Dialogic® CX 2000 Station Interface Board Installation and Developer's Manual

December 2009 64-0486-02

www.dialogic.com

Copyright and legal notices

Revision history

Revision	Release date	Notes
9000-62160-10	May 2002	NBS, Natural Access 2002-1
9000-62160-11	April 2003	SRG, Natural Access 2003-1
9000-62160-12	April 2004	SRR, Natural Access 2004-1
64-0486-01	October 2009	LBG, NaturalAccess R9.0
64-0486-02	December 2009	LBG, NaturalAccess R9.0.1
Last modified: December 3, 2009		

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

Table Of Contents

1 Introduction

The *Dialogic® CX 2000 PCI Station Interface Board Installation and Developer's Manual* explains how to:

- Select a proper chassis for safety and heat considerations
- Install a CX 2000 board in a chassis
- Configure external power supplies
- Install the driver software
- Verify that the board has been installed correctly and is operating correctly
- Perform CT bus switching

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

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 CX 2000 board

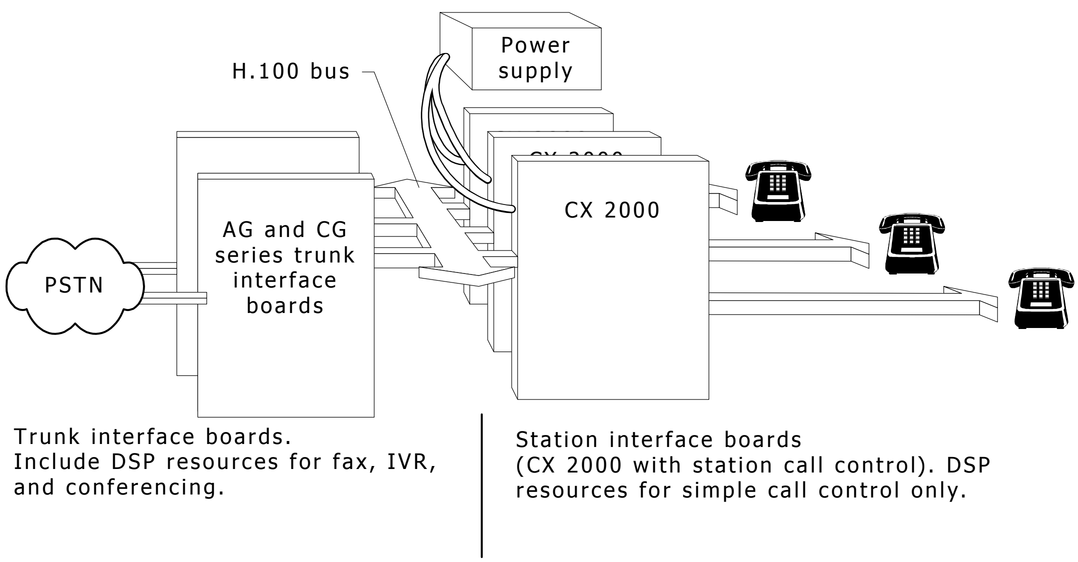
CX 2000 board features

CX 2000 boards are station interfaces for Enterprise markets. They provide analog interfaces to analog devices such as telephones, fax machines, and modems within a private network. They can be used to build such systems as private branch exchanges, automatic call distributors, and IP-PBXs.

In a system containing CX 2000 boards, any communication with the public network is performed by trunk interface boards. CX 2000 boards communicate with these boards over the H.100 bus.

Refer to www.dialogic.com/declarations/default.htm for a list of available CX 2000 board configurations, for a list of countries where Dialogic has obtained approval for the CX 2000 board, and for product updates.

CX 2000 boards have sufficient on-board DSP resources for simple, low-level call control functions. More complex, resource-intensive operations (such as voice play or record functions) must be performed by other boards.



The CX 2000-32 board supports up to 32 stations and provides high ring capacity. It has the following limitations:

- Requires external ring voltage supply
- Requires a chassis with air flow considerations described in *Selecting a PCI chassis* on page 17
- UL and CSA requirements limit cabling to within the building

CX 2000 boards offer a standard set of station call control features. Functions such as playing, recording, and conferencing are performed by the trunk interface boards or other resource boards in the system.

The following table summarizes the CX 2000 board features:

Chassis type	PCI
Number of ports	32
CT bus	H.100
Call center applications	Supported
PBX applications	Supported
Detect on/off hook	Supported
Detect flash-hook	Supported
DTMF detection	Supported
DTMF generation	Supported
Dial tone	Supported
Call progress tones	Supported
CT bus switching API	Supported
Heart beat diagnostic	Supported
Transmit gain	Supported
Receive gain	Supported
Temperature sensors	Supported
On premise extensions	Supported
Off premise extensions	Not supported
Wiring between buildings	Not supported The CX 2000 board is limited to inside cabling, due to both heat and safety power cross certification.
Internal ringing supply	Not supported
Easy chassis selection	Not supported Selecting a PCI chassis with proper air flow is critical for multiple CX 2000-32 boards to operate. For more information, refer to *Selecting a PCI chassis* on page 17.

The CX 2000 fully supports the H.100 bus specification. Switching is implemented with the T8100A chip. The T8100A offers full support for the H.100 bus within the H.100 architecture providing access to all 4096 slots on the bus.

On the boards, switch connections are allowed for up to 128 full duplex connections between local devices and the bus. Non-blocking switch connections are allowed between local devices.

Power supply

To provide power for talk battery and for ringing station telephones (if necessary), an external power supply is required. NMS Communications supplies a rack mount power supply chassis that can contain up to four interchangeable supply modules. Alternatively, you can obtain a power supply from another source. You can connect the power supply to each board.

For more information on choosing and connecting power supplies, refer to *Using the NMS rack mount power supply chassis* on page 25.

Developer's cable kit

To make connecting telephones to CX 2000 boards easier, a developer's cable kit is available. It consists of the following components:

- Two RJ-21, twenty-five pair, 10 feet cables
- Two breakout boxes RJ-21 to 25 RJ-11

For more information about the developer's cable kit, refer to *Connecting to station telephones* on page 20.

Software components

CX 2000 boards require the following software components:

- The Natural Access development environment that provides services for call control, voice store and forward, and other functions.
- NMS OAM (Operations, Administration, and Maintenance) software and related utilities.
- The CX 2000 software package that includes the:
    - CX board plug-in
    - Configuration files
    - CDI service DLLs and libraries that provide the call control functions on CX 2000 boards
    - CX drivers and downloadable firmware

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.*

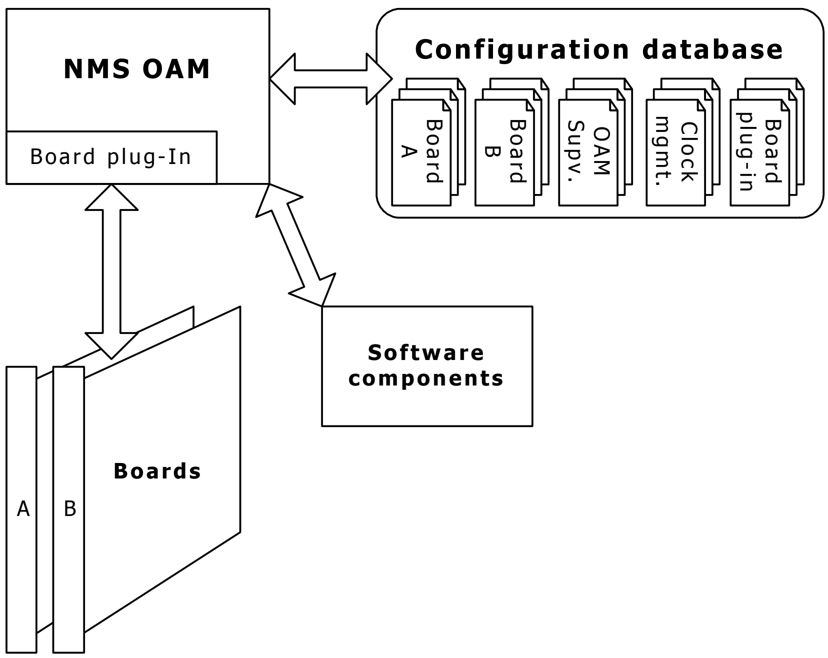
NMS OAM

NMS OAM manages and maintains telephony resources in a system. These resources include hardware components (including CX 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 (boot), stop (shut down), 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.



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

Note: Before using NMS OAM or any related utility, verify that the Natural Access Server (*ctdaemon*) is running. For more information about *ctdaemon*, refer to the *Natural Access Developer's Reference Manual*. For general information about NMS OAM and its utilities, refer to the *NMS OAM System User's Manual*.

CX board plug-in

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

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.

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

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

CDI service

The CX Devices Interface (CDI) service is a Natural Access service that performs low-level station-oriented call control and board management functions for CX boards. These functions include tone generation, DTMF detection, signaling, on-board timer actuation, temperature monitoring, power detection, and station module detection.

CX driver software

The following drivers are installed with Natural Access for operating CX 2000 boards:

Operating system	Driver names
Windows	*cxddrv.sys*
UNIX	*cx* *cxsw*
Red Hat Linux	*cx.o* *cxsw.o*

Installation summary

The following table summarizes the steps required to install CX 2000 hardware and software components:

Step	Description
1	Ensure that your PC system meets the *system requirements* on page 17.
2	Install the board and connect it to station telephones.
3	Connect a power supply. Refer to the Connecting a power supply section.
4	Install Natural Access. Refer to the Natural Access installation booklet for more information.
5	Configure the system.
6	Verify that your installation is operational.

4 Installing a CX 2000 board

System requirements

To install and use CX 2000 boards, your system must have:

- An available PCI bus slot.
- The PCI version 2.2 compliant bus and BIOS.
- Natural Access installed.
- An uninterruptable power supply (UPS). Although a UPS is not strictly required, it is strongly recommended for increased system reliability. The UPS does not need to power the PC video monitor except in areas prone to severe lightning storms.
- An H.100 bus cable if you are connecting to any other H.100 boards.
- A grounded chassis with a three-prong power cord.
- Adequate cooling for the chassis. Refer to *Selecting a PCI chassis* on page 17 for more information.
- A power supply. For more information, refer to *Using the NMS rack mount power supply chassis* on page 25 or Using an alternative power supply.

Caution:	Each CX board is shipped in a protective anti-static container. Leave the board in its original container until you are ready to install it. Handle the board carefully and hold it only by its handles. We recommend that you wear an anti-static wrist strap connected to a good earth ground whenever you handle the board.

Selecting a PCI chassis

Use the following guidelines when choosing a chassis for the CX 2000 board:

- CX 2000 boards must be oriented vertically on the backplane to aid convection cooling. Avoid using a PC tower if you have more than two CX 2000 boards.
- In a large system (five or more slots) use at least one fan for every four slots. Use fans with a minimum rating of 40 cubic feet per minute (CFM) for blowing or drawing air lengthwise along the boards.
- In a smaller system (four or fewer slots) use fans that total at least 100 CFM for blowing or drawing air lengthwise along the boards.

Each chassis is different, and cooling is affected by such factors as:

- The distance between the fans on the boards
- The total volume of the chassis
- The pressure differential between the inside and outside of the chassis

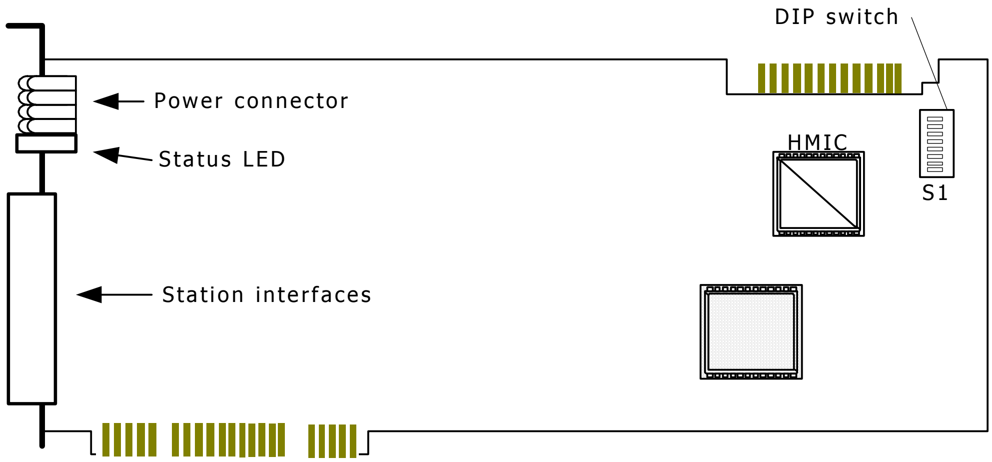
These guidelines are for a typical application. In some cases, more airflow may be necessary to ensure the board is operating at an acceptable temperature.

If you install an uninterrupted power supply, and use it to back up the NMS rack mount power supply (described in *Using the NMS rack mount power supply chassis* on page 25), it should be rated for a minimum of 1.8 kW.

Warning:	This product will not boot in a PC chassis that does not conform to PCI specification version 2.2. If a PC was made before 1999, it probably does not conform to this specification.

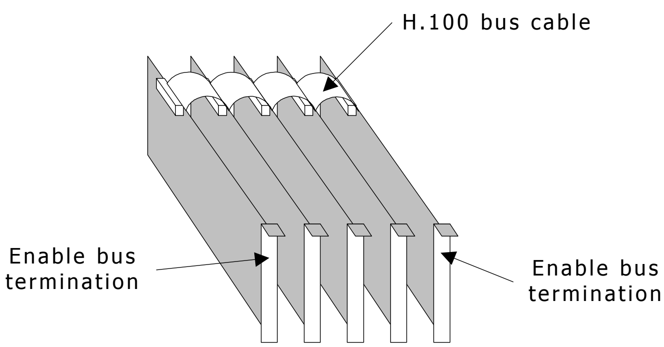
Board components

The following illustration shows where various components are located on a CX 2000 board:



Terminating the H.100 bus

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:



DIP switch S1 controls the H.100 bus termination. The DIP switch is located on the component side of the CX 2000 board. By default, all switches are set to OFF (H.100 bus termination disabled). Setting all S1 switches to ON enables H.100 bus termination. Set all S1 switches to ON for the boards that are on the ends of the H.100 bus.

