TAM-1001-004

DMS-100 Family

PMDEBUG

Technical Assistance Manual

BCS36 and up Standard 09.03 April 1996

northern telecom

DMS-100 Family

PMDEBUG

Technical Assistance Manual

Publication number: TAM-1001-004
Product release: BCS36 and up
Document release: Standard 09.03
Date: April 1996

Publication history

April 1996

BCS36 standard 09.03 updated to reflect editorial changes.

April 1996

BCS36 standard 09.02 updated to include history level and commands for history level.

October 1993

BCS36 preliminary 09.01 updated to reflect changes.

March 1993

BCS35 standard 08.01 updated to reflect changes.

July 1992

BCS34 standard 07.01 updated to reflect changes. Various levels were added. Existing levels were updated with current commands, parameters, and variables.

October 1991

BCS33 standard 06.01 updated to reflect changes.

March 1991

BCS32 standard 05.01 updated to reflect changes and restructure.

Contents

List of Figures

LIst of Tables

About this document

This technical assistance manual (TAM) describes the Peripheral Module Debugging (PMDEBUG) tool set and provides instructions for the use of PMDEBUG. This manual is designed for maintenance personnel who have a high level understanding of the Digital Multiplex System (DMS) switch and code. Maintenance personnel should be aware of the technical information agreement (TIA) between their employer and Northern Telecom (NT).

When to use this document

Northern Telecom (NT) software releases are referred to as batch change supplements (BCS) and are identified by a number, for example, BCS29. This document is written for DMS-100 Family offices that have BCS36 and up.

More than one version of this document may exist. The version and issue are indicated throughout the document, for example, 01.01. The first two digits increase by one each time the document content is changed to support new BCS-related developments. For example, the first release of a document is 01.01, and the next release of the document in a subsequent BCS is 02.01. The second two digits increase by one each time a document is revised and rereleased for the same BCS.

To determine which version of this document applies to the BCS in your office, check the release information in *DMS-100 Family Guide to Northern Telecom Publications,* 297-1001-001.

How to identify the software in your office

The *Office Feature Record (D190)* identifies the current BCS level and the NT feature packages in your switch. You can list a specific feature package or patch on the MAP (maintenance and administration position) terminal by typing

>PATCHER;INFORM LIST identifier

and pressing the Enter key.

where

identifier is the number of the feature package or patch ID

You can identify your current BCS level and print a list of all the feature packages and patches in your switch by performing the following steps. First, direct the terminal response to the desired printer by typing

>SEND printer_id

and pressing the Enter key.

where

printer_id is the number of the printer where you want to print the data

Then, print the desired information by typing

>PATCHER;INFORM LIST;LEAVE

and pressing the Enter key.

Finally, redirect the display back to the terminal by typing

>SEND PREVIOUS

and pressing the Enter key.

Where to find information

The chart below lists the documents that you require to understand the content of this document, or to perform the tasks it describes. These documents are also referred to in the appropriate places in the text.

More than one version of these documents may exist. To determine which version of a document applies to the BCS in your office, check the release information in *DMS-100 Family Guide to Northern Telecom Publications*, 297-1001-001.

Number	Title
297–1001–100	*DMS–100 Family System Description*
TAM-1001-000	*Technical Assistance Manual Index of Documents*
297-1001-001	*Master Index of Practices*
297-1001-106	*Maintenance System Description*
297-1001-107	Maintenance and Administration Tools Description

What precautionary messages mean

Danger, warning, and caution messages in this document indicate potential risks. These messages and their meanings are listed in the following chart.

Message	Significance
DANGER	Possibility of personal injury
WARNING	Possibility of equipment damage
CAUTION	Possibility of service interruption or degradation

Examples of the precautionary messages follow.

DANGER
Risk of electrocution
The inverter contains high voltage lines. Do not open the front panel of the inverter unless fuses F1, F2, and F3 have been removed first. Until these fuses are removed, the high voltage lines inside the inverter are active, and you risk being electrocuted.

WARNING
Damage to backplane connector pins
Use light thumb pressure to align the card with the connectors. Next, use the levers to seat the card into the connectors. Failure to align the card first may result in bending of the backplane connector pins.

CAUTION
Loss of service
Subscriber service will be lost if you accidentally remove a card from the active unit of the peripheral module (PM). Before continuing, confirm that you are removing the card from the inactive unit of the PM.

Command format conventions

In this TAM, a uniform system of notation is used to illustrate system commands and responses. It shows the order in which command elements appear, the punctuation, and the options. Where the conventions are not used, an explanation is given in the text.

In this document, commands, parameters, and responses are represented according to the following conventions.

Input prompt (>)

An input prompt (>) indicates that the information that follows is a command.

Type the command that follows the input prompt and press ENTER.

COMMAND	

Capital letters or special characters

Capital letters show constants, commands, or keywords that the system accepts when entered as written.

Enter the command or fixed parameter exactly as it appears on the page.

Lowercase letters

Lowercase letters show a user– or system–supplied parameter. Definitions are given for each parameter.

For commands and parameters, enter the letters or numbers that the variable represents. In most instances, the name that is used for the variable indicates clearly what you must enter. Where it does not, further explanations are provided.

In responses (which are presented in capital letters), lowercase letters represent a range of values.

Brackets [ ] or **[ ]**

Brackets enclose optional parameters. A vertical list enclosed in brackets means that one or more of the parameters may be selected.

Underlined parameter

Is a default. If no choice is entered, the system acts as though the underlined parameter had been entered.

Underscore connecting words

Means the words are to be treated as one item, for example, pm_type or #_one_two.

•••

Indicates repeated steps or items.

In addition, the following conventions are used.

n (lowercase n)

Is a number from 0 to 9.

a (lowercase a)

Is a letter from A to Z.

h (lowercase h)

Is a hexadecimal integer from 0 to F.

How commands, parameters, and responses are represented

Commands, parameters, and responses in this document conform to the following conventions.

Input prompt (>)

An input prompt (>) indicates that the information that follows it is a command:

```
>BSY
```

Commands and fixed parameters

Commands and fixed parameters that are entered at a MAP are shown in uppercase letters:

```
>BSY LINK
```

Variables

Variables are shown in lowercase letters:

```
>BSY LINK ps_link
```

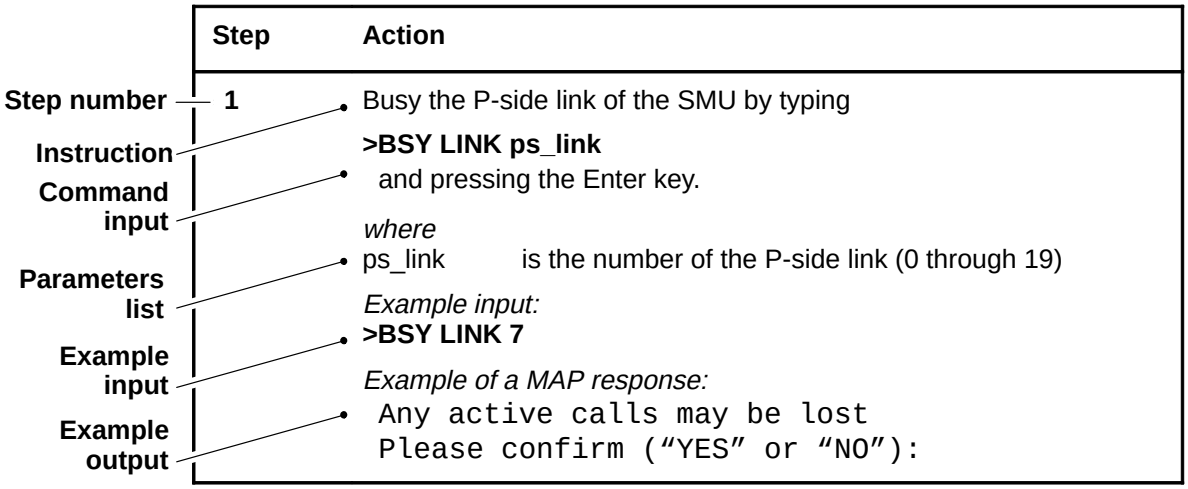
The letters or numbers that the variable represents must be entered. Each variable is explained in a list that follows the command string.

Responses

Responses correspond to the MAP display and are shown in a different type:

```
Any active calls may be lost
Please confirm (“YES” or “NO”):
```

The following example illustrates the command syntax used in this document.



The PMDEBUG utility

This chapter describes the PMDEBUG subsystem, PMDEBUG uses, and PMDEBUG restrictions and limitations.

PMDEBUG is a low-level internal diagnostic tool used to debug Extended Multiprocessor System (XMS) based Peripheral Modules (XPM), which use a Motorola 68000 microprocessor.

PMDEBUG subsystem

The PMDEBUG command interpreter (CI) command is resident to the central control (CC). With it, you can access PMDEBUG commands from any terminal on the CC.

PMDEBUG dumps software error (SWERR), trap, and call-processing data. It also displays the contents of XPM memory and performs call traces.

Note: Only two users can use PMDEBUG simultaneously.

PMDEBUG commands are resident in the peripheral software. Since the memory capacity of XMS-based peripheral module (XPM)s differs, all XPM loads do not have the same commands.

Once an XPM is loaded and running, the PMDEBUG utility can be used to communicate with the peripheral through the network message links.

The PMDEBUG subsystem is accessed when the PMDEBUG CI command is entered with the proper parameters. (Refer to “PMDEBUG command” on page 3-1). Once the PMDEBUG subsystem is entered, the following message appears:

```
PMDEBUG MODE - CONNECTING TO PM
WARNING: You now have access to the PM monitor...Proceed with
             caution
LTCMP>
```

PMDEBUG commands can be entered once the PMDEBUG subsystem is accessed.

PMDEBUG command structure

PMDEBUG commands can be entered in an abbreviated format. On the PMDEBUG command line, commands are displayed with a combination of uppercase and lowercase letters:

Chnls	

As a short cut, you can enter only the uppercase letters to execute the PMDEBUG command; therefore, in the previous example, you can type **C** to access the channels (CHNLS) level.

Commands also can be displayed with the uppercase letters separated from the lowercase letters:

TI(me)	

You can enter the letters to the left of the open parenthesis to execute the PMDEBUG command; therefore, in the previous example, you can type **TI** to execute the TIME command.

You also can type a string of commands on the command line. For example, to access the ASSOC command at the UNPROT level of the CHNLS level, you can enter **C U AS** from the LTCMP level.

PMDEBUG hierarchy

PMDEBUG contains a hierarchy of access levels, analogous to the access levels of MAPCI (Maintenance and Administration Position Command Interpreter). PMDEBUG has two main levels: the master processor (MP) and the signaling processor (SP).

The three XPM processors have a third main level, the function processor (FP), while emergency stand alone (ESA) processors have only one main level, the ESA processor (EP).

Note: This document describes only the commands available in the MP and the SP levels.

Master processor

The (MP) contains instructions that implement the tasks assigned by the CC software. A task is an independent program that accomplishes a specific function, such as auditing, timing, or integrity checking. The MP processes primitives and execs, and carries out all high-level peripheral processor (PP) code calculations.

The MP also supports the following call-processing functions

- digit collection

- channel assignment
- CC message interpretation
- PM message interpretation.

Signaling processor

The SP supports the real-time critical functions of the XPM. These real-time functions include the following:

- A/B-bit scanning
- time-switch control
- channel supervision message (CSM) transmission and reception
- message transmission to, and reception from, the network module (NM) and peripherals

The SP processes the peripheral code and handles all message protocol transactions, such as tones and CSM functions.

The MP and SP communicate through direct memory access (DMA). DMA lets the SP write to and read from MP memory, but the MP cannot access SP memory.

The SP can communicate with all peripheral hardware; the MP cannot communicate with all peripheral hardware.

PMDEBUG uses

PMDEBUG can be used for the following:

- displaying CSM, trap, and SWERR (software errors) information
- displaying channel data
- performing diagnostics
- performing a call trace
- communicating with a peripheral through monitor commands

PMDEBUG restrictions and limitations

Note the following restrictions and limitations:

- PMDEBUG can only be used on a peripheral that is inservice.
- If an XPM load traps and drops into read-only memory (ROM) mode, PMDEBUG will not operate.
- If the C-side links to the XPM are down, PMDEBUG cannot be used.
- If a switch of activity (SwAct) occurs during a PMDEBUG session on the active unit, you will be forced out of PMDEBUG due to the reinitialization of the peripheral.

Calculating terminal identifiers

A terminal is an external connection to the DMS-100, such as a line, a trunk, or a data link. PMDEBUG determines which terminal to monitor by the terminal identifier (TID) entered by the PMDEBUG user. A terminal identifier (TIO) is composed of a node number and a terminal number.

A node number is a unique number assigned by the DMS to a node. A terminal number is a number assigned to a specific terminal attached to a node. Terminal 0 is reserved for maintenance messaging, and the remaining terminals are associated with individual lines, trunks, and other data links.

Each node has a terminal 0 that sends and receives messages specific to the peripheral processor (PP). When a terminal requires a maintenance activity to be performed, such as Return-to-Service, a message is sent from terminal 0 to the Central Controller (CC). When a peripheral module (PM) controller receives maintenance action commands from the Maintenance and administration position (MAP), such as load, busy or test, messages are sent to terminal 0.

Calculating external terminal identifiers

Before calculating internal node and internal terminal numbers, you must determine the external node and external terminal numbers.

To determine the PM node number and terminal number that makes a TID, you can use one of the following methods.

Method 1. Use the PMIST command CONVERT to determine the terminal identifier, as follows:

```
CONVERT  DN   dn
         or
CONVERT  LEN  len
         or
CONVERT  TRK  clli  external_trk_name
```

See Figure 2-1 on page 2-2 for an example of the CONVERT command.

Figure 2-1
CONVERT command example

```
                    PMIST MULTI USER:
                    >convert dn 6211234

                    NN= 0023  TN= 013C

                    PMIST MULTI USER:
                    >
```

The external node number in hexadecimal is 23, and the external terminal number in hexadecimal is 13C.

Method 2. Use the Command Interpreter (CI) command Query Directory Number (QDN) with the directory number of the line to be traced as follows:

QDN <directory number>

Figure 2-2 on page 2-2 shows an example of using the QDN command. The external node number is 21 and the external terminal number is 27.

Figure 2-2
QDN command example

```
CI:
>qdn 6213010
------------------------------------------------------------
DN:      6213010
TYPE: SINGLE PARTY LINE
SNPA: 613
LINE EQUIPMENT NUMBER:    REM1 00 0 00 23
LINE CLASS CODE:   1FR
SIGNALLING TYPE:  DIGITONE
LINE TREATMENT GROUP:     0
LINE ATTRIBUTE INDEX:        0
CARDCODE:   2X17AB   GND: N  PADGRPL STDLN  BNV: NL MNO: N
PM NODE NUMBER      :    21
PM TERMINAL NUMBER  :    27
OPTIONS:
DGT
------------------------------------------------------------
```

Method 3. Use the CI command Query Line Equipment Number (QLEN) subsystem with the LEN as follows:

QLEN <len>

Figure 2-3 on page 2-3 shows an example of the QLEN command. The external node number is 63, and the terminal number is 354.

Figure 2-3
QLEN command example

```
CI:
>qlen 1 0 11 1
------------------------------------------------------------
LEN:     HOST 01 0 11 01
TYPE: SINGLE PARTY LINE
SNPA: 613
DIRECTORY NUMBER:   6215111
LINE CLASS CODE:  1FR
SIGNALLING TYPE:  DIGITONE
LINE ATTRIBUTE INDEX:         32
CARDCODE  6X17    GND N PADGRP STDLN BNV NL MNO N
OPTIONS:
DGT
PM NODE NUMBER     :  63
PM TERMINAL NUMBER :  354
------------------------------------------------------------
```

Method 4. Use the PM command Query perpherial module (QUERYPM) to find the external node number as follows:

POST <node type> <device number>
QUERYPM

Figure 2-4 on page 2-4 shows an example of the QUERYPM command. The external node number is 24.

Figure 2-4
QUERYPM command example

```
PM:
>post dtc 0
POST:
>querypm
  PM TYPE DTC PM NO.: 0 Int.  No.: 1  Node_No.:24
PMs Equipped: 38  Loadname: DT723AY1E,CHKSUM: 018 1
WARM SWACT is supported: VALID FNAME: BTMIA01
DTC 0 is included in the REX schedule.
REX on DTC 0 has not been performed
Node Status: {OK, FALSE}
Unit 0   Act, Status: {OK, FALSE}
Unit 1 Inact, Status: {OK, FALSE}
  Site Flr RPos  Bay_Id   Shf Description  Slot  EqPEC
  HOST  00 C00   LTE  00  51  DTC : 000          6X02AA
```

Method 5. Use the PMIST command NODENO to find the external node number as follows:

1 NODENO <node type> <device class> <device number>

Figure 2-5 on page 2-4 shows an example of the NODENO command.

Figure 2-5
NODENO subcommand example

```
PMIST MULTI USER:
>nodeno tm_node tm8 1
NODENO=16
PMIST MULTI USER:
>
```

2 Calculate the terminal number for Digital Trunk Controller (DTCs) and Line Trunk Controller (LTCs) as follows:

terminal number = (carrier * 32) + <channel> + 1

Note: Determine the carrier number, channel number, circuit number, and similar information by posting the peripheral.

Method 6. To determine the terminal number on a Digital Carrier Module (DCM), post the DCM at the Trunk Test Position (TTP) level of the MAP. Locate the carrier number and time slot associated with the DCM. Then, refer to Figure 2-6 on page 2-6.

In the CCT column, locate the carrier number and the time slot associated with the posted DCM. The external terminal number associated with the carrier and the time slot number is given in the Terminal Number (TN) column.

For example, if DCM 1 is posted at the TTP level of the MAP, and the following is displayed:

```
DCM 1 0 09
```

The carrier number is 0 and the time slot is 09. According to Figure 2-6 on page 2-5, 0-09 is associated with terminal number 11.

Figure 2-6
DCM carrier and time slot to terminal number cross-reference

CCT	TN	CCT	TN	CCT	TN	CCT	TN	CCT	TN
0-01	01	1-01	31	2-01	61	3-01	91	4-01	02
0-02	32	1-02	62	2-02	92	3-02	03	4-02	33
0-03	63	1-03	93	2-03	04	3-03	34	4-03	64
0-04	94	1-04	05	2-04	35	3-04	65	4-04	95
0-05	06	1-05	36	2-05	66	3-05	96	4-05	07
0-06	37	1-06	67	2-06	97	3-06	08	4-06	38
0-07	68	1-07	98	2-07	09	3-07	39	4-07	69
0-08	99	1-08	10	2-08	40	3-08	70	4-08	100
0-09	11	1-09	41	2-09	71	3-09	101	4-09	12
0-10	42	1-10	72	2-10	102	3-10	13	4-10	43
0-11	73	1-11	103	2-11	14	3-11	44	4-11	74
0-12	104	1-12	15	2-12	45	3-12	75	4-12	105
0-13	16	1-13	46	2-13	76	3-13	106	4-13	17
0-14	47	1-14	77	2-14	107	3-14	18	4-14	48
0-15	78	1-15	108	2-15	19	3-15	49	4-15	79
0-16	109	1-16	20	2-16	50	3-16	80	4-16	110
0-17	21	1-17	51	2-17	81	3-17	111	4-17	24
0-18	52	1-18	82	2-18	112	3-18	23	4-18	53
0-19	83	1-19	113	2-19	24	3-19	54	4-19	84
0-20	114	1-20	25	2-20	55	3-20	85	4-20	115
0-21	26	1-21	56	2-21	86	3-21	116	4-21	27
0-22	57	1-22	87	2-22	117	3-22	28	4-22	58
0-23	88	1-23	118	2-23	29	3-23	59	4-23	89
0-24	119	1-24	30	2-24	60	3-24	90	4-24	120

To determine the node number of the DCM, post the DCM at the PM level of the MAP and issue the QUERYPM command. (An example of the QUERYPM command is provided in Figure 2-4 on page 2-4.) The node number is in the output of the QUERYPM command.

Calculating internal terminal identifiers

To determine internal node and internal terminal numbers, do the following.

1 Determine the external node and external terminal numbers to be traced. Refer to Calculating External Terminal Identifiers on page 2-1.

2 Access PMDEBUG by entering: **PMDEBUG pm_type pm_number subunit.**

Following is an example :

PMDEBUG LCM 1 0 0 1

This command will put you in the MP level.

Note 1: Refer to PMDEBUG command on page 3-1 for the proper command syntax for each peripheral type.

Note 2: If the LTCMP prompt does not appear, enter an asterisk (*) until no more levels can be quit. Enter **MP**. Refer to "Master processor" on page 1-2 for more information on the MP level.

3 To determine the internal terminal number, perform the following steps:

a. Access the Call Processing (CP) level by entering **CP**.

b. Determine the internal node and internal terminal number by entering: **E external_node_num external_terminal_num** where E converts external node and external terminal numbers to internal node and internal terminal numbers.

Following is an example of converting an external TID to an internal TID.

```
LTCMP>

TIme,TAsk,Load,Xprompt,Debug,Swerr,Ipc,Patches,Msgtr,
Uspace,Chnls,MTc,DIagnose,Audit,CP,SE,PEg,Rcvrmon.
LTCMP>

>>>cp

Dump,Unprot,Trmnl,Athb,Xdb,tVa,Chb,Fnb,Mod,Brk,Extint,Swct,
Idlq,abtrK,Rmt,*
MP:CP>

>>>e 24 4

INTERNAL NODE     NUMBER = 1
INTERNAL TERMINAL NUMBER = 5

Dump,Unprot,Trmnl,Athb,Xdb,tVa,Chb,Fnb,Mod,Brk,Extint,Swct,
Idlq,abtrK,Rmt,*
MP:CP>

>>>*

LTCMP>
```

Converting internal TID to external TID

To convert internal node and internal terminal numbers to external node and external terminal numbers, do the following.

1 If the internal TID is in hexadecimal, convert the TID to decimal.

2 Access PMDEBUG by entering: **PMDEBUG pm_type pm_number subunit**.

Following is an example:

PMDEBUG LGC 1 0 0 1

This command will put you in the MP level.

Note 1: Refer to "PMDEBUG command" on page 3-1 for the proper command syntax for each peripheral type.

Note 2: If the LTCMP prompt does not appear, enter an asterisk (*) until you have quit all levels. Enter: **MP**. Refer to "Master Processor" on page 1-2 for more information on the MP level.

3 Access the CHNLS level by entering: **C(hnls)**.

4 Access the protected data level by entering: **P(rot**).

5 Dump the node information by entering: **N(ode)**.

Note 1: The NODE command generates the node table.

6 Search the output table for the column labeled INT. This column contains the internal node number.

7 Locate the row containing the internal node number you are converting. Locate the column in the row labeled S_T. This column contains the start terminal number. Use the start terminal number in the following calculations.

8 Determine the drawer and line number of the Line Equipment Number (LEN) by performing the following calculation:

$$\text{drawer} = \frac{(\text{internal terminal}) - (\text{start terminal}) - 1}{32}$$

If the result of this calculation is not an integer, truncate the result.

The line number can be determined by using the following formula:

line = (internal terminal) - (start terminal) - (drawer*32) - 1

9 Search the node table for the EXTNODE field. This field is on the same row as the internal node number. The EXTNODE field is the external node number.

PMDEBUG command syntax

This section describes the command syntax for the PMDEBUG Command Interpreter (CI) command.

Refer to “Command format conventions” on page xvii for information on the command notation used in this section.

PMDEBUG command

The PMDEBUG command accesses the PMDEBUG utility.

Note: This list of Peripheral Module (PM) types appears when command syntax information for PMDEBUG is displayed at the Maintenance Administration Position (MAP).

PMDEBUG	N pm type	nodeno pm number	subunit unit number

N indicates a node number will be specified instead of a peripheral type. If this parameter is not entered, the peripheral module type is expected.

nodeno is the node number.

subunit is the unit number. The unit numbers are as follows:

- **0** indicates unit 0
- **1** indicates unit 1
- **2** indicates the active unit
- **3** indicates the inactive unit

pm type is the peripheral module type, which may be one of the following:

— ADTC Austrian Digital Trunk Controller

— ALCM	Austrian Line Concentrating Module
— ALGC	Austrian Line Group Controller
— ARCC	Austrian Remote Cluster Controller
— CSC	Cell Site Controller
— CSL	Command and Status Link
— DLM	digital line module
— DTC	digital trunk controller
— ELCM	Enhanced Line Concentrating Module
— ESA	Emergency Stand Alone
— GIC	Generic Interface Controller
— ICP	intelligent cellular peripheral
— IDTC	integrated digital trunk controller
— ILCM	international line concentrating module
— ILGC	international line group controller
— ISLM	Integrated Service Line Module
— LCM	line concentrating module
— LCMI	line concentrating module for ISDN
— LGC	line group controller
— LGCI	line group controller for ISDN
— LTC	line trunk controller
— LTCI	line trunk controller for ISDN
— MSB6	Message Switch Buffer (#6 Protocol)
— MSB7	Message Switch Buffer (#7 Protocol)
— PDTC	Digital Trunk Controller for PCM30
— PLGC	Line Group Controller for PCM30
— RCC	remote cluster controller
— RCCI	remote cluster controller for ISDN
— RMM	remote maintenance module
— RMSC	Remote Maintenance Switching Center
— SMR	Subscriber Carrier Module Rural
— SMS	Subscriber Carrier Module SLC-96
— SMSR	Subscriber Carrier Module SLC-96 Remote
— SMU	Subscriber Carrier Module Urban
— STC	Signaling Terminal Controller
— TAC	Test Access Controller
— TDTC	MOC (Licensee) Digital Trunk Controller

— TLGC MOC (Licensee) Line Group Controller
— TLTC MOC (Licensee) Line Trunk Controller
— TMS TOPS Messaging Switch
— TRCC MOC (Licensee) Remote Cluster Controller
— VSR Very Small Remote

Refer to “command syntax” on page 3-3 for the syntax of each peripheral type.

pm number is the peripheral module number.

unit number The unit numbers are as follows:

0 indicates unit 0

1 indicates unit 1

2 indicates the active unit

3 indicates the inactive unit

Note 1: If no unit number is specified, the active unit is accessed.

Note 2: To determine the unit number of the line to be queried on an Line Concentrating Module (LCM), the unit number is 0 if the drawer number is even and 1 if the drawer number is odd.

Command syntax

Following is the command syntax for each peripheral type:

ADTC	**number subunit**
ALCM	**site frame unit subunit**
ALGC	**number subunit**
ARCC	**number subunit**
CSC	**number subunit**
CSL LGC	**LGCno CSLSTno subunit**
DTC	**DTCno CSLSTno subunit**
LTC	**LTCno CSLSTno subunit**
DLM	**site frame unit subunit**
DTC	**number subunit**
ELCM	**site frame unit subunit**
ESA	**number**
GIC	**number subunit**
ICP	**number subunit**
IDTC	**number subunit**
ILCM	**site frame unit subunit**
ILGC	**number subunit**
ISLM	**site frame unit**
LCM	**site frame unit subunit**
LCMI	**site frame unit subunit**
LGC	**number subunit**
LGCI	**number**
LTC	**number subunit**
LTCI	**number subunit**
MSB6	**number subunit**
MSB7	**number subunit**
PDTC	**number subunit**
PLGC	**number subunit**
RCC	**number subunit**
RCCI	**number subunit**
RMM	**number**
RMSC	**number subunit**
SMR	**number subunit**
SMS	**number subunit**
SMSR	**number subunit**
SMU	**number subunit**
STC	**msb_type**
	MSB6 msbno STCM circuit
	MSB7 msbno STCM circuit
TAC	**number subunit**
TDTC	**number subunit**
TLTC	**number subunit**
TMS	**number subunit**
TRCC	**number subunit**
TLGC	**number subunit**
VSR	**site frame unit subunit**

Responses:

`ERROR: ACT/INACT DOES NOT APPLY`

Explanation: The selection of an active or inactive subunit is not possible for the PM specified.

User action: Verify the command syntax for the peripheral type, and reenter the entire command string.

`ERROR: DEATH OF INPUT PROCESS`

Explanation: The process in the Central Control (CC) that monitors user input has ended. The process should re-create itself.

User action: If the process does not recreate itself, try again when the CC is not as busy. If the problem persists, initiate a service report.

`ERROR: FAILED TO GET ACT/INACT`

Explanation: **ABORT** was entered when PMDEBUG was prompted for the subunit number. An active unit has not been specified or cannot be determined. In the case of peripherals that must communicate through another peripheral (such as an LCM through an LGC), an active unit cannot be found in the peripheral being passed through (such as the LGC).

User action: Verify the subunit number and reenter the entire command string. In the case of peripherals that must communicate through another peripheral (such as an LCM through an LGC), wait and try again.

`ERROR: FAILED TO GET LCM SUB-UNIT PARM`

Explanation: **ABORT** was entered when PMDEBUG was prompted for the subunit number. A subunit parameter was not entered with either the ILCM, LCM, or DLM parameters.

User action: Verify the LCM, ILCM, or DLM subunit number and reenter the entire command string.

`ERROR: FAILED TO GET MOD NUMBER PARM`

Explanation: **ABORT** was entered when PMDEBUG was prompted for the PM number. A valid module (peripheral) number was not entered.

User action: Verify the data to be entered. Reenter the entire command string.

`ERROR: FAILED TO GET MODULE TYPE PARM`

Explanation: **ABORT** was entered after PMDEBUG prompted for the module type. A valid peripheral type was not entered.

User action: Verify the module type, and reenter the entire command string.

ERROR: FAILED TO GET NODE NUMBER PARM

Explanation: **ABORT** was entered when PMDEBUG was prompted for the node number. An invalid node number has been entered.

User action: Verify the node number and unit number to be entered and reenter the entire command string.

ERROR: FAILED TO GET STCID

Explanation: In the case of the CSL parameter, the CSL parameter was missing either the LGC, DTC, or LTC parameter, or the CSLST number or subunit number. In the case of the STC parameter, the STC parameter was missing either the MSB6 or MSB7 parameter, or the STCM or circuit numbers.

User action: Verify the command string to be entered, and reenter the entire command string.

ERROR: FAILED TO GET STCID

Explanation: **ABORT** was entered when PMDEBUG was prompted for the STC module number.

User action: Verify the MSB type, MSB number, STC module number, and circuit number. Reenter the entire command string.

ERROR: FAILED TO GET STCID

Explanation: **ABORT** was entered when PMDEBUG was prompted for the subunit number.

User action: Verify the subunit number, and reenter the entire command string.

ERROR: FAILED TO GET SUB-UNIT PARM

Explanation: **ABORT** was entered when PMDEBUG was prompted for the subunit number. A valid subunit parameter was not entered.

User action: Verify subunit number. Reenter entire command string.

ERROR: FAILED TO OPEN MESSAGE LINKS

Explanation: No maintenance channels are available on the selected peripheral. No C-side/P-side channel connection can be made to route from one peripheral to another, such as, from an LGC to an LCM.

User action: Post the peripheral to determine the state of that peripheral. If the peripheral is not system busy (SYSB) or manually busy (MBSY), wait and try again later. If the problem persists, initiate a service report.

ERROR: LACK OF SYSTEM RESOURCES

Explanation: PMDEBUG cannot allocate the resources required for the subsystem to function.

User action: Try again when the CC is less busy.

ERROR: INPUT PROCESS NOT STARTED

Explanation: The CC is unable to create the process necessary to monitor the connection between the CC and the PM.

User action: Try again when the CC is less busy. If software error (SWERR)s or traps occur, initiate a service report.

ERROR: INVALID MODULE

Explanation: The node number entered with the PMDEBUG N parameter does not exist. An invalid peripheral type was entered.

User action: Verify the node number, and reenter the entire command string.

ERROR: INVALID MODULE TYPE

Explanation: Module type parameter entered with the PMDEBUG command does not exist. An invalid module type was entered.

User action: Verify the module type, and reenter the entire command string.

ERROR: MESSAGING FLOODED, PLEASE WAIT THEN TRY AGAIN

Explanation: The mailbox is too busy to process the message fast enough.

User action: Wait a few minutes and try again.

ERROR: MODULE ALREADY ACCESSED

Explanation: PMDEBUG is already active on the unit specified. Only one user can be in a particular unit of a peripheral at one time.

User action: Wait and try again later.

ERROR: MSG LOST ON EXTRACT FAIL

Explanation: The peripheral was unable to extract the message information from the message sent by the monitor task in the XMS-based peripheral module (XPM).

User action: If the messages keep occurring, log off and then log on. If the problem persists, initiate a service report.

ERROR: MSG LOST ON FAIL TO SEND

Explanation: The peripheral did not receive the message/command from the CC. The CC will attempt to resend the message.

User action: If the messages keep occurring, logout and log back on. If problem persists, initiate a service report.

ERROR: MSG NOT FROM MONITOR

Explanation: The XPM message received is not a PMDEBUG message from the XPM monitor.

User action: If messages keep occurring, logout and log back on. If problem persists, initiate a service report.

ERROR: NODE NUMBER INVALID

Explanation: The node number given with the N parameter is invalid.

User action: Verify the node number, and reenter the entire command string.

ERROR: ONLY 3 USERS ALLOWED

Explanation: Only three users are allowed in the PMDEBUG environment at the same time.

