartPlaying**	***encoding*** = ADI_ENCODE_MULAW
rvoice.m54	A-law Play	0.63	**adiStartPlaying**	***encoding*** = ADI_ENCODE_ALAW
rvoice.m54	WAVE Play, 8 kHz, 16-bit	0.63	**adiStartPlaying**	***encoding*** = ADI_ENCODE_PCM8M16

DSP file	Function	MIPS	Related API function	Related arguments
rvoice.m54	mu-law Record	0.63	**adiStartRecording**	***encoding*** = ADI_ENCODE_MULAW
rvoice.m54	A-law Record	0.63	**adiStartRecording**	***encoding*** = ADI_ENCODE_ALAW
rvoice.m54	WAVE Record, 8 kHz, 16-bit	0.63	**adiStartRecording**	***encoding*** = ADI_ENCODE_PCM8M16
rvoice_vad.m54	mu-law Play	0.63	**adiStartPlaying**	***encoding*** = ADI_ENCODE_MULAW
rvoice_vad.m54	A-law Play	0.63	**adiStartPlaying**	***encoding*** = ADI_ENCODE_ALAW
rvoice_vad.m54	WAVE Play, 8 kHz, 16-bit	0.63	**adiStartPlaying**	***encoding*** = ADI_ENCODE_PCM8M16
rvoice_vad.m54	mu-law Record	0.88	**adiCommandRecord** **adiStartRecording**	***encoding*** = ADI_ENCODE_MULAW
rvoice_vad.m54	A-law Record	0.88	**adiCommandRecord** **adiStartRecording**	***encoding*** = ADI_ENCODE_ALAW
rvoice_vad.m54	WAVE Record, 8 kHz, 16-bit	0.88	**adiCommandRecord** **adiStartRecording**	***encoding*** = ADI_ENCODE_PCM8M16
tone.m54	Tone generator	0.75	**adiStartDial** **adiStartDTMF** **adiStartTones**	
voice.m54	NMS Play 16 kbit/s	3.13	**adiStartPlaying**	***encoding*** = ADI_ENCODE_NMS_16, ***maxspeed*** = 100
voice.m54	NMS Play 24 kbit/s	3.13	**adiStartPlaying**	***encoding*** = ADI_ENCODE_NMS_24, ***maxspeed*** = 100
voice.m54	NMS Play 32 kbit/s	3.13	**adiStartPlaying**	***encoding*** = ADI_ENCODE_NMS_32, ***maxspeed*** = 100
voice.m54	NMS Play 64 kbit/s	0.63	**adiStartPlaying**	***encoding*** = ADI_ENCODE_NMS_64, ***maxspeed*** = 100
voice.m54	NMS Play 16 6 kHz 1.5X	5.63	**adiStartPlaying**	***encoding*** = ADI_ENCODE_NMS_16, ***maxspeed*** = 150
voice.m54	NMS Play 24 6 kHz 1.5X	5.81	**adiStartPlaying**	***encoding*** = ADI_ENCODE_NMS_24, ***maxspeed*** = 150
voice.m54	NMS Play 32 6 kHz 1.5X	5.81	**adiStartPlaying**	***encoding*** = ADI_ENCODE_NMS_32, ***maxspeed*** = 150
voice.m54	NMS Play 64 6 kHz 1.5X	2.31	**adiStartPlaying**	***encoding*** = ADI_ENCODE_NMS_64, ***maxspeed*** = 150

DSP file	**Function**	**MIPS**	**Related API function**	**Related arguments**
voice.m54	NMS Play 16 6 kHz 2.0X	7.19	**adiStartPlaying**	***encoding*** = ADI_ENCODE_NMS_16, ***maxspeed*** = 200
voice.m54	NMS Play 24 6 kHz 2.0X	7.50	**adiStartPlaying**	***encoding*** = ADI_ENCODE_NMS_24, ***maxspeed*** = 200
voice.m54	NMS Play 32 6 kHz 2.0X	7.44	**adiStartPlaying**	***encoding*** = ADI_ENCODE_NMS_32, ***maxspeed*** = 200
voice.m54	NMS Play 64 6 kHz 2.0X	2.81	**adiStartPlaying**	***encoding*** = ADI_ENCODE_NMS_64, ***maxspeed*** = 200
voice.m54	NMS Record 16 kbit/s	3.38	**adiStartRecording**	***encoding*** = ADI_ENCODE_NMS_16
voice.m54	NMS Record 24 kbit/s	3.38	**adiStartRecording**	***encoding*** = ADI_ENCODE_NMS_24
voice.m54	NMS Record 32 kbit/s	3.38	**adiStartRecording**	***encoding*** = ADI_ENCODE_NMS_32
voice.m54	NMS Record 64 kbit/s	0.63	**adiStartRecording**	***encoding*** = ADI_ENCODE_NMS_64
wave.m54	WAVE Play 11 kHz 8-bit	1.56	**adiStartPlaying**	***encoding*** = ADI_ENCODE_PCM11M8
wave.m54	WAVE Play 11 kHz 16-bit	1.44	**adiStartPlaying**	***encoding*** = ADI_ENCODE_PCM11M16
wave.m54	WAVE Record 11 kHz 8-bit	1.5	**adiStartRecording**	***encoding*** = ADI_ENCODE_PCM11M8
wave.m54	WAVE Record 11 kHz 16bit	1.13	**adiStartRecording**	***encoding*** = ADI_ENCODE_PCM11M16