Installing the hardware

To install a CX 2000 board:

1. If necessary, configure bus termination as described in *Terminating the H.100 bus* on page 18.
2. Turn off the computer and disconnect it from the power source.
3. Remove the cover and set it aside.
4. 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.
    - 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.

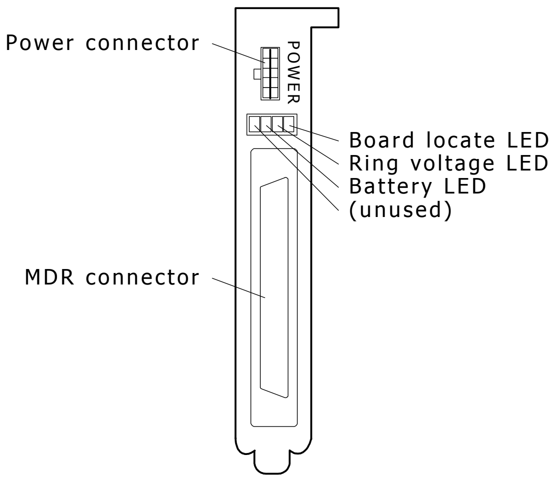
5. Arrange the CX 2000 board and other H.100 boards in adjacent PCI bus slots.
6. Make sure each board's PCI bus connector is seated securely in a slot.
7. Secure the end bracket on the CX 2000 board to the PC.
8. Connect the H.100 bus cable to the CX 2000 board.
9. If you have multiple H.100 boards, connect the H.100 bus cable to each of the H.100 boards.
10. Replace the cover, and connect the computer to its power source.
11. Install Natural Access as described in the Natural Access installation booklet.
12. Connect station telephones to the board as described in *Connecting to station telephones* on page 20.
13. Connect a power supply to the board as described in *Using the NMS rack mount power supply chassis* on page 25 or *Using an alternative power supply* on page 29.

Connecting to station telephones

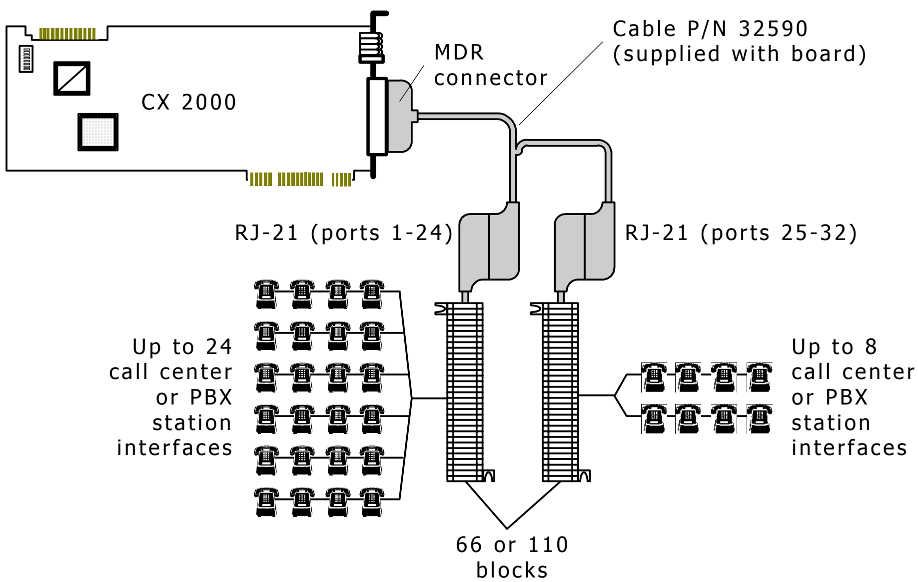
This topic provides information for connecting telephones to the CX 2000 board.

The CX 2000 board can connect to local telephones through up to 2000 feet of cable. Lines from local telephones to the CX 2000 board cannot run outside the building.

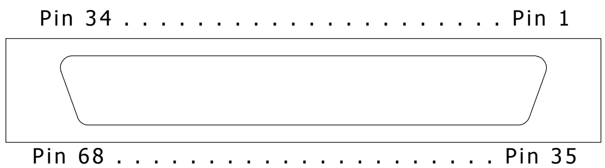
The station interface connector on the CX 2000 is a single MDR 68 pin connector on the end bracket (shown in the following illustration):



The CX 2000 board ships with one 3-foot cable (NMS P/N 32590) with an MDR 68 connector on one end and two RJ-21 connectors on the other. The stations are connected to the RJ-21 connectors using 66 or 110 blocks, as shown in the following illustration:



The following illustration shows the pin locations for each RJ-21 connector on the cable:



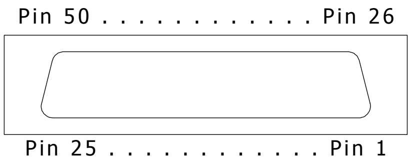
Pinouts for MDR-68 connector on CX 2000 board

The following table shows the pinouts for the MDR 68 connector:

Station	Ring pin	Tip pin	Station	Ring pin	Tip pin
1	2	3	17	36	37
2	4	5	18	38	39
3	6	7	19	40	41
4	8	9	20	42	43
5	10	11	21	44	45
6	12	13	22	46	47
7	14	15	23	48	49
8	16	17	24	50	51
9	18	19	25	52	53
10	20	21	26	54	55
11	22	23	27	56	57
12	24	25	28	58	59
13	26	27	29	60	61
14	28	29	30	62	63
15	30	31	31	64	65
16	32	33	32	66	67

Note: Pins 1 and 68 are not used.

The following illustration shows the pin locations for each RJ-21 connector on the cable:



The following table lists the pinouts for the first RJ-21 connector on the cable:

Station	Ring pin	Tip pin	Station	Ring pin	Tip pin
1	1	26	13	13	38
2	2	27	14	14	39
3	3	28	15	15	40
4	4	29	16	16	41
5	5	30	17	17	42
6	6	31	18	18	43
7	7	32	19	19	44
8	8	33	20	20	45
9	9	34	21	21	46
10	10	35	22	22	47
11	11	36	23	23	48
12	12	37	24	24	49

Note: Pins 25 and 50 are not used on this connector.

The following table lists the pinouts for the second RJ-21 connector on the cable:

Station	Ring pin	Tip pin
25	1	26
26	2	27
27	3	28
28	4	29
29	5	30
30	6	31
31	7	32
32	8	33

Note: Pins 9 - 25 and 34 - 50 are not used on this connector.

Developer's cable kit

NMS provides an optional developer's cable kit. The kit contains two 10-foot RJ-21 cables and two breakout boxes. Each breakout box connects one RJ-21 to 24 standard RJ-11 (POTS) jacks for individual telephones. Use the cables to connect to the breakout boxes or to standard 66 or 110 blocks.

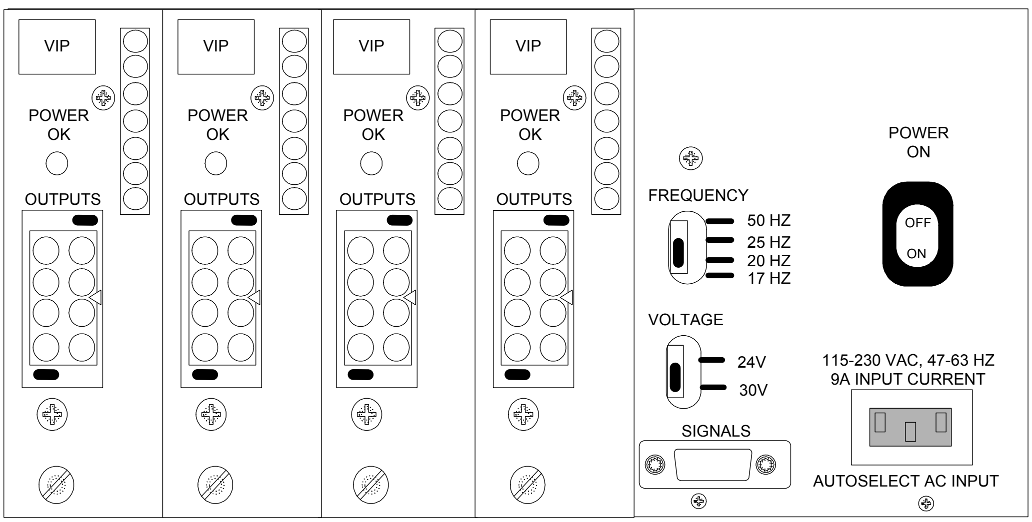
All components of the developer's cable kit sold by NMS are also commercially available from telephone product distributors such as Graybar and Anixter. These distributors can provide variations in cable lengths.

5 Connecting a power supply

Using the NMS rack mount power supply chassis

To supply talk battery power to the station telephones and to power ringing (if necessary), an external power supply is required.

NMS supplies a rack mount power supply chassis that can contain up to four interchangeable supply modules. Each module can power up to two CX 2000 boards. Four modules produce a total combined output of 8.8A for -48 V and -30V/-24 V. The ring output total is 0.68A. The supply outputs are isolated from ground and rely on the CX 2000 board to ground the return line. This provides the best EMI performance. The following illustration shows a rack mount power supply chassis with four modules:



The power supply autoranges for global power standards and can be configured for local ring frequency standards to satisfy global deployment requirements.

Normal configuration

The following table indicates the required number of power supply chassis and modules based upon the number of CX 2000 boards in your system. The table assumes a normal configuration, in which all stations are active on each board. Sufficient ring signal is supplied so that for short (not continuous) peak demand periods, more than 20 telephones rated at 1.0 REN can ring simultaneously.

Number of CX boards	Power supply chassis required (Each chassis includes one power supply module)	Expansion modules required
1	1	0
2	1	0
3	1	1
4	1	1
5	1	2
6	1	2
7	1	3
8	1	3

Redundant power supply configuration

To provide redundancy, or to supply additional ring power to your system, install one more power supply module then you need. The module-to-board connectors on all modules are wired in parallel, so if one module fails, another module supplies power to the first module's board connector. This helps ensure uninterrupted power to any connected boards in the unlikely event that a module fails.

If you connect the power supply to a UPS, the contribution of a fully populated power supply chassis is 1.8 kW.

The following table indicates the required number of power supply chassis and modules in a configuration in which an extra power supply module is installed:

Number of CX boards	Power supply chassis required (Each chassis includes one power supply module)	Expansion modules required
1	1	1
2	1	1
3	1	2
4	1	2
5	1	3
6	1	3
7	N/A	N/A
8	N/A	N/A

In a system containing seven or eight CX boards, there is a maximum of four modules per chassis.

Rack mount considerations

Consider the following items when installing a power supply in a rack:

- Do not block the power supply vents, or otherwise restrict airflow when installing the unit into a rack.
- Ensure that the rack is properly secured, so the rack is stable and cannot easily tip.
- Ensure that the electrical requirements of the system do not exceed the capacity of the electrical circuit.
- If an uninterrupted power supply is used to back up the rack mount supply, it should be rated for at least 1.8 kW.

Note: In the unlikely event that the power supply current exceeds the current rating, the power supply output clamps to zero to protect the supply. The power supply may need to be turned off momentarily and then turned back on to restore normal operation.

Connecting the NMS power supply

You can connect power supply modules directly to CX 2000 boards.

NMS supplies two cables for these connections:

- Shipped with the module - a cable with a male 8-pin Positronic connector on one end (to connect to the module), and two 10-pin MOLEX mini junior connectors on the other end to connect to the TELCO POWER connectors on CX 2000 boards.
- Can be ordered separately - a cable with a male 8-pin Positronic connector on one end (to connect to the module), and #8 spade lugs on the other end to connect to the chassis telecom power bus.

Connecting directly to boards

To connect the NMS power supply directly to each board:

1. On the power supply chassis, set the VOLTAGE switch to 24 V.
2. On the power supply, set the FREQUENCY switch to a ringing frequency (default = 20 Hz).

   The default ringing frequency setting (20 Hz) operates correctly for most applications. However, you can change this setting if a station does not ring when directed, or to change the sound of the ringer to match that of other devices in the target country or region.

Warning:	Do not change the frequency or voltage while the power supply is operating.

3. Plug the Y end of the cable into the TELCO POWER connectors on the CX 2000 boards.

4. Plug the other end of the cable into the power supply.
5. When you have finished configuring the power supply, plug it into a power source.

Alarm signal connector

The NMS rack mount power supply has a DB9 connector on the rear panel that can be used to indicate an alarm condition. The following table lists the pinouts of this connector:

Pin	Description
1	Chassis ground
2	1.5K resistor to +12 V DC
3	4.7K resistor to +5 V DC
4	Alarm signal output. This is an open collector NPN transistor with the emitter connected to COMMON. The transistor is normally on. It is turned off for an alarm condition. The transistor is rated for 20 V DC and 5 mA. The 4.7K resistor on pin 3 or pin 7 can provide pull-up to +5 V DC.
5	Optional signal
6	+5 V DC @ 3 mA
7	4.7K resistor to +5 V DC
8	COMMON
9	COMMON

Powering up the power supply

To power up the supply, turn on the POWER ON switch located on the rear panel of the unit. When the unit is operating properly, the green POWER ON indicator on the front panel glows. In addition, the POWER ON indicator on each module glows (visible on the rear panel of the unit).