User action: Wait and try again when fewer people are using PMDEBUG.

ERROR: REPEAT COUNT MUST BE TWO HEX DIGITS

Explanation: The REPEAT command must be followed by two hexadecimal digits. For example, if you wish to repeat a command five times, you must type **REPEAT #05 <command>**.

User action: Reenter the REPEAT command followed by a hexadecimal sign and two digits.

LACK OF CSC LINK SOURCES

Explanation: No CSC link sources are available.

User action: Verify CSC link sources.

LINKS CLOSED, WILL TRY TO REOPEN

Explanation: No C-side links are open.

User action: Post the peripheral to determine the state of that peripheral. If the peripheral is not SYSB or MBSY, try again later. If the problem persists, initiate a service report.

NO ROUTE AVAILABLE TO NODE

Explanation: No maintenance message links are open to route to the peripheral.

User Action: Post the peripheral to determine the state of that peripheral. If the peripheral is not system busy (SYSB) or manually busy (MBSY), try again later. If problem persists, initiate a service report.

NOTE: PMDEBUG WILL TERMINATE WHEN LAST REQUEST IS COMPLETE

Explanation: PMDEBUG monitors messages for a few seconds after the QUIT command is issued to prevent MTCB logs.

User Action: None.

PMDEBUG MODE - CONNECTING TO PM

WARNING: You now have access to the PM monitor...proceed with caution

Explanation: This message indicates you have accessed the PMDEBUG subsystem.

User Action: None.

PMDEBUG TERMINATES

Explanation: The PMDEBUG process has terminated. You are returned to the CI level.

User Action: None.

UNABLE TO CONVERT PM TYPE TO NODE TYPE

Explanation: An internal routine for converting the peripheral type to a node type has failed.

User Action: Initiate a service report.

WARNING: LOGON OR LOGOFF CHANGES PM PMDEBUG STATE

Explanation: The LOGON or LOGOFF command sends logon or logoff instructions to the peripheral. PMDEBUG cannot be used if the LOGOFF command is sent to the peripheral.

User Action: QUIT out of PMDEBUG before logging off.

PMDEBUG commands in the master processor

This section describes the commands available in the master processor (MP). PMDEBUG commands that are unique to the signaling processor (SP) are described in Chapter 5 on page 5-1.

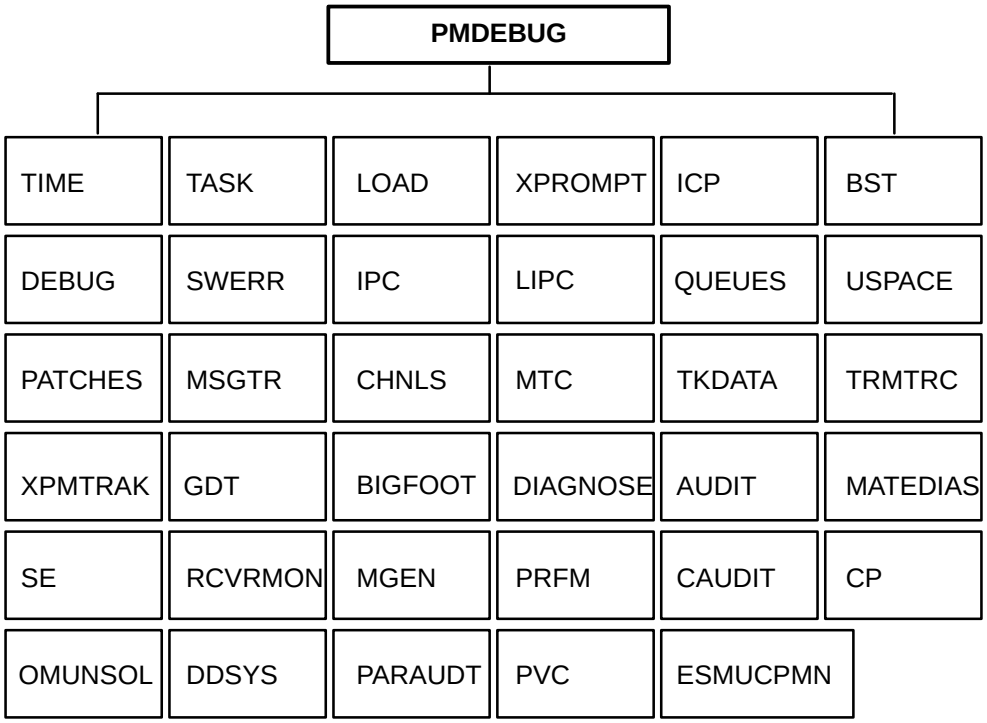
PMDEBUG commands are available to you following successful access to the PMDEBUG subsystem. See "Accessing PMDEBUG" on page 7-1 for a description of this procedure.

> ***Note:*** The top level of PMDEBUG varies depending on the peripheral being accessed. For simplicity in this document, the top level of PMDEBUG in the MP will be referred to as the LTCMP level.

LTCMP level

LTCMP is the first level accessed when the PMDEBUG subsystem is entered in a DTC. Figure 4-1 on page 4-2 contains an example of some of the commands available at the LTCMP level of a digital trunk controller (DTC).

Figure 4-1
Commands at the LTCMP level of a DTC



Note 1: All debugging levels contained in this document may not be present in every load. Some loads have less memory and are not large enough to contain every level.

Note 2: All commands contained in a debugging level in this document may not be present in every load. Some loads have less memory and are not large enough to contain every command.

Note 3: Some commands contained in a debugging level may be slightly different from the descriptions in this document. Memory constraints of various loads cause some commands to be altered.

Note 4: Commands that are not relevant to a specific XPM load are not included in that load. For instance, a remote cluster controller (RCC) does not have a channel supervision message (CSM) card. Therefore, an RCC does not have the CSM monitor level.

TIME level

The TIME level is accessed when the TIME command is entered at the LTCMP level.

The TIME level displays the system time.

TIME	

The TIME command provides access to the DISPLAY command that displays the current system time.

Example:

```
LTCMP>

TIme,TAsk,Load,Xprompt,Debug,Swerr,LIpc,Ipc,Patches,Msgtr,
Uspace,Chnls,MTc,DIagnose,Audit,PKtldr,CP,SE,C7tu,PEg,Rcvrmon.
LTCMP>

>>>ti

Display,*
MP:TIme>

>>>d

Slip count = 10 .  System time: 00:01:22:57.33  00 07 98 45
MP:TIme>
```

TASK level

The TASK level is accessed when the TASK command is entered at the LTCMP level.

A task is an independent program that accomplishes a specific function, such as auditing, timing, or integrity checking.

The TASK level displays information about specified tasks.

TASK	**task name** **ALL**

Example:

```
LTCMP>
TIme,TAsk,Load,Xprompt,Debug,Swerr,LIpc,Ipc,Patches,Msgtr,
Uspace,Chnls,MTc,DIagnose,Audit,PKtldr,CP,SE,C7tu,PEg,Rcvrmon.
LTCMP>

>>>ta

INPUT TASK NAME (TASKNAME, ALL)
MP:TAsk>

>>>tpt

TASKID   TCBPTR   PRIO SPACE  STKUSED HEAPUSED AVAIL TASKNAME
0029 0029  0006 83B0  4   3000     90     144   2766  TPT

TOTALS:              3000     90     144   2766
```

The field names indicate the following:

- TASKID task identifier
- TCBPTR Task Control Block pointer
- PRIO task priority
- SPACE total amount of space allocated for the task
- STKUSED amount of stack used
- HEAPUSED amount of heap used
- AVAIL amount of space that has not been used
- TASKNAME name of the task

LOAD level

The LOAD level is accessed when the LOAD command is entered at the LTCMP level.

The LOAD level provides information about the current software load in the XPM.

LOAD	

Example:

```
LTCMP>

TIme,TAsk,Load,Xprompt,Debug,Swerr,LIpc,Ipc,Patches,Msgtr,
Uspace,Chnls,MTc,DIagnose,Audit,PKtldr,CP,SE,C7tu,PEg,Rcvrmon.
LTCMP>

>>>l

RAM LOAD NAME =  DC725AT

MP ROM NAME =  XPMRFA10, SP ROM NAME = XPMRFA10
```

XPROMPT level

The XPROMPT level is accessed when the XPROMPT command is entered at the LTCMP level.

The XPROMPT level allows you to change the PMDEBUG prompt.

XPROMPT	

Example:

```
LTCMP>

TIme,TAsk,Load,Xprompt,Debug,Swerr,LIpc,Ipc,Patches,Msgtr,
Uspace,Chnls,MTc,DIagnose,Audit,PKtldr,CP,SE,C7tu,PEg,Rcvrmon.
LTCMP>

>>>x

ENTER SPECIAL CHARACTER
MP:XPROMPT>

>>>c

LTCMP>C
```

In the previous example the *C* is removed from the prompt by accessing the XPROMPT level and pressing `ENTER` when the system asks for a special character.

ICP level

The Intelligent Cellular Peripheral (ICP) level allows manipulation of all key ICP data structures and dynamic state tables.

CAUTION
This monitor level should not be used by a customer, unless specifically directed to and guided by a BNR or NT representative.

Misuse of this level can adversely affect the functioning and sanity of the ICP peripheral. The loss of call processing and communications activity can occur on up to eight cellsites connected to the affected ICP.

BST level

The Base Station Transceiver (BST) level allows manipulation of all key ICP data structures and dynamic state tables.

CAUTION
This monitor level should not be used by a customer, unless specifically directed to and guided by a BNR or NT representative.

Misuse of this level can adversely affect the functioning and sanity of the ICP peripheral. The loss of call processing and communications activity can occur on up to eight cellsites connected to the affected ICP.

DEBUG level

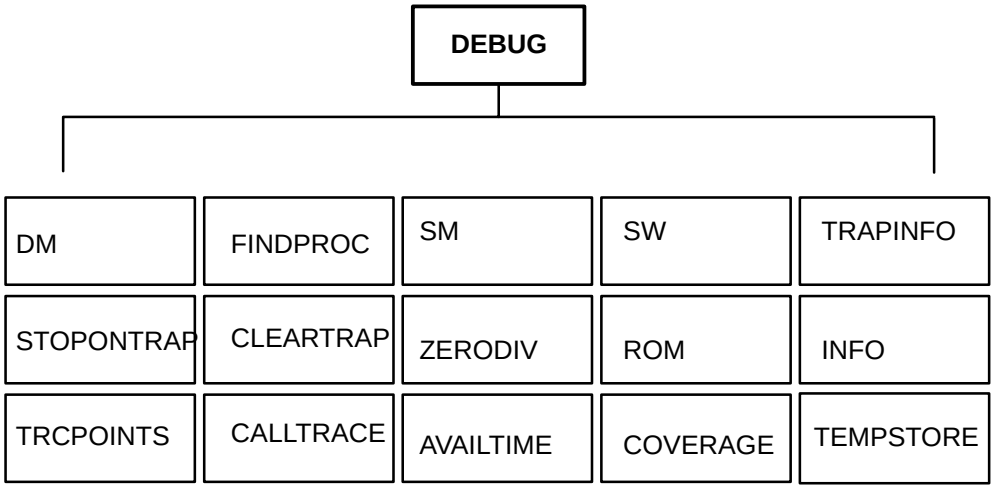
The DEBUG level is accessed when the DEBUG command is entered at the LTCMP level.

The DEBUG level contains commands that allow you to perform such functions as the following:

- disassemble an address in memory
- find the location of a procedure in memory
- perform word writes to memory
- display trap information
- set and display tracepoints
- perform call traces
- display available memory
- display available temporary storage
- perform mathematical functions

Figure 4-2 on page 4-7 contains the commands available at the DEBUG level.

Figure 4-2
Commands in the DEBUG level



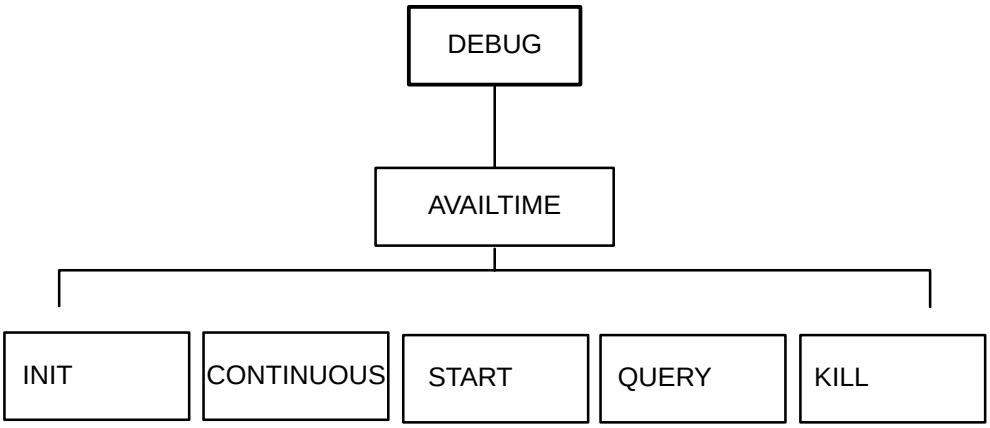
The following commands are available in the DEBUG level:

AVAILTIME level

The AVAILTIME level is accessed when the AVAILTIME command is entered from the DEBUG level.

Figure 4-3 on page 4-8 contains the commands available at the AVAILTIME level.

Figure 4-3
Commands in the AVAILTIME level



The AVAILTIME level contains commands for measuring unused processor real time on the XPM.

Following are descriptions of the commands available in the AVAILTIME level:

INIT command

The INIT command allocates storage and initializes measurement tasks. The INIT command does not start any processor time measurements.

The INIT command must be issued before any other commands can be entered at the AVAILTIME level.

INIT	

CONTINUOUS command

The CONTINUOUS command displays the amount of real time available in the peripheral processor (PP) at designated time intervals. The CONTINUOUS command prompts for the time interval in seconds, from 10 to 100.

CAUTION
ILGC goes system busy (SysB) during PMDEBUG AVAILTIME scanning.
Using the CONTINUOUS command in the AVAILTIME level can cause the XPM to go SysB. The preferred tool to use is the PERFORM level in the MAP.

If the START command is issued before the CONTINUOUS command, the measurement tasks must be terminated with the KILL command and reinitialized with the INIT command before the CONTINUOUS command can be issued.

In addition, the START command cannot be issued while the CONTINUOUS command is active.

To halt the display of real-time information, enter the KILL command.

CONTINUOUS

Example

```
MP:Debug> DM,FIndproc,SM,SW,Trapinfo,STopontrap,Cleartrap,
Zerodiv,Rom,Info,TRCpoints,CAlltrace,Availtime,COverage,TEmpstore

MP:Debug>

>>>a

Init,Continuous,Start,Query,Kill.
MP:Availtime>

>>>i
MP:Availtime>

>>>c
time interval in seconds ( 10-100 )
MP:Continuous>

>>>10
```

```
MP:Availtime>

%T 1 - availtime in MP 71 % ( 7 .   66 sec )
%T 2 - availtime in MP 65 % ( 7 .  119 sec )
%T 3 - availtime in MP 65 % ( 7 .  156 sec )
%T 4 - availtime in MP 65 % ( 7 .   85 sec )
%T 5 - availtime in MP 65 % ( 7 .  159 sec )
%T 6 - availtime in MP 65 % ( 7 .  160 sec )

<cr>

Init,Continuous,Start,Query,Kill.
MP:Availtime>

>>>k
MP:Availtime>
```

START command

The START command starts processor time measurement. The START command prompts for the duration of the measurement and for the frequency (the number of periods) to measure.

START	

QUERY command

The QUERY command displays the amount of real time available in the PP.

This command is used after the START command to display the results of the processor time measurements.

QUERY	

KILL command

The KILL command stops the measurements, ends measurement tasks, and deallocates storage.

KILL	

Example:

```
MP:Debug> DM,FIndproc,SM,SW,Trapinfo,STopontrap,Cleartrap,
Zerodiv,Rom,Info,TRCpoints,CAlltrace,Availtime,COverage,TEmpstore

MP:Debug>
```

```
>>>a

Init,Continuous,Start,Query,Kill.
MP:Availtime>

>>>i
MP:Availtime>

>>>s

TIME INTERVAL IN SECONDS ( 1-100 )
MP:Start>

>>>1

REPETITION FACTOR ( 1-100 )
MP:Start>

>>>10

MP:Availtime>

>>>q
1 - AVAILTIME IN MP 68 % ( 0 .678 sec )
2 - AVAILTIME IN MP 72 % ( 0 .718 sec )
3 - AVAILTIME IN MP 72 % ( 0 .719 sec )
4 - AVAILTIME IN MP 72 % ( 0 .717 sec )
5 - AVAILTIME IN MP 72 % ( 0 .721 sec )
6 - AVAILTIME IN MP 72 % ( 0 .716 sec )
7 - AVAILTIME IN MP 71 % ( 0 .714 sec )
8 - AVAILTIME IN MP 63 % ( 0 .634 sec )

MP:Availtime>

>>>k

MP:Availtime>

>>>*

MP:Debug> DM,FIndproc,SM,SW,Trapinfo,STopontrap,Cleartrap,
Zerodiv,Rom,Info,TRCpoints,CAlltrace,Availtime,COverage,TEmpstore

MP:Debug>
```

CALLTRACE level

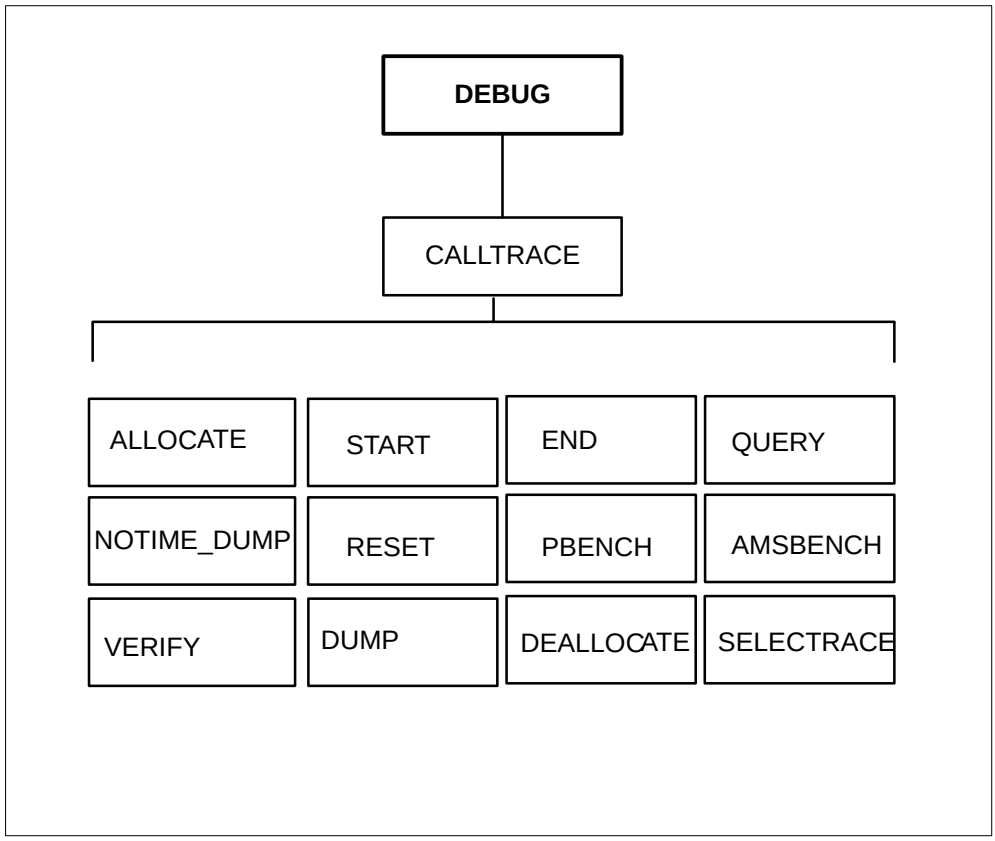
The CALLTRACE level is accessed when the CALLTRACE command is entered at the DEBUG level.

Commands at the CALLTRACE level are used to display the flow of call processing through specified segments in a given task. The output consists of segment and procedure names and numbers, include and exclude times, a time stamp, include and exclude percentages, the number of procedure calls for each procedure, and the depth of the procedure call.

The CALLTRACE level requires at least 64K of free storage in the processor being examined (such as the MP or the SP). The tools at the CALLTRACE level cannot be used if sufficient free memory is not available.

Figure 4-4 on page 4-12 contains the commands available in the CALLTRACE level.

Figure 4-4
Commands in the CALLTRACE level



The following descriptions are of the commands available in the CALLTRACE level:

ALLOCATE command

The ALLOCATE command allocates and initializes procedure trace buffers.

ALLOCATE	D

Where:

D indicates that the default number of parameters will be allocated

START command

The START command starts the call trace.

START	

The START command prompts for the segments to be traced (0, 1, 6, ...29).

1 Enter the task name, `ALL` if all tasks are to be traced, or the task identifier in hex (first word only). Entering `30` provides all user segments as well as system segment 0 (0, 1, 6, ... 29). Entering `31` provides all user segments and will prompt for the tasks to be traced.

To reduce the amount of output, the information entered should be as selective as possible.

2 To start the trace immediately, enter `0` when prompted for the number of procedure calls to skip.

END command

The END command stops call tracing.

END	

QUERY command

The QUERY command displays the number of buffers in use.

QUERY	

Note: Approximately two buffers are used for each line of CALLTRACE output and for each procedure call.

VERIFY command

The VERIFY command compares how much time is taken to enter and exit a Pascal procedure against how much time is taken to enter and exit an assembler procedure.

The VERIFY command cannot be used while a trace is in progress.

Once the VERIFY command is processed, you are prompted to enter BASEMON 0.

VERIFY	

DUMP command

The DUMP command dumps the contents of tasks. A task is an independent program that accomplishes a specific function, such as auditing or parity checking.

The DUMP command prompts for the task name, which can be either the first word of the task number in hexadecimal or the task name. If the task name is entered, the amount of unused storage is displayed.

While a table is being processed, asterisks are displayed on the screen for every 50 table entries, indicating that the process is not in an infinite loop and has not ended.

DUMP	**TRACE** **SUMMARY** **BOTH** **VOMIT**

Where:

TRACE displays a trace of procedures, including segment and procedure names and numbers, as well as the include time, exclude time, a time stamp, and the level of the procedure calls represented graphically with asterisks.

Note 1: The include time is the number of seconds, milliseconds, and microseconds spent in a procedure, including procedure calls.

Note 2: The exclude time is the number of seconds, milliseconds, and microseconds spent in a procedure, excluding procedure calls.

SUMMARY displays the include and exclude times, as well as the number of times a procedure is called, and the total time for the call trace and procedure counts. This command also displays the

percentage of include time over total time and the percentage of exclude time over total time.

BOTH displays both the trace information and the summary information

VOMIT displays the segment names and numbers, the procedure number called, and the time stamp

Note 1: A plus sign (+) displayed next to the include time for a procedure indicates that the procedure did not return.

Note 2: A switch of activity (SwAct) may disrupt a call trace in the SP.

Note 3: Allocate the least amount of buffers possible to reduce system degradation.

NOTIME_DUMP command

The NOTIME_DUMP command dumps call-trace data without displaying the time.

NOTIME_DUMP	

RESET command

The RESET command resets call-trace buffers for reuse.

RESET	

PBENCH command

The PBENCH command allows benchmark testing for Pascal-to-Pascal calls. This procedure is used by the VERIFY command. You should not process this command.

PBENCH	

AMSBENCH command

The AMSBENCH command allows benchmark testing for Pascal-to-assembler calls. This procedure is used by the VERIFY command. You should not process this command.

AMSBENCH	

DEALLOCATE command

The DEALLOCATE command returns temporary storage for use by other processes. The ALLOC command must be reissued before another trace can be started.

DEALLOCATE	

SELECTRACE command

The SELECTRACE command performs a CALLTRACE for a specific set of terminal identifiers (TID).

The SELECTRACE command allows you to select up to three specific terminals in the TPT task to trace. Only procedure calls pertaining to the selected terminals are traced.

This command is used to perform a call-processing call trace on a specific terminal in a specific task.

The information displayed with the DUMP command does not indicate which terminal number is being traced. Therefore, tracing one terminal at a time is most useful.

SELECTRACE	ON n OFF QUERY

Where:

ON turns SELECTRACE on

n is the internal terminal number to be traced. Valid terminal numbers are 0 to 7055

OFF turns SELECTRACE off

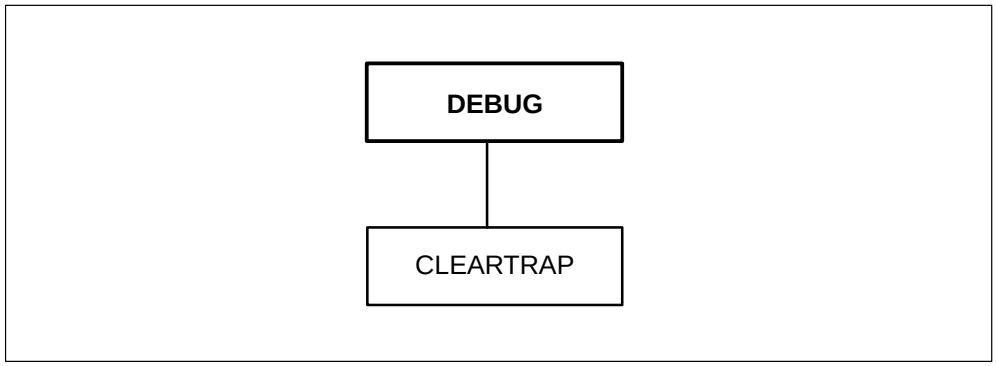
QUERY displays information on the status of the SELECTRACE

CLEARTRAP command

The CLEARTRAP command is entered from the DEBUG level. The CLEARTRAP command clears the trap buffer.

Figure 4-5 on page 4-17 diagrams how the CLEARTRAP command is accessed.

Figure 4-5
The CLEARTRAP command



Following is the CLEARTRAP command syntax:

CLEARTRAP	All	Prev

Where:

All clears all traps from the trap buffer

Prev clears the traps from previous loads

Example:

```
MP:Debug> DM,FIndproc,SM,SW,Trapinfo,STopontrap,Cleartrap,
Zerodiv,Rom,Info,TRCpoints,CAlltrace,Availtime,COverage,TEmpstore

MP:Debug>

>>>c

CLEAR ALL TRAPS OR PREVIOUS TRAPS (‘A’ or ‘P’):
MP:Cleartrap>

>>>p

PREVIOUS TRAPS CLEARED
```

DM command

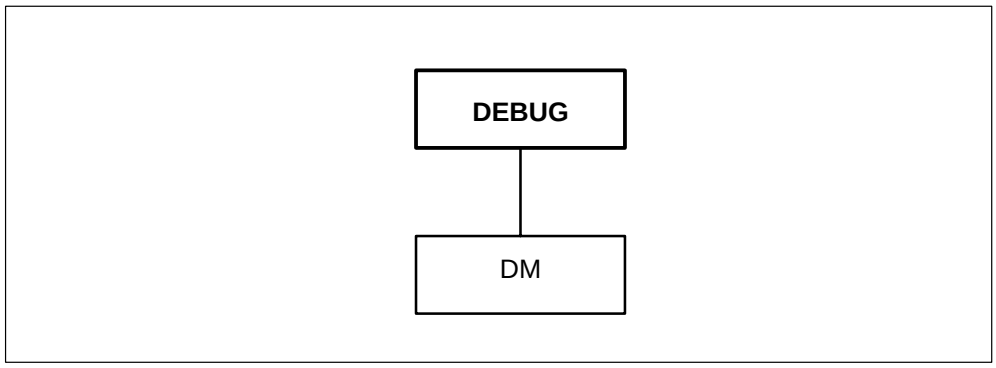
The display memory (DM) command is issued from the DEBUG level.

The DM command displays memory locations in hexadecimal and displays an ASCII interpretation of the memory locations. The DM command

searches for a segment and a procedure and performs symbolic disassembly of memory.

Figure 4-6 on page 4-18 diagrams how the DM command is accessed.

Figure 4-6
The DM command



Following is the DM command syntax:

DM	**address <count>** **G offset <count>** **R address <count>** **P procedure <offset> <count>** **S segment <offset> <count>**

Where:

G displays memory at the specified offset into the global variable memory space

R disassembles the number of instructions specified beginning at the absolute address specified. If no number is entered in the count field, the command disassembles code until you press Enter. Entering Y at the prompt continues disassembly. Pressing Enter again ends disassembly.

P disassembles instructions beginning at the specified offset in the specified procedure. This command searches all available segments for the procedure. A procedure entered as a decimal number represents the procedure number. If no entry is made for the offset parameter, the default is 0. If no number is entered in the count field, the command disassembles code until you press Enter. Entering Y at the

prompt continues disassembly. Pressing Enter again ends disassembly.

S disassembles instructions beginning at the specified offset in the specified procedure in the specified segment. If no entry is made for the offset parameter, the default is 0. If no number is entered in the count field, the command disassemble code until you press Enter. Entering `Y` at the prompt continues disassembly. Pressing Enter again ends disassembly.

address is the absolute address in the processor memory. The address is entered in hexadecimal.

procedure is the procedure name or decimal number. To find the procedure number, refer to the "FINDPROC command" on page 4-21.

segment is the segment name or decimal number. To find the segment name or number, refer to the "FINDPROC command" on page 4-21.

count when used without the R, P, or S options specifies the number of bytes to be displayed. The number entered for the count parameter is assumed to be in hexadecimal. The number entered is rounded up to the nearest multiple of 16. If no number is entered, 16 is the default and 16 bytes are displayed.

When a number is entered for the count parameter with the R, P, or S options, the specified number of assembly language lines are displayed.

If no number is entered for the count parameter, disassembly continues until you press `ENTER`.

offset is the offset byte in the procedure. To determine the offset byte, refer to the FINDPROC command on page 4-21.

Note: If no count parameter is given with the R, P, or S options, disassembling continues until you press `ENTER`. You are then prompted whether you can continue disassembly or stop.

Example:

```
MP:Debug> DM,FIndproc,SM,SW,Trapinfo,STopontrap,Cleartrap,
Zerodiv,Rom,Info,TRCpoints,CAlltrace,Availtime,COverage,TEmpstore

MP:Debug>

>>>fi 123456

Seg : TPTWSWCT  97    Proc : FILLSYNC  12   Offset : 88

MP:Debug> DM,FIndproc,SM,SW,Trapinfo,STopontrap,Cleartrap,
Zerodiv,Rom,Info,TRCpoints,CAlltrace,Availtime,COverage,TEmpstore

MP:Debug>

>>>dm 123456

123456  0805 0000 6700 0288 266C 5310 4DEB 0002 ....g...&1S.M...

MP:Debug> DM,FIndproc,SM,SW,Trapinfo,STopontrap,Cleartrap,
Zerodiv,Rom,Info,TRCpoints,CAlltrace,Availtime,COverage,TEmpstore

MP:Debug>

>>>dm p fillsync 0 3

Seg : TPTWSWCT  97    Proc : FILLSYNC  12
1233CE          (000):      JSR                 14EC        4EB8 14EC
1233D2(004):    ORI.B       #00,-(A0)           0020 0000
1233D6(008):    BSR.B       233E4               610C

MP:Debug> DM,FIndproc,SM,SW,Trapinfo,STopontrap,Cleartrap,
Zerodiv,Rom,Info,TRCpoints,CAlltrace,Availtime,COverage,TEmpstore

MP:Debug>

>>>dm s 97 12 0 3

Seg : TPTWSWCT  97    Proc : FILLSYNC  12
1233CE          (000):      JSR                 14EC        4EB8 14EC
1233D2(004):    ORI.B       #00,-(A0)           0020 0000
1233D6(008):    BSR.B       233E4               610C

MP:Debug> DM,FIndproc,SM,SW,Trapinfo,STopontrap,Cleartrap,
Zerodiv,Rom,Info,TRCpoints,CAlltrace,Availtime,COverage,TEmpstore

MP:Debug>
```

```
>>>dm r 123456 6

123456 (000):   BTST      #00,D5          0805 0000
12345A (004):   BEQ       236E4           6700 0288
12345E (008):   MOVEA.L   5310(A4),A3     266C 5310
123462 (00C):   LEA       0002(A3),A6     4DEB 0002
123466 (010):   MOVE.L    A6,002A(A5)     2B4E 002A
12346A (014):   MOVEA.L   5310(A4),A0     206C 5310

MP:Debug> DM,FIndproc,SM,SW,Trapinfo,STopontrap,Cleartrap,
Zerodiv,Rom,Info,TRCpoints,CAlltrace,Availtime,COverage,TEmpstore

MP:Debug>

>>>dm g 1233ce

098738  0000 0000 0009 976A 009C 0066 0003 0000 .......j...f....
```

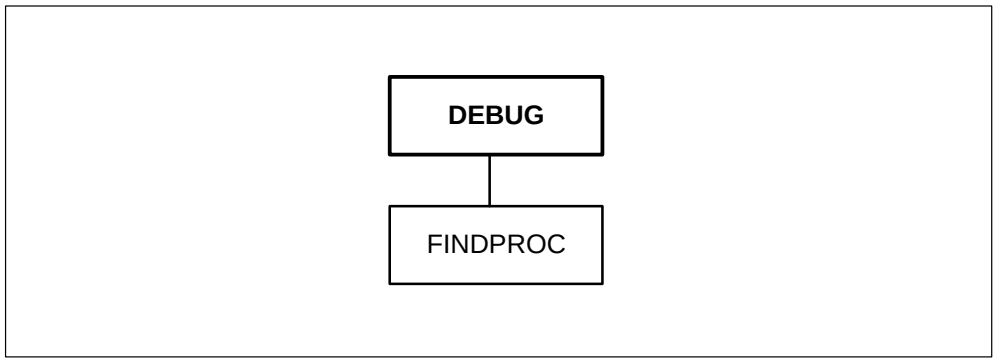
FINDPROC command

The FINDPROC command is issued from the DEBUG level.