The following table shows the correspondence between the filter and adapt values used for the echo canceller and MIPS consumption:

DSP file	**Filter length (ms)**	**Adapt time (ms)**	**MIPS**
echo.m54	2	100	2.75
echo.m54	2	200	2.38
echo.m54	2	400	2.25
echo.m54	2	800	2.13
echo.m54	4	100	3.13
echo.m54	4	200	2.63
echo.m54	4	400	2.38
echo.m54	4	800	2.25
echo.m54	6	100	3.50
echo.m54	6	200	2.88

DSP file	Filter length (ms)	Adapt time (ms)	MIPS
echo.m54	6	400	2.63
echo.m54	6	800	2.50
echo.m54	8	100	3.88
echo.m54	8	200	3.13
echo.m54	8	400	2.88
echo.m54	8	800	2.75
echo.m54	10	100	4.25
echo.m54	10	200	3.50
echo.m54	10	400	3.00
echo.m54	10	800	2.88
echo.m54	16	100	5.25
echo.m54	16	200	4.25
echo.m54	16	400	3.63
echo.m54	16	800	3.38
echo.m54	20	100	5.63
echo.m54	20	200	4.50
echo.m54	20	400	3.88
echo.m54	20	800	3.38
echo_v3.m54	24	100	8.56
echo_v3.m54	24	200	6.13
echo_v3.m54	24	400	4.88
echo_v3.m54	24	800	4.25
echo_v3.m54	32	100	10.75
echo_v3.m54	32	200	7.56
echo_v3.m54	32	400	5.94
echo_v3.m54	32	800	5.13
echo_v3.m54	40	100	13.00
echo_v3.m54	40	200	9.00
echo_v3.m54	40	400	7.00
echo_v3.m54	40	800	6.00
echo_v3.m54	48	100	15.25
echo_v3.m54	48	200	10.44
echo_v3.m54	48	400	8.06

DSP file	Filter length (ms)	Adapt time (ms)	MIPS
echo_v3.m54	48	800	6.88
echo_v3.m54	64	100	19.69
echo_v3.m54	64	200	13.31
echo_v3.m54	64	400	10.19
echo_v3.m54	64	800	8.56
echo_v4.m54	2	100	4.125
echo_v4.m54	2	200	3.938
echo_v4.m54	2	400	3.875
echo_v4.m54	2	800	3.813
echo_v4.m54	4	100	4.438
echo_v4.m54	4	200	4.188
echo_v4.m54	4	400	4.063
echo_v4.m54	4	800	4.000
echo_v4.m54	6	100	4.750
echo_v4.m54	6	200	4.438
echo_v4.m54	6	400	4.313
echo_v4.m54	6	800	4.188
echo_v4.m54	8	100	5.063
echo_v4.m54	8	200	4.688
echo_v4.m54	8	400	4.500
echo_v4.m54	8	800	4.438
echo_v4.m54	10	100	5.375
echo_v4.m54	10	200	4.938
echo_v4.m54	10	400	4.750
echo_v4.m54	10	800	4.625
echo_v4.m54	16	100	6.313
echo_v4.m54	16	200	5.688
echo_v4.m54	16	400	5.375
echo_v4.m54	16	800	5.188
echo_v4.m54	20	100	6.938
echo_v4.m54	20	200	6.188
echo_v4.m54	20	400	5.813
echo_v4.m54	20	800	5.625

DSP file	**Filter length (ms)**	**Adapt time (ms)**	**MIPS**
echo_v4.m54	24	100	10.375
echo_v4.m54	24	200	7.938
echo_v4.m54	24	400	6.750
echo_v4.m54	24	800	6.125
echo_v4.m54	32	100	12.625
echo_v4.m54	32	200	9.375
echo_v4.m54	32	400	7.813
echo_v4.m54	32	800	7.000
echo_v4.m54	40	100	14.813
echo_v4.m54	40	200	10.875
echo_v4.m54	40	400	8.875
echo_v4.m54	40	800	7.875
echo_v4.m54	48	100	17.063
echo_v4.m54	48	200	12.313
echo_v4.m54	48	400	9.938
echo_v4.m54	48	800	8.750
echo_v4.m54	64	100	21.500
echo_v4.m54	64	200	15.188
echo_v4.m54	64	400	12.000
echo_v4.m54	64	800	10.438

AG 2000-BRI board processing

In most applications, all DSP functions can run on all DSPs on the board. Complex functions such as WAVE speech, echo cancellation, and variable speech rates can result in reduced number of ports.

Use the following table as a guideline for determining board functionality. There are additional constraints such as memory and queue sizes in determining required MIPS:

AG board	Total DSPs	MIPS per DSP	Operating system overhead per DSP (MIPS)	Available MIPS
AG 2000/200-2BRI	2	100	10	180
AG 2000/400-4BRI	4	100	10	360

Index