Using an alternative power supply

You can use a power supply other than the NMS power supply. This power supply must provide:

- DC voltage to provide talk battery power to the station telephones.
- AC and DC ring voltage, if your application involves ringing station telephones. The AC voltage provides the ringing power. The DC voltage provides loop current that signals the CX board when the telephone goes on or off hook.

This topic specifies the power supply requirements for different boards and describes how to connect an alternative power supply.

Power supply requirements

The tables in this topic specify power supply requirements for different boards, cable lengths, and resistive loads.

Cables between the power supply and the board must be rated for 2 A per board or greater. Twisted pair cabling is recommended for noise reduction.

Warning:	In the worst case, the ring voltage must not exceed 92 V AC, and the DC voltage must not exceed 52 V DC.

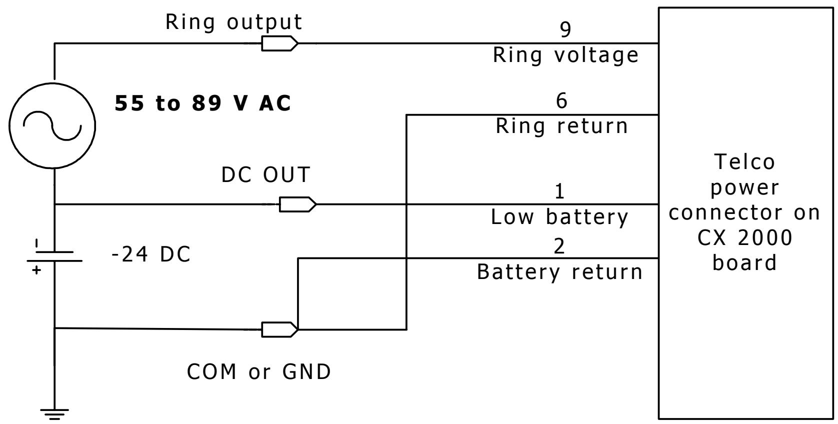
An AG 2000 power supply can be substituted for the rack mount supply for one CX 2000 board. The cable supplied with the AG 2000 power supply will mate with the connector on the board.

CX 2000 power supply requirements

For CX 2000 boards, AC voltage is required only if you are enabling ringing of station telephones.

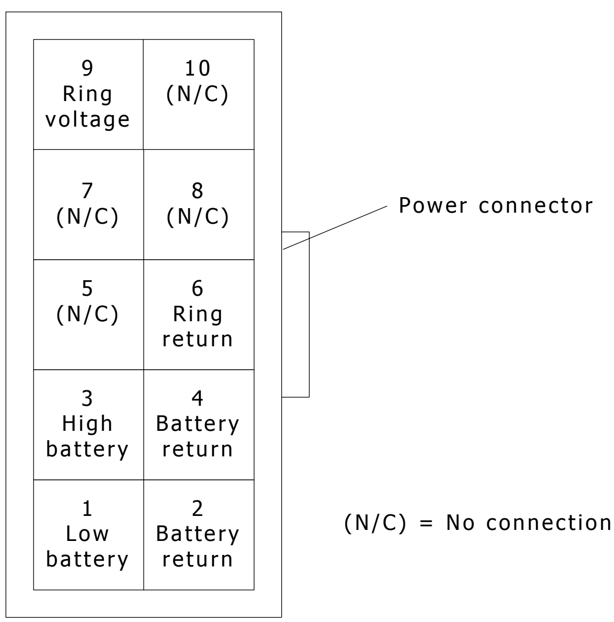
Length of 24 AWG cable	Max resistive load	Recommended output	
		Talk battery	**Ring voltage(only if ringing required)**
0 to 2000 feet	600 Ohms	-24 V DC	55 to 89 V AC and -24 V DC
> 2000 feet	Not supported.		

The ring signal circuitry in the power supply must be equivalent to the following illustration:



Connecting an alternative power supply

Connect the power supply to the TELCO POWER connector on the end bracket of the board. The following illustration shows the power connector pinouts for the CX 2000 board:



The mating connector is Molex 43025-1000 with Molex 43030-0001 or Molex 43030-007 pins.

If only one DC output is available, it must be connected to both the high battery input and the low battery input.

6 Configuring the system

Referencing the CDI manager for Natural Access

For the CDI manager component to be available to the Natural Access server when it boots, the CDI manager must be referenced in the Natural Access configuration file, *cta.cfg*, as shown below:

```
[ctasys]
Service = ncc, adimgr
Service = adi, adimgr
Service = cdi, cdimgr
Service = ais, aismgr
Service = dtm, adimgr
Service = ppx, ppxmgr
Service = swi, swimgr
Service = vce, vcemgr
Service = oam, oammgr
```

For more information about *cta.cfg* and its contents, refer to the *Natural Access Developer's Reference Manual*.

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, Encoding = MuLaw). 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 configuration functions.
oaminfo	Displays keywords and settings for one or more components. Can also set individual keywords.

Applications can use OAM service functions to retrieve and modify configuration parameters. For more information, refer to the *NMS OAM Service Developer's Reference Manual*.

For general documentation of NMS OAM utilities, refer to the *NMS OAM System User's Manual*.

Configuring and starting the system using oamsys

To configure a system using *oamsys*:

Step	Action
1	Install the boards as described in *Installing the hardware* on page 19.
2	Determine which board keyword file you will use, or edit one of the sample CX 2000 board keyword files, to specify appropriate configuration information for each board. For more information, refer to *Using keywords* on page 61.
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. For more information about *pciscan*, refer to the *NMS OAM System User's Manual*.
4	Create a system configuration file, or edit a sample system configuration file, to point to all the board keyword files for your system. Specify a unique name and board number for each board. A sample system configuration file is provided.
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 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 34.

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 about 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 CX 2000 board-specific settings to include in the system configuration file for each board:

Keyword	Description	Allowed values for CX 2000 products
[***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.	CX 2000-16 CX 2000-32 CX_2000
Number	Board number you use in the application to refer to the board.	Any integer from 0 to 31. 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.	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 sent in the order 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 CX 2000 boards:

- Board number 0 is located at bus 0, slot 15. It is assigned a keyword file named *cx-master.cfg*.
- Board number 1 is located at bus 0, slot 16. It is assigned a keyword file named *cx-slave.cfg*.

```
[CX-0]
Product = CX 2000-32
Number  = 0
Bus     = 0
Slot    = 15
File    = c:\nms\cx\cfg\cx-master.cfg


[CX-1]
Product = CX 2000-32
Number  = 1
Bus     = 0
Slot    = 16
File    = c:\nms\cx\cfg\cx-slave.cfg
```

Running oamsys

To run *oamsys*, enter the 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.

Specify parameters in board keyword files as name/value pairs, such as AutoStart = NO.

To change a parameter:

- Use of modify one of the sample board keyword files corresponding to your country and board type. Refer to the *NMS OAM System User's Manual* for information about the syntax of NMS OAM board keyword files.
- Specify parameter settings using the *oamcfg* utility. 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.

A sample board keyword file, cx2000.cfg, is installed by Natural Access. You can copy this file and modify it. The file is located in one of the following paths, depending upon your operating system:

Operating system	Path to sample file
Windows	*\nms\cx\cfg*
UNIX	*/opt/nms/cx/cfg*

The contents of *cx2000.cfg* are shown in the following example. For information about NMS OAM board keyword files, refer to the *NMS OAM System User's Manual.*

```
  #
  #  Standalone operation
  #
Clocking.HBus.ClockMode   = STANDALONE
Clocking.HBus.ClockSource = OSC

  #
  #  Master the CT Bus (drive clock A)
  #
#Clocking.HBus.ClockMode   = MASTER_A
#Clocking.HBus.ClockSource = OSC

  #
  #  Slave to the CT Bus (slave from clock A)
  #
#Clocking.HBus.ClockMode   = SLAVE
#Clocking.HBus.ClockSource = A_CLOCK
```

You can customize additional features:

- Configuring the ring cadence
- Configuring board clocking

Configuring ring cadences

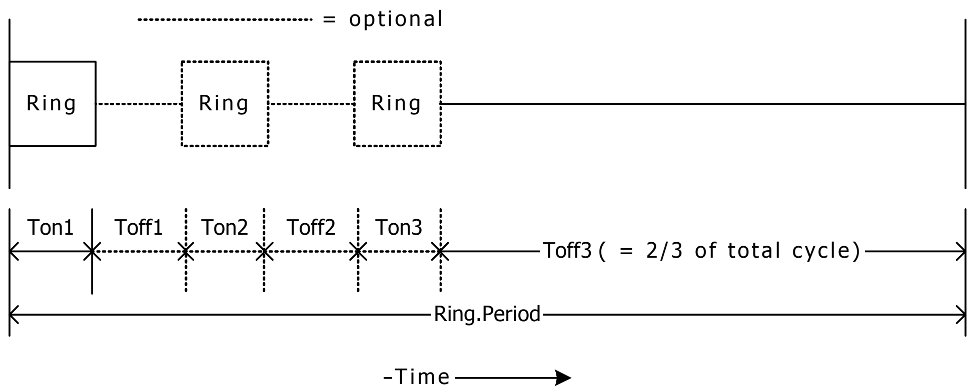
For CX 2000 boards, you can specify up to three different ring patterns (cadences) to use at different times. For example, you can configure one cadence to signify an extension-to-extension call, another cadence to signify an outside call, and another cadence to signify a callback.

Each cadence can have up to three rings per cycle. For example, your first cadence could consist of one 2000 ms ring followed by 4000 ms of silence (like a typical ring tone in the United States). Your second cadence could sound more like the ring tone in the UK (ring ring...ring ring...). Your third cadence could have three rings (ring ring ring...ring ring ring...).

Ring cadencing is controlled with board keywords. Cadencing keywords have default values that specify three distinctive ring cadences. The following keywords determine each cadence:

Keyword	Description
Ring.Cadences[x].Ton1	Determines the length (in ms) of the first ring in the cadence.
Ring.Cadences[x].Toff1	Determines the length (in ms) of the silence between the first and second rings in the cadence.
Ring.Cadences[x].Ton2	Determines the length (in ms) of the second ring in the cadence.
Ring.Cadences[x].Toff2	Determines the length (in ms) of the silence between the second and last rings in the cadence.
Ring.Cadences[x].Ton3	Determines the length (in ms) of the last ring in the cadence.
Ring.Cadences[x].Toff3	Determines the length (in ms) of the silence between the last ring in the cadence and the first ring of the next cadence. This value must be equal to 2/3 of the total length of the cadence.
Ring.Period	Must be set to the total length of the cadence (in ms).

The following illustration shows the role of each keyword in determining a cadence:



You can omit the third ring, or both the second and third rings, by setting their keywords to 0. However, Ring.Cadences[x].Ton1 and Ring.Cadences[x].Toff3 must always be set. Also, Ring.Cadences[x].Toff3 must always equal at least 2/3 of the total length of the cadence. This is so the ring phasing algorithm works correctly.

All cadences must be of the same length. For example, the total length of the following cadences must be the same for each cadence. Set the Ring.Period keyword to this length.

```
  Ring.Cadences[x].Ton1
+ Ring.Cadences[x].Toff1
+ Ring.Cadences[x].Ton2
+ Ring.Cadences[x].Toff2
+ Ring.Cadences[x].Ton3
+ Ring.Cadences[x].Toff3
```

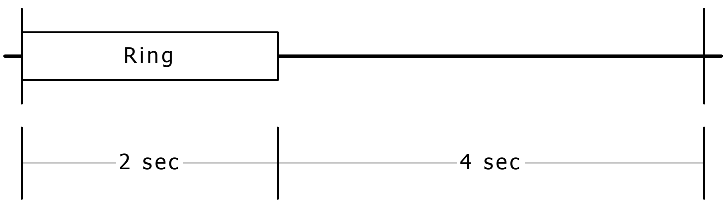
Default ring cadences

Cadencing keywords have default values that specify three distinctive ring cadences. The following table lists the default values for the keywords:

x	Ton1	Toff1	Ton2	Toff2	Ton3	Toff3	Total ms	Ring pattern
0	2000	0	0	0	0	4000	6000	ring...(silence)...
1	600	800	600	0	0	4000	6000	ring...ring...(silence)...
2	400	400	400	400	400	4000	6000	ring...ring...ring...(silence)...

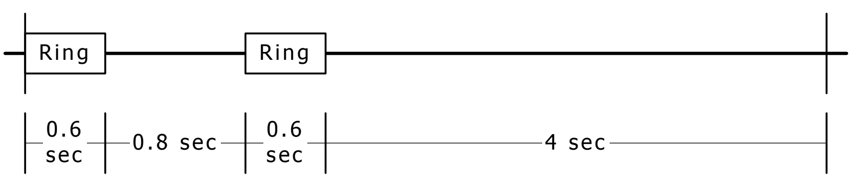
The following illustrations show the three default cadences.