The FINDPROC command displays the segment name and number, procedure name and number, and offset of all loaded procedures having that name. This command also displays the procedure name of the input address.

Figure 4-7 on page 4-21 diagrams how the FINDPROC command is accessed.

Figure 4-7
The FINDPROC command



Following is the FINDPROC command syntax:

FINDPROC	**name**	**address**

Where:

name is a procedure name with no underscores. This command searches all available segments for the procedure and displays the segment and procedure names and numbers.

address is an absolute address. This command scans all loaded procedures and verifies that the specified address falls inside a procedure body. The segment name and number, procedure name and number, and the procedure offset are displayed.

Example:

```
MP:Debug> DM,FIndproc,SM,SW,Trapinfo,STopontrap,Cleartrap,
Zerodiv,Rom,Info,TRCpoints,CAlltrace,Availtime,COverage,TEmpstore

MP:Debug>

>>>fi 12a420

Seg: UTRNAMP  129    Proc: STARTUTR  2  Offset: 15A
```

INFO level

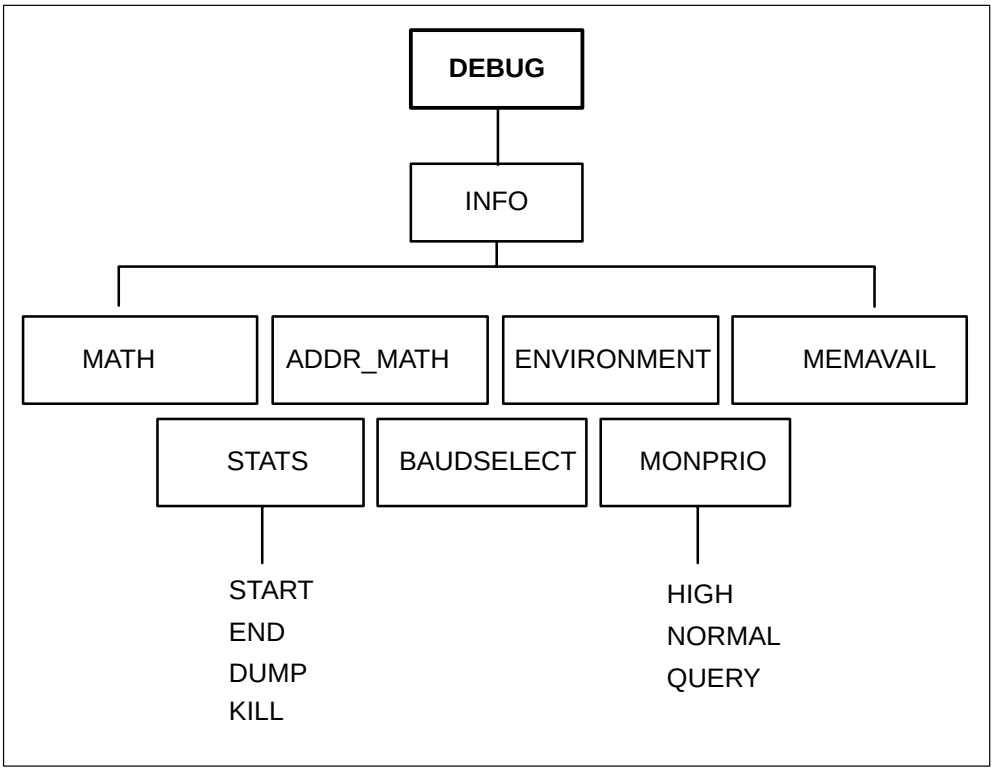
The INFO (information) level is accessed when the INFO command is entered from the DEBUG level.

The INFO level contains commands for the following functions:

- performing mathematical functions
- displaying system statistics
- displaying the current baud rate
- displaying a parameter in both hexadecimal and decimal
- displaying a list of the modules in an XPM
- displaying the priority of the BASEMON and MONIO tasks
- displaying the amount of memory available in the MP.

Figure 4-8 on page 4-23 contains the commands available in the INFO level.

Figure 4-8
Commands in the INFO level



Following are descriptions of the commands available in the INFO level:

MEMAVAIL command

The MEMAVAIL command displays the amount of memory available in the MP and the SP.

At low memory levels, many PMDEBUG tools cannot operate.

MEMAVAIL	

Example:

```
Math,Addr_math,Environment,Stats,Baudselect,MOnprio,MEmavail.
MP:INFO>

>>>me
```

```
MP --> 770618  bytes
SP -->  99240  bytes

MP:INFO>

>>>*

MP:DEBUG>
```

MATH command

The MATH command enters the MATH level. You can add, subtract, multiply, and divide single-word integers in the MATH level. You can enter integers in hexadecimal, decimal, or a mixture of both. The output is displayed in both hexadecimal and decimal.

MATH	

Example:

```
Math,Addr_math,Environment,Stats,Baudselect,MOnprio,MEmavail.
MP:INFO>

>>>math

<EXPR>, *
MP:MATH>

>>>7 + 3

#A = 10
<EXPR>, *
MP:MATH>

>>>*
```

ADDRESS MATH command

The ADDRESS MATH command adds and subtracts hexadecimal double word numbers. Integers are input in hexadecimal or decimal format, and the output is in hexadecimal.

ADDRESS MATH	

Example:

```
Math,Addr_math,Environment,Stats,Baudselect,MOnprio,MEmavail.
```

```
MP:INFO>
 >>>a

Math,Addr_math,Environment,Stats,Baudselect,MOnprio,MEmavail.
MP:INFO>

>>>a

<EXPR>, *
MP:ADDR_MATH>

>>>12 + 4

00016
<EXPR>, *
MP:ADDR_MATH>

>>>*
```

ENVIRONMENT command

The ENVIRONMENT command displays a list of the modules loaded in the XPM, the version code, the start address, and the length in bytes. The list is in numerical order according to segment number.

ENVIRONMENT	

Example:

```
Math,Addr_math,Environment,Stats,Baudselect,MOnprio,MEmavail.
MP:INFO>

>>>e

RTSS      AG05

SEG NO   NAME        VERSION      START ADDRESS   LENGTH

   0     SYS.PASCAL                                00171AB2
6876
   1     OSXPM       AG03         001581FC           256
   8     UTILMOD     AD09         0017369E         10514
   9     UTLCALLP    AC07         00173590           268
   .
   .
   .
```

STATS command

The STATS command dumps statistics on the system.

STATS	

Following are the commands at the STATS level:

START
starts collecting statistics

END stops the collection of statistics

DUMP displays the statistics. Pressing ENTER halts the output.

KILL releases the temporary storage acquired by the START command

Example:

```
Math,Addr_math,Environment,Stats,Baudselect,MOnprio,MEmavail.
MP:INFO>

>>>s

Start,End,Dump,Kill,*
MP:STATS>

>>>s

INTERVAL (IN 10 MSEC UNITS)
MP:STATS>

>>>1

CYCLIC BUFFERS (Y/N)
MP:STATS>

>>>y

OK!

Start,End,Dump,Kill,*
MP:STATS>

>>>d
```

```
Time : 00:02:18:01.83

inter    prio0    prio1    prio2    prio3    prio4    prio5    prio6    prio7  pmach intdf
  0        1        0        0        0        0        0        0        1        2    1

lckdf    slice    ipc     in   out   inuse    csin    csout    psin   psout   imcin   imcot
  0        0      0     0     0      24       0        0       0       0       0       0

ipmli ipmlo queue oavtr 1avtr 2avtr 3avtr
 0    0    0    0     0    0    0

IPC profile
csmac csmht cont tones chanl mpcsm mpab mpchn mpcon mtcac diags
  0    0    0    0     0    0    0    0    0    0    0    0

timeo timec
  0    0

cp: tptrq servr   msb: msbin msbot   trks: c6inc c6otg n6inc
      0    0          0    0            0    0    0

n6otg
   0

MTS/TPS: isend trans msgrs bldnt sched imsgs i2msg tmsgs
         0    0    0     0    0    0     0    0

>>>e

>>>k
```

Note: In the previous example, the sequence of statistics is repeated every millisecond.

Following is a list of abbreviations that appear in the output from the STATS command:

inter number of interrupts

prio0 number of task scheduled with priority 0

prio1 number of task scheduled with priority 1

prio2 number of task scheduled with priority 2

prio3 number of task scheduled with priority 3

prio4	number of task scheduled with priority 4
prio5	number of task scheduled with priority 5
prio6	number of task scheduled with priority 6
prio7	number of task scheduled with priority 7
pmach	number of P-machine instructions
intdf	number of scheduler interrupts deferred
lckdf	number of scheduler locks deferred
slice	number of tasks time-sliced out
ipc	number of interprocessor communication IPC actions (such as get, send, release)
in	number of IPCs into the processor
out	number of IPCs out of the processor
inuse	number of IPC buffers in use
csin	number of C-side messages incoming
csout	number of C-side messages outgoing
psin	number of P-side messages incoming
psout	number of P-side messages outgoing
imcin	number of IMC messages incoming
imcot	number of IMC messages outgoing
ipmli	number of IPML messages incoming
ipmlo	number of IPML messages outgoing
queue	number of queue operations
0avtr-3avtr	message capacity for available transceivers
csmac	number of SP CSM requests

csmht	number of SP CSM single-plane hits
cont	number of SP continuity requests
tones	number of SP tone requests
chanl	number of SP channel requests
mpcsm	number of MP CSM requests
mpab	number of MP A/B bits
mpchn	number of MP channel requests
mpcon	number of MP continuity requests
mtcac	number of MTC requests
diags	number of diagnostics run
timeo	number of timeouts
timec	number of timer cancellations
tptrq	number of TPT requests
servr	number of server message decode sequences
msbin	number of MSB messages incoming
msbot	number of MSB messages outgoing
c6inc	number of C6 trunk incoming messages
c6otg	number of C6 trunk outgoing messages
n6inc	number of N6 trunk incoming messages
n6otg	number of N6 trunk outgoing messages
isend	number of international messages sent
trans	number of cross processor messages
mstrs	number of masters called
bldnt	number of networks created

sched number of SCBs processed

imsgs number of IMSGs processed

i2msg number of I2MSGs processed

tmsgd international message traffic in TPS.

BAUDSELECT command

The BAUDSELECT command displays the current baud rate and allows you to increase or decrease the baud rate.

BAUDSELECT	

The baud rate options are as follows:

0 300

1 1200

2 2400

3 4800

4 9600

Example:

```
Math,Addr_math,Environment,Stats,Baudselect,MOnprio,MEmavail.
MP:INFO>

>>>b

CURRENT BAUD RATE IS 9600
0-300, 1-1200, 2-2400, 3-4800, 4-9600
MP:Baudselect>

>>>*
```

MONPRIO command

The MONPRIO command displays the priority of the BASEMON and the MONIO tasks. You may change the priority of these tasks.

MONPRIO	

Example:

```
Math,Addr_math,Environment,Stats,Baudselect,MOnprio,MEmavail.
MP:INFO>

>>>mo

High,Normal,Query
MP:MOnprio>

>>>q

BASEMON3
MONIO3
MP:MOnprio>

>>>h

MP:MOnprio>

>>>q

BASEMON7
MONIO7
MP:MOnprio>

>>>n

MP:MOnprio>

>>>q

BASEMON3
MONIO3
MP:MOnprio>

>>>*
```

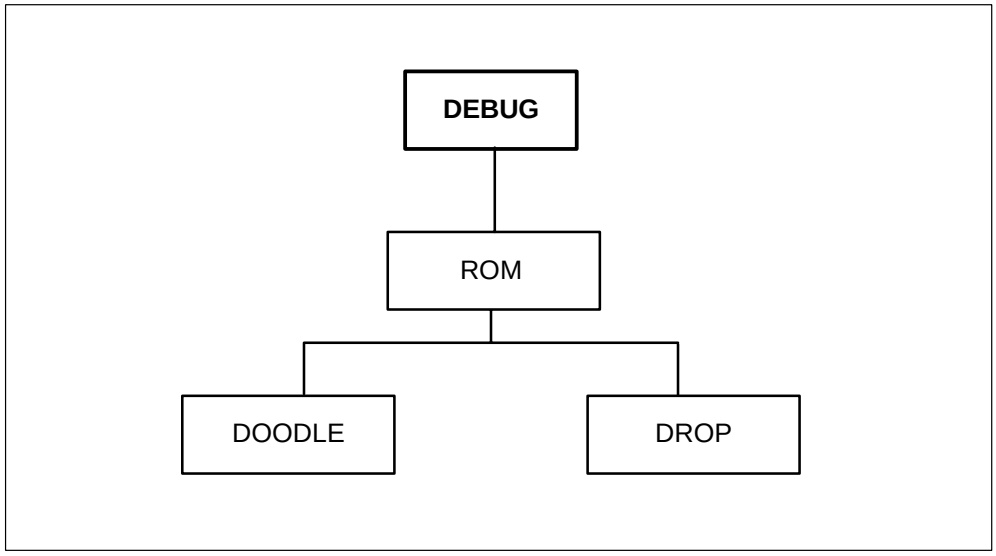
ROM level

CAUTION
Do *not* use the ROM level on a live switch.
Commands at the ROM level alter peripheral memory, requiring the peripheral to be reloaded. The peripheral is taken out of service.

The ROM level is accessed when the ROM command is entered from the Debug level. The ROM level gives you access to the MP read only memory, and the peripheral is taken out of service. No XPM processing occurs.

Figure 4-9 on page 4-32 contains the commands available at the ROM level.

Figure 4-9
Commands in the ROM level



Following are descriptions of the commands available in the ROM level:

DOODLE command

CAUTION
Do not use the DOODLE command on a live switch.
The DOODLE level alters peripheral memory, requiring the peripheral to be reloaded. The peripheral is taken out of service.

The DOODLE command enters the DOODLEBUG ROM and allows you to return to the DEBUG level by issuing the ROM G command.

The DOODLE command takes full control of the central processing unit (CPU).

DOODLE	

DROP command

CAUTION
Do *not* use the DROP command on a live switch.
The DROP command alters peripheral memory, requiring the peripheral to be reloaded.

The DROP command sets a flag that signals the program to drop into doodlebug before initialization prior to a system restart. This command is used when breakpoints need to be set before initialization begins.

DROP	

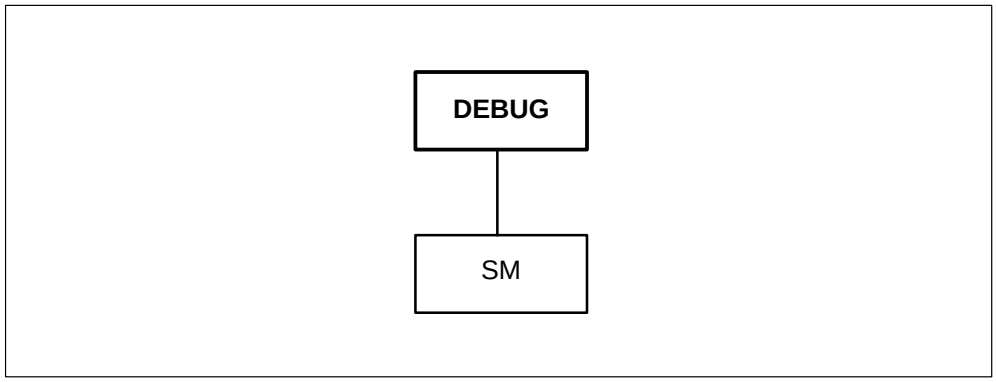
SM command

The SM command is entered from the DEBUG level.

The SM command sets memory address data, up to 16 bytes. Only valid addresses can be written to. This command only performs byte writes. Refer to “SW command” on page 4-35 for information on word writes.

Figure 4-10 page 4-33 diagrams how the SM command is accessed.

Figure 4-10
The SM command



Following is the SM command syntax:

SM	address	data

Where:

address is the absolute address, up to 16 bytes

data is the data to be written in memory, up to 16 bytes

Note: Data must be entered one byte at a time, for example, **SM 1000 12 34** where 12 and 34 are two hexadecimal digits. Pressing ENTER halts the display of output.

Example:

```
DM,FIndproc,SM,SW,Trapinfo,STopontrap,Cleartrap,Zerodiv,
Rom,Info,TRCpoints,CAlltrace,Availtime,COverage,TEmpstore.
MP:Debug>

>dm r 1230de

1230DE          (000):  ADDA            A4,A4           D8CC
1230E0 (002):   Invalid Opcode          02D9
1230E2 (004):   CMPA.L  A6(A6,A5),A4    B9F6 D7A6
1230E6 (008):   ADDI    #474E,A1        0649 474E
1230EA(00C):    LEA     (A2),A7         4F52
1230EC          (00E):  LEA             -(A7),A2        45E7
1230EE(010):    MOVEP   B9F2(A2)D7      0F0A B9F2
1230F2 (014):   ADD.L   D3,-(A6)        D7A6

1230F4 (016):   ADDI    #4E49,D4        0644 4E49
Cont.?(Y,<cr>)
MP:DM>

MP:Debug>

DM,FIndproc,SM,SW,Trapinfo,STopontrap,Cleartrap,Zerodiv,Rom,
Info,TRCpoints,CAlltrace,Availtime,COverage,TEmpstore.
MP:Debug>

>sm 1230e0 4e 71

Old values:
1230E0  02D9                    ..
'y' to verify

MP:SM>

>y

1230E0  4E71                    NqMP:Debug>

DM,FIndproc,SM,SW,Trapinfo,STopontrap,Cleartrap,Zerodiv,Rom,
Info,TRCpoints,CAlltrace,Availtime,COverage,TEmpstore.
```

```
MP:Debug>

>dm r 1230e0

1230E0 (000):   NOP                          4E71
1230E2 (002):   CMPA.L   A6(A6,A5),A4        B9F6 D7A6
1230E6 (006):   ADDI     #474E,A1            0649 474E
Cont.?(Y,<cr>)
MP:DM>
```

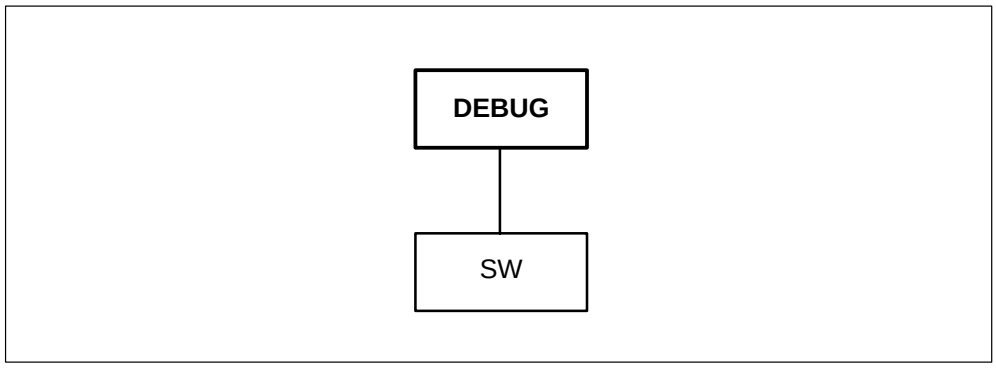
SW command

The SW command is entered from the DEBUG level.

The SW command sets word-address data (one word performs a word write). This command performs a full word write to the address specified by the data parameter. Odd addresses and protected memory cannot be written to.

Figure 4-11 on page 4-35 diagrams how the SW command is accessed.

Figure 4-11
The SW command



Following is the SW command syntax:

SW	address data

Where:

address is the one-word absolute address

data is one word of data

Example:

```
MP:Debug>

>dm r 1230e0

1230E0 (000):   NOP                         4E71
1230E2 (002):   CMPA.L   A6(A6,A5),A4       B9F6 D7A6

1230E6 (006):   ADDI     #474E,A10649       474E
Cont.?(Y,<cr>)
MP:DM>

MP:Debug>

>sw 1230e0 02d9

Old val's:
1230E0  0002                        ..
'Y' to verify

MP:SW>

>y

1230E0  02D9                        ..MP:Debug>

DM,FIndproc,SM,SW,Trapinfo,STopontrap,Cleartrap,Zerodiv,Rom,
Info,TRCpoints,CAlltrace,Availtime,COverage,TEmpstore.
MP:Debug>

>dm r 1230e0

1230E0 (000):   Invalid  Opcode             02D9
1230E2 (002):   CMPA.L   A6(A6,A5),A4       B9F6 D7A6
1230E6 (006):   ADDI     #474E,A1           0649 474E

1230EA(00A):    LEA      (A2),A7            4F52
1230EC          (00C):   LEA                -(A7),A2    45E7
Cont.?(Y,<cr>)
MP:DM>

>*

MP:Debug>
```

STOPONTRAP command

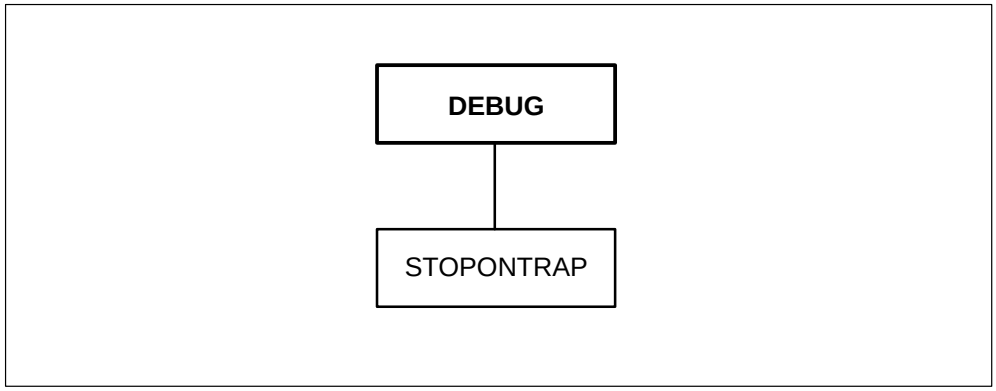
The STOPONTRAP command is entered from the DEBUG level.

CAUTION
Do *not* use the STOPONTRAP level on a live switch.
Commands at the STOPONTRAP level alter peripheral memory, requiring the peripheral to be reloaded.

The STOPONTRAP command specifies that the next occurring trap causes the peripheral to drop into doodlebug. Turning STOPONTRAP ON stops the machine in the exact state in which the trap occurred, allowing in-depth debugging.

Figure 4-12 on page 4-37 diagrams how the STOPONTRAP command is accessed.

Figure 4-12
The STOPONTRAP command



Following is the STOPONTRAP command syntax:

STOPONTRAP	**ON**	**OFF**

Where:

ON turns STOPONTRAP on

OFF turns STOPONTRAP off

TRAPINFO level

The TRAPINFO level is accessed when the TRAPINFO command is entered at the DEBUG level.

The TRAPINFO level displays trap information saved in the trap buffer. This level provides information such as the type of trap that occurred, the

task that was processing when the trap occurred, the trap count (number of traps since the last reload), the location of an error trap, and a traceback detailing the procedures that called the trapped procedure. A copy of the various registers at the time of the trap and a portion of the stack are also displayed.

For examples of using the commands in the TRAPINFO level, refer to Figure 7-2 on page 7-8.

The TRAPINFO command enters the trap level only if traps have been saved in the trap buffer.

Two types of traps occur in the MP and the SP. The first type of trap is caused by software errors such as:

- dividing by zero
- range checks
- user-defined traps
- task errors

The second type of trap is hardware traps, such as:

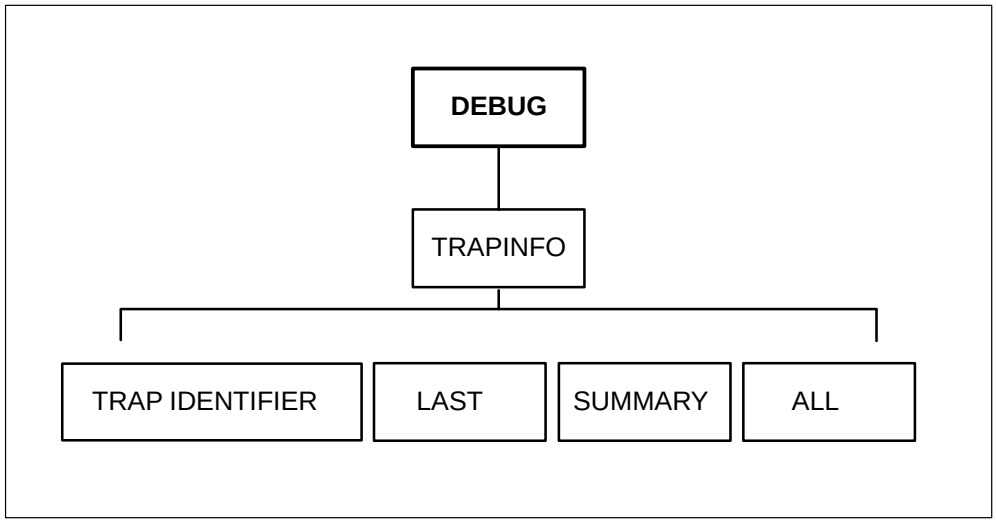
- parity errors
- bus errors
- memory management unit (MMU) errors
- addressing errors

Up to eight traps can be saved in the trap buffer. Traps are saved over resets and reloads if the new load is the same software release as the previous one. This ensures that the trap buffer location has not changed. If the XPM is reloaded with a different load, the traps may not be available. If the trap buffer is located in the same place in memory as the previous load, and if program store (PS) is larger or smaller than the previous load, tracebacks of old traps are not displayed as procedures and offsets. Only hexadecimal offsets are displayed.

Each trap produced has a time stamp indicating when the trap occurred. This time stamp reflects the central control (CC) time, if the peripheral is in-service. CC time is the day of the month followed by the time of day. Text indicates whether the time stamp is the CC or the peripheral (XPM) time.

Figure 4-13 on page 4-39 shows the commands available in the TRAPINFO level.

Figure 4-13
Commands in the TRAPINFO level



The following commands are contained in the TRAPINFO level:

Trap identifier command

Each trap stored in the trap buffer has a unique trap identifier. When the trap identifier is entered, information for that trap, including a traceback, is displayed. If the XPM is reloaded, traps remaining in the trap buffer have their trap identifiers renumbered sequentially with the least recent trap assigned as 1.

END	

Note: The trap identifier is from 1 to 8.

LAST command

The LAST command displays the complete trap information on the most recently occurring trap in the trap buffer.

LAST	

SUMMARY command

The SUMMARY command displays a list of all traps in the trap buffer. The information displayed includes the trap identifier, the segment number, and an error indicator.

SUMMARY	

ALL command

The ALL command dumps all trap information stored in the trap buffer, starting with the most recent trap.

ALL	

TRCPOINTS level

The TRCPOINTS level is accessed when the TRCPOINTS command is entered from the DEBUG level.

You are prompted for a tracepoint number (0-15), the address where the tracepoint will be inserted, and a memory location or register number for conditioning the tracepoint.

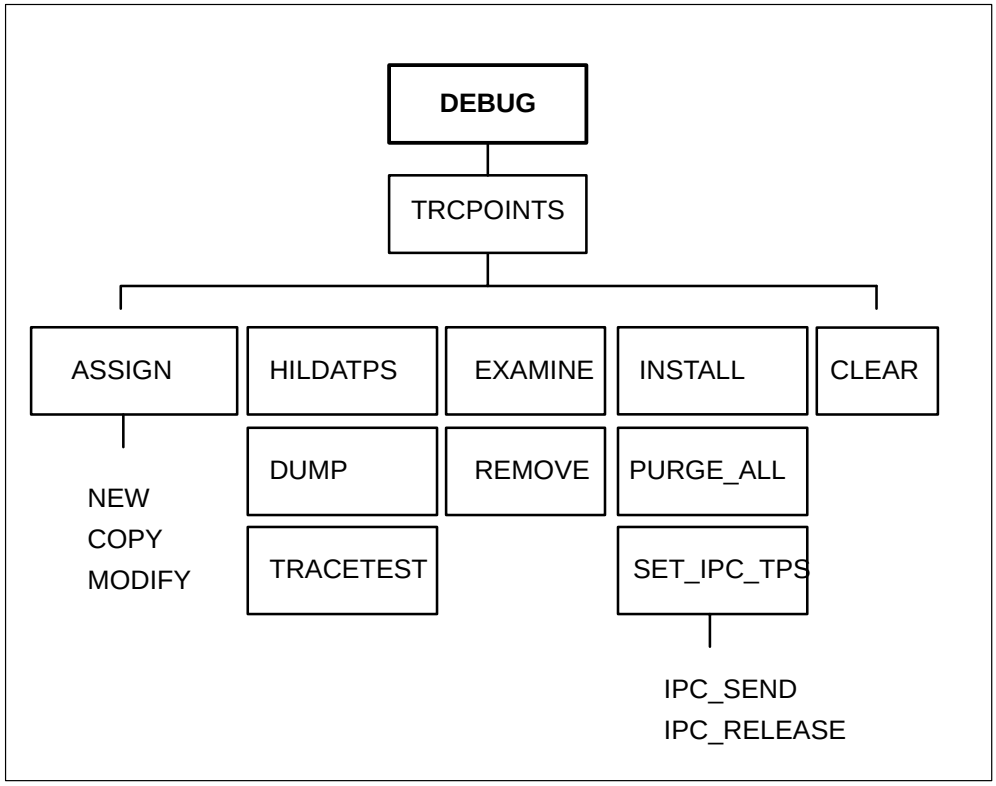
If a tracepoint is assigned twice, the tracepoint is overwritten and previously gathered information is lost.

The address where the tracepoint is inserted must be a word boundary where a valid 68000 opcode resides. Addresses can be determined with the FINDPROC command (refer to) or with the DM command (refer to).

> ***Note:*** An opcode is the identifier of a specific operation to be performed. Each primitive is assigned an opcode.

Figure 4-14 on page 4-41 contains the commands available at the TRCPOINTS level.

Figure 4-14
Commands in the TRCPOINTS level



CAUTION

Observe the following cautions:

- To prevent two different tools from inspecting the same instructions, the Tracepoint tool should not be used simultaneously with other tools that inspect code.
- Tracepoints can be set in any code other than the tracepoint code or code used by both the MP and the SP or both the MP and the FP.
- Tracepoints should not be placed in P-code.

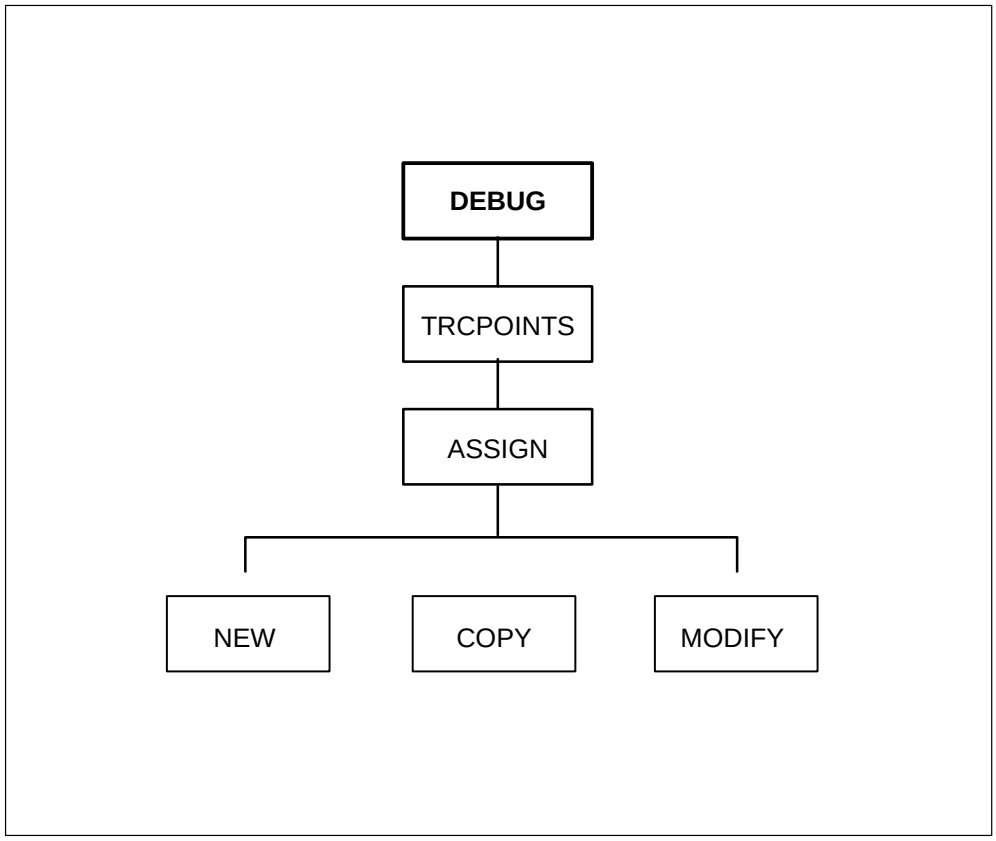
Refer to Figure 4-14 on page 4-41 for an example of using the commands at the TRCPOINTS level.

The following descriptions are of the commands available in the TRCPOINTS level:

ASSIGN level

The ASSIGN level is accessed when the ASSIGN command is entered at the TRCPOINTS level.

Figure 4-15
Commands in the ASSIGN level



Following are descriptions of the commands available in the ASSIGN level:

NEW command

The NEW command assigns a new tracepoint. This command does not implement the tracepoint. Tracepoint implementation is processed with the INSTALL command. Refer to “INSTALL” on page 4-44.

COPY command

The COPY command copies an old tracepoint record into a new tracepoint record. You are asked if any changes will be made to the tracepoint before the tracepoint is assigned. You are then prompted for the number of hits.

MODIFY command

The MODIFY command allows you to change the specifications of an existing tracepoint entered with the ASSIGN NEW command. The MODIFY command cannot be used once a tracepoint has been inserted into the code with the INSTALL command.

> ***Note:*** **When the MODIFY command is initiated, all the buffers that were previously assigned are lost. Therefore, this command should be used with caution.**

The MODIFY command provides access to the following commands:

ADDRESS allows you to modify the tracepoint address

TRACEBACK allows you to modify the traceback option

REGISTER_DUMP allows you to modify the register dump option

MEMORY_DISPLAY allows you to modify the memory location display option

CONDITIONAL allows you to modify the conditional output option

STACK_DISPLAY allows you to modify the long word or stack display

EXAMINE allows you to display all information associated with a tracepoint

STACK_FRAME_DISPLAY allows you to select whether stack frame data is to be displayed in a tracepoint.

SURVIVE_RESTARTS allows you to modify the survive restart option

Note: Each command in the MODIFY level corresponds to a field associated with the ASSIGN NEW command.

Fields associated with the ASSIGN command are modified whenever one of the MODIFY commands are entered. For example, when the ADDRESS command is entered, the tracepoint address is modified. All buffers previously belonging to the address are then lost.

EXAMINE command

The EXAMINE command dumps the assigned tracepoint and displays the information that will be received from the tracepoint.

EXAMINE	trcptno

Where:

trcptno is the tracepoint number to be examined.

INSTALL command

The INSTALL command inserts a tracepoint into the code. Only tracepoints created with the ASSIGN command can be inserted into the code with the INSTALL command.

INSTALL	trcptno

Where:

trcptno is the tracepoint number to be inserted

CLEAR command

The CLEAR command reinitializes the data buffer pointers. All previous data is lost and temporary store is reused.

CLEAR	trcptno

Where:

trcptno is the tracepoint number to be cleared

DUMP command

The DUMP command displays the data gathered at a tracepoint.

DUMP	trcptno

Where:

trcptno is the tracepoint number to be displayed

REMOVE command

The REMOVE command removes a tracepoint that has been inserted into the code with the INSTALL command.

REMOVE	trcptno

Where:

trcptno is the tracepoint number to be removed

PURGE_ALL command

The PURGE_ALL command removes all tracepoints inserted with the INSTALL command.

PURGE_ALL	

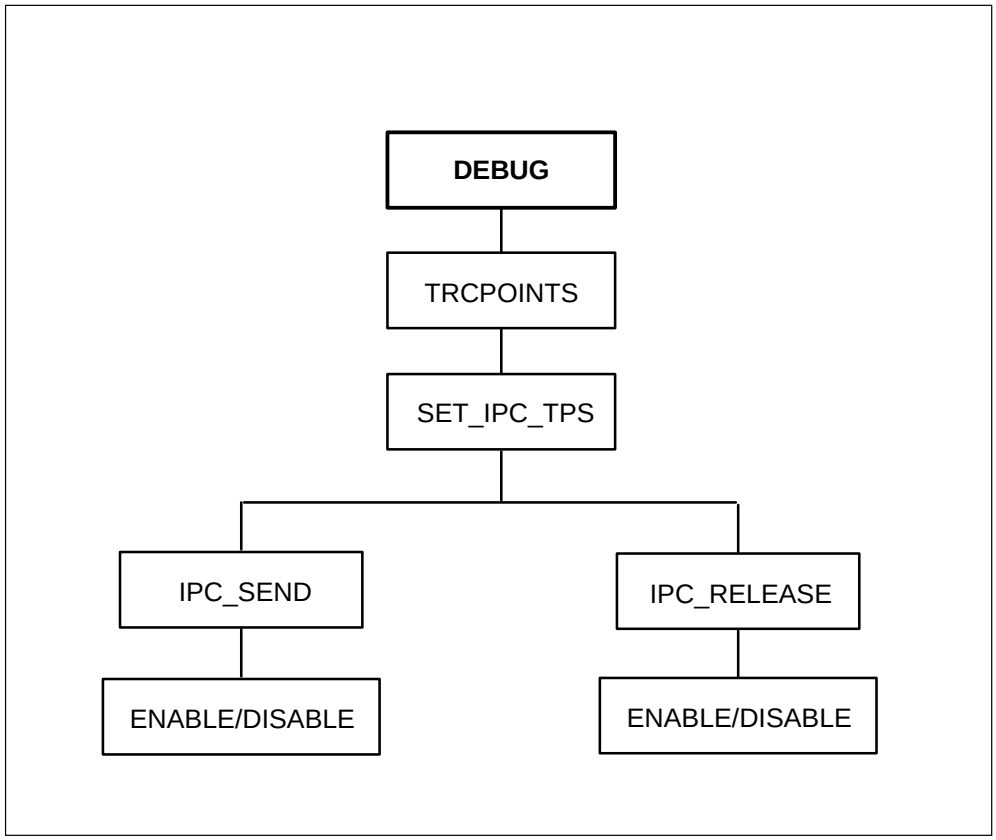
SET_IPC_TPS level

You cancollect IPC communication data by setting tracepoints in the selective message trace code of the assembly procedures IPCSEND and IPCRELEASE. These tracepoints are used in conjunction with message trace (MSGTRC) facility. When a message trace is enabled, messages that match the message trace criteria will affect the IPC tracepoint and pertinent information will be collected. This data gives insight into the task that originated a particular IPC message and may prove beneficial in debugging task communication problems. The SET_IPC_TPS level is accessed when the SET_IPC_TPS command is entered at the TRCPOINTS level.

> ***Note:*** SET_IPC_TPS will produce no results unless the message trace is enabled and an IPC message matches the criteria defined in MSGTRC.

Set_ipc_tps	

Figure 4-16
Commands in the SET_IPC_TPS level



IPC_SEND command

The IPC_SEND command automatically installs a tracepoint in the selective message trace code of this assembly procedure. This tracepoint is used in conjunction with the message trace utility (MSGTRC). Messages that match the selective message trace criteria will hit the tracepoint and pertinent data will be captured.

IPC_SEND	

IPC_RELEASE command

The IPC_RELEASE command automatically installs a tracepoint in the selective message trace code of this assembly procedure. This tracepoint is used in conjunction with the message trace utility (MSGTRC). Messages that match the selective message trace criteria will hit the tracepoint and pertinent data will be captured.

IPC_RELEASE	

The following commands, ENABLE/DISABLE, are available for the IPC_SEND and IPC_RELEASE levels.

ENABLE command

The ENABLE command enables the appropriate IPC_SEND or IPC_RELEASE tracepoint by installing this tracepoint in the selective message trace found in each of these procedures. This tracepoint will provide a procedure traceback and will not survive restarts.

ENABLE	

DISABLE command

The DISABLE command disables the appropriate IPC_SEND or IPC_RELEASE tracepoint by removing this tracepoint from the code.

DISABLE	

Following are examples of the ASSIGN and MODIFY commands for the TRACEPOINTS level:

Example: ASSIGN command

```
MP:Debug> DM,FIndproc,SM,SW,Trapinfo,STopontrap,Cleartrap,
Zerodiv,Rom,Info,TRCpoints,CAlltrace,Availtime,COverage,TEmpstore

MP:Debug>

>>>t

Assign,Examine,Install,Clear,Dump,Remove,Purge_all,Tracetest.
MP:TRCpoints>

>>>a

New,Copy,Modify
MP:Assign>

>>>n

TP # (0-15)
MP:New>

>>>0

Enter absolute address or S seg(#,name) proc(#,name) offset
MP:New>
```

```
>>>17d6b2

Select the options you want
-Traceback (y/n), and stack traceback (s)
MP:New>

>>>y

-Reg dump (y/n)
MP:New>

>>>y

-Mem location display (y/n)
MP:New>

>>>y

Do you want a (L)ongword or (B)yte
MP:New>

>>>l

Enter absolute address, or G offset
MP:New>

>>>g

Enter offset
MP:New>

>>>329a

Do you want a (L)ongword or (B)yte
MP:New>

>>>*

-Display longword on stack (y/n)
MP:New>

>>>y

Offset (hexadecimal)
MP:New>
```

```
>>>10

-Conditional (y/n)
MP:New>

>>>n

-Survive Restarts (y/n)
MP:New>

>>>n
```

Example: MODIFY command

```
MP:Debug> DM,FIndproc,SM,SW,Trapinfo,STopontrap,Cleartrap,
Zerodiv,Rom,Info,TRCpoints,CAlltrace,Availtime,COverage,TEmpstore

MP:Debug>

>>>t

New,Copy,Modify
MP:Assign>

>>>m

TP # (0-15)
MP:Modify>

>>>0

Address,Traceback,Mem_dsply,Reg_dmp,Conditions,Examine,Stk_dsply,
surVive
MP:Modify>

>>>c

Do you want (M)em or (R)eg equality
MP:Modify>

>>>m
```

```
Enter absolute address or S seg(#,name) proc(#,name) offset

>>> <cr>

MP:Assign>
```

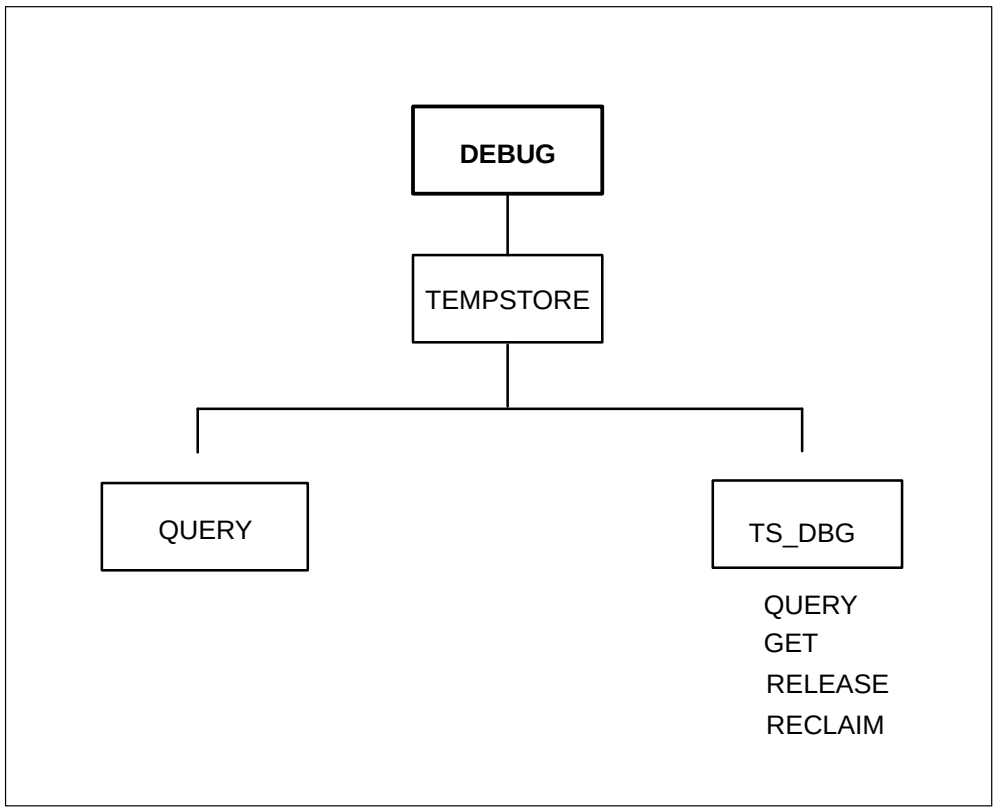
TEMPSTORE level

The TEMPSTORE level is accessed when the TEMPSTORE command is entered from the DEBUG level.

The TEMPSTORE level contains commands for querying the amount of used and free temporary storage as well as commands for allocating, reclaiming, and releasing temporary store.

Figure 4-17 on page 4-50 contains the commands available at the TEMPSTORE level.

Figure 4-17
Commands in the TEMPSTORE level



Following are descriptions of the commands available in the TEMPSTORE level:

QUERY command

The QUERY command displays information concerning the occupied segments of temporary storage and the amount of free temporary storage. The size of temporary storage is the logically accessible storage.

This command can be used to determine which tools to disable if temporary storage is full or fragmented.

QUERY	

TS_DBG command

The TS_DBG command is a detailed version of the QUERY command. The TS_DBG command provides access to temporary storage debugging tools.

TS_DEBUG	

The tools at the TS_DBG level are as follows:

QUERY displays the name of the tool that has access to temporary storage, the address and size of temporary storage, the processor being queried (MP, SP, FP), the procedure variable (ts_nil_pvar, nil_pvar, recover_pvar), and the temporary storage recoverability (recovered, pending).

Note: The procedure variable is passed to temporary storage. It points to the routine to be used for cleanup during the reclaiming of storage.

Note: Temporary storage is recoverable between restarts. All temporary storage memory is set to PENDING after a restart. If a tool reclaims the same piece of memory, the status is changed to RECOVERED.

GET is a debugging tool that requests temporary storage. You are prompted for the size of temporary storage and a pseudotool name. If a restart occurs during the use of this command, temporary storage is not recoverable.

RELEASE is a debugging tool that allows a pseudotool to release temporary storage so the storage segment can be made available to other tools. You are prompted for the pseudotool name.

RECLAIM is a debugging tool that allows a pseudotool to attempt to reaccess the same segment of temporary storage that the tool

owned prior to a restart. You are prompted for the pseudotool name.

Example:

```
MP:Debug> DM,FIndproc,SM,SW,Trapinfo,STopontrap,Cleartrap,
Zerodiv,Rom,Info,TRCpoints,CAlltrace,Availtime,COverage,TEmpstore

MP:Debug>

>>>te

Query,Ts_dbg
MP:TEmpstore>

>>>q

                    SEGMENT          SIZE          PROC
                    ____________________________________
                    FREE             00002454      --
                    TRACEPOINTS      00007D00      MP
                    CALLTRACE        00007F26      MP
                    FREE             0004BEF7      --

MP:TEmpstore>

>>>t

Query,Get,Release,REClaim.
MP:Ts_dbg>

>>>q

  TOOL      ADDR      SIZE      PROC     PVAR      DEAD
_____________________________________________________________________
_
0  FREE             000663B6  00002454   MP  ts_nil_pvar  recovered
1  TRACEPOINTS      0006AD34  00007D00   MP  recover pvar recovered
2  CALLTRACE        0007A80A  00007F26   MP  recover pvar recovered
3  FREE             0008A72C  0004BEF7   MP  ts_nil_pvar  recovered
 .
 .
 .
```

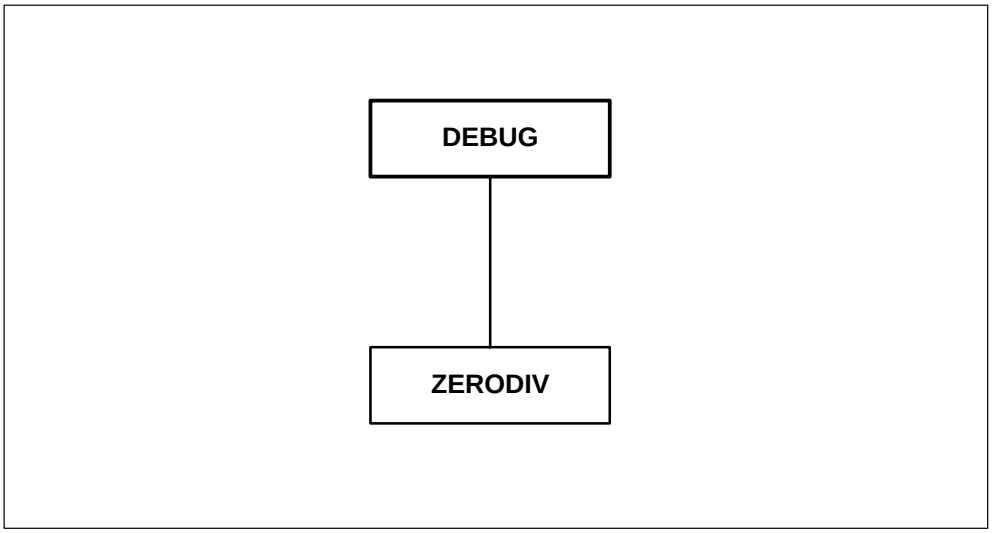
ZERODIV command

CAUTION
Do not use the ZERODIV command on a live switch.
The ZERODIV command causes a trap in the peripheral, which can cause a peripheral to go down.

The ZERODIV command is entered from the DEBUG level. The ZERODIV command divides by zero to force a trap.

Figure 4-18 on page 4-53 diagrams how the ZERODIV command is accessed.

Figure 4-18
The ZERODIV command



Following is the ZERODIV command syntax:

ZERODIV	

SWERR level

The software error (SWERR) level is accessed when the SWERR command is entered at the LTCMP level.

A SWERR indicates an error condition in the software or displays information about a software condition.

SWERRs are inserted into a software program during initial testing for debugging. If a SWERR is generated during testing, the designer must debug the code.

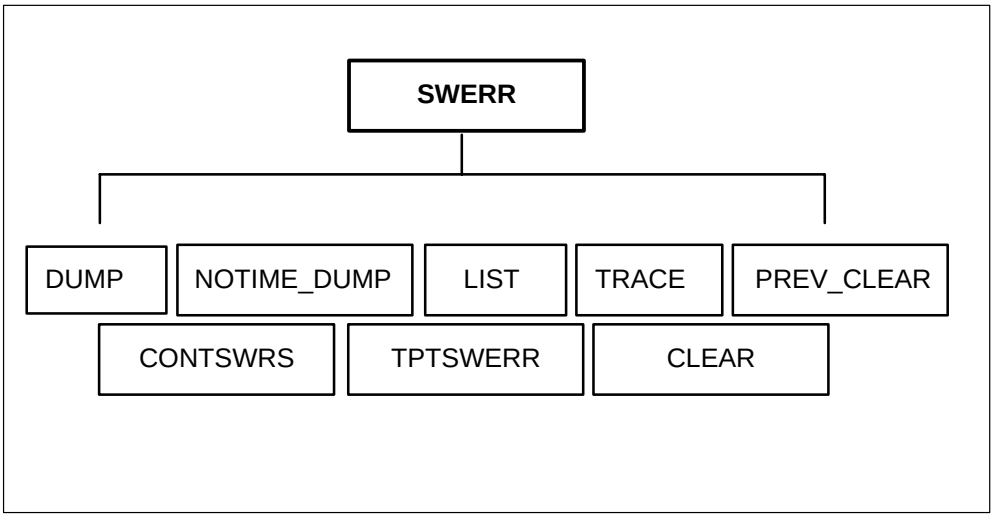
Some SWERRs are left in the code to indicate that an improbable or undesirable event has occurred in the software.

SWERRs contain two parts.

- programmer-supplied
  - — error classification
  - — text message
  - — integer REASON
- LOGSYSTEM-supplied
  - — time and date
  - — traceback, which is a record of the procedures processed by the software, showing the sequence in which the instructions were processed

Figure 4-19 on page 4-55 contains the commands available at the SWERR level.

Figure 4-19
Commands in the SWERR level



The SWERR level contains commands that allow you to perform such functions as collecting SWERRs, listing SWERRs, and dumping SWERR tracebacks.

Following are descriptions of the commands available in the SWERR level:

CLEAR command

The CLEAR command clears the SWERR buffer and activates SWERR collection. One SWERR is left in the SWERR buffer indicating that the SWERRs have been reset.

CLEAR	

> ***Note:*** The CLEAR command clears the SWERR buffer and activates SWERR collection. One SWERR is left in the SWERR buffer indicating that the SWERRs have been reset.

CONTSWRS command

The CONTSWRS command enables continuous collection and display of SWERRs.

CONTSWRS	**ON** **OFF**

Following are the commands available at the CONTSWRS sublevel:

ON turns CONTSWRS on

OFF turns CONTSWRS off.

DUMP command

The DUMP command lists all the SWERRs. The DUMP command does not include a traceback.

DUMP	

LIST command

The LIST command lists the SWERR you specified. The LIST command does not include a traceback.

LIST	selector

Where:

selector is the SWERR to be listed. The following options can be used with this parameter:

- If nothing is entered for this parameter, all SWERRs are listed.
- If an integer **n** is entered, the last **n** SWERRs are listed. For instance, if a **3** is entered, the last three SWERRs are listed.
- If **MARK** is entered, all SWERRs are listed.
- If **MARK n** is entered, the SWERR with that mark is listed.
- If **MARK n1 n2** is entered, all SWERRs with marks in the range between n1 and n2 are listed.

Examples:

L	lists all SWERRs
L 12	lists the last 12 SWERRs
L Mark	lists all SWERRs
L Mark 1208	lists SWERR marked 1208
L Mark 24 36	lists all SWERRs whose marks are in the range 24 to 36

NOTIME_DUMP command

The NOTIME_DUMP command dumps SWERR information without the PP or CC time.

NOTIME_DUMP	

PREV_CLEAR command

The PREV_CLEAR command clears the list of SWERRs that occurred in the previous load or that occurred prior to the last return to service (RTS).

PREV_CLEAR	

TPTSWERR level

The TPTSWERR level collects extra data when a SWERR in the TPT task occurs.

TPTSWERR	

Following are the commands at the TPTSWERR level:

TID DUMP when set to ON, displays the last four bytes of the SWERR, which is the (TID)

When set to OFF, this command displays the last four bytes of a primitive trace or a message trace

CAPTURE allows the selection of specific SWERRs associated with the data block to be captured when a TPT SWERR occurs

REPEAT when set to ON, captures a SWERR when it reoccurs

DUMP displays all blocks captured

QUERY displays the status of the options available at the TPTSWERR level

KILL disables selective capturing of SWERR information

TRACE command

The TRACE command lists the load information followed by the selected SWERR. The TRACE command includes associated tracebacks.

If the TRACE command is entered without the count parameter, all SWERRs are traced.

TRACE	count

Where:

count the number of SWERRs to be traced, starting with the most recent SWERR

Examples:

```
MP:Debug> DM,FIndproc,SM,SW,Trapinfo,STopontrap,Cleartrap,
Zerodiv,Rom,Info,TRCpoints,CAlltrace,Availtime,COverage,TEmpstore

MP:Debug>

>>>s

Dump,Notime_dump,List,Trace,Clear,Prev_clear,COntswrs.
MP:Swerr>

>>>t

RAM LOAD NAME = ST722BH
MP ROM NAME = XPMRFA10, SP ROM NAME = XPMRFA10
SWERR SEQUENCE #  4746   ...during restart...
 Task: nilname    Restart
 PP Time: 00:00:00:27.56      current load
 Data: 0A 00 3F 00  00 00 00 00 00 00
 called from :    001BC93A    MONID        19     MAINLINE   109    offset: #36
                  0015823A    OSXPM         1     INITXPMO     1    offset: #2A
                  00128F29    Addr invalid

SWERR Sequence #  4745
 Task : MMTCTASK   Bad Selector
 PP Time: 00:00:00:39.56      current load
 Data: 52 81 2D 81
 called from : 001A7206       MPMTC        53     MTCREQHA 54    offset: #8D2
                  001A767E    MPMTC        53     MTCTASK    11    offset: #EE
  .
  .
  .

Dump,Notime_dump,List,Trace,Clear,Prev_clear,COntswrs.
MP:Swerr>

>>>l

SWERR Sequence # 4748
 Task : TPT   trk recover
 CC Time : 09:20:31:38.77
 PP Time : 00:00:26:10.04     current load
 Data: 01 29 00 95 07
```

```
SWERR Sequence # 4747
 Task : TPT   trk recover
 CC Time : 09:20:31:38.77
 PP Time : 00:00:26:10.04     current load
 Data: 01 28 00 84 07

SWERR Sequence # 4746
 Task : nilname  Restart
 PP Time : 00:00:00:27.56     current load
 Data: 0A 00 3F 00  00 00 00 00  00

SWERR Sequence # 4745
 Task : MTCTASK  bad selector
 PP Time : 00:00:00:39.56     current load
 Data: 52 81 2D 81

Dump,Notime_dump,List,Trace,Clear,Prev_clear,COntswrs.
MP:Swerr>

>>>n

SWERR Sequence # 4748
 Task : TPT     trk recover
                current load
 Data: 01 29 00 95 07

SWERR Sequence # 4747
 Task : TPT     trk recover
                current load
 Data: 01 28 00 84 07

SWERR Sequence # 4746    ...during restart...
 Task : nilname   Restart
                current load
 Data: 0A 00 3F 00  00 00 00 00  00

SWERR Sequence # 4745
 Task : MTCTASK   bad selector
                current load
 Data: 52 81 2D 81

Dump,Notime_dump,List,Trace,Clear,Prev_clear,COntswrs.
MP:Swerr>

>>>co
```

```
ON,OFf
MP:COntswrs>

>>>*
```

Interprocessor communications level

IPC level

The interprocessor communications (IPC) level is accessed when the IPC command is entered at the LTCMP level.

Tasks are independent programs that accomplish specific functions. Tasks can communicate with one another by using IPC software. IPC software allows tasks to communicate by sending and receiving buffers containing messages.

Each task is identified by one or more unique communication identifiers (CID). A CID identifies a task, just as an address identifies a house. A task may have more than one CID associated with it, but one CID cannot identify more than one task. Every message a task sends is sent to a CID.

The IPC level of PMDEBUG allows you to obtain working buffers with the GET and DEQUEUE commands and release the buffers with the SEND, RELEASE, and QUEUE commands.

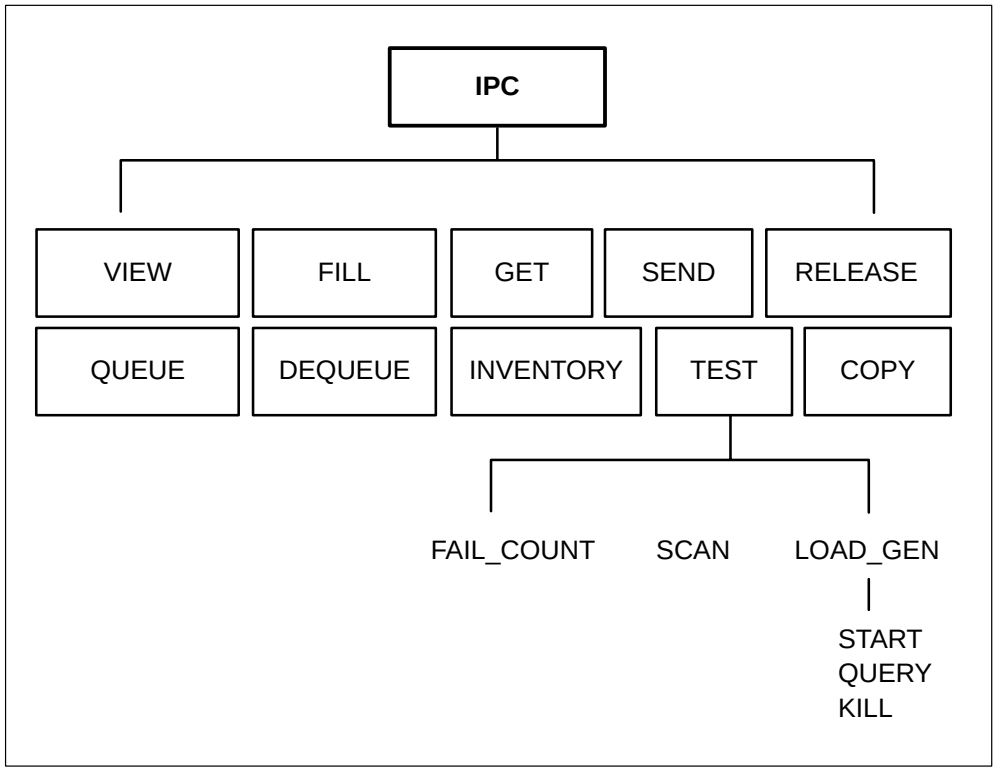
Only one working buffer can be obtained at a time.

All numbers used in the IPC level are entered and displayed in hexadecimal.

Figure 4-20 on page 4-62 illustrates the commands available at the IPC level.

For an example of using the commands at the IPC level, refer to Figure 7-13 on page 7-63.

Figure 4-20
Commands in the IPC level



CAUTION
Do not use the IPC level on a live switch.
Users of this level should have a thorough knowledge of XPM messaging. Incorrect data provided in the message can cause a peripheral to trap or to go down, causing severe service interruption.

Following are the commands available in the IPC level:

COPY command

The COPY command prompts for source and destination buffers and copies from source to destination.

COPY	**FROM working buffer_num** **TO working buffer_num**

Where:

FROM	the original working buffer that will be copied
working buffer_num	the working buffer number
TO	the buffer that receives the copy of the original buffer

DEQUEUE command

The DEQUEUE command removes a buffer from the storage/destination queue. This command prompts for TOP or BUFFER ITEM. If TOP is specified, the first item in the queue becomes the working buffer. If a buffer number is specified, that buffer becomes the working buffer if it is on the queue.

DEQUEUE	**TOP** **cid**

Where:

TOP	indicates the first item in the queue will become the working buffer
cid	is the buffer number that will become the working buffer

FILL command

You can enter data into a communications buffer with the FILL command. You are then prompted for new information.

FILL	**V** **D cc** **S cc** **C bb mm ... mm**

Where:

V	displays the buffer being filled
D	allows you to write to the destination CID
S	allows you to write to the source CID
C	allows you to write information into the IPC buffer
cc	is the CID

bb the start byte. Numbering begins with 0

mm the data in the communications buffer

Note: **Do *not* add any data after the cc parameters. Any additional input will be interpreted as data.**

GET command

The GET command obtains an IPC buffer from the IPC utility. This buffer becomes the working buffer. This command cannot be used if a working buffer exists or if no buffers are available. You receive appropriate messages when one of these conditions occur.

GET	

INVENTORY command

The INVENTORY command lists the buffer numbers for all buffers in the storage queue or the destination queue for a given CID. The working buffer is listed as the last item.

INVENTORY	**HOLDING** **cid**

Where:

HOLDING the holding queue

cid the communication identifier of the task

QUEUE command

The QUEUE command queues the working buffer on the storage/destination queue.

QUEUE	

RELEASE command

The RELEASE command releases the working buffer.

RELEASE	

SEND command

The SEND command sends the working buffer into the IPC system.

SEND	

TEST level

The TEST command accesses tools used for testing.

TEST	

Following are the commands available at the TEST level:

FAILCOUNT displays the number of times the IPC system failed to get a buffer

SCAN continuously displays the destination CID. You are prompted for a destination identifier

LOAD_GEN generates a load for the IPC system

VIEW command

The VIEW command prompts for a buffer number, or the working buffer, and displays the contents of the specified buffer.

VIEW	

Example:

```
LTCMP>

TIme,TAsk,Load,Xprompt,Debug,Swerr,LIpc,Ipc,Patches,Msgtr,
Uspace,Chnls,MTc,DIagnose,Audit,PKtldr,CP,SE,C7tu,PEg,Rcvrmon.
LTCMP>

>>>i
```

```
View,Fill,Get,Send,Release,Queue,Dequeue,Inventory,Copy,Test.
MP:Ipc> g

IPC BUFFER # = 25
MP:Ipc> f

WORKING, OR BUFFER NUMBER
MP:Ipc> w

View,Dest_cid,Src_cid,Contents.
MP:Fill> D 07

MP:Fill> S A4

MP:Fill>

View,Dest_cid,Src_cid,Contents.
MP:Fill> c

INPUT CONTENTS START & DATA
MP:Contents> 3 11 22 33 44

MP:Fill> v

BUFFER 25   DEST:      7 NILNAME       SRC:     A4   TPTP3
00  07  00  11  22  33  44  00  FF  FF  FF  FF  FF  FF  FF  FF
FF  FF  FF  FF  FF  FF  FF  FF  FF  FF  FF  FF  FF  FF  FF  FF
FF  FF  FF  FF  FF  FF  FF  FF  FF  FF  FF  FF  FF  FF  FF  FF
FF  FF  FF  FF  FF  FF  FF  FF  FF  FF  FF  FF  00  08  00  08

MP:Fill> *

MP:Ipc>

View,Fill,Get,Send,Release,Queue,Dequeue,Inventory,Copy,Test.
MP:Ipc> *
```

LIPC level

The LIPC command is entered from the LTCMP level. The LIPC command accesses the long interprocessor communication (LIPC) level.