Default cadence (x=0)



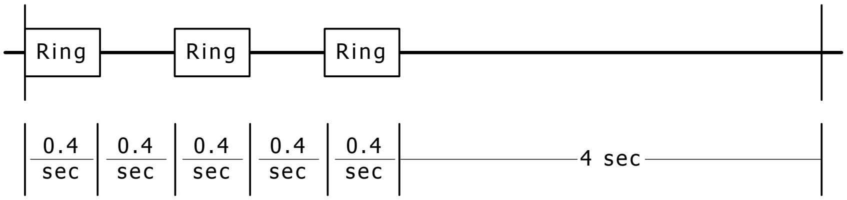

```
Ring.Cadences[0].Ton1  =   2000
Ring.Cadences[0].Toff1 =      0
Ring.Cadences[0].Ton2  =      0
Ring.Cadences[0].Toff2  =     0
Ring.Cadences[0].Ton3  =      0
Ring.Cadences[0].Toff3  =  4000
                        -------
    Ring.Period =          6000
```

Default cadence (x=1)




```
Ring.Cadences[1].Ton1  =    600
Ring.Cadences[1].Toff1 =    800
Ring.Cadences[1].Ton2  =    600
Ring.Cadences[1].Toff2  =     0
Ring.Cadences[1].Ton3  =      0
Ring.Cadences[1].Toff3  =  4000
                        -------
    Ring.Period =          6000
```

Default cadence (x=2)




```
Ring.Cadences[2].Ton1  =    400
Ring.Cadences[2].Toff1 =    400
Ring.Cadences[2].Ton2  =    400
Ring.Cadences[2].Toff2  =   400
Ring.Cadences[2].Ton3  =    400
Ring.Cadences[2].Toff3  =  4000
                          -------
    Ring.Period =          6000
```

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.

This topic describes:

- Clocking capabilities
- Clocking configurations
- Configuring with keywords
- Examples
- Clocking exceptions

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

CX 2000 clocking capabilities

This topic describes the rules and limitations that apply to setting up CT bus clocking on CX 2000 boards.

CX 2000 boards do not have direct access to any external source to derive a timing reference. Thus the NETWORK timing reference is not directly available to these boards. The only timing source available to CX 2000 boards is OSC.

Note: It is also possible to configure a CX 2000 board to use NETREF as a timing reference. However, a simpler solution is to have the board driving NETREF serve as the clock master instead, and eliminate use of these signals.

If another board has access to an outside clock signal, use this board as the clock master. CX 2000 boards are best used as clock masters only if none of the boards on the H.100 bus have any access to an outside digital clock signal (for example, if your system contains only boards with analog trunk interfaces). In this case, the CX 2000 board can drive A_CLOCK or B_CLOCK using its internal oscillator (OSC) as the timing reference. Refer to *Examples* on page 43 for a sample system configuration with one CX 2000 board and two AG 4000 or AG 4040 boards.

When a CX 2000 board is configured as the system primary clock master:

- The board's first timing reference must be set to a NETREF clock or OSC.
- The board's fallback timing reference must be set to a NETREF reference or OSC.

When a CX 2000 board is configured as the system secondary clock master:

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

When a CX 2000 board is configured as a clock slave:

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

Refer to *Other clocking capabilities* on page 40 for more options.

The following tables summarize the CT bus clocking capabilities of the CX 2000 board:

Clocking capabilities as primary master

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

Clocking capabilities as secondary master

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

Clocking capabilities as slave

Capability	Yes/No	Comments
Serve as slave	Yes	
Slave to A_CLOCK	Yes	
Slave to B_CLOCK	Yes	
Available fallback timing references:		
A_CLOCK	Yes	
B_CLOCK	Yes	

Other clocking capabilities

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

Clocking configurations

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

Method	Description
Using *clockdemo* application model	Create an application that assigns each board its clocking mode, monitors clocking changes, and reconfigures clocking if clock fallback occurs. A sample clocking application, *clockdemo*, is provided with Natural Access. *clockdemo* provides a robust fallback scheme that suits most system configurations. *clockdemo* source code is included, allowing you to modify the program if your clocking configuration is complex. For more information about *clockdemo*, refer to the *NMS OAM System User's Manual*. **Note:** Most clocking applications (including *clockdemo*) require all boards on the CT bus to be started in standalone mode.
Using board keywords (with or without application intervention)	For each board on the CT bus, set the board keywords to determine the board's clocking mode and to determine how each board behaves if clock fallback occurs. This method is documented in this topic. Unlike the *clockdemo* application, which allows you to specify several boards to take over mastery of the clock when another board fails, the board keyword method allows you to specify only a single secondary master. For this reason, the board keyword method is best used to implement clock fallback in your system, or in test configurations where clock reliability is not a factor. The board keyword method does not create an autonomous clock timing environment. If you implement clock fallback using this method, an application must still intervene when clock fallback occurs to reset system clocking before other clocking changes occur. If both the primary and secondary clock masters stop driving the clocks, and an application does not intervene, the boards default to standalone mode.

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

Configuring CX 2000 board clocking using keywords

Board keywords enable you to specify the clocking role of each CX 2000 board in a system in the following ways:

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

You can also use board keywords to establish clock fallback sources.

The following tables describe how to use board keywords to specify clocking configurations on multiple-board or multiple-chassis systems. Refer to *Examples* on page 43 for sample configurations.

Configuring the CX 2000 as primary clock master

Use the following board keywords to configure a CX 2000 board as the primary clock master.

Note: A CX 2000 board should not be used as primary or secondary clock master unless no board in the system has access to an external timing reference. Use these settings only if another board has access to an external timing reference, and the CX board must act as clock master. This configuration is not recommended.

Keyword	Description
Clocking.HBus.ClockSource	Specifies the source from which this board derives its timing. Set this keyword to a network source (NETREF or OSC).
Clocking.HBus.ClockMode	Specifies the CT bus clock that the board drives. Set this keyword to either A_CLOCK (MASTER_A) or B_CLOCK (MASTER_B).
Clocking.HBus.AutoFallBack	Enables or disables clock fallback on the board. Set to YES if Clocking.HBus.ClockSource is set to NETREF. Otherwise, set to NO.
Clocking.HBus.FallbackClockSource	Specifies an alternate timing reference to use when the master clock source fails. Set this keyword to a timing source other than the one specified with Clocking.HBus.ClockSource: NETREF or OSC.

Note: If the primary master's first source fails and then returns, the board's timing reference switches back to the first timing source. This is not true for the secondary clock master.

Configuring the CX 2000 as secondary clock master

Use the following board keywords to configure a CX 2000 board as the secondary clock master.

Note: A CX 2000 board should not be used as primary or secondary clock master unless no board in the system has access to an external timing reference. Use these settings only if another board has access to an external timing reference, and the CX board must act as clock master. This configuration is not recommended.

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

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

Configuring the CX 2000 as a clock slave

Use the following board keywords to configure a CX 2000 board as a clock slave:

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

Configuring the CX 2000 as a standalone board

To configure a CX 2000 board in standalone mode so the board references its own clocking information, set Clocking.HBus.ClockMode to STANDALONE. In standalone mode, the board uses only its own oscillator as a timing signal reference. However, the board cannot make switch connections to the CT bus.

Examples

Example 1: System with mixed board types

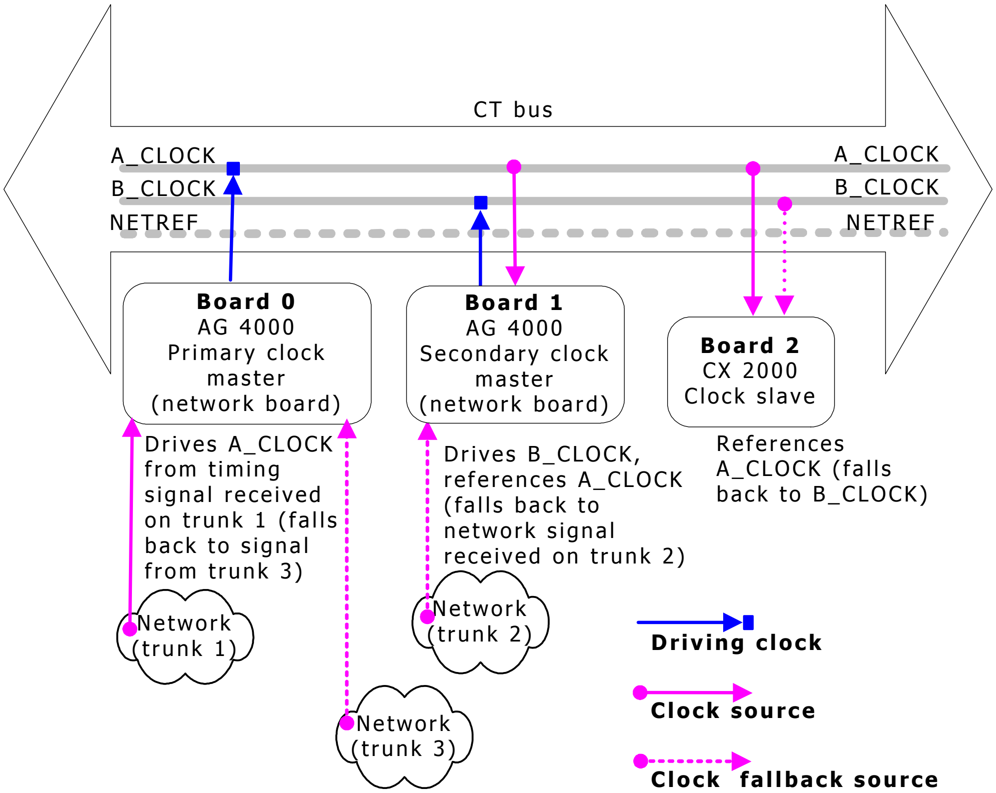
The following example assumes a system configuration in which one CX 2000 board and two AG 4000 or AG 4040 boards reside in a single chassis. The boards are configured in the following way:

Board	Configuration
Board 0	AG 4000 or AG 4040 board. Primary bus master. Drives A_CLOCK, based on signal from network (trunk 1). Falls back to signal from network (trunk 3).
Board 1	AG 4000 or AG 4040 board. Secondary bus master. Drives B_CLOCK, based on signal from A_CLOCK. Falls back to signal from network (trunk 2).
Board 2	CX 2000 board. Clock slave to A_CLOCK (auto-fallback enabled).

This configuration assigns the following clocking priorities:

Priority	Timing reference
First	Board 0, digital trunk 1. A network signal from a digital trunk provides the primary master clock source.
Second	Board 0, digital trunk 3. A network signal from a digital trunk provides the primary master clock source.
Third	Board 1, digital trunk 2. A network signal from a digital trunk provides the secondary master clock fallback source.

The following illustration shows this configuration:



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

Board	**Role**	**Clocking keyword settings**
0	Primary clock master	Clocking.HBus.ClockMode = MASTER_A Clocking.HBus.ClockSource = NETWORK Clocking.HBus.ClockSourceNetwork = 1 Clocking.HBus.AutoFallBack = YES Clocking.HBus.FallBackClockSource = NETWORK Clocking.HBus.FallBackNetwork = 3
1	Secondary clock master	Clocking.HBus.ClockMode = MASTER_B Clocking.HBus.ClockSource = A_CLOCK Clocking.HBus.AutoFallBack = YES Clocking.HBus.FallBackClockSource = NETWORK Clocking.HBus.FallBackNetwork = 2
2	Clock slave	Clocking.HBus.ClockMode = SLAVE Clocking.HBus.ClockSource = A_CLOCK Clocking.HBus.AutoFallBack = YES Clocking.HBus.FallBackClockSource = B_CLOCK

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 first timing reference. Board 0 references its timing from a network timing signal received on its own trunk 1. Board 0 also uses the network timing signal from its own trunk 3 as its clock fallback source. This means that if the network timing signal derived from its own digital trunks fails, Board 0 continues to drive A_CLOCK based on the timing reference from trunk 3.

If, however, both of the signals used by Board 0 fail, Board 0 stops driving A_CLOCK. The secondary master (Board 1) then falls back to a timing reference received on its own trunk 2, and uses this signal to drive B_CLOCK. B_CLOCK then becomes the timing source for all boards that use B_CLOCK as their backup timing reference. The primary master also attempts to slave to B_CLOCK.

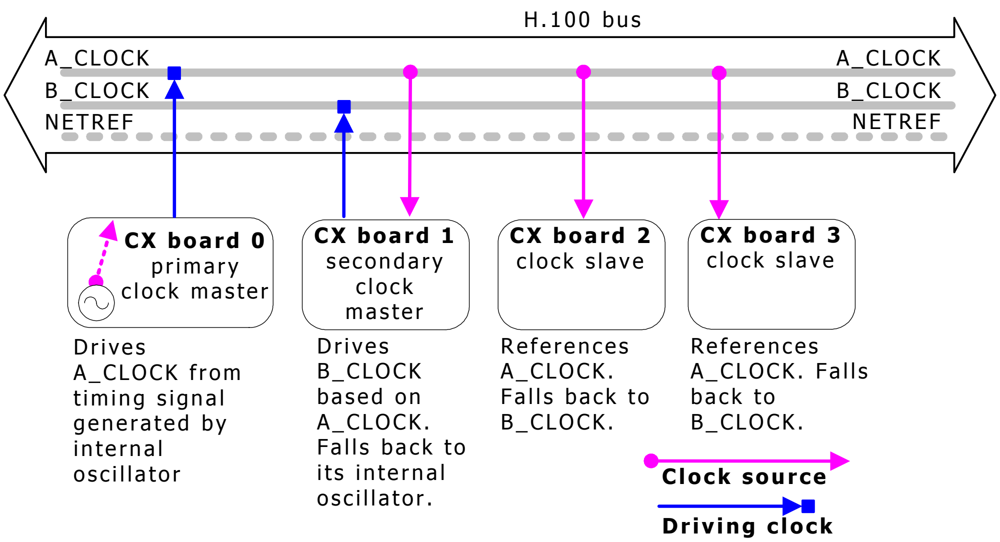
Note: For this clock fallback scheme to work, all the clock slaves must specify A_CLOCK as the clock source, and B_CLOCK as the clock fallback source.

Example 2: System with CX 2000 boards only, CX is master

The following example assumes a system configuration in which four CX 2000 boards reside in a single chassis. The boards are configured in the following way:

Board	Configuration
Board 0	Primary clock master. Drives A_CLOCK, based on signal from internal oscillator. Auto-fallback disabled.
Board 1	Secondary clock master. Drives B_CLOCK, based on signal from A_CLOCK. Falls back to its internal oscillator.
Board 2	Clock slave to A_CLOCK. Falls back to B_CLOCK.
Board 3	Clock slave to A_CLOCK. Falls back to B_CLOCK.