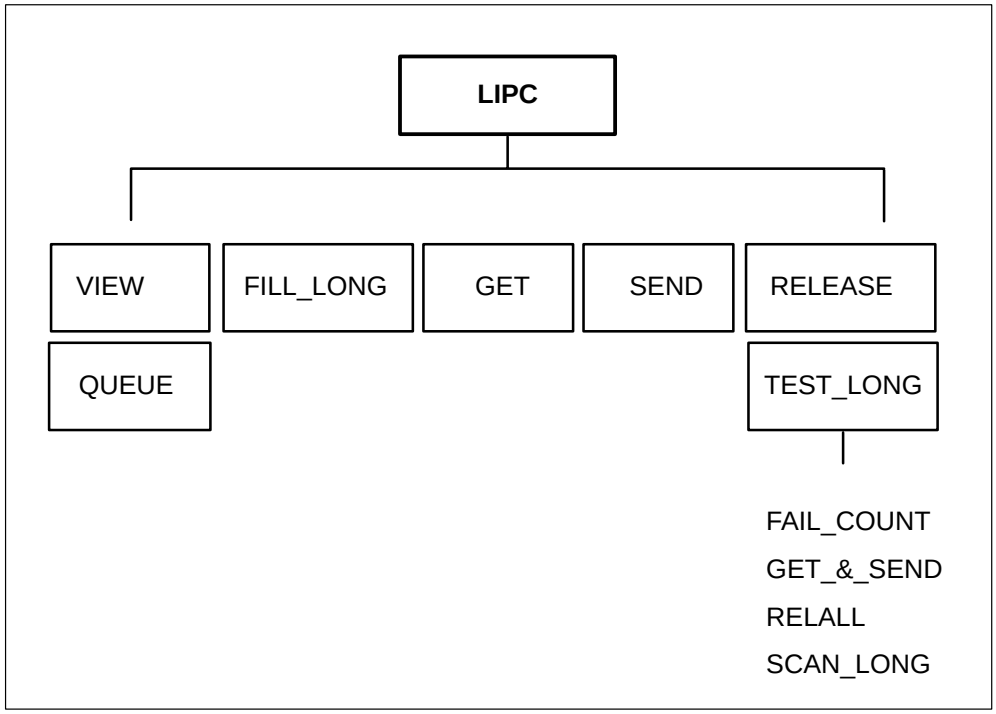
The LIPC buffers provide intertask and intratask communication in XPMs. LIPC is used for messaging in CCS7 and ISDN.

The LIPC level contains commands that allow you to create, view, and send LIPC buffers.

CAUTION
Do not use the LIPC level on a live switch.
Users of this level should have a thorough knowledge of XPM messaging. Incorrect data provided in the message can cause a peripheral to trap or go down, causing severe service interruption.

Figure 4-21 on page 4-67 contains the commands available in the LIPC level.

Figure 4-21
Commands in the LIPC level



Following are descriptions of the commands available in the LIPC level:

COPY command

The COPY command prompts for source and destination buffers and copies from source to destination.

COPY	**FROM**	**working buffer_num**
	TO	**working buffer_num**

Where:

FROM	the original working buffer that will be copied
working buffer_num	the working buffer number
TO	the buffer that receives the copy of the original buffer

DEQUEUE command

The DEQUEUE command removes a buffer from the storage/destination queue.

DEQUEUE	**TOP** **cid**

Where:

TOP	indicates the first item in the queue that will become the working buffer
cid	the buffer number that will become the working buffer

FILL_LONG command

The FILL_LONG command allows you to enter data into a communications buffer. You are then prompted for the new information.

FILL_LONG	**V** **D cc** **S cc** **C bb mm ... mm**

Where:

V	displays the buffer being filled
D	allows you to write to the destination CID
S	allows you to write to the source CID
C	allows you to write information into the IPC buffer
cc	the CID
bb	the start byte. Numbering begins with 0

mm the data in the communications buffer

Note: **Do not add any data after the cc parameters. Any additional input will be interpreted as data.**

GET command

The GET command obtains an LIPC buffer from the LIPC utility. This buffer becomes the working buffer. This command cannot be used if a working buffer exists or if no buffers are available.

GET	

INVENTORY command

The INVENTORY command lists the buffer numbers for all buffers in the storage queue or the destination queue for a given CID. The working buffer is listed as the last item.

INVENTORY	**HOLDING** **cid**

Where:

HOLDING the holding queue

cid the communication identifier of the task

QUEUE command

The QUEUE command queues the working buffer on the storage/destination queue.

QUEUE	

RELEASE command

The RELEASE command releases the working buffer.

RELEASE	

SEND command

The SEND command sends the working buffer into the LIPC system.

SEND	

TEST_LONG level

The TEST_LONG level accesses tools for testing LIPC buffers.

TEST_LONG	

The following tools are in the TEST_LONG level:

FAIL_COUNT provides a count of long IPC buffers that failed to be obtained from the LIPC utility

GET_&_SEND obtains a buffer and sends it into the LIPC system

RELALL releases all buffers

SCAN_LONG displays a buffer

VIEW command

The VIEW command prompts for a buffer number, or the working buffer, and displays the contents of that buffer.

VIEW	

Example:

```
MP:Debug> DM,FIndproc,SM,SW,Trapinfo,STopontrap,Cleartrap,
Zerodiv,Rom,Info,TRCpoints,CAlltrace,Availtime,COverage,TEmpstore

MP:Debug>

>>>li

View,Fill_long,Get,Send,Release,Queue,Dequeue,Inventory,Copy,
Test_long.
MP:LIpc>

>>>g

ipc buffer # = 12C
MP:LIpc>

>>>f

working, or buffer number
LTCMP>

>>>w

View,Dest_cid,Src_cid,Contents.
```

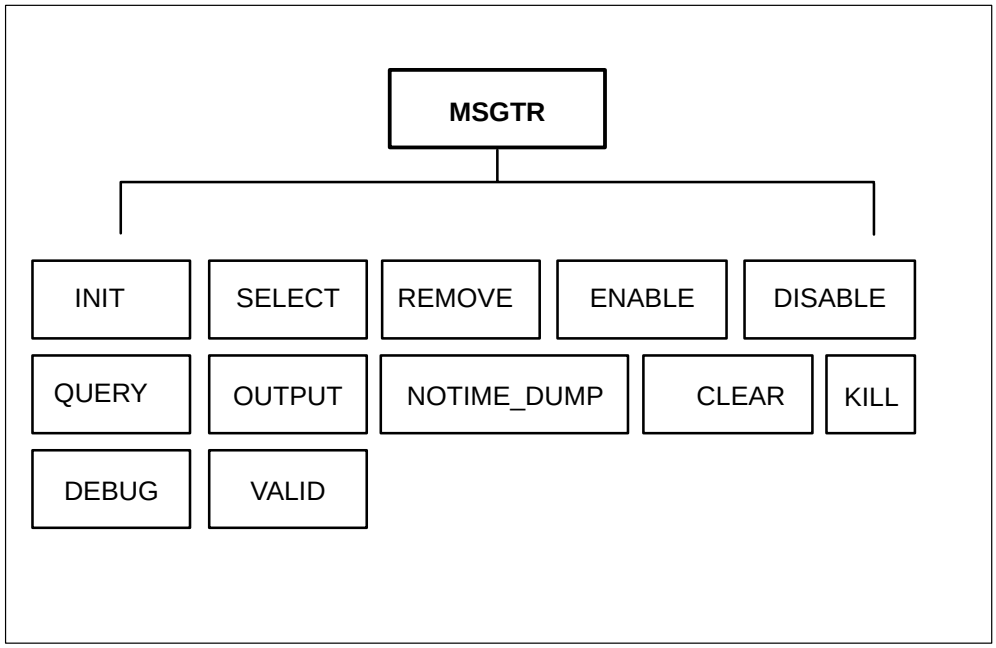
```
MP:Fill_long>

>>>d 81

MP:Fill_long>

>>>s a4

MP:Fill_long>

View,Dest_cid,Src_cid,Contents.
MP:Fill_long>

>>>c

Input contents start & data
MP:Contents>

>>>3 11 22 33 44

MP:Fill_long>

>>>v

buffer 12C  dest: 81 MP MTC   src: A4 TPTP3

00  07  00  11  22  33  44  00  CB F1  CB F4  CB F7  CB FA
CB FD CC 00  CC 03  CC 06  CC 09  CC 0C CC 0F  CC 12
CC 15  CC 18  CC 1B  CC 1E  CC 21  CC 24  CC 27  CC 2A
CC 2D  CC 30  CC 33  CC 36  CC 39  CC 3C CC 3F  CC 42
CC 45  CC 48  CC 4B  CC 4E  CC 51  CC 54  CC 57  CC 5A
CC 5D  CC 60  CC 63  CC 66  CC 69  CC 6C CC 6F  CC 72
CC 75  CC 78  CC 7B  CC 7E  CC 81  CC 84  CC 87  CC 8A
CC 8D  CC 90  CC 93  CC 96  CC 99  CC 9C CC 9F  CC A2
CC A5  CC A8  CC AB  CC AE  CC B1  CC B4  CC B7  CC BA
CC BD  CC C0  CC C3  CC C6  CC C9  CC CC CC CF  CC D2
CC D5  CC D8  DD DB  CC DE  CC E1  CC E4  CC E7  CC EA
CC ED  CC F0  CC F3  CC F6  CC F9  CC FC CC FF  CD 02
CD 05  CD 08  CD 0B  CD 0E  CD 11  CD 14  CD 17  CD 1A
CD 1D  CD 20  CD 23  CD 26  CD 29  CD 2C CD 2F  CD 32
CD 35  CD 38  CD 3B  CD 3E  CD 41  CD 44  CD 47  CD 4A
CD 4D  CD 50  CD 53  CD 56  CD 59  CD 5C CD 5F  CD 62
CD 65  CD 68  CD 6B  CD 6E  CD 71  CD 74  CD 77  CD 7A
CD 7D  CD 80  CD 83  CD 86  CD 89  CD 8C CD 8F  CD 92
CD 95  00  1B  C3  2F  00  09  00  09

MP:Fill_long>
```

Message trace level

The MSGTR (message trace) level is accessed when the MSGTR command is entered from the LTCMP level.

Figure 4-22 on page 4-72 illustrates the commands available at the MSGTR level.

Figure 4-22
Commands in the MSGTR level



Following are descriptions of the commands available in the MSGTR level:

CLEAR command

The CLEAR command purges the snapshot table of any snapshots that have not been printed without disturbing the selection criteria table.

Note: **The OUTPUT must be turned OFF and the trace must be DISABLED before the CLEAR command can be issued.**

CLEAR	

DEBUG level

The DEBUG command allows you to look at snapshots after a trap has occurred. If the tracing structures were initialized when the trap occurred, the tool provides pointer information, snapshot information, and a command to look at snapshots.

DEBUG	

Example:

```
>debug
TRACE STATUS:  INITIALIZED, NOT ENABLED, OUTPUT OFF

MAX SNAPSHOT = 375
PRINT INDEX              = 0
LAST SNAPSHOT USED  = 0
S(napshot,R(ecover,*
```

The fields in the previous example are as follows:

TRACE STATUS indicates the current status of the trace facility

MAX SNAPSHOT indicates the number of snapshots available

PRINT INDEX indicates the next snapshot to be printed

LAST SNAPSHOT USED indicates the last snapshot filled. The difference between PRINT INDEX and LAST SNAPSHOT USED equals the number of snapshots to print.

The SNAPSHOT and RECOVER commands are used when a message being traced causes a trap. Since the snapshot is taken before the message arrives at the destination, the previous snapshot contains the message that caused the trap.

The following descriptions are of the commands used to view snapshots:

SNAPSHOT displays the contents of a snapshot. Before this command can be used, the utility must be initialized. Refer to the INIT command on page 4-74.

RECOVER requires that the snapshot data structures are intact after the trap. The RECOVER command returns the facility status to initialized without changing such facilities as pointers or indexes. After issuing the command, snapshots can be printed as usual. This command should be used *only* after a trap.

DISABLE command

The DISABLE command stops the IPC routines from tracing messages and turns off the output task.

Note: **The DISABLE command can be used *only* when a trace has been activated with the ENABLE command.**

DISABLE	

ENABLE command

The ENABLE command activates the selective trace facility. The IPC routines begin collecting trace information, and the display routine is activated to produce any messages that have been captured.

This command can be used only when a trace has been initialized, when at least one selection criteria exists, and when the trace is not already active.

ENABLE	

INIT command

The INIT command sets up and initializes the main data structures for the selective trace. This command can only be used when a selective trace has not been initialized.

INIT	

If temporary store cannot be obtained, the following message appears:

```
***** WARNING:  NO TEMP STORE - check other trace tools
```

This message indicates that you should verify if another trace tool, such as IPC trace or TRACEPOINTS, is active and is using temporary store.

If no temporary store is available, the following message appears:

```
INSUFFICIENT TEMP STORE
```

This message indicates that the selective trace cannot be used in the XPM until some memory space is freed.

KILL command

The KILL command disables the message trace, kills the display routine, and frees the temporary store for use by other facilities.

Once the message trace is killed, the selective trace facility must be initialized before being used again.

KILL	

NOTIME DUMP command

The NOTIME DUMP command displays a summary of the message trace. The summary does not contain the GET, SEND, or RELEASE times.

NOTIME DUMP	

OUTPUT command

The OUTPUT command either displays messages as they are intercepted or holds intercepted messages without displaying them. Messages that are held can be displayed later.

OUTPUT	**ON**	**on**
	OFF	**off**

Where:

ON indicates that messages are displayed on the VDU as they are intercepted. When OUTPUT is on, circular-table overwrites are not allowed. If an overwrite is attempted, the snapshot is discarded. The lost snapshot is detected by the missing snapshot sequence numbers in the output.

If **OUTPUT ON ON** is entered, the quick-look option is enabled. The quick-look option displays the message buffer up to the length parameter, rather than printing all 64 bytes.

If the quick-look option is not on, all 64 bytes of the message buffer are displayed.

OFF indicates that messages are held and are not displayed on the VDU screen. When OUTPUT is off, circular-table overwrites are allowed. When OUTPUT is on again, printing resumes and the lost snapshots can be detected by the missing snapshot sequence numbers.

Note 1: Traces can be run with OUTPUT off. Once the trace has been disabled, the results can be dumped, and you can perform other work on the terminal.

Note 2: When tracing under heavy IPC traffic, gaps in the numbering sequence of the messages being displayed indicate that messages are being captured more quickly than they can be displayed.

QUERY command

The QUERY command displays the selection criteria you request. The table also shows the entry numbers assigned to each criteria.

The QUERY command can be used once the trace has been initialized.

QUERY	

Example:

```
>q

TRACE STATUS:INITIALIZED,NOT ENABLED,OUTPUT ON,QUICK LOOK OFF
#       SOURCE        DEST              DATA
-----------------------------------------------------------------
1  MP TPTP1  A1      XX                XX XX XX XX XX XX XX XX XX XX

2      XX             XX                XX XX XX XX 02  94  XX XX XX XX
```

REMOVE command

The REMOVE command removes message trace criteria from the message table.

The REMOVE command can only be used when the trace has been initialized but is not active.

REMOVE	**entry number** **ALL**

Where:

entry number the entry number corresponding to the message selection criteria to be removed

ALL all the selection criteria will be removed

Note: After each REMOVE command, the selection table is immediately compressed. If more than one criterion will be removed, check the current entry number for the criterion before issuing the next REMOVE command.

Example:

```
>R 1
```

The previous example removes message trace criterion number 1.

```
>R ALL
```

The previous example removes all the selection criteria.

SELECT command

You can select message trace criteria using the SELECT command. This command creates a selection criteria table entry. The entries are used to select messages for tracing.

Selection criteria should be unique to avoid unnecessary tracing. After selection criteria has been created, it is referenced by an entry number that is obtained by using the QUERY command.

SELECT	**source**	**dest**	**data(n)**

Where:

source is the CID of the task that will send the message

dest is the CID of the task that will receive the message

data(n) is the message header. Up to 10 bytes can be entered in single-byte, hexadecimal format. If a byte is not significant and is entered only as a place holder, an **X** or **XX** is entered. Any unfilled data fields at the end of the command are filled with **XX**.

Refer to Figure 4-23 on page 4-79 for a description of the bytes of the message header.

Note: The MDEST and MSRC fields in Figure 4-23 on page 4-79 are used when the message is being sent to the P-side of the XPM or to the XPM mate unit.

For example, to trace all the messages sent between the TPTP1 task (CID a1) and the TPTP2 task (CID a3), enter **s a1 a3**.

The display will be as follows:

```
#1  SRC:  MP TPTP1     A1   DEST:   MP TPTP2
    DATA: XX XX XX XX XX XX XX XX XX XX
```

To trace all the messages sent to and from terminal number 585, convert the external terminal number to an internal terminal number (in this case 587). Then, convert the internal terminal number to hexadecimal (in this case 24B). Enter **s xx xx xx xx xx xx 02 4b**.

The display will be as follows:

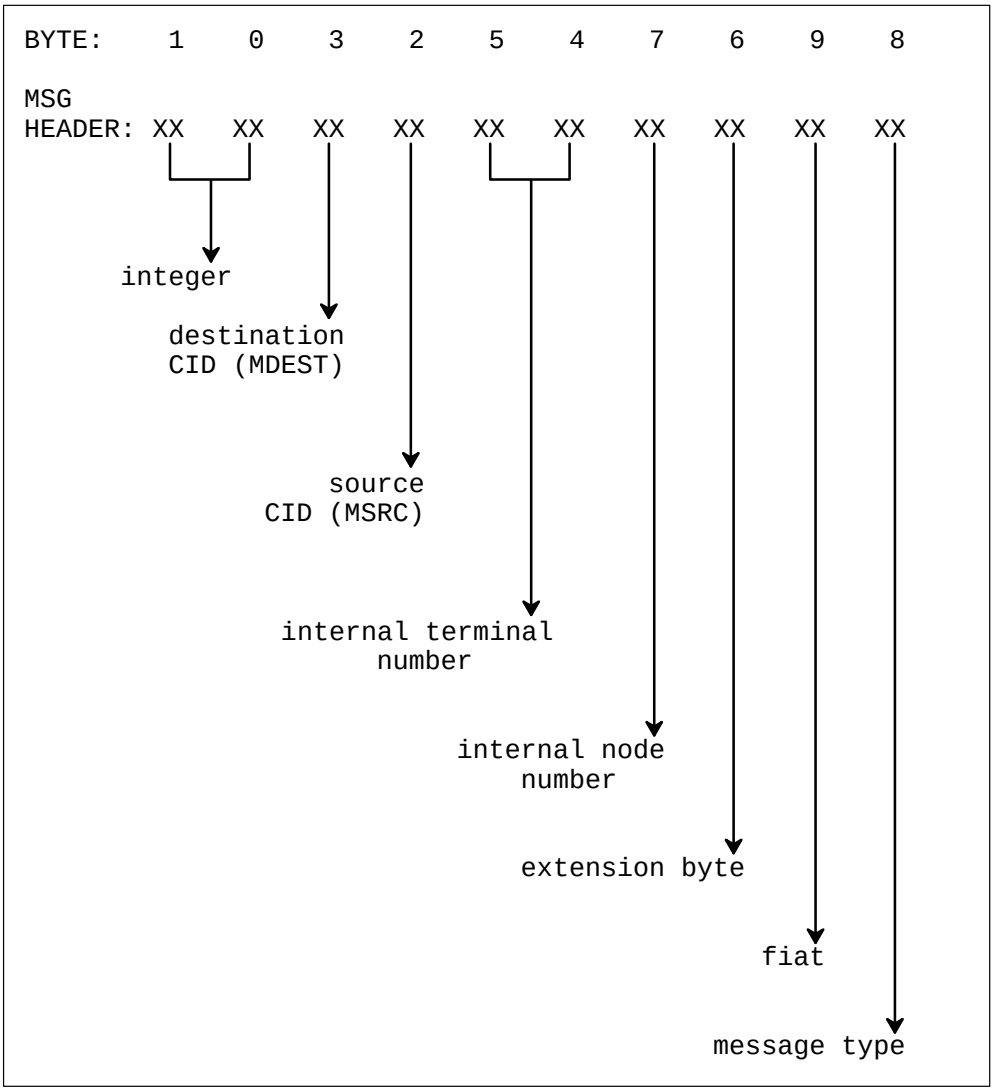
```
#1  SRC:   XX    DEST:     XX
    DATA: XX XX XX XX 02 4B XX XX XX XX
```

Note 1: A list of CIDs can be displayed by entering the VALID command at the MSGTR level. Refer to “VALID command” on page 4-79.

Note 2: A list of fiats can be found in module UTILCONS.

Note 3: The main list of message-type overlays can be found in module UTILTYPE. However, many peripherals have message types declared in other modules.

Figure 4-23
Byte assignments for SELECT command data parameter



VALID command

The VALID command displays the valid (CIDs).

VALID	

USPACE level

The USPACE (user space) level is accessed when the USPACE command is entered from the LTCMP level.

The USPACE level contains memory management information on the section of memory set aside for user-level tasks to access. This level does not contain information related to memory available for heap or for tools.

The MMU provides virtual addressing in the MP by converting the virtual addresses generated by a task into actual addresses. The actual address is used to access real memory. The virtual address is equal to the real actual address; in other words, the addresses have a one-to-one translation in the MMU.

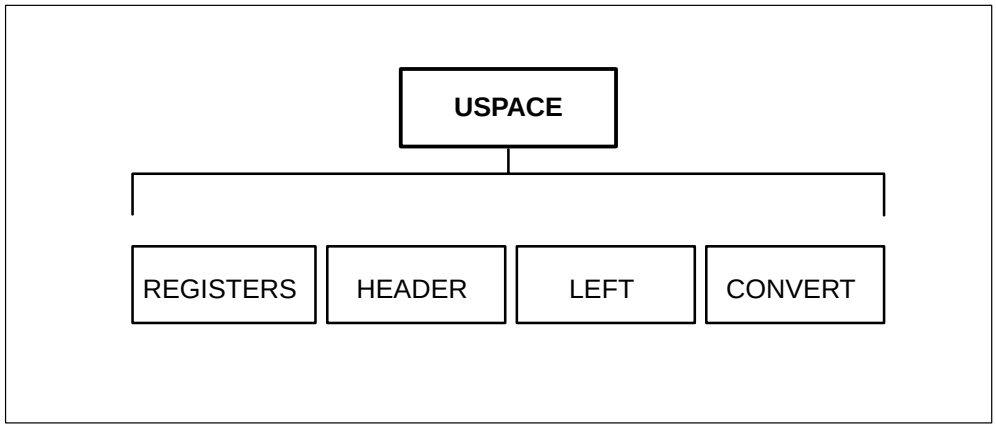
The MMU divides memory into segments. A segment starts at a 1 kilobyte address and can occupy several related memory frames. Virtual addresses are calculated using an offset from the start of a segment.

Each segment has an MMU register associated with it called a segment register. There are 64 segment registers, each containing the following information:

- address of the segment to which the offsets are applied
- range field, in decimal, defining the size of the segment in kilobytes
- flag field defining the protection status of the segment and whether the segment is present in memory
- address field defining the starting address of the segment in memory

Figure 4-24 on page 4-81 contains the commands available at the USPACE level.

Figure 4-24
Commands in the USPACE level



Following are descriptions of the commands available in the USPACE level:

CONVERT command

The CONVERT command converts a virtual address to a real address.

CONVERT	

HEADER command

The HEADER command displays the segment descriptor as well as the address and size of the free blocks in the user data area.

HEADER	

Example:

```
Registers,Header,Left,Convert,*
MP:Uspace>

>>>h
```

```
DUMP OF  SEGMENT   DESCRIPTORS:
Segment      Name        Address     Size      Perm      Prot      Usd
      0                   157A18        0         T         F         T
      1                   2D7A30        0         T         F         T
      2                   457A48        0         T         F         T
      3                   5D7A60        0         T         F         T
      4                   757A78        0         T         F         T
      5                   8D7A90        0         T         F         T
      6                   A57AA8        0         T         F         T
      7                   BD7AC0        0         T         F         T
      8         patch     1E0000        0         T         F         T
      9         trmprot   1E1400        0         T         T         T
     10         glbprot   1E4C00        0         T         F         T
     11         chprot    1E6C00        0         T         T         T
      .
      .
      .
     58                   857F70        0         F         F         F
     59                   9D7FA0        0         F         F         F

FREE BLOCKS:  ADDRESS SIZE
              1EC400   15

Registers,Header,Left,Convert,*

MP:Uspace>
```

LEFT command

The LEFT command displays the amount of free space in the USPACE memory. The free space is measured in kilobytes.

LEFT	

Example:

```
Registers,Header,Left,Convert,*
MP:Uspace>

>>>l

AMOUNT OF FREE SPACE IS   15   K

Registers,Header,Left,Convert,*

MP:Uspace>
```

REGISTERS command

The REGISTERS command dumps the segment registers.

REGISTERS	

Example:

```
Registers,Header,Left,Convert,*
MP:Uspace>

>>>r

DUMP OF SEGMENT REGISTERS:
 SEG #        BASE        RANGE(k)      WP        UA       VIRT
     0      000000             255       F         T      000000
     1      040000             255       F         T      040000
     2      080000             255       F         T      080000
     3      0C0000             255       F         T      0C0000
     4      100000             255       F         T      100000
     5      140000             255       F         T      140000
     6      180000             255       F         T      180000
     7      1C0000             255       F         T      1C0000
     8      1E0000               4       F         T      200000
     9      1E1400              13       T         T      240000
    10      1E4C00               7       F         T      280000
    11      1E6C00               7       T         T      2C0000
     .
     .
     .
    58      280000             255       F         T      E80000
    59      2C0000             255       F         T      EC0000

Registers,Header,Left,Convert,*
MP:Uspace>
```

In the previous example, the field names indicate the following:

BASE the base address field, defining the starting address of the segment in memory

RANGE the size of the segment

WP the write-protect flag. A **T** in this field indicates that you cannot write to the segment

UA the user-access-allowed flag. A **T** in this field indicates access to the segment is permitted when the CPU is running in user mode

VIRT the virtual address

PATCHES level

The PATCHES level is accessed when the PATCHES command is entered from the LTCMP level.

The PATCHES level contains the list of patches applied to the XPM software load.

PATCHES	

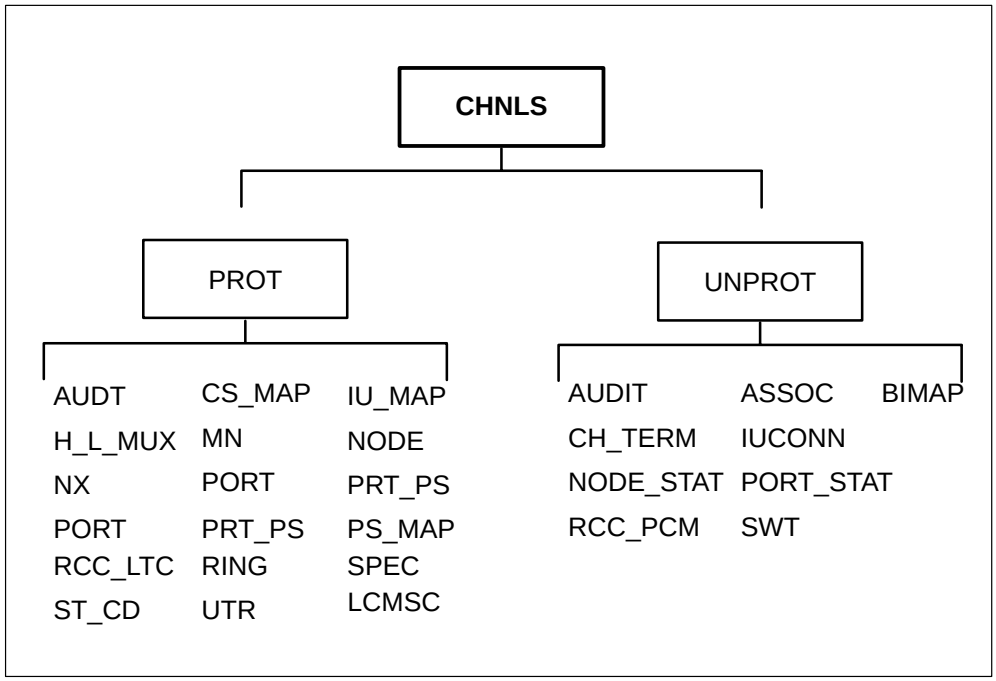
Channels level commands

CHNLS level

The Channels level is accessed when the CHNLS command is processed from the LTCMP level.

The Channels level contains commands that allow you to display information about the channel associated with the XPM. Figure 4-25 on page 4-85 shows the commands available at the CHNLS level.

Figure 4-25
Commands in the CHNLS level



Following are descriptions of the commands available in the CHNLS level:

PROT level

The PROT command accesses the protected channels (PROT) level. The PROT level contains commands for displaying protected channel information, such as the configuration of the XPM and the related P-side node.

PROT	

Following are descriptions of the commands available in the PROT level:

NODE command

The NODE command dumps the node table.

NODE	**internal node**

Where:

internal node is the internal node number, which is the index to the node table.

Example:

```
Node,Port,Spec,H_l_mux,Utr,NX,PRt_ps,Audt,Mn,Cs_map,PS_map,
Iu_map,Ring,ST_cd,RCc_ltc.

MP:Prot>

>>>n

NODE_TABLE
int   extnode  n_t  type   msg_pcl  relation  n_p  s_p  e_p  s_t  rct  host
 1         19    1    11   pcl_ds30  slave      16    0   15    1    0     1
 2         24  641    18   pcl_dmsx  master     16   16   31    2    0     1
 3         30   31     3   pcl_dmsx  master      1    0    0    0    0     2
 4         27  641    12   pcl_dmsx  master      8    0    0    0    0     2
 5         28  641    12   pcl_dmsx  master      8    0    0    0    0     2
 6         29  641    12   pcl_dmsx  master      8    0    0    0    0     2
 7         25  641    12   pcl_dmsx  master      8   32   39  643    0     1
 8         26  641    12   pcl_dmsx  master      8   40   47 1284    0     1

MP:Prot>
```

The fields of the node table are as follows:

int internal node number

extnode external node number

Note: The external node number can also be determined with the QUERYPM command.

n_t number of terminals

type type of node, one of the following:

3 RMM

11 LTC, DTC, LGC

12 LCM

13 MSB

15 CSC

18 RCC

22 ESA

p_t PM type. Enter **DECL PM_TYPE** for details.

msg_pcl message protocol, one of the following:

- pcl_nil
- pcl_ds30
- pcl_dmsx
- pcl_hdlc
- pcl_spare

relation relationship, master or slave

n_p number of ports

s_p start port, which is an index to the port table of the node

e_p end port, which is an index to the port table of the node

s_t start terminal, which is the index to the terminal data table

rct rct index (for SMU only)

host the internal node number of the peripheral host node

ipml the interperipheral message link (IPML) flag, which is an indication of whether the node has IPML capability to the host node

PORT command

The PORT command displays the port table.

PORT	**range**

Where:

range the port table index range. Enter the start port (s_p) and end port (e_p) indexes of the node.

The port table fields are as follows:

index port table index

p_host physical port number on the host

side indicates the port is one of the following types

- **port_pside** P-side port
- **port_cside** C-side port
- **port_interunit** interunit port
- **port_rmm_side** remote maintenance module (RMM) side port

spch speech capability, one of the following:

- spch_cap
- no_spch

msg messaging capability, one of the following:

- msg_cap
- no_msg

msg_chnl message channel number

type_of_port port type, one of the following:

- prt_nil
- prt_ds30
- prt_ds30a
- prt_ds1

SPEC command

The SPEC command displays the special connections table, which contains a map of the time switch. You are then prompted for the P-side port number.

SPEC	

Note: When you are prompted for the P-side port number, an entry of 20 will display the entire special connection table (ports 0 to 19).

Example:

Node,Port,Spec,H_l_mux,Utr,NX,PRt_ps,Audt,Mn,Cs_map,PS_map,
lu_map,Ring,ST_cd,RCc_ltc.

MP:Prot>

>>>s

SPECIFY PS_PORT (PS_PORT = 20 FOR ALL PORTS)
MP:Spec>

>>>2

SPEC_CON_TABLE <entries with special conn will be displayed>

ps_port	ps_chnl	cs_port	cs_chnl	dir	path_type
2	1	18	2	two_way	nailedup_path
2	3	2	2	two_way	nailedup_path
2	4	3	3	two_way	nailedup_path
2	5	0	5	two_way	nailedup_path
2	6	1	6	two_way	nailedup_path
2	8	2	7	two_way	nailedup_path
2	9	3	8	two_way	nailedup_path
2	10	0	10	two_way	nailedup_path
2	11	1	11	two_way	nailedup_path
2	13	2	12	two_way	nailedup_path
2	14	3	13	two_way	nailedup_path
2	15	0	15	two_way	nailedup_path
2	17	1	17	two_way	nailedup_path
2	19	2	18	two_way	nailedup_path
2	20	3	19	two_way	nailedup_path
2	21	0	21	two_way	nailedup_path
2	22	1	22	two_way	nailedup_path
2	24	2	23	two_way	nailedup_path
2	25	3	24	two_way	nailedup_path
2	26	0	26	two_way	nailedup_path
2	27	1	27	two_way	nailedup_path
2	29	2	28	two_way	nailedup_path
2	30	3	29	two_way	nailedup_path
2	31	0	31	two_way	nailedup_path

Node,Port,Spec,H_l_mux,Utr,NX,PRt_ps,Audt,Mn,Cs_map,PS_map,
Iu_map,Ring,ST_cd,RCc_ltc.

MP:Prot>

The SPEC_CON_TABLE fields are as follows:

ps_port P-side port number

ps_chnl P-side channel number

cs_port C-side port number

cs_chnl C-side channel number

dir direction, one of the following:

- two_way
- to_ps
- from_ps

path_type path type, one of the following:

- nailed_up_path
- reserved_path

LCMSC command

The LCMSC command allows you to display information on a specified port.

Lcmsc	port	val1	val2	val3

where

port You can specify a port in the range 0 - 19 by entering the desired port number.

Note: If this is a CPM peripheral, the pside port range is 0 - 53

val1 By entering the value 20, you can display information on all pside ports.

Note: If this is a CPM peripheral, enter the value 54.

val2 By entering the value 21, you can display TDM connection information for the LCME.

Note: 1 When you enter the value 21, TDM information for the LCME is displayed. The information does not pertain to the LCMI

Note: 2 If this is a CPM peripheral, enter the value 55.

val3 By entering the value 22, a list of all DTA connections residing in the pside peripheral is displayed.

Note: 1 The value 22 displays information related to an LCME or LCMI.

Note: 2 DTA information is not displayed for CPM peripherals.

H_L_MUX command

The H_L_MUX command displays a SAVE_LOW_MUX and a SAVE_HIGH_MUX cross-reference of port numbers to link types.

H_L_MUX	

Example:

```
Node,Port,Spec,H_l_mux,Utr,NX,PRt_ps,Audt,Mn,Cs_map,PS_map,
Iu_map,Ring,ST_cd,RCc_ltc.

MP:Prot>

>>>h_l_mux
```

```
SAVE_LOW_MUX & SAVE_HIGH_MUX
port        type
   0        ps_ds1
   1        ps_ds1
   2        ps_ds1
   3        ps_ds1
   4        ps_ds1
   5        ps_ds1
   6        ps_ds1
   7        ps_ds1
   8        ps_ds1
   9        ps_ds1
  10        ps_ds1
  11        ps_ds1
  12        ps_ds30a
  13        ps_ds30a
  14        ps_ds30a
  15        ps_ds30a
  16        ps_ds30a
  17        ps_ds30a
  18        ps_ds30a
  19        ps_ds30a

MP:Prot>
```

UTR command

The UTR command displays the spare card slots and the card type in each slot.

UTR	

Example:

```
Node,Port,Spec,H_l_mux,Utr,NX,PRt_ps,Audt,Mn,Cs_map,PS_map,
lu_map,Ring,ST_cd,RCc_ltc.

MP:Prot>

>>>u
```

```
MOD_COM_AREA
slot        card        address (msw)
  0          28             1A
  1          0               0
  2          0               0
  3          0               0
  4          0               0

CHNL_PROT
slot        card        address (msw)
  0          28             1A
  1          0               0
  2          0               0
  3          0               0
  4          0               0

UTR INFO = TRUE
```

In the previous example, data from both the MOD_COM_AREA and from channel data is displayed. The value in the MOD_COM_AREA should be identical to those from channel data. The most significant word (MSW) of the card address is displayed next to the card and slot numbers.

Refer to module UTILTYPES for card number definitions.

NX command

The NX command displays the global variables contained in module MPCHNLS. This command is used to check invalid pointer values.

NX	

Example:

```
Node,Port,Spec,H_l_mux,Utr,NX,PRt_ps,Audt,Mn,Cs_map,PS_map,
Iu_map,Ring,ST_cd,RCc_ltc.

MP:Prot>

>>>nx
```

```
Variable of interest in MPCHNLS:
xfertblid(hex)    =  0
xfersrc(hex)      = C3
xfer_index        = 9
chnl_prot    .next_port = 48
chnl_prot    .next_trml = 1925
chnl_prot    .next_card = 34
chnl_prot    .next_rct  = 0
chnl_prot    .nextnode  = 9
chnl_prot    .next_iuport= 4
chnl_prot    .next_ns_tbl = 1
chnl_prot    .next_glb_tbl = 4
chnl_prot    .next_xtd_xec = 1

MP:Prot>
```

Note: The *xfertblid, xfersrc,* and *xfer_index* fields are used for data transfer between mates.

PRT_PS command

The PRT_PS command displays P-side port information.

PRT_PS	

Example:

```
Node,Port,Spec,H_l_mux,Utr,NX,PRt_ps,Audt,Mn,Cs_map,PS_map,
Iu_map,Ring,ST_cd,RCc_ltc.

MP:Prot>

>>>pr
```

PORT	PORT_TYPE	INTNODE	MCA_P	MCA_N
0	2	2	2	2
1	2	2	2	2
2	2	2	2	2
3	2	2	2	2
4	2	2	2	2
5	2	2	2	2
6	2	2	2	2
7	2	2	2	2
8	2	2	2	2
9	2	2	2	2
10	2	2	2	2
11	2	2	2	2
12	1	7	1	7
13	1	7	1	7
14	1	7	1	7
15	1	7	1	7
16	1	8	1	8
17	1	8	1	8
18	1	8	1	8
19	1	8	1	8

MP:Prot>

Note: The values in field PORT_TYPE should be identical to the values in field MCA_P. The values in field INTNODE should be identical to the values in field MCA_N.

The fields of the PRT_PS command display are as follows:

PORT is the P-side port number (from channel data)

PORT_TYPE one of the following:

0 NIL

1 DS30A

2 DS1

3 PCM30

4 DS30

5 Spare1

6 Spare2

7 Spare3

INTNODE P-side internal node number (from channel data)

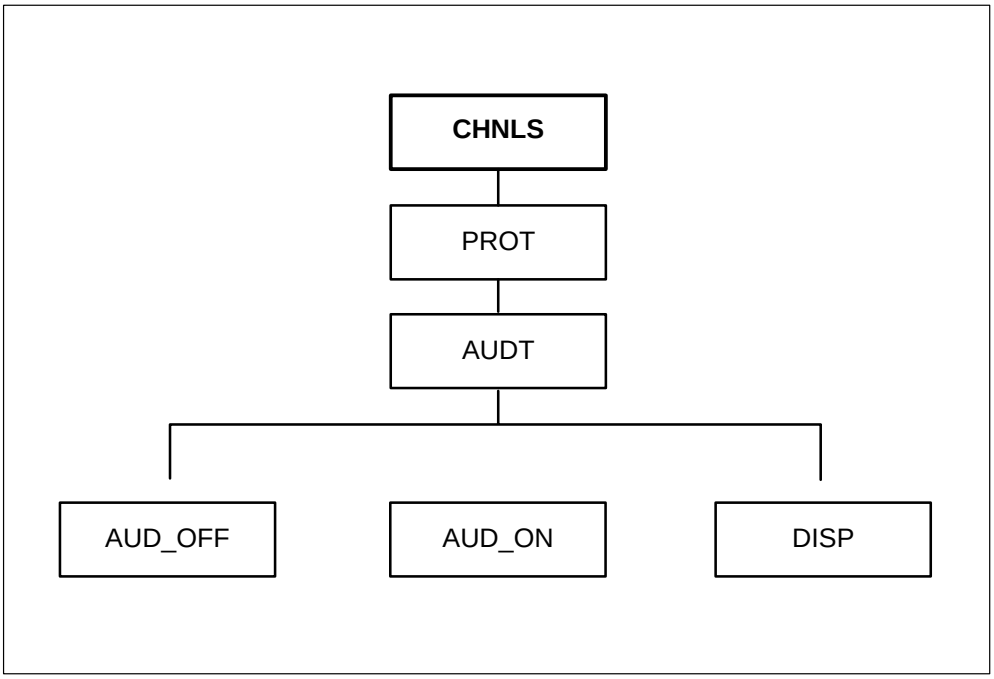
MCA_P port type (from the Mod_Com_Area).

MCA_N internal node number (from the Mod_Com_Area)

AUDT level

The AUDT command accesses the static data audit information level. The AUDT command is processed from the PROT sublevel of the CHNLS level. Figure 4-26 on page 4-96 shows the commands on the AUDT level.

Figure 4-26
Commands in the AUDT level



Following are the commands available in the AUDT level:

DISP displays audit information, as follows

Example:

```
Disp,aud_On,aud_oFf,*

MP:Audt>

>>>d

Common tables to be audited
 Audit table     Name          Prot          Cksum
      0          chprot        TRUE            175
      1          trmprot       TRUE            157
      2          xecdatat      TRUE            151
Non-std tables to be audited
 Audit table     Name          Prot          Cksum
      0          execbl0       TRUE            213
Glb data to be audited
  data ptr       size          chksum        aud_on        seg_sane
 10F822          256           26             TRUE           TRUE
 114300          120           0              TRUE           TRUE
 114402          64            128            TRUE           TRUE
 110DCC          132           184            TRUE           TRUE
data audit turned on = TRUE
data updating in progress = FALSE

Disp,aud_On,aud_oFf,*
MP:Audt>
```

AUD_OFF disables the data audit

AUD_ON enables the data audit

MN level

CAUTION

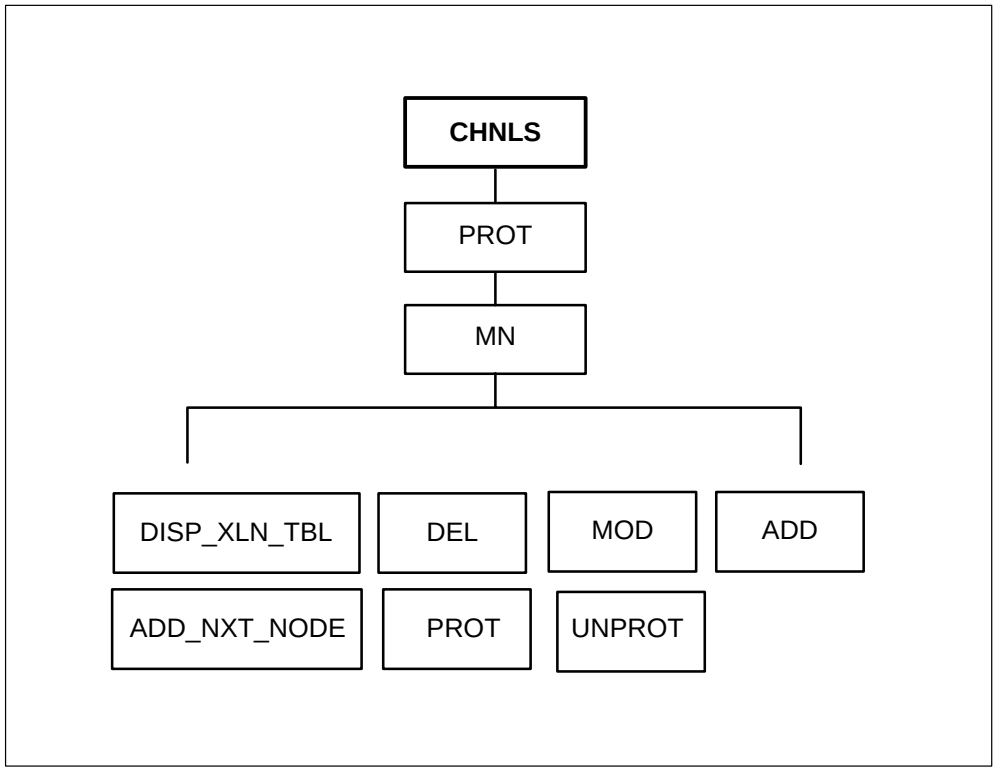
Do not use the MN level on a live switch.

The MN level contains commands for modifying the node table.

MN	

Figure 4-27 on page 4-98 contains the commands available at the MN level.

Figure 4-27
Commands in the MN sublevel



Following are the commands available at the MN sublevel:

DISP_XLN_TBL	displays the node translation table
DEL	deletes the internal node for testing purposes
MOD	modifies the external node number when you supply the internal node number
ADD	adds a node to the node table
ADD_NXT_NODE	increments the index to the node table

CS_MAP command

The CS_MAP command displays a C-side port map that indicates which C-side ports are equipped and which are unequipped.

CS_MAP	

Example:

```
Node,Port,Spec,H_l_mux,Utr,NX,PRt_ps,Audt,Mn,Cs_map,PS_map,
Iu_map,Ring,ST_cd,RCc_ltc.

MP:Prot>

>>>c

Usable  rcc_intra_chnl for equipped csports are
  2       7   12      16   18      23      28

CSIDE PORT MAP
Index              equipped/unequipped
  0                      equipped
  1                      equipped
  2                      equipped
  3                      equipped
  4                      equipped
  5                      equipped
  6                      equipped
  7                      equipped
  8                      equipped
  9                      equipped
 10                      equipped
 11                      equipped
 12                      equipped
 13                      equipped
 14                      equipped
 15                      equipped

MP:Prot>
```

PS_MAP command

The PS_MAP command displays P-side port information. The display indicates the P-side node off the port and the C-side port number of the P-side node (such as an LCM) to which the host port (such as an LGC) is connected.

PS_MAP	**range**

Where:

range the port number, from 0 to 99

Example:

```
Node,Port,Spec,H_l_mux,Utr,NX,PRt_ps,Audt,Mn,Cs_map,PS_map,
Iu_map,Ring,ST_cd,RCc_ltc,Fcard,Lcmsc.
MP:Prot>

>ps

PSPORT_MAP
port        psnode          csp_on_psnode
  0             2                      0
  1             2                      3
  2             2                      1
  3             2                      4
  4             3                      0
  5             3                      3
  6             3                      1
  7             3                      4
  8             0                      0
  9             0                      0
 10             0                      0
 11             0                      0
 12             0                      0
 13             0                      0
 14             0                      0
 15             0                      0
 16             0                      0
 17             0                      0
 18             0                      0
 19             0                      0

MP:Prot>
```

IU_MAP command

The IU_MAP command displays the inter-unit port (IUPORT) map.

IU_MAP	

Example:

```
Node,Port,Spec,H_l_mux,Utr,NX,PRt_ps,Audt,Mn,Cs_map,PS_map,
Iu_map,Ring,ST_cd,RCc_ltc.

MP:Prot>

>>>i

IUPORT_MAP
port        psnode
  0              7
  1              7
  2              8
  3              8
  4              0
  5              0
  6              0
  7              0
  8              0
  9              0
 10              0
 11              0
 12              0
 13              0
 14              0
 15              0
 16              0
 17              0
 18              0
 19              0

MP:Prot>
```

The fields of the IUPORT_MAP display are as follows:

PORT the index to the IUPORT map.

PSNODE internal node number of the P-side node; 0 indicates the port is not used

RING command

The RING command displays the RING table.

RING	

Example:

```
Node,Port,Spec,H_l_mux,Utr,NX,PRt_ps,Audt,Mn,Cs_map,PS_map,
Iu_map,Ring,ST_cd,RCc_ltc.

MP:Prot>

>>>r

SPECIFY INDEX 0...9
MP:Ring>

>>>0

RING TABLE for index = 0
INTNODE  =  7
code =       0   : 0060
code =       1   : 001C
code =       2   : 8003
code =       3   : 7000
code =       4   : 00E0
code =       5   : 1000
code =       6   : 6000
code =       7   : 0080
code =       8   : 0010
code =       9   : 0002
code =      10   : 4000
code =      11   : 0000
code =      12   : 0000
code =      13   : 0000
code =      14   : 0000
code =      15   : 0000
ring scheme      : frequency

MP:Prot>
```

ST_CD command

The ST_CD command displays the external node number and card type of the card in each slot.

ST_CD	

Example:

```
Node,Port,Spec,H_l_mux,Utr,NX,PRt_ps,Audt,Mn,Cs_map,PS_map,
lu_map,Ring,ST_cd,RCc_ltc.

MP:Prot>

>>>st

 slot_number          extnode          card_type
           0                0                  0
           1                0                  0
           2                0                  0
           3                0                  0
           4                0                  0
           5                0                  0

MP:Prot>
```

RCC_LTC command

The RCC_LTC command displays the mapping of RCC C-side ports to the LTC P-side ports.

RCC_LTC	

UNPROT level

The UNPROT command accesses the unprotected channels (UNPROT) Level.

The UNPROT level contains commands for displaying unprotected channel information in memory.

UNPROT	

Following are descriptions of the commands available in the UNPROT level:

BIMAP command

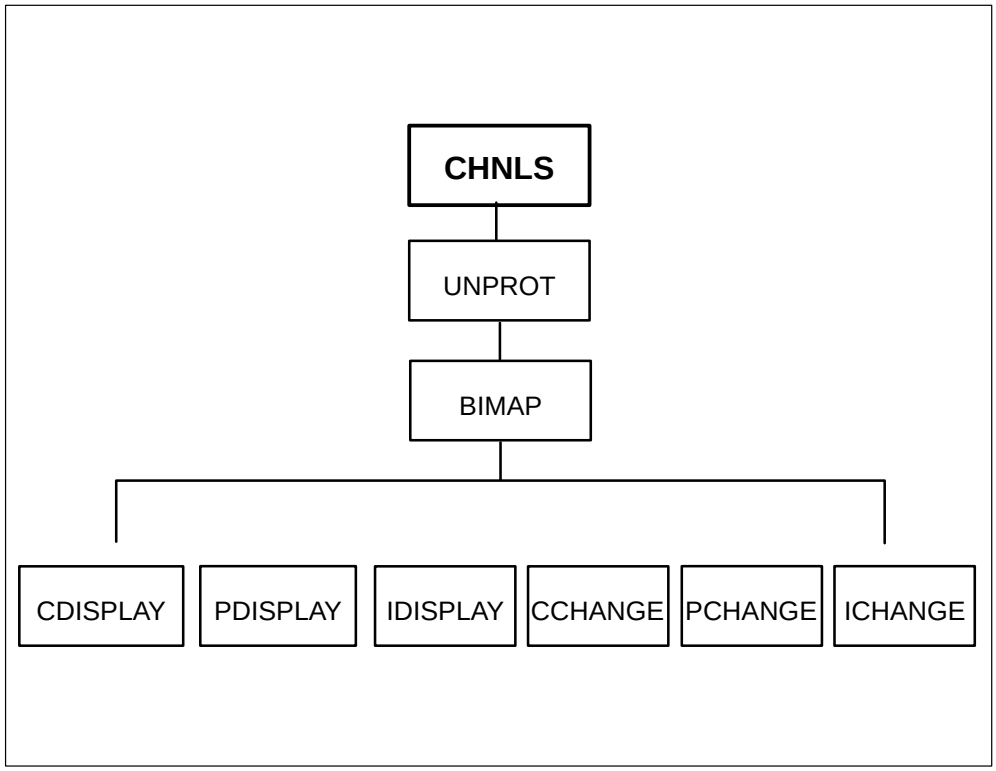
The BIMAP command accesses the Channel Busy Idle Map (BIMAP) level.

The BIMAP level displays port and channel bit maps.

Figure 4-28 on page 4-104 contains the commands available at the BIMAP level.

BIMAP	

Figure 4-28
Commands in the BIMAP sublevel



Following are the commands available in the BIMAP sublevel:

CDISPLAY The CDISPLAY command displays a C-side port and channel idle bit map. The values for the fields are as follows:

3 two_way

2 to_ps

1 from_ps

0 no_path (idle)

Following is an example:

Example:

```
CDisplay,PDisplay,IDisplay,CChange,PChange,IChange,*

MP:Bimap>

>>>cd 2

PORT        CHNLS (0-31)
  2          3333   3333   3333   3333   0333   3333   3333   3333

MP:Bimap>

>>>cd

PORT        CHNLS (0-31)
  0          0333   3333   3333   3333   0333   3333   3333   3333
  1          0033   3333   3333   3333   0333   3333   3333   3333
  2          3333   3333   3333   3333   0333   3333   3333   3333
  3          0333   3333   3333   3333   0333   3333   3333   3330
  4          0333   3333   3333   3333   0333   3333   3333   3333
  5          0333   3333   3333   3333   0333   3333   3333   3333
  6          0333   3333   3333   3333   0333   3333   3333   3333
  7          0333   3333   3333   3333   0333   3333   3333   3333
  8          0330   0033   0003   3000   0330   0033   0003   3000
  9          0033   0003   3000   3300   0033   0003   3000   3300
 10          0003   3000   3300   0330   0003   3000   3300   0330
 11          0000   3300   0330   0033   0000   3300   0330   0033
 12          0000   0000   0000   0000   0000   0000   0000   0000
 13          0000   0000   0000   0000   0000   0000   0000   0000
 14          0000   0000   0000   0000   0000   0000   0000   0000
 15          0000   0000   0000   0000   3000   0000   0000   0000
 16          0000   0000   0000   0000   0000   0000   0000   0000
 17          0000   0000   0000   0000   0000   0000   0000   0000
 18          0333   3330   0000   0000   0000   0000   0000   0000
 19          0000   0000   0000   0000   0000   0000   0000   0000

CDisplay,PDisplay,IDisplay,CChange,PChange,IChange,*

MP:Bimap>
```

PDISPLAY The PDISPLAY command displays a P-side port and channel idle bit map. The values for the fields are as follows:

3 two_way

2 to_ps

CDISPLAY The CDISPLAY command displays a C-side port and channel idle bit map. The values for the fields are as follows: **1** from_ps

0 no_path (idle)

Following is an example:

Example:

```
CDisplay,PDisplay,IDisplay,CChange,PChange,IChange,*

MP:Bimap>

>>>pd

PORT      CHNLS (0-31)
  0       3333 3333 3333 3333 3333 3333 3333 3333
  1       3333 3333 3333 3333 3333 3333 3333 3333
  2       3333 3333 3333 3333 3333 3333 3333 3333
  3       3333 3333 3333 3333 3333 3333 3333 3333
  4       3333 3333 3333 3333 3333 3333 3333 3333
  5       3333 3333 3333 3333 3333 3333 3333 3333
  6       3333 3333 3333 3333 3333 3333 3333 3333
  7       3333 3333 3333 3333 3333 3333 3333 3333
  8       3333 3333 3333 3333 3333 3333 3333 3333
  9       3333 3333 3333 3333 3333 3333 3333 3333
 10       3333 3333 3333 3333 3333 3333 3333 3333
 11       3333 3333 3333 3333 3333 3333 3333 3333
 12       3300 0000 0000 0000 3000 0000 0000 0000
 13       3300 0000 0000 0000 3000 0000 0000 0000
 14       3300 0000 0000 0000 3000 0000 0000 0000
 15       3300 0000 0000 0000 3000 0000 0000 0000
 16       3300 0000 0000 0000 3000 0000 0000 0000
 17       3300 0000 0000 0000 3000 0000 0000 0000
 18       3300 0000 0000 0000 3000 0000 0000 0000
 19       3300 0000 0000 0000 3000 0000 0000 0000

CDisplay,PDisplay,IDisplay,CChange,PChange,IChange,*

MP:Bimap>
```

IDISPLAY The IDISPLAY command displays an inter-unit port and channel idle bit map (ports 6 and 7 of the LCM). The values for the fields are as follows:

3 two_way

2 to_ps

1 from_ps

0 no_path (idle)

Following is an example:

Example:

```
CDisplay,PDisplay,IDisplay,CChange,PChange,IChange,*
MP:Bimap>

>>>id

PORT      CHNLS (0-31)
  0       3000  0000  0000  0000  3333  3333  3333  3333
  1       3000  0000  0000  0000  3333  3333  3333  3333
  2       3000  0000  0000  0000  3333  3333  3333  3333
  3       3000  0000  0000  0000  3333  3333  3333  3333
  4       3333  3333  3333  3333  3333  3333  3333  3333
  5       3333  3333  3333  3333  3333  3333  3333  3333
  6       3333  3333  3333  3333  3333  3333  3333  3333
  7       3333  3333  3333  3333  3333  3333  3333  3333
  8       3333  3333  3333  3333  3333  3333  3333  3333
  9       3333  3333  3333  3333  3333  3333  3333  3333
 10       3333  3333  3333  3333  3333  3333  3333  3333
 11       3333  3333  3333  3333  3333  3333  3333  3333
 12       3333  3333  3333  3333  3333  3333  3333  3333
 13       3333  3333  3333  3333  3333  3333  3333  3333
 14       3333  3333  3333  3333  3333  3333  3333  3333
 15       3333  3333  3333  3333  3333  3333  3333  3333
 16       3333  3333  3333  3333  3333  3333  3333  3333
 17       3333  3333  3333  3333  3333  3333  3333  3333
 18       3333  3333  3333  3333  3333  3333  3333  3333
 19       3333  3333  3333  3333  3333  3333  3333  3333

CDisplay,PDisplay,IDisplay,CChange,PChange,IChange,*

MP:Bimap>
```

CAUTION

Do not use the CCHANGE command on a live switch.

CCHANGE The CCHANGE command changes the specified idle map C-side port information to a user-specified value.

CCHANGE	value csport cschnl

Where:

value the new C-side port value, from 0 through 3

csport the C-side port to receive the change, from 0 through 19

cschnl the channel to receive the change, from 0 through 31

Examples:

To change all channels to 0, enter **CC 0.**

To change all channels on port 15 to 0, enter **CC 0 15.**

To change port 15, channel 17, to 0, enter **CC 0 15 17.**

PCHANGE The PCHANGE command changes the specified idle map P-side port information to a user-specified value.

PCHANGE	value psport pschnl

Where:

value the new P-side port value, from 0 through 3

psport the P-side port to receive the change, from 0 through 19

pschnl the channel to receive the change, from 0 through 31

ICHANGE The ICHANGE command changes the specified idle map international port information to a user specified value.

ICHANGE	value iuport iuchnl

Where:

value the new port value, from 0 through 3

iuport the port to receive the change, from 0 through 19

iuchnl the channel to receive the change, from 0 through 31

NODE_STAT command

The NODE_STAT command displays the status of the host P-side node.

NODE_STAT	

Example:

```
Bimap,Node_stat,Port_stat,Ch_term,Audit,Iuconn,ASsoc,Rcc_pcm,Swt.
MP:Unprot>

>>>n

NODE_STATUS (slave nodes only)
node        ns          ms[0]           ms[1]
 2          3           ls_busy         ls_busy
 3    --- HOST NODE IS RCC ---
 4    --- HOST NODE IS RCC ---
 5    --- HOST NODE IS RCC ---
 6    --- HOST NODE IS RCC ---
 7          2           ls_running      ls_running
 8          2           ls_running      ls_running

MP:Unprot>
```

PORT_STAT command

The PORT_STAT command displays status information for C-side ports, P-side ports, and IU ports.

PORT_STAT	

Example:

Due to size limitations, the following example has been altered to fit on the width of this page:

```
Bimap,Node_stat,Port_stat,Ch_term,Audit,Iuconn,ASsoc,Rcc_pcm,Swt.
MP:Unprot>

>>>p

     CS_PORT_STATUS                    PS_PORT_STATUS
PORT   PORT_STAT     MSG_CHNL     PORT   PORT_STAT     MSG_CHNL

 0         busy          1         0         busy          1
 1         busy          0         1         busy          0
 2         okay          1         2         busy          1
 3         busy          0         3         busy          1
 4         busy          0         4         busy          1
 5         busy          0         5         busy          0
 .
 .
 .
17         uneq          0        17         okay          0
18         uneq          0        18         okay          1
19         uneq          0        19         okay          1

                    IU_PORT_STATUS
               PORT   PORT_STAT     MSG_CHNL
               0         busy          0
               1         busy          0
               2         busy          0
               3         busy          0
               4         uneq          0
               5         uneq          0
               .
               .
               .
              17         uneq          0
              18         uneq          0
              19         uneq          0

MP:Unprot>
```

CH_TERM command

The CH_TERM command displays information on a specific terminal.

CH_TERM	

Example:

```
Bimap,Node_stat,Port_stat,Ch_term,Audit,Iuconn,ASsoc,Rcc_pcm,Swt.
MP:Unprot>

>>>c

SPECIFY RANGE 0 - 7055
MP:Ch_term>

>>>0 1

TERMINAL        TYPE        KSET            CONFIG
    1              1       FALSE               0

MP:Prot>

>>>c 0 3

TERMINAL    TYPE     KSET      CONFIG
    1              1       FALSE               0
    2              1       FALSE               0

MP:Prot>
```

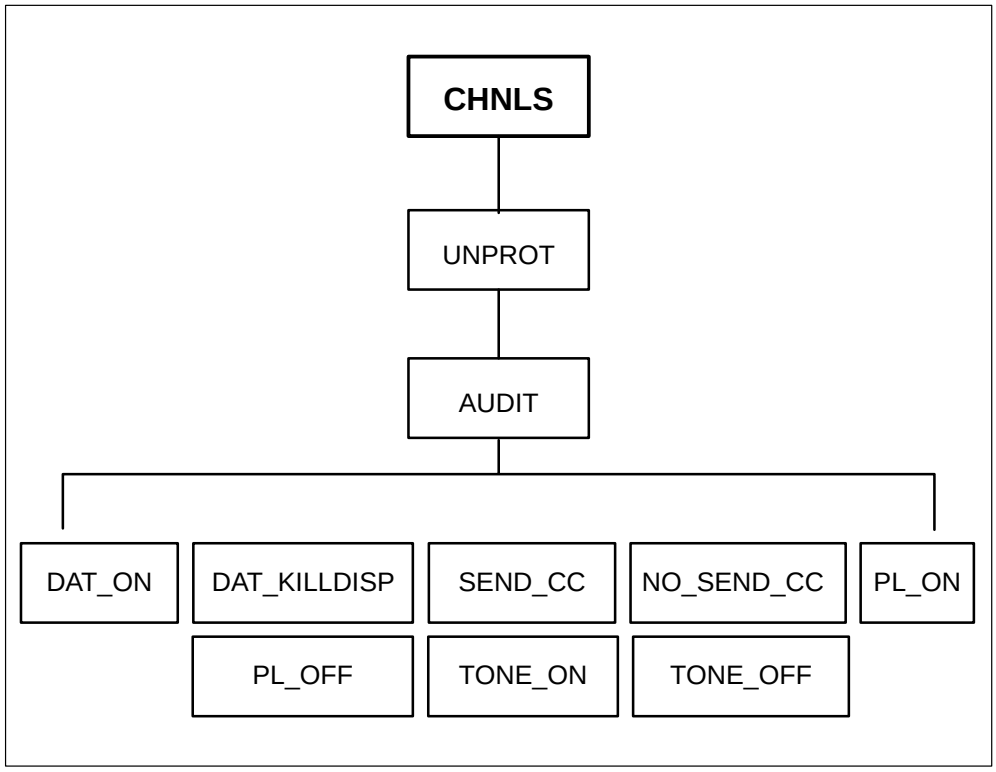
AUDIT level

The AUDIT command accesses the P-side link and tone audits level, which contains commands that allow you to select parameters for the audit display.

AUDIT	

Following are descriptions of the commands contained in the AUDIT sublevel:

Figure 4-29
Commands in the AUDIT level



DAT_ON displays the status of the audits

DAT_KILLDISP does not display the status of the audits

SEND_CC flag to send audit information to the CC

NO_SEND_CC flag not to send audit information to the CC

PL_ON flag to turn P-side link audit ON

PL_OFF flag to turn P-side link audit OFF

TONE_ON flag to turn the tone audit ON

TONE_OFF flag to turn the tone audit OFF

Example:

dAt,dat_KillDisp,Send_cc,No_send_cc,Pl_on,pL_off,Tone_on,
Tone_oFf,*

```
MP:Audit>

>>>A

nx_audt_port          = 0
aud_pl_slip_count     = 0
aud_tn_slip_count     = 0
plink audit comp      = TRUE
tone audit comp       = TRUE
send fault to CC      = FALSE
plink  audit on       = TRUE
tone audit on         = TRUE
data audit enable     = TRUE

dAt,dat_KillDisp,Send_cc,No_send_cc,Pl_on,pL_off,Tone_on,
Tone_oFf,*
MP:Audit>

>>>S

dAt,dat_KillDisp,Send_cc,No_send_cc,Pl_on,pL_off,Tone_on,
Tone_oFf,*
MP:Audit>

>>>L

dAt,dat_KillDisp,Send_cc,No_send_cc,Pl_on,pL_off,Tone_on,
Tone_oFf,*
MP:Audit>

>>>F

dAt,dat_KillDisp,Send_cc,No_send_cc,Pl_on,pL_off,Tone_on,
Tone_oFf,*
MP:Audit>

>>>A

nx_audt_port          =  0
aud_pl_slip_count     =  0
aud_tn_slip_count     =  0
plink audit comp      =  TRUE
tone audit comp       =  TRUE
send fault to CC      =  TRUE
plink  audit on       =    FALSE
tone audit on         =    FALSE
data audit enable     =  TRUE
```

```
dAt,dat_KillDisp,Send_cc,No_send_cc,Pl_on,pL_off,Tone_on,
Tone_oFf,*
MP:Audit>
```

IUCONN command

The IUCONN level displays the interunit connections that are in use.

IUCONN	

ASSOC command

The ASSOC command displays the Call Header Blocks (CHBs) associated with specific channels.