The following illustration shows this configuration:



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

Board	Role	Clocking keyword settings
0	Primary clock master	Clocking.HBus.ClockMode = MASTER_A Clocking.HBus.ClockSource = OSC Clocking.HBus.AutoFallBack = NO
1	Secondary clock master	Clocking.HBus.ClockMode = MASTER_B Clocking.HBus.ClockSource = A_CLOCK Clocking.HBus.AutoFallBack = YES Clocking.HBus.FallBackClockSource = OSC
2	Clock slave	Clocking.HBus.ClockMode = SLAVE Clocking.HBus.ClockSource = A_CLOCK Clocking.HBus.AutoFallBack = YES Clocking.HBus.FallBackClockSource = B_CLOCK
3	Clock slave	Clocking.HBus.ClockMode = SLAVE Clocking.HBus.ClockSource = A_CLOCK Clocking.HBus.AutoFallBack = YES Clocking.HBus.FallBackClockSource = B_CLOCK

In this configuration, Board 0 is the primary master and drives A_CLOCK. All slave boards on the system use A_CLOCK as their first timing reference. Board 0 references its timing from a signal derived from its oscillator. Auto-fallback is disabled for this board.

Board 1 is the secondary master, driving B_CLOCK based on A_CLOCK. If Board 0 stops driving A_CLOCK, Board 1 continues driving B_CLOCK based upon its internal oscillator.

All other boards are slaves to A_CLOCK. If Board 0 stops driving the clock, all boards fall back to B_CLOCK, which is driven by Board 1. If Board 1 stops driving B_CLOCK, all boards fall back to their internal oscillators.

CX 2000 clocking exceptions

Applications can poll clock status with **swiGetBoardClock** periodically to capture snapshots of the board clock status and to detect clocking events, such as the loss of a source. While most boards provide an instantaneous clock status, CX boards provide a latched clock status, which locks in the clock status until it is cleared. When polling the clock status on a CX 2000 board, **swiGetBoardClock** reports a status of BAD on each clock source that experienced an error any time since the last configuration command was issued. To clear the errors and refresh the status information, an application must call **swiConfigBoardClock**. For information about using these functions, refer to the *Switching Service Developer's Manual*.

The sample *swish* script that follows shows a strategy for obtaining the most current clock status:

```
#
#   Obtaining fresh clock status on CX 2000 boards.
#
#   When querying clocks on most boards, the query returns an
#   instantaneous clock status.  CX 2000 is different in that it latches
#   clock errors when they occur.  Errors remain latched until the next
#   configuration command is issued.  In some cases the latched data
#   is stale and fresher status is desired.  This example swish script
#   shows how to use a query-config-query strategy for obtaining fresh
#   status.
#
#   Initialize clocking
#
OpenSwitch b1 = cxsw 1
ConfigBoardH100Clock b1 type=h100 source=h100_a h100mode=slave fallback=enable
fallbacksource=h100_b
#  When polling clock status:
#    Query clocks to obtain current clock configuration, ignoring status
#    Re-issue same clock configuration for purpose of clearing error latches
#    Query clocks to obtain fresh status
#
QueryBoardClock      b1 type=h100
ConfigBoardH100Clock b1 type=h100 source=h100_a h100mode=slave fallback=enable
fallbacksource=h100_b
QueryBoardClock      b1 type=h100
```

Notes on modem connections

The CX 2000 board interface can provide the same grade of connection to high-speed modems (such as V.34 and V.90) as PBXs and telephone office switches. However, the speed of the connections is not guaranteed to be at the highest rates. The following system factors are important in obtaining optimum modem performance:

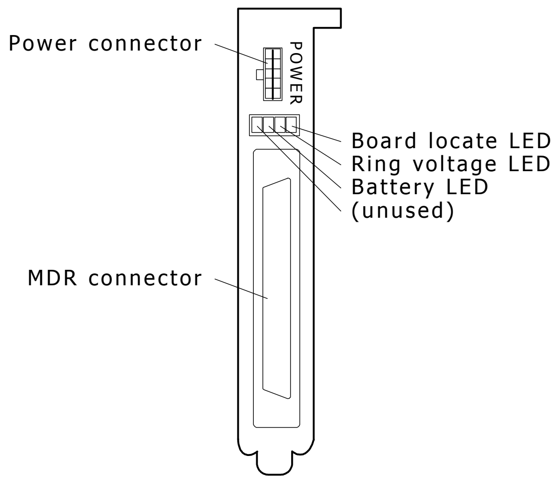
- Cables from the board to the modem must be short, telephone grade twisted pair. Avoid routing cables near noise sources. Avoid moisture in cables.
- There should be only one 2-wire analog loop connection from the modem to the ISP. Also, there should be at most one analog-to-digital conversion in the link from the modem to the ISP. Digital trunks to the public network are preferred for V.34 and are required by V.90 technology.
- Add loss in the uplink connection to speed up the downlink connection if analog trunks are used. This reduces the echo signal.

Even with these precautions, network impairments such as noise, echo, or distortion can continue to limit modem performance, causing slower transfer speeds than desired. These are limitations of the network and modem technologies.

7 Verifying the installation

CX 2000 status indicator LEDs

As shown in the following illustration, the CX 2000 board has LEDs located on the end bracket:



The following table describes each LED:

LED	Description
Board locate	Locates a board using *pciscan*.
Ring voltage	LED on verifies that a ring signal is available to the board.
Battery	LED on verifies -24 V DC is available to the board.

The fourth LED is not used. It is on when the battery LED is on.

Verifying the board installation

To verify that you have installed a CX 2000 board correctly:

1. Install the CX 2000 board, as described in *Installing the hardware* on page 19. For simplicity, ensure that no other telephony boards are driving bus clocks.
2. Install the software. Refer to the Natural Access installation booklet for more information.
3. Connect the power supply to the rear power connector as described in *Using the NMS rack mount power supply chassis* on page 25.
4. Run *pciscan* to determine the location of NMS boards on the system.

   To run *pciscan*, enter:

   ```
   pciscan
   ```

   pciscan displays the PCI bus and PCI slot locations of the boards that are configured in the system.

   To flash an LED on a specific board under Windows, run *pciscan* with the PCI bus and PCI slot locations. For example:

   ```
   pciscan 2 14
   ```

   The Board Locate LED begins flashing. Press any key to stop the flashing LED. For more information about *pciscan*, refer to the *NMS OAM System User's Manual*.
5. Edit the system configuration file to reflect the PCI settings. For information about this file, refer to *Configuring and starting the system using oamsys* on page 32.
6. Configure the target board to operate in standalone mode by driving clocks with the internal oscillator. To do so, add the following keyword statements to the board keyword file:

   ```
   Clocking.HBus.ClockMode = STANDALONE
   Clocking.HBus.ClockSource = OSC
   SwitchConnections = Auto
   ```
7. Attach a telephone to the port for station number 1. Port numbering is 1-based; timeslot numbering is 0-based. To determine the timeslot for a port, subtract 1 from the port number.

   For information on attaching telephones to the board, refer to *Connecting to station telephones* on page 20.
8. Run the *oammon* utility to monitor for board errors and other events.
9. Run *oamsys* to boot the board. *oamsys* interprets the system configuration file and loads the parameters in the keyword files to the boards. *oamsys* searches for configuration files in the AGLOAD path.

   To run *oamsys*, open a command window and enter `oamsys`.

   For information about *oamsys*, refer to the *NMS OAM System User's Manual*.
10. Examine the *oammon* output for errors and other events.

Verifying the board's operation

Once you have verified that the board is properly installed (as described in *Verifying the board installation* on page 50), use the *cditest* utility to check that the board is operating correctly. Using *cditest* and a telephone, you can see off-hook/on-hook events, play dial tone, see DTMF events, ring the telephone and more.

Refer to *Interactive test program: cditest* on page 112 for more information.

Follow this procedure to perform a simple board operation test:

1. Set up the board, and verify that it is working correctly in standalone mode as described in *Verifying the board installation* on page 50.
2. Run the *cditest* utility. *cditest* is found in one of these directories:

Operating system	Path
Windows	*\nms\ctaccess\demos\cditest*
UNIX	*/opt/nms/ctaccess/demos/cditest*

   On the *cditest* command line, specify the address of the DSP port corresponding to the attached telephone's line interface port. For example, if the telephone is attached to port 1 (timeslot 0) on board 0, and the DSP is attached to stream 4, run *cditest* by entering:

   ```
   cditest -b 0 -s 4:0
   ```

3. Type the following commands at the prompt:
   a. Type op to open the port.
   b. Type et to enable talk battery power.
   c. Type eb to start the signaling detector.
   d. Take the phone off-hook. The event CDIEVN_OFF_HOOK is displayed.
   e. Type ed to start the DTMF detector.
   f. Type gn, and press the **Return** key to generate a dial tone.
4. Dial digits on the telephone. As you do so, digit events are displayed as follows:

   ```
   Event: CDIEVN_DTMF_STARTED, digit 1
   Event: CDIEVN_DTMF_ENDED
   Event: CDIEVN_DTMF_STARTED, digit 2
   Event: CDIEVN_DTMF_ENDED
   Event: CDIEVN_DTMF_STARTED, digit 3
   Event: CDIEVN_DTMF_ENDED
   ```

5. Place the phone on-hook. The event CDIEVN_ON_HOOK is displayed.
6. Type sr to start ringing the phone. The phone rings.
7. Type ar to stop ringing the phone.
8. Type cp to close the port.
9. Type q to quit cditest.

Verifying the board's operating temperature

The CX Devices Interface (CDI) service provides API functions for temperature monitoring on CX 2000 boards. Refer to the *CDI Service Developer's Reference Manual* for information about these functions.

Readings should be taken after running under a typical load (with a number of stations off-hook) for one hour. The following table indicates the maximum safe operating temperatures for various environments:

On-board temperature sensor ID	Maximum temperature reading in temperature controlled laboratory environment	Maximum field operating temperature
0	65° C	90° C
1	65° C	90° C
2	60° C	90° C
3	60° C	90° C
4	60° C	90° C

Exceeding these readings will cause warnings of overheating. Reduce the temperature in one of the following ways:

- Clean the chassis air filters.
- Replace a failed or underrated fan.
- Replace the chassis with one that provides more air flow. For chassis recommendations, refer to *Selecting a PCI chassis* on page 17.
- Improve room temperature controls.

CX boards that operate beyond the maximum field operating temperatures may exhibit one or more of the following symptoms:

- Events are sent to the application to warn of overheating. For more information about these events, refer to the *CDI Service Developer's Reference Manual*.
- New calls receive a strange tone in place of the dial tone.
- The loop current may be reduced. This reduction in current may impact the operation of telephones or other attached devices.

8 Implementing switching

CX 2000 switch model

This topic describes:

- The specific use of each stream, as shown for H.100 streams and local streams
- An illustration of the CX 2000 switch model
- Lucent T8100A switch blocking

H.100 streams

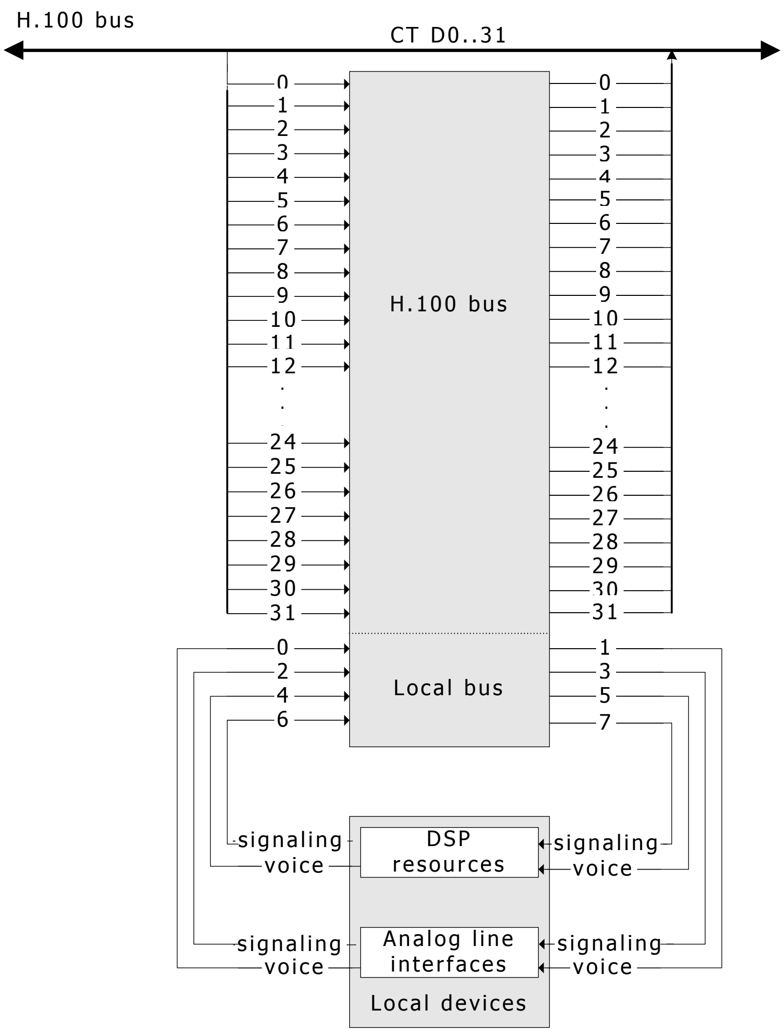
H.100 streams	
H.100 Bus	Streams 0..31, timeslots 0..127 (Streams clocked at 8 MHz)

Local streams

Local streams	
Station voice information	Stations 0 - 47: Streams 0 and 1, timeslots 0..47 for 48 ports Stations 0 - 31: Streams 0 and 1, timeslots 0..31 for 32 ports
Station signaling information	Stations 0 - 47: Streams 2 and 3, timeslots 0..47 for 48 ports Stations 0 - 31: Streams 2 and 3, timeslots 0..31 for 32 ports
DSP voice information	Streams 4 and 5, timeslots 0..47 for 48 ports Streams 4 and 5, timeslots 0..31 for 32 ports
DSP signaling information	Streams 6 and 7, timeslots 0..47 for 48 ports Streams 6 and 7, timeslots 0..31 for 32 ports

Switch model

The following illustration shows the CX 2000 switch model:



Lucent T8100A switch blocking

Switching on the CX 2000 board is implemented by the Lucent T8100A chip (HMIC). The Lucent T8100A 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 the:

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

Default connections for a standalone board

For a standalone CX 2000 board, disable H.100 connectivity in the configuration file (Clocking.HBus.ClockMode = DISABLE). In this case, default connections are made on the board to connect voice and signaling information to DSP resources.

Station type	Setting
Full duplex voice station	Local:0:0..47 => Local:5:0..47, Local:4:0..47 => Local:1:0..47 for 48 ports Local:0:0..31 => Local:5:0..31, Local:4:0..31 => Local:1:0..31 for 32 ports
Full duplex signaling station	Local:2:0..47 => Local:7:0..47, Local:6:0..47 => Local:3:0..47 for 48 ports Local:2:0..31 => Local:7:0..31, Local:6:0..31 => Local:3:0..31 for 32 ports

Using the Switching service

To use the Natural Access Switching service (SWI) with CX 2000 boards, applications must create a context and open the Switching service on that context. Since switching is a board-level function, applications typically open the Switching service on a non-DSP port, such as 0:0.

Refer to the *Natural Access Developer's Reference Manual* and the *Switching Service Developer's Reference Manual* for additional information and examples of opening services.

Opening the switch

After opening the Switching service, applications can open the switch block on the board to obtain a switch handle for further Switching service calls. To open the switch block on a board, specify the switching driver name in the call to **swiOpenSwitch**. For CX 2000 boards, the driver name is *cxsw*. The following example shows how to use *cxsw* in an application:

```
//Open the switchblock for the board using the proper driver
dwRetValue = swiOpenSwitch(hContext,
                           "cxsw",
                           BoardNumber,
                           0x0,
                           &hSwitch);
```

Configuring local devices

Local device configuration on CX 2000 boards is controlled by the Switching service. The Switching service provides generic API functions for accessing device configuration parameters defined by the underlying hardware and device driver.

Applications can use **swiConfigLocalTimeslot** and **swiGetLocalTimeslotInfo** to configure a device on a given local stream and timeslot by specifying a particular parameter and providing a data structure specific to that parameter. For more information about these functions, refer to the *Switching Service Developer's Reference Manual*.

Accessing the line gain

CX 2000 boards support input and output gain configurations on network voice ports (timeslots) from -6 dB to +6 dB in one dB increments.

Input gain is applied to the signal received from the network. Output gain is applied to the signal transmitted to the network. The default value for both input line gain and output line gain on CX 2000 boards is nominally 0 dB.

Caution:	Increasing gain can also increase noise, echo, degrade DTMF detection, and possibly cause oscillations on the telephone network. There also may be regulatory authority implications. Use gain with caution.

Decreasing gain may reduce echo and other noise.

This topic describes:

- Getting the line gain
- Setting the line gain

Getting the line gain

Use **swiGetLocalTimeslotInfo** to query the input or output line gain. Set the arguments for this function as follows:

Argument	Field	Value
swihd		Handle returned by **swiOpenSwitch**.
args	localstream	0 or 1. Refer to the *CX 2000 switch model* on page 53.
	localtimeslot	0..47. Refer to the *CX 2000 switch model* on page 53.
	deviceid	MVIP95_ANALOG_LINE_DEVICE
	parameterid	MVIP95_INPUT_GAIN or MVIP95_OUTPUT_GAIN
buffer		Points to the NMS_LINE_GAIN_PARMS structure.
size		Size of buffer, in bytes.

The NMS_LINE_GAIN_PARMS structure is:

```
typedef struct
{
     INT32 gain;

} NMS_LINE_GAIN_PARMS;
```

The value returned in the gain component of NMS_LINE_GAIN_PARMS represents the gain in dB multiplied by 1000. For example, if the input gain on a particular network timeslot is currently set to -3 dB, after calling **swiGetLocalTimeslotInfo** for parameter MVIP95_INPUT_GAIN, the gain field is -3000.

The following sample code shows how to retrieve line gain applied to a signal received from the network:

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

DWORD myGetReceiveGain ( SWIHD swihd, SWI_TERMINUS terminus, INT32*
                                                              gain_dB )
{
     SWI_LOCALTIMESLOT_ARGS  args;
     NMS_LINE_GAIN_PARMS     device ;
     DWORD                   rc ;

     args.localstream      = terminus.stream ;
     args.localtimeslot    = terminus.timeslot ;
     args.deviceid         = MVIP95_ANALOG_LINE_DEVICE ;
     args.parameterid      = MVIP95_INPUT_GAIN ;

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

    *gain_dB  =  device.gain / 1000  ;

     return rc ;
}
```

The following sample code shows how to retrieve line gain applied to a signal transmitted to the network:

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

DWORD myGetTransmitGain ( SWIHD swihd, SWI_TERMINUS terminus,
                                                     INT32* gain_dB )
{
     SWI_LOCALTIMESLOT_ARGS  args;
     NMS_LINE_GAIN_PARMS     device ;
     DWORD                   rc ;

     args.localstream      = terminus.stream ;
     args.localtimeslot    = terminus.timeslot ;
     args.deviceid         = MVIP95_ANALOG_LINE_DEVICE ;
     args.parameterid      = MVIP95_OUTPUT_GAIN ;


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

    *gain_dB  =  device.gain / 1000  ;

     return rc ;

}
```

Setting the line gain

Use **swiConfigLocalTimeslot** to set the input or output line gain. Set the arguments for this function as follows:

Argument	Field	Value
swihd		Handle returned by **swiOpenSwitch**.
args	localstream	0 or 1. Refer to the *CX 2000 switch model* on page 53.
	localtimeslot	0..47 (maximum 31 in 32 station models). Refer to the *CX 2000 switch model* on page 53.
	deviceid	MVIP95_ANALOG_LINE_DEVICE
	parameterid	MVIP95_INPUT_GAIN or MVIP95_OUTPUT_GAIN
buffer		Points to the NMS_LINE_GAIN_PARMS structure.
size		Size of buffer, in bytes.

The NMS_LINE_GAIN_PARMS structure is:

```
typedef struct
{
     INT32 gain;

} NMS_LINE_GAIN_PARMS;
```

Multiply the desired gain setting in dB by 1000. For example, to set the input line gain on a network voice port to -4 dB, set the gain field of NMS_LINE_GAIN_PARMS to -4000.

The following sample code shows how to configure gain applied to a signal received from the network:

```
#include "swidef.h"  /*  Natural Access Switching service           */
#include "mvip95.h"  /*  MVIP-95 definitions                        */
#include "nmshw.h"   /*  NMS hardware-specific definitions          */
*/
DWORD mySetReceiveGain ( SWIHD swihd, SWI_TERMINUS terminus, INT32 gain_dB )
{
     SWI_LOCALTIMESLOT_ARGS  args;
     NMS_LINE_GAIN_PARMS     device ;

     args.localstream      = terminus.stream ;
     args.localtimeslot    = terminus.timeslot ;
     args.deviceid         = MVIP95_ANALOG_LINE_DEVICE ;
     args.parameterid      = MVIP95_INPUT_GAIN ;

     device.gain  =  gain_dB * 1000  ;

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

}
```

The following sample code shows how to configure line gain applied to a signal transmitted to the network:

```
#include "swidef.h"  /*  Natural Access Switching service           */
#include "mvip95.h"  /*  MVIP-95 definitions                        */
#include "nmshw.h"   /*  NMS hardware-specific definitions          */
*/
DWORD mySetTransmitGain ( SWIHD swihd, SWI_TERMINUS terminus, INT32 gain_dB )
{
     SWI_LOCALTIMESLOT_ARGS  args;
     NMS_LINE_GAIN_PARMS     device ;

     args.localstream      = terminus.stream ;
     args.localtimeslot    = terminus.timeslot ;
     args.deviceid         = MVIP95_ANALOG_LINE_DEVICE ;
     args.parameterid      = MVIP95_OUTPUT_GAIN ;

     device.gain  =  gain_dB * 1000  ;

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

}
```

9 Keyword summary

Using keywords

The keywords for a CX 2000 board describe that board's configuration. Some keywords are read/write and 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.*

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. 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 board keyword file syntax.

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

- Create a new board keyword file, either with additional keywords or with 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 CX plug-in level keywords, specify the CX plug-in name (*cx.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. From 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 CX plug-in level keywords, specify the CX plug-in name on the command line:

  ```
  oaminfo -n cx.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 you can change:

To...	Use these keywords...
Specify whether the board is started or stopped automatically	AutoStart AutoStop
Specify information about the board	Encoding Location.PCI.Bus Location.PCI.Slot Name Number
Set up clocking information	Clocking.HBus.AutoFallBack Clocking.HBus.ClockMode Clocking.HBus.ClockSource Clocking.HBus.ClockSourceNetwork Clocking.HBus.FallbackClockSource Clocking.HBus.NetRefSource Clocking.HBus.NetRefSpeed Clocking.HBus.SClockSpeed Clocking.HBus.Segment Clocking.Type
Configure ring cadences	Ring.Cadences[x].Ton1 Ring.Cadences[x].Toff1 Ring.Cadences[x].Ton2 Ring.Cadences[x].Toff2 Ring.Cadences[x].Ton3 Ring.Cadences[x].Toff3 Ring.Period
Configure switching	SwitchConnections SwitchDriver.Name
Configure debugging information	DebugMask
Specify files to download to the board	DefaultQslacFile DSPFile
Configure the DSP	DSP.Image
Enable or disable power to station telephones	ExternalRingerEnable HighBatteryEnable LowBatteryEnable RingVoltageEnable SignalingLoopbackEnable

Informational keywords

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

- Board
- EEPROM

Retrieving board information

Keyword	Type	Description
Location.Type	String	Bus type.
State	String	State of the physical board.
Driver.Name	String	Operating system independent root name of the driver.
Product	String	Product type of the CX board.

Retrieving EEPROM information

Keyword	Type	Description
Eeprom.AssemblyRevision	Integer	Hardware assembly level.
Eeprom.Family	Integer	Board family.
Eeprom.MFGWeek	Integer	Week of the last full test.
Eeprom.MFGYear	Integer	Year of the last full test.
Eeprom.SerialNum	Integer	Serial number unique to each board. This number is factory configured.
Eeprom.SoftwareCompatibility	Integer	Minimum software revision level.
Eeprom.TestLevel	Integer	Test level of the EEPROM.
Eeprom.TestLevelRev	Integer	Test level revision of the EEPROM.

Plug-in keywords

The CX plug-in keywords include:

- Boards[x]
- BootDiagnosticLevel
- DetectedBoards[x]
- Products[x]
- Version.Major
- Version.Minor

10 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 or integer
Default	Default value
Allowed values	A list of all possible values
Example	An example of usage
Description	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 = ***argument***

Access

Read/Write

Type

String

Default

NO

Allowed values

YES | NO

Example

```
AutoStart = NO
```

Details

The Supervisor 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 details, 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 = ***argument***

Access

Read/Write

Type

String

Default

NO

Allowed values

YES | NO

Example

```
AutoStart = NO
```

Details

The Supervisor 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 details, refer to the *NMS OAM System User's Manual*.

See also

AutoStart

Boards[x]

Contains a list of all boards managed by the plug-in (the list of all CX 2000 boards that have managed objects in the NMS OAM database).

Syntax

Boards[***x***] = ***board_name***

Access

Read-only (plug-in)

Type

String

Allowed values

Any valid board name.

Details

The NMS OAM supervisor managed object also contains a Boards[x] array keyword. All values in each plug-in Boards[x] array keyword are added to the keyword at the Supervisor level. You can retrieve the values in the Boards[x] array keyword at the Supervisor level to determine the names of boards currently managed by NMS OAM.

You can retrieve the value of the Supervisor Boards.Count keyword to determine the number of items in the Supervisor Boards[x] array keyword. Retrieve the value of the board plugin Boards.Count keyword to determine the number of items in the plugin Boards[x] array keyword.

For details, refer to the *NMS OAM Service Developer's Reference Manual*.

BootDiagnosticLevel

Specifies the level of diagnostics performed during initialization of the board. When disabled (set to 0) the board ignores any diagnostic errors returned while it is being initialized.

Syntax

BootDiagnosticLevel = ***level***

Access

Read/Write (plug-in level)

Type

Integer

Default

1

Allowed values

-65535 to 65535

Example

```
BootDiagnosticLevel = 1
```

Details

The valid values for ***level*** are 0, and 1. Zero (0) indicates that no diagnostics are performed, and 1 is the maximum level.

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

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

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

Clocking.HBus.AutoFallBack

Enables or disables clock fallback on the board. This keyword specifies whether or not the board automatically switches to a secondary timing reference if its primary timing reference fails.

For information about setting up CT bus clocking, and rules and restrictions for configuring CT bus clocking, refer to *Configuring board clocking* on page 38.

Syntax

Clocking.HBus.AutoFallBack = ***argument***

Access

Read/Write

Type

String

Default

NO

Allowed values

YES | NO

Example