ASSOC	

SWT command

The SWT command displays the LCM stable call busy idle map. The map is used during XPM warm SwAct.

SWT	

Example:

```
Bimap,Node_stat,Port_stat,Ch_term,Audit,Iuconn,ASsoc,Rcc_pcm,Swt.
MP:Unprot>

>>>s

SPECIFY POINTER 0..9
MP:Swt>

>>>0
7
0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000 0000 0000

MP:Unprot>
```

In the previous example, the SPECIFY POINTER value is the LCM identifier.

A *0* in the SWT display indicates no call is active; a *1* indicates a call is active.

MTC level

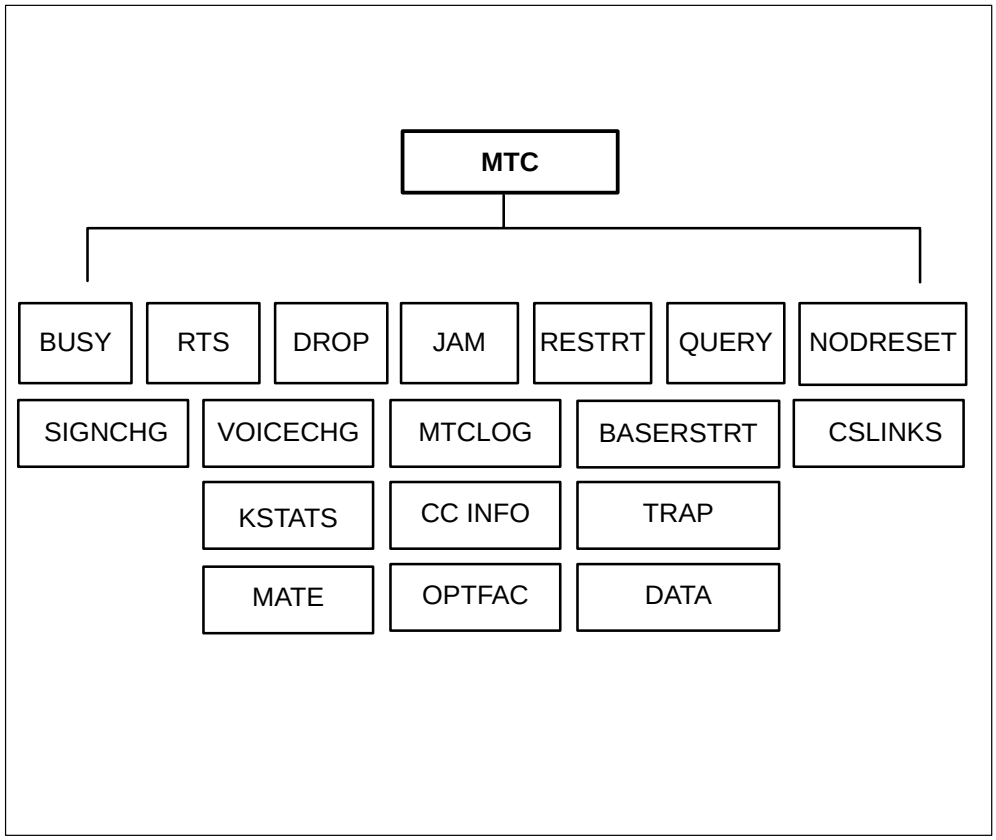
CAUTION
Do not use the MTC level on a live switch.
Commands at the MTC level take a peripheral out-of-service, which can cause severe service interruption.

The MTC (Maintenance) level is accessed when the MTC command is entered at the LTCMP level.

The MTC level contains commands for performing maintenance on the XPM.

Figure 4-30 on page 4-117 contains the commands available at the MTC level.

Figure 4-30
Commands in the MTC level



The commands at the MTC level produce a display as follows:

```
STATE  UCISSGOIFSJR    LAST CMD     RESPONSE
-------------------------------------------------
BUSY      X        INACTIVE BUSY    REQ INVALID
```

The fields are as follows:

STATE is the state of the peripheral. The possible states are as follows:

- busy
- nil
- running
- ready
- sync
- restart
- esa

- readyforesa

UCISSGOIFSJR

are maintenance state attribute booleans. An *X* under an abbreviation indicates the condition is true. The abbreviations are as follows:

U SWERRs (PM180 and PM189 logs) are being sent to the CC when the boolean is set to TRUE.

C CC_link_ok indicates that the peripheral is communicating with the CC when the boolean is set to TRUE.

I IMC_ok indicates that the fixed link between the units of the XPM is operating when the boolean is set to TRUE.

S In_super_sync indicates that the unit is in superframe sync when the boolean is set to TRUE.

Note: Peripherals downline from the XPM expect a constant framing pulse and a superframe every 240 frames. The inactive unit uses the mate superframe phase comparator to compare the superframe it generates with the superframe generated by the active unit.

S In_sync indicates the unit is in sync with the synchronizing source when the boolean is set to TRUE.

G Gained indicates gain is high when an activity interrupt is received when the boolean is set to TRUE.

O Overload indicates the peripheral is highidle or local overload when the boolean is set to TRUE.

I Initial indicates a restart is required when the boolean is set to TRUE.

F Freeze indicates the peripheral is frozen after an uncontrolled SwAct when the boolean is set to TRUE.

S SwAct indicates the peripheral is under controlled SwAct when the boolean is set to TRUE.

J Jam indicates the peripheral is jammed inactive when the boolean is set to TRUE.

R Reload indicates a reload is required when the boolean is set to TRUE.

activity inactive or active indicates whether the terminal is active for the unit on which PMDEBUG is being used.

LAST CMD displays the last command sent to the XPM monitor. This field allows you to verify the command is correct.

RESPONSE can be one of the following:

- req invalid
- invalid
- nil
- acknowledge
- fail
- ok

Following are descriptions of the commands available in the MTC level:

BUSY command

The BUSY command manually busies (MBsy) the peripheral.

BUSY	

RTS command

The RTS command returns the peripheral to service.

RTS	

DROP command

The DROP command causes the side of the peripheral PMDEBUG is being used on to give activity to the mate.

DROP	

JAM command

The JAM command jams the peripheral inactive.

JAM	

RESTRT command

The RESTRT command causes the peripheral to restart.

RESTRT	

QUERY command

The QUERY command requests a query message to XPM maintenance.

QUERY	

NODRESET command

The NODRESET command resets the P-side node.

NODRESET	ext_node unit

Where:

ext_node is the external node number

unit is the unit number

SIGNCHG command

The SIGNCHG command changes the P-side node signal on a specified port and channel number.

SIGNCHG	port chnl signal side

Where:

port is the port number

chnl is the channel number

signal is the signal you send to the specified port and channel. The signal is one of the following:

- open
- mtc_open
- close

side indicates the side of the peripheral on which the new signal will enter. The side is one of the following:

- cside
- pside

Note: The ABORT command can be used to interrupt the prompting sequence.

VOICECHG command

The VOICECHG command changes the speech signal for a specified port.

VOICECHG	port signal side

Where:

port is the port number

signal is the signal you send to the specified port and channel. The signal is one of the following:

- busy
- ok

side indicates the side of the peripheral on which the new signal will enter. The side is one of the following:

- cside
- pside

Note: The ABORT command can be used to interrupt the prompting sequence.

MTCLOG command

The MTCLOG command displays the last ten maintenance log messages.

MTCLOG	

CSLINKS command

The CSLINKS command displays the C-side link's status.

CSLINKS	

KSTATS command

The KSTATS command is not implemented.

KSTATS	

CC INFO command

The CC INFO command displays information about the CC.

CC INFO	

TRAP command

The TRAP command causes the XPM to trap.

TRAP	

MATE level

The MATE level contains commands for communicating with the mate unit. Following are the commands at the MATE level:

OPEN COMMAND The OPEN command opens the mate links.

RESET COMMAND The RESET command resets the mate unit over the IMC link 0 or 1. You are prompted for the IMC number.

STATUS COMMAND The STATUS command sends the mate unit the status message over the IMC link 0 or 1. You are prompted for the IMC number.

INITIALIZE COMMAND The INITIALIZE command sends the mate a run message.

OPTFAC command

The OPTFAC command displays the optional facilities identifier registers.

OPTFAC	

XPMTRAK level

The XPMTRAK level is accessed when the XPMTRAK command is entered at the LTCMP level.

The XPMTRAK level contains three tools and various commands that allow you to trace procedures affecting the tools. XPMTRAK tools capture data based on specific events relating to call processing. Following is a list of the tools:

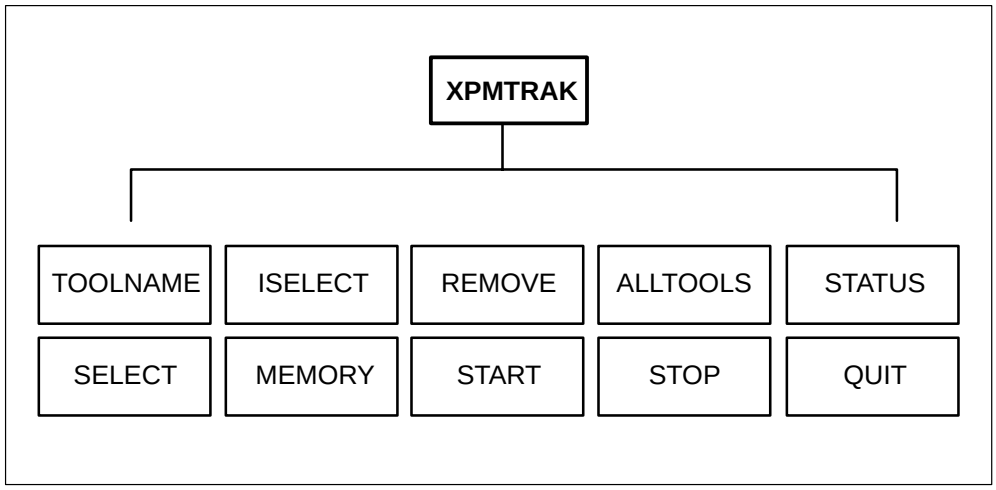
Pgmtrc program trace captures procedure call events in the TPT task.

Msgtrc message trace captures messages between the CC and XPM.

Trmtrc Terminal trace captures call processing events from hooks in the code.

Figure 4-31 on page 4-123 contains the commands available at the XPMTRAK level.

Figure 4-31
Commands in the XPMTRAK level



Toolname command

You can select or manipulate any or all of the tools. You must type one of the following commands to select either program trace (Pgmtrc), message trace (Msgtrc), or terminal trace (Trmtrc):

- Pgmtrc
- Msgtrc
- Trmtrc.

TOOLNAME	ON OFF Bufsize n Query Clear Display [Full | Brief]

Where:

ON turns the tool on

OFF turns the tool off

Bufsize changes the buffer size

n the amount of buffer storage (**n** applies to **Bufsize** only)

Example:

```
P B 600
>
The new buffer size for PGMTRC is 600.
New buffer size will take effect the next time the tool is turned on.
>
MP:XPMtrak>
```

Query displays buffer information

Example:

```
P Q
** PGMTRC query **
>
1660 of 2550 buffers used for PGMTRC.
>
Current buffer size : 800
>
Minimum buffer size : 400
>
Maximum buffer size : 1525
>
Display is set to FULL
>
MP:XPMtrak>
```

Clear clears the contents of the tool's buffers

CAUTION
Use caution with the Clear command; the data you have captured will be erased.

Display displays the tool's procedures during the specified time span

- **Full** displays the tool's list of procedures, including the segment name, procedure name, include time, exclude time, time stamp, and depth. Figure 4-32 on page 4-125 shows an example of the PROGRAM DISPLAY FULL command.
- **Brief** displays the tool's list of procedures, including the segment name, procedure name, time stamp, and depth.

Figure 4-32
Example of PROGRAM DISPLAY FULL command

```
P D F
>
633 of 800 table entries used.
>
Thinking
>
#
>
#
>
#
Proc call trace for TID:  0025 0025 - TPT
>

>
     Segment          Procedure          Include time      Exclude Time    Time Stamp   Depth

110  TPTTASK     27   TPTORIGC    0    0  508    0    0  508    16:42:44.80  *********
>
110  TPTTASK     26   TPTHDLR     5  151  164    0    0  755    16:42:44.80  *********
>
110  TPTTASK     24   CHECKCPB    0    0  132    0    0  132    16:42:44.80  **********
>
110  TPTTASK     22   RAISESWA    0    0   92    0    0   92    16:42:44.80  **********
>
115  TPTLCCA     32   SOCTRAFF    0    0  412    0    0  142    16:42:44.80  **********
>
115  TPTLCCA     24   GDTRETXO    0    0  168    0    0  168    16:42:44.80  ***********
>
115  TPTLCCA     16   SOCXMITL    0    0  101    0    0  101    16:42:44.80  ***********
>
 24  DEBUGPER    23   DISABLEC    5  149  771    0    0  188    16:42:44.80  **********
>
 24  DEBUGPER    15   TRACE       5  149  582    5  149  582    16:42:44.80  ***********
>
110  TPTTASK      5   FIATSTAR    0    5  381    0    0  205    16:42:49.95  *********
>
14   LNGIPC       5   GETPTRPR    0    0  113    0    0  113    16:42:49.95  **********
>
 78  TPTSUTL     34   SETMDBPT    0    0  385    0    0  217    16:42:49.95  **********
>
 78  TPTSUTL     33   SETLINEC    0    0  167    0    0  167    16:42:49.95  ***********
>
109  TPT          2   LCMSTART    0    4  677    0    0   78    16:42:49.95  **********
>
 87  TPTPOTS      4   POTSINTE    0    0  279    0    0  279    16:42:49.95  ***********
```

Figure 4-32
Example of PROGRAM DISPLAY FULL command

```
100  SERPRIM      39   EVEROUT    0    4  318    0    0  333    16:42:49.95  ***********
>
100  SERPRIM      40   DIRECTMS   0    0  213    0    0  213    16:42:49.95  ************
>
 87  TPTPOTS       5   POTSARBI   0    1  411    0    0  245    16:42:49.95  ************
>
 87  TPTPOTS      10   ARBDNINI   0    1  166    0    0  159    16:42:49.95  ************
*>
 79  TPTUTIL       8   LINKCHB    0    0  684    0    0  156    16:42:49.95  **************
>
 79  TPTUTIL      38   VALIDMDB   0    0  169    0    0  169    16:42:49.95  ***************
>
 79  TPTUTIL      39   GETNEWCH   0    0  358    0    0  358    16:42:49.95  ***************
>
 79  TPTUTIL      10   LINKFNBL   0    0  322    0    0  322    16:42:49.96  **************
>
 90  TPTDIRNM     12   DNFTN      0    2  361    0    0  110    16:42:44.86  ************
>
 90  TPTDIRNM     24   DNLCM2     0    2  155    0    0   88    16:42:49.96  **************
>
 90  TPTDIRNM     19   DNOBSKEY   0    2   66    0    0  172    16:42:49.96  ***************
>

MP:XPMtrak>
```

SELECT command

The SELECT command selects a terminal. Once the terminal is selected, you can process commands about the specified terminal. The tools collect data for the selected terminals.

SElect	**[internal term] [ext node] [ext term]**

Where:

internal term is the terminal number you want to select.

ext node You can select an external node and external terminal number; the system will convert them to the internal terminal number. You must specify both parameters.

ext term You can select an external terminal number and external node; the system will convert them to the internal terminal number. You must specify both parameters.

REMOVE command

The REMOVE command deselects a terminal.

Remove	[internal term] [ext node] [ext term] [All]

Where:

internal term is the terminal number you want to deselect

ext node You can select an external node and terminal number; the system will convert them to the internal terminal number

ext term You can select an external node and terminal number; the system will convert them to the internal terminal number

All You can remove multiple terminals from the system with the All command

ALLTOOL command

You can access all three tools using the ALLTOOL command.

Alltool	ON OFF Query Clear Display [Full | Brief]

Where:

ON turns the tools on

OFF turns the tools off

Query displays buffer information

Clear clears the contents of the buffers for all tools

CAUTION
Use caution with the Clear command; the data you have captured will be erased.

Display displays the tool's procedures during the specified time span

- **Full** displays detailed information about each tool. Figure 4-33 on page 4-129 shows an example of the ALLTOOL DISPLAY FULL command.

  - — **Pgmtrc** displays the tool's list of procedures, including the segment name, procedure name, include time, exclude time, stamp, and depth.
  - — **Msgtrc** displays information transferred between the MP and SP.
  - — **Trmtrc** displays the occurrence of a particular event.
- **Brief** displays limited information about each tool. Figure 4-34 on page 4-134 shows an example of the **Alltool Display Brief** command.
  - — **Pgmtrc** displays the tool's list of procedures, including the segment name, procedure name, stamp, and depth.
  - — **Msgstrc** displays information transferred between the MP and SP; however, all of the hex code is not displayed.
  - — **Trmtrc** displays the occurrence of a particular event. The output is in hex code. Most users will select the full display because the hex code is translated.

Figure 4-33
Example of ALLTOOL DISPLAY FULL command

```
A D F
>
******* Tracing data gathered for PGMTRC *******
>
801 of 1600 bufs processed
>
Thinking
<
#
>
#
>
#
>
Proc call trace for TID:  0025 0025 - TPT
>
    Segment              Procedure          Include time     Exclude Time   Time Stamp    Depth
>
110  TPTTASK        26   TPTHDLR      7  922  372     0    0  568     17:00:18.64   ***
>
110  TPTTASK        24   CHECKCPB     0    0  155     0    0  155     17:00:18.64   ****
>
 14  LNGIPC          5   GETPTRPR     0    0  113     0    0  113     17:00:18.64   ****
>
110  TPTTASK        11   CHCKSLCT     7  910  690     0    0  270     17:00:18.64   ****
>
 14  LNGIPC          5   GETPTRPR     0    0  113     0    0  113     17:00:18.64   *****
>
24   DEBUGPER       23   DISABLEC     7  910  306     0    0  187     17:00:18.64   *****
>
24   DEBUGPER       15   TRACE        7  910  118     7  910  118     17:00:18.64   ******
>
101  TRMTRC         57   TTRACESC     0    0  213     0    0  118     17:00:26.55   ****
>
101  TRMTRC          8   USERSTAT     0    0   94     0    0   94     17:00:26.55   *****
>
101  TRMTRC         58   TTRCCAPT     0    0  149     0    0  149     17:00:26.55   ****
>
101  TRMTRC         58   TTRCCAPT     0    0  149     0    0  149     17:00:26.57   *******
>
101  TRMTRC         57   TTRACESC     0    0  209     0    0  115     17:00:26.57   *******
>
101  TRMTRC          8   USERSTAT     0    0   94     0    0   94     17:00:26.57   ********
>
101  TRMTRC         58   TTRCCAPT     0    0  149     0    0  149     17:00:26.57   *******
```

Figure 4-33
Example of ALLTOOL DISPLAY FULL command (continued)

```
>
 90  TPTDIRNM      12  DNFTN       0    3  911    0    0  117    17:00:26.57  *******
>
101  TRMYRC        57  TTRACESC    0    0  209    0    0  115    17:00:26.57  ********
>
101  TRMTRC         8  USERSTAT    0    0   94    0    0   94    17:00:26.57  *********
>
101  TRMTRC        58  TTRCCAPT    0    0  150    0    0  150    17:00:26.57  ********
>
 90  TPTDIRNM,     18  DNLCMFTN    0    3  433    0    0   97    17:00:26.57  ********
>
 90  TPTDIRNM      24  DNLCM2      0    3  335    0    0   89    17:00:26.57  *********
>
 90  TPTDIRNM      19  DNOBSKEY    0    0  246    0    0  174    17:00:26.57  **********
>
 87  TPTPOTS        2  PLOGWRKP    0    0  638    0    0  112    17:00:26.57  ***********
>
101  TRMTRC        57  TTRACESC    0    0  213    0    0  118    17:00:26.57  ************
>
101  TRMTRC         8  USERSTAT    0    0   94    0    0   94    17:00:26.57  *************
>
101  TRMTRC        58  TTRCCAPT    0    0  149    0    0  149    17:00:26.57  ************
>
87   TPTPOTS        6  POTSQPHY    0    0  162    0    0  162    17:00:26.57  ************
>
78   TPTSUTL        2  SENDCCMS    7    2  433    0    0  855    17:00:26.57  ***********
>
247  LAMMP         10  TRGSTORE    0    0  104    0    0  104    17:00:26.57  ************
>
115  TPTLCCA       26  SOCTRANS    0    0  654    0    0  217    17:00:26.57  ************
>
115  TPTLCCA        5  DQIDLE      0    0  142    0    0  142    17:00:26.57  *************
>
115  TPTLCCA       10  QGDT        0    0  295    0    0  136    17:00:26.57  ************
>
115  TPTLCCA        9  QFLINK      0    0  158    0    0  158    17:00:26.57  **************
>
247  LAMMP         13  TRGPEGSE    0    0  384    0    0  119    17:00:26.57  ************
>
247  LAMMP         11  SHOULDPE    0    0  265    0    0  265    17:00:26.57  *************
>
101  TRMTRC        57  TTRACESC    0    0  243    0    0  149    17:00:26.57  ************
>
101  TRMTRC         8  USERSTAT    0    0   94    0    0   94    17:00:26.57  *************
>
```

Figure 4-33
Example of ALLTOOL DISPLAY FULL command (continued)

```
101  TRMTRC      58  TTRCCAPT   0    0  190   0    0  190   17:00:26.58  ***********
>
101  TRMTRC      57  TTRACESC   0    0  214   0    0  119   17:00:26.58  *******
>
101  TRMTRC       8  USERSTAT   0    0   95   0    0   95   17:00:26.58  ********
>
101  TRMTRC      58  TTRCCAPT   0    0  149   0    0  149   17:00:26.58  *******
>
87   TPTPOTS      3  PPHYWRKP   0    1  501   0    0  572   17:00:26.58  ***
>
101  TRMTRC      57  TTRACESC   0    0  211   0    0  115   17:00:26.58  ***
>
101  TRMTRC       8  USERSTAT   0    0   96   0    0   96   17:00:26.58  *****
>
101  TRMTRC      58  TTRCCAPT   0    0  149   0    0  149   17:00:26.58  ***
>
 87  TPTPOTS      8  BUNDLEMS   0    0  568   0    0  258   17:00:26.58  ****
>
 87  TPTPOTS      9  INITIPCH   0    0  128   0    0  128   17:00:26.58  *****
>
 87  TPTPOTS      7  FORMATMS   0    0  180   0    0  180   17:00:26.58  *****
>
247  LNGIPC       4  BUFFRELE   0    0  265   0    0  265   17:00:26.58  ***
>
 24  DEBUGPER    23  DISABLEC   0   67  146   0    0  188   17:00:26.58  ***
>
 24  DEBUGPER   1 5  TRACE      0   66  957   0    0  142   17:00:26.58  ****
>
*******  Tracing data gathered for MSGTRC  *******
>
<0001>  SRC:  SP nilname 0A    DEST:  SP  FLWO    B7    BUF#:  025
>
GET:  00:17:00:26.52 SEND:  00:17:00:26.53 RELEASE:  -  -  -  -  -  -  -
>
DATA:  0010 B70A 02A4 0400 0B46 4A00 464A 0026
>
        B1FF  FFFF  FFFF  FFFF  FFFF  FFFF  FFFF  FFFF
>
         FFFF  FFFF  FFFF  FFFF  FFFF  FFFF  FFFF  FFFF
>
           FFFF  FFFF  FFFF FFFF  FFFF  FFFF  0008  0008
>
<0002>    SRC:    SP    nilname  0A    DEST:    MP    TPTP3    A4    BUF#:  025
>
GET:  -  -  -  -  -  -  -       SEND:  00:17:00:26.55  RELEASE:  00:17:00:26.58M
>
```

Figure 4-33
Example of ALLTOOL DISPLAY FULL command (continued)

```
DATA:   0015  B70A  02A4  0400  0B46  4A00  464A  0026
>
        B1FF  6BE8  FFFF FFFF FFFF FFFF FFFF FFFF
>
        FFFF  FFFF  FFFF  FFFF  FFFF  FFFF  FFFF  FFFF
>
        FFFF  FFFF  FFFF  FFFF  FFFF  FFFF  0008  0008
>
<0003>   SRC:   MP   TPTP3   A4   DEST:   MP  NETMSG   CC   BUF#:  008
>
GET:  00:17:26.57  SEND:  00:17:00:26.58  RELEASE:  00:17:00:26.59
>
DATA:   000A  0000  02A4  0400  0C00  FFFF  FFFF  FFFF
>
       FFFF  FFFF  FFFF  FFFF  FFFF  FFFF  FFFF  FFFF
>
       FFFF  FFFF  FFFF  FFFF  FFFF  FFFF  FFFF  FFFF
>
        FFFF  FFFF  FFFF  FFFF  FFFF  FFFF  0025  0025
>
<0004>   SCR:   MP   TPTP3   A4   DEST:   MP   MSGDMSX  84   BUF:   026
>
GET:  00:17:00:26.58  SEND:  00:17:00:26.58 RELEASE:  00:17:00:26.59
>
DATA:   000B  01A4  0400 040A  0601  FFFF FFFF
>
       FFFF  FFFF  FFFF  FFFF  FFFF  FFFF  FFFF
>
        FFFF  FFFF  FFFF  FFFF  FFFF  FFFF  FFFF
>
        FFFF FFFF FFFF FFFF FFFF 0025  0025
>
*******  Tracing data gathered for TRMTRC  *******
>
>
XPM TIME  ->> 00:17:42:35.47  CC TIME  ->>  30:10:19:55.17
>
00:17:00:26.55  676  ,- -  A7‹ ATHB  allocated:  446  ;  Tpt_Queue_Hdlr
>
00:17:00:26.55  676  ,- -  00>  ***************   Scr  =  Idle  Queue   ******************
>
00:17:00:26.55  676  ,- -  3D>  POTS  msg  rcvd:  OFFHOOK:  RC=  CONTINUE;  AA=  #00
>
00:17:00:26.56  676  ,- -  D7>  Event rtr1;  Feat type= ARBLTRAT; MT=  #10
>
00:17:00:26.56  676  ,- -  DE>  POTS  arb1;  fiat= FIATLCM;  MT=  #10;  state= NILSTATE
```

Figure 4-33
Example of ALLTOOL DISPLAY FULL command (continued)

```
>
00:17:00:26.56  676  ,- -  DF>  POTS  arb2;  MT=  #10action=  DNINIT
>
00:17:00:26.56  676  ,- -  A9>  CHB  allocated:  217
>
00:17:00:26.56  676  ,- -  AB>  FNB allocated:  548  ;  feat=  DN
>
00:17:00:26.57  676  ,- -  D8>  Event rtr2;  Fn  blk  rc=  SAMEBLK;  AA=  #00
>
00:17:00:26.57  676  ,- -  D7>  Event  rtr1;  Feat type=  DN;  MT=  #15
>
00:17:00:26.57  676  ,- -  D9>  Dn  ftn;  fiat=  FIATLCM;  MT=  #15;  intnl  st=  DKIDLE
>
00:17:00:26.57  676  ,- -  3F>  POTS  log wrk  STONHKSC;  AA=  #00
>
00:17:00:26:58  676  ,- -  02>  CC MSG:  CCORIGIN;  MT  (PMIST)=  #0C00
>
00:17:00:26.58  676  ,- -  D8>  Event  rtr2;  Fn blk  rc=  DONE;  AA=  #00
>
00:17:00:26.58  676  ,- -  E1>  POTS  phys wrk:  SCANONHO
```

Figure 4-34
Example of ALLTOOL DISPLAY BRIEF command

```
A  D  B
>
******* Tracing data gathered for PGMTRC *******
>
801 of 1600 bufs processed
>
Thinking
<
#
>
#
>
#
>
Proc call trace for TID:  0025 0025 - TPT
>
    Segment              Procedure          Include time      Exclude Time   Time Stamp    Depth
>
110   TPTTASK        26    TPTHDLR                                          17:00:18.64   ***
>
110   TPTTASK        24    CHECKCPB                                         17:00:18.64   ****
>
 14   LNGIPC          5    GETPTRPR                                         17:00:18.64   ****
>
110   TPTTASK        11    CHCKSLCT                                         17:00:18.64   ****
>
 14   LNGIPC          5    GETPTRPR                                         17:00:18.64   *****
>
24    DEBUGPER       23    DISABLEC                                         17:00:18.64   *****
>
24    DEBUGPER       15    TRACE                                            17:00:18.64   ******
>
101   TRMTRC         57    TTRACESC                                         17:00:26.55   ****
>
101   TRMTRC          8    USERSTAT                                         17:00:26.55   *****
>
101   TRMTRC         58    TTRCCAPT                                         17:00:26.55   ****
>
101   TRMTRC         58    TTRCCAPT                                         17:00:26.57   *******
>
101   TRMTRC         57    TTRACESC                                         17:00:26.57   *******
>
101   TRMTRC          8    USERSTAT                                         17:00:26.57   ********
>
101   TRMTRC         58    TTRCCAPT                                         17:00:26.57   *******
>
```

Figure 4-34
Example of ALLTOOL DISPLAY BRIEF command (continued)

```
 90  TPTDIRNM      12   DNFTN                17:00:26.57  *******
>
101  TRMYRC        57   TTRACESC             17:00:26.57  ********
>
101  TRMTRC         8   USERSTAT             17:00:26.57  *********
>
101  TRMTRC        58   TTRCCAPT             17:00:26.57  ********
>
 90  TPTDIRNM,     18   DNLCMFTN             17:00:26.57  ********
>
 90  TPTDIRNM      24   DNLCM2               17:00:26.57  *********
>
 90  TPTDIRNM      19   DNOBSKEY             17:00:26.57  **********
>
 87  TPTPOTS        2   PLOGWRKP             17:00:26.57  ***********
>
101  TRMTRC        57   TTRACESC             17:00:26.57  ************
>
101  TRMTRC         8   USERSTAT             17:00:26.57  *************
>
101  TRMTRC        58   TTRCCAPT             17:00:26.57  ************
>
87   TPTPOTS        6   POTSQPHY             17:00:26.57  ************
>
78   TPTSUTL        2   SENDCCMS             17:00:26.57  ***********
>
247  LAMMP         10   TRGSTORE             17:00:26.57  ************
>
115  TPTLCCA       26   SOCTRANS             17:00:26.57  ************
>
115  TPTLCCA        5   DQIDLE               17:00:26.57  *************
>
115  TPTLCCA       10   QGDT                 17:00:26.57  ************
>
115  TPTLCCA        9   QFLINK               17:00:26.57  **************
>
247  LAMMP         13   TRGPEGSE             17:00:26.57  ************
>
247  LAMMP         11   SHOULDPE             17:00:26.57  *************
>
101  TRMTRC        57   TTRACESC             17:00:26.57  ************
>
101  TRMTRC         8   USERSTAT             17:00:26.57  *************
>
101  TRMTRC        58   TTRCCAPT             17:00:26.58  ************
>
```

Figure 4-34
Example of ALLTOOL DISPLAY BRIEF command (continued)

```
101  TRMTRC        57   TTRACESC                          17:00:26.58  *******
>
101  TRMTRC         8   USERSTAT                          17:00:26.58  ********
>
101  TRMTRC        58   TTRCCAPT                          17:00:26.58  *******
>
87   TPTPOTS        3   PPHYWRKP                          17:00:26.58  ***
>
101  TRMTRC        57   TTRACESC                          17:00:26.58  ****
>
101  TRMTRC         8   USERSTAT                          17:00:26.58  *****
>
101  TRMTRC        58   TTRCCAPT                          17:00:26.58  ****
>
 87  TPTPOTS        8   BUNDLEMS                          17:00:26.58  ****
>
 87  TPTPOTS        9   INITIPCH                          17:00:26.58  *****
>
 87  TPTPOTS        7   FORMATMS                          17:00:26.58  *****
>
247  LNGIPC         4   BUFFRELE                          17:00:26.58  ***
>
 24  DEBUGPER      23   DISABLEC                          17:00:26.58  ***
>
 24  DEBUGPER      15   TRACE                             17:00:26.58  ****
>
*******  Tracing data gathered for MSGTRC  *******
>
<0001>  SRC:  SP nilname 0A    DEST:  SP  FLWO    B7    BUF#:  025
>
GET:  00:17:00:26.52 SEND:  00:17:00:26.53 RELEASE:  - - - - - - -
>
DATA:  0010 B70A 02A4 0400 0B46 4A00 464A 0026
>
         B1FF
>
<0002>    SRC:    SP    nilname  0A    DEST:    MP    TPTP3    A4    BUF#:  025
>
GET:  - - - - - - -       SEND:  00:17:00:26.55  RELEASE:  00:17:00:26.58M
>
DATA:    0015  B70A  02A4  0400  0B46  4A00  464A  0026
>
          B1FF  6BE8  FFFF
>
<0003>    SRC:    MP    TPTP3    A4    DEST:    MP  NETMSG    CC    BUF#:  008
>
```

Figure 4-34
Example of ALLTOOL DISPLAY BRIEF command (continued)

```
GET:  00:17:26.57  SEND:  00:17:00:26.58  RELEASE:  00:17:00:26.59
>
DATA:    000A  0000  02A4  0400  0C00  FFFF
>

<0004>    SCR:    MP    TPTP3    A4    DEST:    MP    MSGDMSX  84    BUF:    026
>
GET:  00:17:00:26.58  SEND:  00:17:00:26.58 RELEASE:  00:17:00:26.59
>
DATA:    000B  03A4  02A4 0400 040A  0601
>
*******  Tracing data gathered for TRMTRC  *******
>
XPM TIME  ->> 00:17:08:53.84  CC TIME  ->>  30:09:46:13.33
>
00:17:00:26.55    676  ,- -  A7>    #01    #BE    #00    #00
>
00:17:00:26.55  676  ,- -    00>     #02    #00   #00    #00
>
00:17:00:26.55  676  ,- -  3D>     #10     #00    #00    #00
>
00:17:00:26.56  676  ,- -  D7>     #00     #10    #00    #00
>
00:17:00:26.56  676  ,- -  DE>    #0B    #10    #00    #00
>
00:17:00:26.56  676  ,- -  DF>     #10    #01    #00    #00
>
00:17:00:26.56  676  ,- -  A9>     #00    #D9    #00    #00
>
00:17:00:26.56  676  ,- -  AB>     #02    #24    #01    #00
>
00:17:00:26.57  676  ,- -  D8>     #05    #00    #00    #00
>
00:17:00:26.57  676  ,- -  D7>     #01    #15    #00    #00
>
00:17:00:26.57  676  ,- -  D9>    #0B    #15    #00    #00
>
00:17:00:26.57  676  ,- -  3F>    #3D    #00    #00    #00
>
00:17:00:26.58  676  ,- -  02>    #0C    #00    #01    #00
>
00:17:00:26.58  676  ,- -   D8>   #02    #00    #00    #00
>
00:17:00:26.58  676  ,- -   E1>   #06    #00    #00    #00
>
```

MEMORY command

The MEMORY command displays information about memory: free memory, the tool name, its current memory buffer size, the minimum memory buffer size, and the status of the tool (in use or not in use).