```
Clocking.HBus.AutoFallBack = NO
```

Details

The primary timing reference is specified by the Clocking.HBus.ClockSource keyword. The secondary timing reference is specified by the Clocking.HBus.FallbackClockSource keyword.

Note: Use the *swish* command **queryBoardClock** to determine what timing reference the board is actively using.

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

See also

Clocking.HBus.ClockMode, Clocking.HBus.NetRefSource

Clocking.HBus.ClockMode

Specifies whether the board is a clock master driving A_CLOCK or B_CLOCK, or is a clock slave driven by one of these clocks.

For information about setting up CT bus clocking, and rules and restrictions for configuring CT bus clocking, refer to *Configuring board clocking* on page 38.

Syntax

Clocking.HBus.ClockMode = ***setting***

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 this keyword include:

Value	Description
MASTER_A	The board is a clock master that drives A_CLOCK.
MASTER_B	The board is a clock master that drives B_CLOCK.
SLAVE	The board is a clock slave that derives its timing from the primary bus master.
STANDALONE	The board does not drive any CT bus clocks. Connections are not allowed to the board's CT bus timeslots in standalone mode. For more information about this mode, refer to *CX 2000 clocking capabilities* on page 38.

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

See also

Clocking.HBus.AutoFallBack, Clocking.HBus.ClockSource, Clocking.HBus.FallbackClockSource, Clocking.HBus.NetRefSource

Clocking.HBus.ClockSource

Specifies the primary timing reference for the board.

For information about setting up CT bus clocking, and rules and restrictions for configuring CT bus clocking, refer to *Configuring board clocking* on page 38.

Syntax

Clocking.HBus.ClockSource = ***argument***

Access

Read/Write

Type

String

Default

OSC

Allowed values

OSC | A_CLOCK | B_CLOCK | NETREF

Example

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

Details

Valid entries for this keyword are:

Value	Description
OSC	Valid only if the board is the primary clock master or in standalone mode. OSC causes the board to drive the bus clock using the signal from its on-board oscillator. Use this setting only when no external timing reference is available. The on-board oscillator is accurate to 32 ppm (parts per million) and meets the requirements for a Stratum 4E clock.
A_CLOCK	Valid only if the board is a clock slave or secondary master. This setting causes the board to act as a slave to A_CLOCK.
B_CLOCK	Valid only if the board is a clock slave or secondary master. This setting causes the board to act as a slave to B_CLOCK.
NETREF	Valid only if the board is the primary clock master. NETREF causes the board to drive the bus clock using a signal from the NETREF carrier on the CT bus. Another source is driving NETREF. This source is specified using the Clocking.HBus.NetRefSource keyword.

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

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

See also

Clocking.HBus.AutoFallBack, Clocking.HBus.ClockMode, Clocking.HBus.FallbackClockSource

Clocking.HBus.ClockSourceNetwork

Specifies the number of the trunk that the board uses as its external network timing reference for its internal 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 38.

Syntax

Clocking.HBus.ClockSourceNetwork = ***networknum***

Access

Read/Write

Type

Integer

Default

0

Allowed values

0

Example

```
Clocking.HBus.ClockSourceNetwork = 0
```

Details

Since CX 2000 boards do not have digital trunks, this keyword is always set to 0.

See also

Clocking.HBus.ClockSource

Clocking.HBus.FallbackClockSource

Specifies the secondary clock reference to use when the primary clock reference fails.

For information about setting up CT bus clocking, and rules and restrictions for configuring CT bus clocking, refer to *Configuring board clocking* on page 38.

Syntax

Clocking.HBus.FallbackClockSource = ***argument***

Access

Read/Write

Type

String

Default

OSC

Allowed values

OSC | A_CLOCK | B_CLOCK | NETREF

Example

```
Clocking.HBus.FallBackClockSource = OSC
```

Details

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

Valid entries for this keyword are:

Value	Description
OSC	Valid only if the board is a clock master. OSC causes the board to use its on-board oscillator as its secondary timing reference. Use this setting only when no external timing reference is available. The on-board oscillator is accurate to 32 ppm (parts per million) and meets the requirements for a Stratum 4E clock.
A_CLOCK	Use the setting if the board is a clock slave to B_CLOCK, and a secondary clock master is driving A_CLOCK. This setting causes the board to use A_CLOCK as its secondary timing reference.
B_CLOCK	Use the setting if the board is a clock slave to A_CLOCK, and a secondary clock master is driving B_CLOCK. This setting causes the board to use B_CLOCK as its secondary timing reference.
NETREF	Valid only if the board is a clock master. NETREF causes the board to use the signal from the NETREF carrier on the CT bus as its secondary timing reference. Another source is driving NETREF. This source is specified using the Clocking.HBus.NetRefSource keyword.

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

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

See also

Clocking.HBus.ClockMode, Clocking.HBus.ClockSource

Clocking.HBus.NetRefSource

Specifies a source to drive the NETREF timing signal on the H.100 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 38.

Syntax

Clocking.HBus.NetRefSource = ***argument***

Access

Read/Write

Type

String

Default

STANDALONE

Allowed values

OSC | STANDALONE

Example

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

Details

A CX 2000 board can drive this signal only from its internal oscillator. Use this configuration for development purposes only.

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

See also

Clocking.HBus.AutoFallBack, Clocking.HBus.ClockMode, Clocking.HBus.ClockSource, Clocking.HBus.FallbackClockSource, Clocking.HBus.NetRefSpeed

Clocking.HBus.NetRefSpeed

Specifies 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 38.

Syntax

Clocking.HBus.NetRefSpeed = ***argument***

Access

Read/Write

Type

String

Default

8K

Allowed values

8K | 1544M | 2048M

Example

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

Details

Only 8K is currently supported.

See also

Clocking.HBus.NetRefSource

Clocking.HBus.SClockSpeed

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

For information about setting up CT bus clocking, and rules and restrictions for configuring CT bus clocking, refer to *Configuring board clocking* on page 38.

Syntax

Clocking.HBus.SClockSpeed = ***argument***

Access

Read/Write

Type

String

Default

2M

Allowed values

2M | 4M | 8M

Example

```
Clocking.HBus.SClockSpeed = 2M
```

See also

Clocking.HBus.Segment

Clocking.HBus.Segment

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

For information about setting up CT bus clocking, and rules and restrictions for configuring CT bus clocking, refer to *Configuring board clocking* on page 38.

Syntax

Clocking.HBus.Segment = ***speed***

Access

Read/Write

Type

Integer

Default

1

Allowed values

0 to 65535

Example

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

See also

Clocking.HBus.SClockSpeed

Clocking.Type

Specifies the type of CT bus with which the board is compatible.

For information about setting up CT bus clocking, and rules and restrictions for configuring CT bus clocking, refer to *Configuring board clocking* on page 38.

Syntax

Clocking.Type = ***type***

Access

Read/Write

Type

String

Default

HBus

Allowed values

HBus

Example

```
Clocking.Type = HBus
```

DebugMask

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

Syntax

DebugMask = ***mask***

Access

Read/Write

Type

Integer

Default

0

Allowed values

mask = Any value shown in the following table.

Example

```
DebugMask = 0x00000200
```

Details

You can specify the following DebugMask parameters:

Value	Description
0x00000001	Additional initialization messages.
0x00000002	Legacy initialization messages.
0x00000004	DLM download and start address.
0x00000008	Total resources for each DSP.
0x00000080	DLM resolving and relocation.
0x00000100	Host interface up and down messages.
0x00000200	Inter-manager messages
0x00000400	All manager messages.
0x80000000	Available memory.
0xFFFFFFFF	All of the above.

DebugMask settings takes effect immediately. It is not necessary to reboot the board for these settings to take effect.

DefaultQslacFile

Specifies the QSLAC file.

Syntax

DefaultQslacFile = ***argument***

Access

Read/Write

Type

String

Default

c2allsl6.slc

Allowed values

Any valid file name.

Example

```
DefaultQslacFile = c2allsl6.slc
```

DetectedBoards[x]

Contains a list of all boards detected by the CX board plug-in in response to an invocation of the OAM service function **oamDetectBoards**.

Syntax

DetectedBoards[***x***] = ***board_name***

Access

Read-only (plug-in level)

Type

String

Allowed values

Any valid board name.

Details

The array is empty until this function is called.

Board detection actually takes place at the plug-in level. When **oamDetectBoards** is invoked, the Supervisor directs each installed plug-in to detect all boards in the system of a board type that the plug-in supports. The plug-in creates a name for each board, and adds the name to the plug-in DetectedBoards[x] array keyword.

The NMS OAM supervisor managed object also contains a DetectedBoards[x] array keyword. All values in each plug-in DetectedBoards[x] array keyword are added to the keyword at the Supervisor level. You can retrieve the values in the DetectedBoards[x] array keyword at the Supervisor level to determine the names of all detected boards.

You can retrieve the value of the Supervisor DetectedBoards.Count keyword to determine the number of items in the Supervisor DetectedBoards[x] array keyword. Retrieve the value of the board plug-in DetectedBoards.Count keyword to determine the number of items in the plugin DetectedBoards[x] array keyword.

For details, refer to the *NMS OAM Service Developer's Reference Manual*.

DSPFile

Specifies the name of the file to be loaded into the DSP.

Syntax

DSPFile = ***argument***

Access

Read/Write

Type

String

Default

cx100.dsp

Allowed values

Any valid file name.

Example

```
DSPFile = cx100.dsp
```

DSP.Image

Specifies the digital signal processor (DSP) operating system to use on the DSP.

Syntax

DSP.Image = ***filename***

Access

Read/Write

Type

File name

Default

cx100.dsp

Allowed values

Valid DSP image file name.

Example

```
DSP.Image = cx100.dsp
```

Encoding

Specifies the DSP and CODEC hardware companding mode.

Syntax

Encoding = ***mode***

Access

Read/Write

Type

String

Default

MuLaw

Allowed values

ALaw | MuLaw

Example

```
Encoding = MuLaw
```

ExternalRingerEnable

Enables use of external ringing voltage.

Syntax

ExternalRingerEnable = ***argument***

Access

Read/Write

Type

String

Default

Enable

Allowed values

Enable | Disable

Example

```
ExternalRingerEnable = Enable
```

HighBatteryEnable

Enables or disables high battery.

Syntax

HighBatteryEnable = ***argument***

Access

Read/Write

Type

String

Default

Enable

Allowed values

Enable | Disable

Example

```
HighBatteryEnable = Enable
```

See also

LowBatteryEnable

Location.PCI.Bus

Specifies the board's PCI location.

Syntax

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

Access

Read/Write

Type

Integer

Default

0

Allowed values

0 - 255

Example

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

Details

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

A PCI board's address and interrupt is automatically set by the system. This statement along with the Location.PCI.Slot keyword assigns the board number to the physical board.

Use *pciscan* to determine the logical bus and slot assigned to boards. For more information about this utility, 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 bus and slot number. A PCI board is specified in the system configuration file by specifying its bus and slot number.

A PCI board's address and interrupt is automatically set by the system. This statement along with Location.PCI.Bus assigns a board number to the physical board.

Use *pciscan* to determine the logical bus and slot assigned to the boards. For more information about this utility, refer to the *NMS OAM System User's Manual*.

LowBatteryEnable

Enables or disables low battery

Syntax

LowBatteryEnable = ***argument***

Access

Read/Write

Type

String

Default

Enable

Allowed values

Enable | Disable

Example

```
LowBatteryEnable = Enable
```

See also

HighBatteryEnable

Name

Specifies the board name.

Syntax

Name = ***name***

Access

Read/Write at board level; read-only at plug-in level

Type

String

Default

The product name, followed by a space and then a numeral. For example: CX 2000-32 0.

Allowed values

(At board level) any valid board name.

(At plug-in level) *cx.bpi*

Example

```
Name = My_CX_2000
```

Details

The name can be changed by modifying this keyword.

The plug-in Name keyword is read-only. It contains the name of the plug-in (*cx.bpi*).

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]

Contains a list of all products supported by the CX plug-in.

Syntax

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

Access

Read-only (CX plug-in level)

Type

String

Allowed values

CX 2000-32 | CX 2000-16

Details

Model CX 2000-16 is not available.

The contents of the Products[x] keywords in the CX plug-in (and all other installed plug-ins) are added to the NMS OAM 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.

You can retrieve the value of the Supervisor Products.Count keyword to indicate the number of items in the Supervisor Products[x] array keyword. Retrieve the value of the board plugin Products.Count keyword to determine the number of items in the plugin Products[x] array keyword.

Ring.Cadences[x].Toff1

Determines the length of the interval after the first ring in cadence ***x***. For more information, refer to *Configuring ring cadences* on page 35.

Syntax

Ring.Cadences[***x***].Toff1 = ***n***

Access

Read/Write

Type

Integer

Default

Ring.Cadences[x]	Toff1 default
0	0
1	800
2	400

Allowed values

n = 0 to 32766 ms

x = 0 to 2

Example

```
Ring.Cadences[1].Toff1 = 800
```

See also

Ring.Cadences[x].Toff2, Ring.Cadences[x].Toff3, Ring.Cadences[x].Ton1, Ring.Cadences[x].Ton2, Ring.Cadences[x].Ton3, Ring.Period

Ring.Cadences[x].Toff2

Determines the length of the interval after the second ring in cadence ***x***. For more information, refer to *Configuring ring cadences* on page 35.

Syntax

Ring.Cadences[***x***].Toff2 = ***n***

Access

Read/Write

Type

Integer

Default

Ring.Cadences[x]	Toff2 default
0	0
1	0
2	400

Allowed values

n = 0 to 32766 ms

x = 0 to 2

Example

```
Ring.Cadences[1].Toff2 = 0
```

See also

Ring.Cadences[x].Toff1, Ring.Cadences[x].Toff3, Ring.Cadences[x].Ton1, Ring.Cadences[x].Ton2, Ring.Cadences[x].Ton3, Ring.Period

Ring.Cadences[x].Toff3

Determines the length of the interval after the third ring in cadence ***x***. Ring.Cadences[x].Toff3 must be at least 2/3 of the duration of Ring.Period. For more information, refer to *Configuring ring cadences* on page 35.

Syntax

Ring.Cadences[***x***].Toff3 = ***n***

Access

Read/Write

Type

Integer

Default

Ring.Cadences[x]	Toff3 default
0	4000
1	4000
2	4000

Allowed values

n = 0 to 32766 ms

x = 0 to 2

Example

```
Ring.Cadences[1].Toff3 = 4000
```

See also

Ring.Cadences[x].Toff1, Ring.Cadences[x].Toff2, Ring.Cadences[x].Ton1, Ring.Cadences[x].Ton2, Ring.Cadences[x].Ton3, Ring.Period

Ring.Cadences[x].Ton1

Determines the length of the first ring in cadence ***x***. For more information, refer to *Configuring ring cadences* on page 35.

Syntax

Ring.Cadences[***x***].Ton1 = ***n***

Access

Read/Write

Type

Integer

Default

Ring.Cadences[x]	Ton1 default
0	2000
1	600
2	400

Allowed values

n = 0 to 32766 ms

x = 0 to 2

Example

```
Ring.Cadences[1].Ton1 = 600
```

See also

Ring.Cadences[x].Toff1, Ring.Cadences[x].Toff2, Ring.Cadences[x].Toff3, Ring.Cadences[x].Ton2, Ring.Cadences[x].Ton3, Ring.Period

Ring.Cadences[x].Ton2

Determines the length of the second ring in cadence ***x***. For more information, refer to *Configuring ring cadences* on page 35.

Syntax

Ring.Cadences[***x***].Ton2 = ***n***

Access

Read/Write

Type

Integer

Default

Ring.Cadences[x]	Ton2 default
0	0
1	600
2	400

Allowed values

n = 0 to 32766 ms

x = 0 to 2

Example

```
Ring.Cadences[1].Ton2 = 600
```

See also

Ring.Cadences[x].Toff1, Ring.Cadences[x].Toff2, Ring.Cadences[x].Toff3, Ring.Cadences[x].Ton1, Ring.Cadences[x].Ton3, Ring.Period

Ring.Cadences[x].Ton3

Determines the length of the third ring in cadence ***x***. For more information, refer to *Configuring ring cadences* on page 35.

Syntax

Ring.Cadences[***x***].Ton1 = ***n***

Access

Read/Write

Type

Integer

Default

Ring.Cadences[x]	Ton3 default
0	0
1	0
2	400

Allowed values

n = 0 to 32766 ms

x = 0 to 2

Example

```
Ring.Cadences[1].Ton3 = 0
```

See also

Ring.Cadences[x].Toff1, Ring.Cadences[x].Toff2, Ring.Cadences[x].Toff3, Ring.Cadences[x].Ton1, Ring.Cadences[x].Ton2, Ring.Period

Ring.Period

Specifies the duration of a full cycle of rings (usually six seconds). For more information, refer to *Configuring ring cadences* on page 35.

Syntax

Ring.Period = ***n***

Access

Read/Write

Type

Integer

Default

6000

Allowed values

n = 6 to 32766 ms

Example

```
Ring.Period = 6000
```

See also

Ring.Cadences[x].Toff1, Ring.Cadences[x].Toff2, Ring.Cadences[x].Toff3, Ring.Cadences[x].Ton1, Ring.Cadences[x].Ton2, Ring.Cadences[x].Ton3

RingVoltageEnable

Enables or disables ring voltage.

Syntax

RingVoltageEnable = ***argument***

Access

Read/Write

Type

String

Default

Enable

Allowed values

Enable | Disable

Example

```
RingVoltageEnable = Enable
```

SignalingLoopbackEnable

Enables or disables signaling loopback.

Syntax

SignalingLoopbackEnable = ***argument***

Access

Read/Write

Type

String

Default

Disable

Allowed values

Enable | Disable

Example

```
SignalingLoopbackEnable = Disable
```

SwitchConnections

Specifies whether the board nails up default switch connections when initialized.

Syntax

SwitchConnections = ***mode***

Access

Read/Write

Type

String

Default

Auto

Allowed values

Yes | No | Auto

Example

```
SwitchConnections = No
```

Details

Valid entries include:

Value	Description
No	Does not nail up switch connections.
Yes	Nails up switch connections regardless of the Clocking.HBus.ClockMode keyword setting.
Auto	Nail up connections automatically if the Clocking.HBus.ClockMode keyword is set to STANDALONE.

When running the Point-to-Point Switching service, set SwitchConnections = No. Use the *ppx.cfg* file to define default connections. For more information, refer to the *Point-to-Point Switching Service Developer's Reference Manual*.

SwitchDriver.Name

Specifies the operating system independent root name of the switching driver.

Syntax

SwitchDriver.Name = ***filename***

Access

Read/Write

Type

String

Default

cxsw

Allowed values

Any valid switch driver name.

Example

```
SwitchDriver.Name = cxsw
```

See also

SwitchConnections

Version.Major

Indicates the major version number of the plug-in. The keyword value is incremented when a change is made to the plug-in.

Syntax

Version.Major = ***number***

Access

Read-only (plug-in level)

Type

Integer

Allowed values

Any integer.

See also

Version.Minor

Version.Minor

Indicates the minor version number of the plug-in. The keyword value is incremented when a change is made to the plug-in.

Syntax

Version.Minor = ***number***

Access

Read-only (plug-in level)

Type

Integer

Allowed values

Any integer.

See also

Version.Major

11 Demonstration program

Using CX demonstration programs

The following demonstration programs are provided with the CX software:

Program	Description
cditest	Verifies that the CDI service is operational and demonstrates CDI service functions.
cdicc	Demonstrates a call center application using the CDI service, with mixed board support in a single application.
cdipbx	Demonstrates a PBX application using the CDI service.

Refer to the *CDI Service Developer's Reference Manual* for information about *cdicc* and *cdipbx*.

Before you start a demonstration program, ensure that:

- Natural Access is properly installed.
- The boards are properly installed.
- One or more boards are booted.

Interactive test program: cditest

Name

cditest

Purpose

Demonstrates CDI service functions executing in asynchronous mode. *cditest* is used to:

- Verify proper installation and operation of the CDI service.
- Expose working examples of Natural Access and CDI service functions.

Usage

`cditest [`***options***`]`

where ***options*** are:

Option	Description	Default
-b ***n***	Board number ***n***.	0
-s [***strm:***]***slot***	DSP [stream] and timeslot.	4:0
-?	Help	

Featured functions

Natural Access system functions and CDI service functions are featured.

Description

cditest is a menu-driven interactive program. Enter one- and two-letter commands to execute Natural Access and CDI service commands.

cditest operates only if default switch connections are nailed up on the board (SwitchConnections=Yes, or SwitchConnections=Auto and Clocking.HBus.ClockMode=STANDALONE, or the connections are made in another way).

Procedure

The following procedure assumes that you are testing on a CX 2000 board with an external power supply and an attached telephone.

To run *cditest*:

1. Navigate to the demonstration program directory:

Operating system	Path
Windows	*/opt/nms/cx/cfg*
UNIX	*opt/nms/ctaccess/demos/cditest*

2. Start *cditest* by entering the following at a command prompt:

```
cditest -b n -s [stream:]slot
```

   Where ***n***, ***stream*** and ***slot*** are the number and PCI stream and slot of the CX board. For example, to open port 01 on board 0, enter:

```
ditest -b0 -s4:0
```

   A menu of commands is displayed.

3. Enter OP to create a context and open the CDI service.

   CTAEVN_OPEN_SERVICES_DONE is displayed on your screen.

4. Enter any additional commands that you want to use.

   For example, the ET command enables the battery. EB enables the bit detector.

   The stop event fetch (SE), get one event (GE), and continue event fetch (CE) commands allow you to step through board operations one at a time, retrieving events with each step. You can use these commands to answer questions you may have relating to state/event combinations.

12 Hardware specifications

General hardware specifications

This topic describes:

- Mechanical specifications
- Host interface
- Telephone interface
- H.100 compliant interface
- Environment
- Maximum board operating temperature
- Power requirements including the telco power per board
- Signaling module
- Rack mount ringing power supply specifications

Mechanical specifications

Feature	Specification
TDM Bus	Features one complete H.100 bus interface with MVIP-95 enhanced-compliant switching
Processing power	One TMS320C549 DSP
Board weight	Main board: .50 lb (.18 kg) Daughterboard: .15 lb (.08 kg)
Software	Natural Access

Host interface

Feature	Specification
Electrical	5 V PCI bus interface compliant with the PCI specification, version 2.2. The PCI interface is a 33 MHz, 32-bit target device
Mechanical	Designed to the PCI specification
Bus Speed	33 MHz maximum
I/O Mapped Memory	Memory mapped interface for efficient block data transfers
Addresses/Interrupts	Automatically configured by PCI BIOS (no jumpers or switches)
BIOS	Required conformance to PCI specification version 2.2

Telephone interface

At the end of the adapter cable on the CX 2000 board, there are two RJ-21 connectors with 24 circuits on the first, and eight circuits on the second. Refer to *Connecting to station telephones* on page 20 for the RJ-21 connector pinouts and the ring pin and tip pin table.

H.100 compliant interface

- Switchable access to any of 4096 H.100 timeslots.
- H.100 clock master or clock slave (software-selectable).
- Compatible with any H.100-compliant telephony interface.

Environment

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

Maximum board operating temperature

Thermometer ID	In temperature controlled laboratory environment	In the field
0	65° C	90° C
1	65° C	90° C
2	60° C	90° C
3	60° C	90° C
4	60° C	90° C

For more information, refer to *Verifying the board's operating temperature* on page 52.

Power requirements

State	Requirement
BD_SEL# Active/CX 2000 Active	1 A maximum @ 5 V

Telco power per board

Input power	Current	Maximum voltage
-24 to-30 V DC (low battery)	1.0 A maximum	30.5 V DC
-24 to -48 V DC (high battery)	1.0 A maximum (with 32 stations active)	52.0 V DC
Ring voltage	0.25 A (with 20 ports active)	92.0 V AC, 52.0 V DC

Signaling module

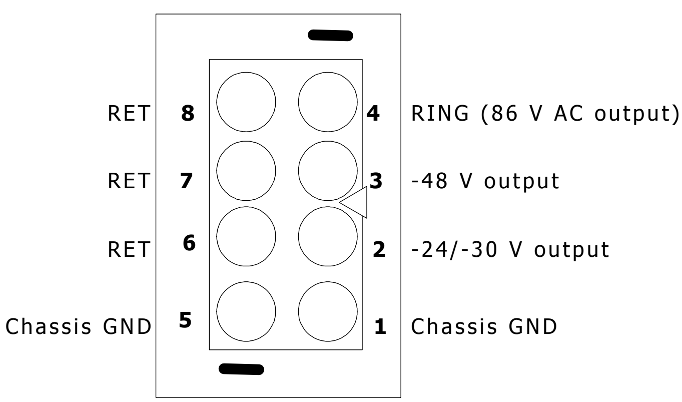
Specification	Value
Return loss (ref. 600 Ohms +2.2 μF standard)	20 dB minimum (ERL)
4 to 2 wire gain tolerance	+/- 1 dB
4 to 2 wire gain range	+6 to -6 dB
2 to 4 wire gain tolerance	+/- 1 dB
2 to 4 wire gain range	+6 to -6 dB
Frequency response 300 Hz - 3200 Hz. reference to 1 kHz	+/- 1 dB
Trans-hybrid loss	20 dB minimum @ 300 Hz - 3400 Hz into 600 Ohms
Signal overload level	+3 dBm at 0 dB gain
T-R input impedance (300 - 3200 Hz)	600 Ohms
Idle channel noise through connection	< 20 dB rnC
Crosstalk transmit to receive channels	< -70 dB @ 1 kHz
Operating loop current	Maximum: 25 to 30 mA Minimum: 10 mA
Maximum ringer equivalence load	1.5
Ringing voltage output	CX 2000 power supply module: 86 V AC, -48 V DC

Rack mount ringing power supply specifications

The specifications in this topic apply to the NMS rack mount ringing power supply.

Description	A 19" w x 5.25" h rack mount chassis containing four separate modules, each rated for 2.2 A (DC) and 0.1 7 A (DC) output current. The modules operate in a parallel mode output current.
Input power	90-132/180-264 V AC 47-63 Hz automatic range selection.
DC output	24V DC/ 30 V DC and -48 V DC @ 2.2 A/module total.
DC output regulation	Less than 1%.
DC output ripple	Less than 0.5% peak to peak.
Output isolation	24 V DC and -48 V DC isolated from chassis ground. AC output is referenced by -48 V DC output.
AC output	0.17A/module with 100% duty cycle.
AC output frequency	17, 20, 25, or 50 Hz +/-1 0% switch selectable.
AC output regulation	Less than 10% for the full input voltage range and no load to full load. 90 V AC maximum.
AC output wave form	Simulated sine wave with less than 20% distortion.
Current limiting	All outputs have current limiting with full protection and auto recovery.
Output indicator	Green LED on the module indicates that all outputs are operating. External signal indicates an alarm condition.
Module failure protection	A failure in any module results in its outputs being automatically taken offline.
Temperature range	Ambient temperature range is 0 to 50 degrees C for full load operation.
EMI design standards	Approved to FCC 20780, Part 15, Class B, EN55022, Class B, and EN50082-1.
Safety design standards	Approved to EN60950, UL1950 3rd edition and 1/24/00 CSA C22.2-950.

The following illustration shows the NMS power supply pinouts:



The mating connector is Positronics PLBO8M0050 with MC116N pins.

Index