Memory	

Example:

```
ME
****  Current XPMTRAK memory status  ****
>
Free memory  :       424948 bytes
Tool          Current mem.(bufsize)    Minimum mem.(bufsize)     Status
>
===           ================     =================      ====
>
PGMTRC            46 K ( 800 )              23 K ( 400 )          not in use
```

STATUS command

The STATUS command displays the status of the selected tools and whether they have been started or stopped.

STatus	

START command

The START command activates collection of data for all tools simultaneously. No data is displayed.

STARt	

STOP command

The STOP command stops tracing. No data is displayed.

STOp	

QUIT command

The QUIT command ends XPMTRAK processing.

Quit	

ISELECT command

The ISELECT command is used to select an ISDN terminal to trace. You can refer to the terminal by specifying parameters.

Iselect	[ext node] [ext term] [ltid] [loop_tid] [set_tid]

In order to trace a specified terminal, you must specify parameters in one of the following sequences:

- **ext node, ext term, ltid**
- **loop_tid, ltid**
- **set_tid**

Where:

ext node is the external node number

ext term is the external terminal number

ltid is the ltid byte

loop_tid is the loop_tid byte

set_tid is the set_tid byte

Example:

```
MP:XPMtrak> ISELECT 3 15  7

MP:XPMtrak> ISELECT 5 2

MP:XPMtrak> ISELECT 7032
```

BIGFOOT level

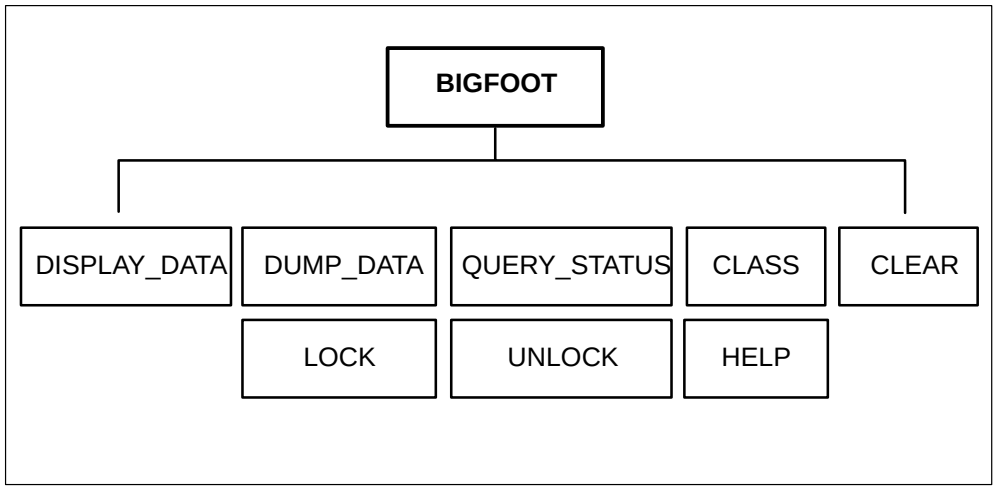
The Bigfoot level is accessed when the BIGFOOT command is entered at the LTCMP level.

You can capture a series of relevant events and dates, which may have led to a XPM switch or loss of activity. The data is stored in memory buffers as events and is capable of surviving most XPM initializations and resets, including XPM program reloads. Bigfoot captures data throughout the normal operation of the XPM and stores the data into memory that contain information about various XPM operations and their results. Bigfoot provides the following:

- information about critical XPM events that may have caused or led to an outage
- a foot print of the XPM unit just before the reinitialization
- information that would otherwise be lost over the XPM unit's initialization

Figure 4-35 on page 4-140 contains the commands available at the BIGFOOT level.

Figure 4-35
Commands in the BIGFOOT level



DISPLAY_DATA command

The DISPLAY_DATA command displays specified, captured data.

Display_data	**[a] [h]**

DUMP_DATA command

The DUMP_DATA command displays all of the captured data.

a is the data in the active buffer

h is the data in the holding buffer

DUmp_data	**[a] [h]**

QUERY_STATUS command

The QUERY_STATUS command displays the current status of BIGFOOT.

a is the data in the active buffer

h is the data in the holding buffer

QUERY_status	

CLASS command

The CLASS command enables the specified class.

CLass	[E]	[D]	[n]	[a]	[q]

where

e will enable the specified class

d will disable the specified class

n is the number of the class

a enables all classes

q lists all enabled and disabled classes

CLEAR command

The CLEAR command clears data in the active buffer or holding buffer

CLEar	[a] [h] [b]

where

a is the data in the active buffer

h is the data in the holding buffer

b is the data in active and holding buffers

LOCK command

The LOCK command locks the active or holding buffers.

Lock	[a] [h]

where

a is the active buffer

h is the holding buffer

UNLOCK command

The UNLOCK command unlocks the active or holding buffers.

where

a is the active buffer

h is the holding buffer

SURVIVE command

The SURVIVE command overrides the default settings and has the unit use the classes that were enabled previous to the RTS.

Note: The only time the defaults will take effect with SURVIVE enabled is after an initial program load.

where

e will enable the specified class

d will disable the specified class

q displays whether survive has been enabled or disabled

Survive	**[E]** **[D]** **[Q]**

HELP command

The HELP command gives information and syntax on the various commands.

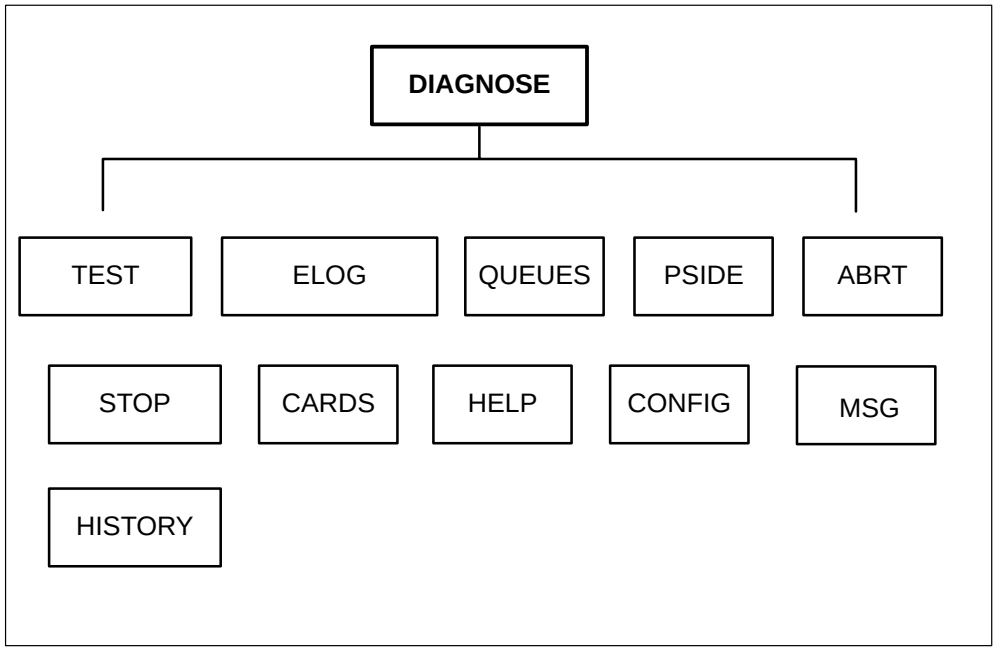
Help	

DIAGNOSE level

The Diagnostics (DIAGNOSE) level is accessed when the DIAGNOSE command is processed at the LTCMP level.

The DIAGNOSE level contains commands for requesting that diagnostics be performed and viewing the results. Figure 4-36 on page 4-143 contains the commands available at the DIAGNOSE level.

Figure 4-36
Commands in the DIAGNOSE level



Note: All numbers input and displayed at the DIAGNOSE level are in hexadecimal. Numbers can be forced to decimal by preceding the number with a percent (%) sign, such as %84.

Following are the commands available in the DIAGNOSE level:

ABRT command

The ABRT command aborts a pending diagnostic or set (one that is running or waiting to run). This command is used after the TEST or CARDS command is issued.

ABRT	r = n

where

n n is a value obtained from the queues command

CARDS command

The CARDS command lists the cards that are currently equipped in the unit. This command displays the shelf configuration table.

CARDS	

CONFIG command

The CONFIG command displays the configuration of the diagnostic system and the current utilization of the system.

CONFIG	

ELOG command

The ELOG command displays the error logs associated with the last diagnostic or set of diagnostics requested by the monitor.

Note: The ELOG command will not diaplay error logs associated with TESTCALLs requested from the MAP.

ELOG	

HELP command

The HELP command displays the format of the commands in the DIAGNOSE level and lists the diagnostics that may be performed. If no parameter is entered with the HELP command, the commands available at the DIAGNOSE level are displayed.

HELP	SETS DIAGS

Where:

SETS displays the names of the sets and the diagnostics associated with each set

DIAGS displays the names of the diagnostics available at the DIAGNOSE level

PSIDE command

The PSIDE command formats and displays the P-side link and node configuration of the XPM unit.

PSIDE	

QUEUES command

The QUEUES command displays the status of the ready queue and the execution queue of the diagnostic system. This command displays the diagnostics that are currently waiting for process (ready queue) and the diagnostics that are currently running (process queue).

QUEUES	

STOP command

The STOP command prematurely terminates the running diagnostic.

STOP	

TEST command

The TEST command allows you to run diagnostics and view the results.

TEST	diag sets options

Note: If the test command is used to run a single diagnostic the "I" (isolation) option should be used so isolation will occur if the diagnostic fails. If diagnostic fails and isolation is not requested, the generated cardlist is longer and is not odored with the most likely suspect card at the top of the list. Isolation is automatically run when a failure occurs while running a diagnostic set such as, INSVALL, OSVCALL, FACAUD.

Where:

diag is an individual diagnostic or a set (for example: osvcall, insvall, facavd). Diagnostic names can be displayed by typing HELP DIAGS.

sets are sets of diagnostics. Diagnostic set names can be displayed by typing HELP SETS.

options is one of the following:

L indicates the number of times to repeat the test. This option is followed by the number of times the test will be repeated. Following is an example of the L option:

T INSVALL (L=n

where n is the number of times the test will be looped. The default is 1.

Note: If you set L = FF, the diag or set will repeat continuously.

I when a single diagnostic is run, indicates that fault isolation will take place if the diagnostic fails. Isolation is executed by an automatic requesting of more diagnostics to run in the event of a failure.

Note 1: Ordering of the cardlist reported by a diagnostic failure occurs only if isolation is requested. An ordered cardlist has the most likely suspect card that caused the fault at the top of the list.

Note 2: Isolation is automatic when a failure occurs while running a diagnostic set such as, INSVALL, OSVCALL, FACAUD.

T INSVALL (I

P indicates parameters will be passed to a diagnostic. The parameters are entered as hexadecimal bytes and must be separated by commas. Only six parameter bytes will be accepted. Following is an example of the P option:

T INSVALL (P=n,n1,n2,...,n5)

where n...n5 are hexadecimal bytes. The parameters entered with the P option are dependent on the diagnostic being run.

S indicates that static data must be present before the diagnostic can run. Following is an example of the S option:

T INSVALL (S

R indicates the resource (such as a port number) to be tested. This option is followed by the resource number, which is dependent on the diagnostic to be performed. The parameters are entered as hexadecimal bytes and must be separated by commas. Only six parameter bytes will be accepted. Following is an example of the R option:

```
T INSVALL (R=n,n1,n2,...,n5)
```

where n is the resource number. More than one resource number may be specified.

HISTORY level

The HISTORY level provides access to commands that record the results of each of the diagnostic which are run as part of the facitilty audit, (FACAUD set). Results are recorded on a service basis for each of the diagnostic.

VIEW command

The VIEW command displays the recorded results for each of the last diagnostics run by service state.

VIEW	

CLEAR command

The CLEAR command resets the recorded results of one of the history entries.

CLEAR	entry state

Where:

entry is a diagnostic index in the history table or “0” to clear all entries.

state is the service state cleared for the entry.

CAUTION
Do not use the PASS command.
It is recommended that the PASS command not be used. It forces a "passed" into the history table for the requested diagnostic entry and state.

PASS command

It is recommended that the PASS command not be used. It forces a "passed" into the history table for the requested diagnostic entry and state.

PASS	entry state

Where:

entry is a diagnostic index in the history table or "0" to clear all entries.

state is the service state cleared for the entry.

CAUTION
Do not use the FAIL command
It is recommended that the FAIL command not be used. It forces a "failed" into the history table for the requested diagnostic entry and state.

FAIL command

It is recommended that the FAIL command not be used. It forces a "failed" into the history table for the requested diagnostic entry and state.

FAIL	entry state

Where:

entry is a diagnostic index in the history table or "0" to clear all entries.

state is the service state cleared for the entry.

CAUTION
Do not use the ADD command
It is recommended that the ADD command not be used because it adds an entry to the history table.

ADD command

It is recommended that the ADD command not be used because it adds an entry to the history table.

ADD	

CAUTION
Do not use the DELETE command
It is recommended that the DELETE command not be used because it deletes an entry in the history table.

DELETE command

It is recommended that the DELETE command not be used because it deletes an entry in the history table.

DELETE	

MSG level

The message (MSG) level

MSG	text r = n

Where:

text a string of up to 20 characters.

n is the number obtained from the queue command output.

The terminal trace level (TRMTRC)

The Terminal Trace (TRMTRC) level is accessed when the TRMTRC command is processed from the LTCMP level.

You can trace the software path of a call and capture specific data at each of the call stages using the TRMTRC level. This level can be used to monitor the stages of line and trunk call processing.

Any number of trunks and lines can be monitored at one time. You can specify a range of terminals to be monitored.

The following data is captured at each key point (hook) during the progression of a line or trunk call:

- event type
- event data, as follows:

  trunks: four data bytes

  lines: three data bytes and an extension byte

- time stamp
- terminal number

An event is defined by an entry in the call-processing code where a tracepoint (hook) is inserted. Each event is identified with a unique hex number. Presently, 84 such hooks exist. A list is available in module UTLCALLP.

A maximum of 32,768 events can be stored in the TRMTRC buffer. Storage size is dependent on the amount of temporary storage available when the TRMTRC tool is initialized.

Each hook you specify is tested against the terminal range. If the terminal number of the hook falls in the specified range, the event type, four bytes of data, the terminal number, and a time stamp are stored in the TRMTRC buffer. You can display the information stored in the buffer in either a hex or a translated dump. If the dump is translated, it is presented to you in a readable format.

You have a choice of detail when selecting the data for TRMTRC to capture. They are broken down as follows:

- LEVEL 1 - Includes high-level information, such as:
  - — trunks
    - – unfiltered AB bit changes received/transmitted
    - – start/stop of AB to SV bit mapping
    - – transmitted AB bit changes
    - – DP digits received/sent
  - — lines
    - – information from logical work processors
    - – information from KSET and POTS physical interfaces
    - – feature messages
    - – digit reports
  - — both
    - – confusion, CC-directed, and *close report* messages sent
    - – TPT and AU (CHB cleanup) SWERRs
    - – tone connections and requests
- LEVEL 2 - Includes level 1 information and slightly more detailed information, such as:
  - — trunks
    - – filtered AB scanner messages
    - – trunk long timers
    - – AB primitives processed
  - — lines
    - – tones applied
    - – channels allocated/deallocated
    - – long timers
    - – integrity events
    - – line long timers
  - — both:
    - – TPT messages
    - – major primitives processed
    - – execs posted
    - – C-side and P-side connections
    - – RCC makeconns
    - – short timers

  - — RCC intraswitch channels allocated
  - — terminal event queue information
- LEVEL 3 - Includes level 1 and 2 information and very detailed information such as:
  - — Trunks: filter timers.
  - — Lines:
    - – event router information
    - – DN functions
    - – POTS and KSET arbitrator information
    - – MADN ringing information.
  - — Both:
    - – starting and killing of timers
    - – SWACT synchronization requests
    - – terminal functions started and killed
    - – CP intent information
    - – block linking and delinking.

After the desired level of detail has been chosen for a range of terminals and the tool is enabled, you can display all the activities captured on the selected range.

Interactions with other tools

TRMTRC can be used in conjunction with other debugging tools. However, TRMTRC can consume most of the temporary storage to hold the captured data. Reduced temporary storage can prevent TRMTRC from running.

Therefore, all debugging tools except TRMTRC should be started, allowing the tools to access the amount of temporary storage necessary. Then, TRMTRC can be started, using the temporary storage remaining. If the temporary storage remaining is not enough for TRMTRC, you must turn off some of the other tools.

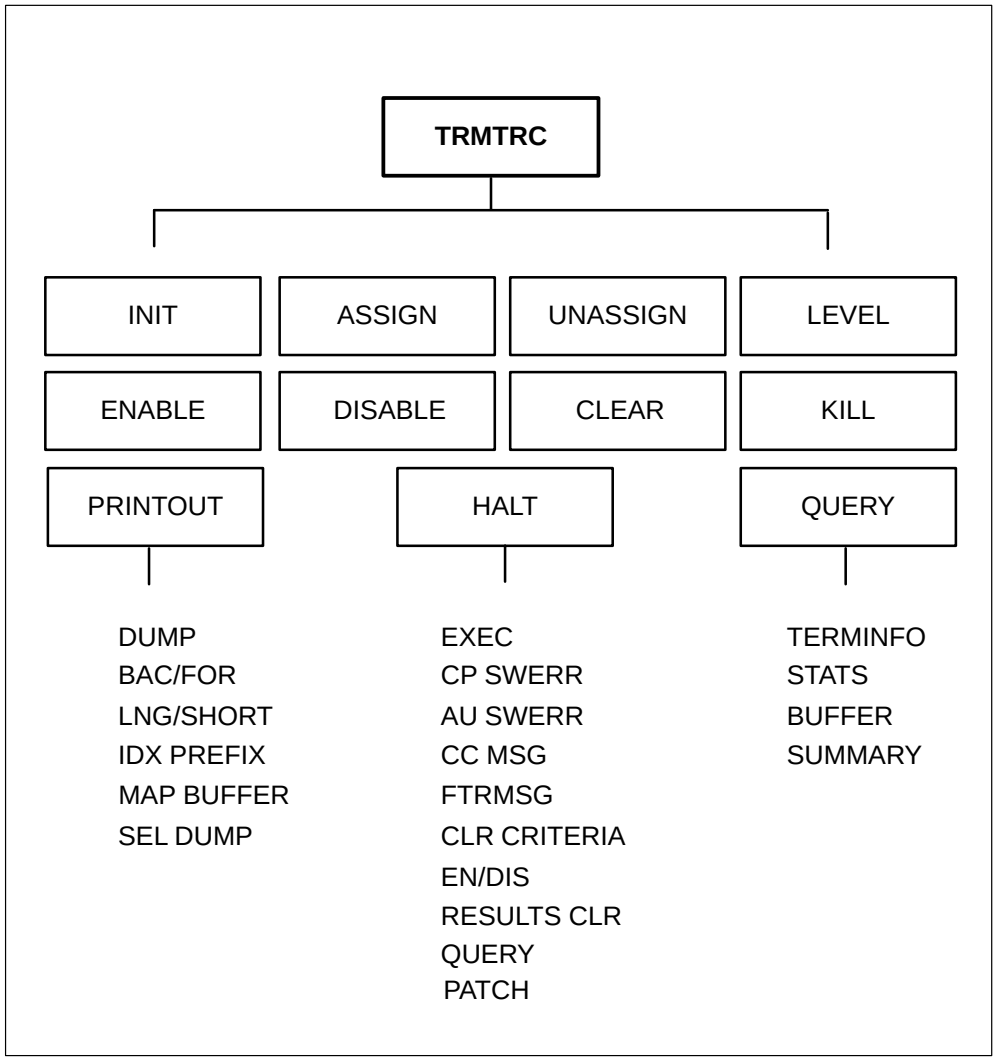
Restrictions

This tool should only be used on large memory loads.

Command descriptions

Figure 4-37 on page 4-153 contains the commands available in the TRMTRC level.

Figure 4-37
Commands in the TRMTRC level



Following are descriptions of the commands available in the TRMTRC level:

Note 1: TRMTRC must be initialized before any commands at the TRMTRC level (except INIT and QUERY TERMINFO) will operate.

Note 2: Any of the commands that generate continuous lines of output can be stopped by pressing **ENTER**.

INIT command

The INIT command initializes the data area used by TRMTRC and sets the default settings on the tool. INIT is the first command processed for the

TRMTRC tool. If the TRMTRC tool has been initialized, the KILL command must be processed before the INIT command can be reissued.

You must specify the working terminal set and the number of physical buffers to be used. The larger the working terminal set, the more temporary storage is used to track the terminals assigned for tracing. Each physical buffer is 5 kilobytes in size and can store 512 terminal events.

INIT	

ASSIGN command

The ASSIGN command activates a range of terminals in the working set.

You must enter the range of internal terminal numbers to activate ASSIGN. (The range consists of the lowest and the highest internal terminal numbers of the set of terminals to be activated.) The internal terminal numbers specified with the ASSIGN command must have been specified in the working set with the INIT command.

ASSIGN	

UNASSIGN command

The UNASSIGN command deactivates a range of terminals in the working set. The parameters are the lowest and the highest internal terminal numbers of the set to be activated. The internal terminal numbers specified with the UNASSIGN command must have been specified in the working set with the INIT command.

UNASSIGN	

LEVEL command

The LEVEL command allows you to set the level of detail to capture events. The default is level 1.

You are prompted to enter **1**, **2**, or **3**.

LEVEL	

ENABLE command

The ENABLE command enables those terminals specified with the ASSIGN command. Once enabled, TRMTRC will continuously capture data from the hooks in the code until you disable tracing or use the HALT command.

ENABLE	

DISABLE command

The DISABLE command disables the terminals currently being traced. After the processing of this command, the terminals are still assigned and can be immediately reenabled.

DISABLE	

CLEAR command

The CLEAR command clears the logical buffer. Tracing must be disabled before the CLEAR command can be issued.

CLEAR	

QUERY level

The QUERY level contains four subcommands, which provide information on tool configuration/status.

QUERY	

Commands available at the QUERY level are as follows:

TERMINFO requires a range of terminals as parameters. TERMINFO displays the following data for each terminal:

- type of terminal
- internal node and terminal number
- external node and terminal number
- for AB trunks, the P-side port and channel
- whether this terminal is assigned

Note 1: The range of terminals specified as parameters for the TERMINFO command does not have to be in the working set.

Note 2: TERMINFO can be processed even if the tool is not initialized for the purpose of getting node and terminal number information.

STATS requires a range of terminals as parameters. STATS displays the number of snapshots (events) captured in the buffer for each terminal in the range.

BUFFER displays the current write pointer in the logical buffer and the number of times the buffer has wrapped. The buffer number displays the next buffer that will be written to.

SUMMARY displays a summary of the current configuration of the tool, including:

- the internal state machine (a state machine keeping track of the tracing enabled, halt criteria, halt enabled, and halt results flags and disallowing illegal combinations)
- number of physical buffers in use
- capacity (number of events) of each physical buffer
- capacity of each logical buffer
- the detail level
- which printing (long or short) is active
- the direction of print
- whether index prefixing is on or off
- the working terminal set
- a list of assigned terminals

HALT level

The HALT level consists of ten subcommands that control the halting system. If one of the terminals being traced hits one of the following conditions, tracing will stop, allowing you to analyze the buffer for error conditions:

HALT	

Following are the commands available at the HALT level:

EXEC stops TRMTRC when one to five user-specified execs are processed. The parameters are in the range #00 to #FF. You can specify ALL to halt on the first exec posted.

CP SWERR stops TRMTRC when one to five user-specified Call Processing (CP) SWERRs are generated. The parameters are in the range #00 to #FF. You can specify ALL to halt on the first occurrence of a CP SWERR.

AU SWERR stops TRMTRC when the cleanup of the CHB for a terminal occurs. This is the only CP block that can be triggered, since it is the only block containing the terminal number.

CC MSG stops TRMTRC when one to five user-specified CC directive messages are generated. The parameters are in the range #0000 to #FFFF. You can also specify ALL to halt on the first CC message sent.

FTRMSG stops TRMTRC when one to five user-specified feature messages are generated. Halting occurs. The parameters are in the range #00 to #FF. You can specify ALL to halt on the first feature message sent.

Note: The parameters of the FTRMSG are the feature type found in a PMIST feature message. A feature message has a PMIST message type of "#96 00 ...," and the feature type is the third byte.

PATCH stops TRMTRC when one of up to five user-specified patch halt conditions occurs. This is a special halt condition that allows you to patch the XPM in many different locations. These patches cause TRMTRC to halt. Each patched location has an identifier. This identifier is entered as a parameter to the PATCH command. These parameters (identifiers) are in the range #00 to #FF. You can also specify ALL to halt on the first patch hit.

CLR CRTRIA clears the TRMTRC halt criteria. You are prompted for the halt condition criteria that should be cleared

EN/DIS is a toggle to enable/disable the halting mechanism

RESULTS CLR clears the results from the last halt that occurred. You can then reenable the halt system.

QUERY displays the current state of the halting system including:

- the state of the TRMTRC state machine (a machine state keeping track of the tracing enabled, halt criteria, halt enabled, and halt results flags and disallowing illegal combinations)
- the results of a halt, if a halt has occurred
- the halt criteria currently defined

PRINTOUT level

The PRINTOUT level consists of six subcommands that handle the printing and scanning of the logical buffer. Tracing should be stopped before any form of printing/scanning can take place. The time stamps in the printout are the MP time stamps.

PRINTOUT	

Following are the commands available at the PRINTOUT level:

DUMP accepts a range of terminals as parameters. The DUMP command displays the information stored in the buffer in the currently defined direction with the current format (long or short). If index prefixing is on, the buffer/offset will be printed before each line of output.

BAC/FOR toggles the direction of print between forward and backward. The default is forward.

LNG/SHORT toggles the direction of print between long (translated) or short (hex dump). The default is long.

IDX PREFIX toggles the index-prefixing mode between on and off. The default is off. If on, the buffer/offset number will be printed before each line displayed.

MAP BUFFER allows you to scan the buffer for information.

The buffer can be scanned for specific information. The minimum amount of information that must be specified is as follows:

st_buf the first buffer to scan (start buffer)

st_off the offset in the start buffer

ed_buf the last buffer to scan (end buffer)

ed_off the offset in the end buffer

To pinpoint the information, restrictions must be specified. The available restrictions are as follows:

t terminal number

l level of detail

e event type (id)

d1 first data byte

d2 second data byte

d3 third data byte

d4 fourth data byte

Two options are also available when mapping the buffer:

p Print the results to the screen. This parameter is used only when the search criteria is by terminal number only; if other criteria are specified, printing is the only option.

nz Do not include terminal 0 when searching for a match on specified criteria. Sometimes in a trace, terminal 0 gets captured. The default is to always include it in the search.

Any combination of parameters can be entered at the command line *in the following order*:

m t <tn> l <lvl> e <ev> d1 <dd> d2 <dd> d3 <dd>
d4 <dd> st_buf st_off ed_buf ed_off p nz

For example, the basic parameters are the start and stop points in the buffer to scan. The command string, **m 0 4c 3 26**, will scans the logical buffer from buffer #0 offset #4C to buffer #3 offset #26. The beginning and end blocks of information for the terminals between the two buffer points are displayed. The information reported includes the terminal number, event type, buffer and offset, and time stamp of each start and stop point.

Restrictions can be placed on the information to be displayed. For example, **m t 483 0 4c 3 26**, scans the logical buffer from buffer #0 offset #4C to buffer #3 offset #26 for internal terminal number 483 (decimal). The beginning and end blocks of information for terminal 483 are displayed.

In the example, **m t 483 e 23 0 4c 3 26**, information for terminal 483 concerning event type #23 is displayed. Instead of displaying start/stop information, the actual line is printed out.

Note: Any search criteria specified in place of, or in addition to, the terminal number causes the specified events to be printed out.

In the example, **m e 23 0 4c 3 26**, only captured events with the with event number #23 are displayed.

In the example, **m t 483 l 2 0 4c 3 26**, only captured events for terminal 483 with detail level 2 or 1 are displayed.

In the example, **m e 23 d1 34 d2 62 0 4c 3 26**, only captured events with event number #23 and data bytes one and two having values #34 and #62 (respectively) are displayed.

In the example, **m t 123 0 0 3 86 p**, the buffer for terminal 123 is mapped and the matching events are displayed.

In the example, **m t 123 0 0 3 83 p nz**, the buffer for terminal 123 is mapped and printed, but information for terminal 0 is not included.

SEL DUMP dumps a selective part of the buffer. No restrictions apply. The parameters are the start buffer and offset and the end buffer and offset. For example, **s 145 2 13**, displays from buffer #1 offset #45 to buffer #2 offset #13.

KILL command

The KILL command deactivates the terminal tracing tool by setting the global variables controlling the hooks to FALSE, releasing the data structures allocated from the INIT command, and cleaning up global variables.

KILL	

The AUDIT level

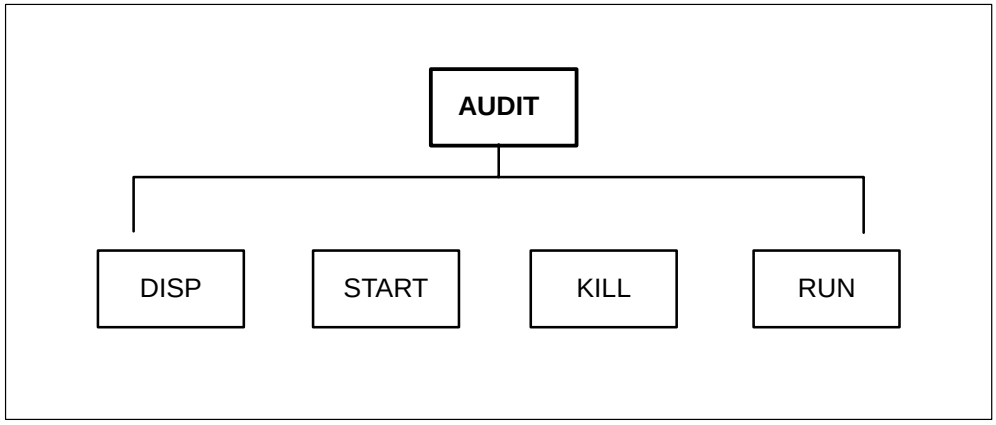
The AUDIT level is accessed when the AUDIT command is processed from the LTCMP level.

The SP and the MP run audits in specific hardware or software components of the XPM to detect faults. Audits can detect such errors as:

- task error
- message error
- sanity time-out

Figure 4-38 on page 4-161 contains the commands available a the AUDIT level.

Figure 4-38
Commands in the AUDIT level



Following are descriptions of the commands available in the AUDIT level:

DISP command

The DISP command displays the audit table and displays whether an audit is active. This command displays how frequently an audit processes and the current count. When the interval count is equal to the current count, the audit processes.

DISP	

KILL command

The KILL command disables all audits.

KILL	

RUN command

The RUN command allows you to process one audit or all audits. The audits do not need to be active for this command to process.

You are prompted for the CID of a particular audit or FF to run all audits.

RUN	cid FF

Where:

cid is the communication identifier of a task

FF indicates all tasks will be run

START command

The START command initializes and activates the audits. The individual audit counts are reset to 0, and a delayed IPC message is sent to start the audits. This command has no effect on audits that are already active.

All audits run automatically.

START	

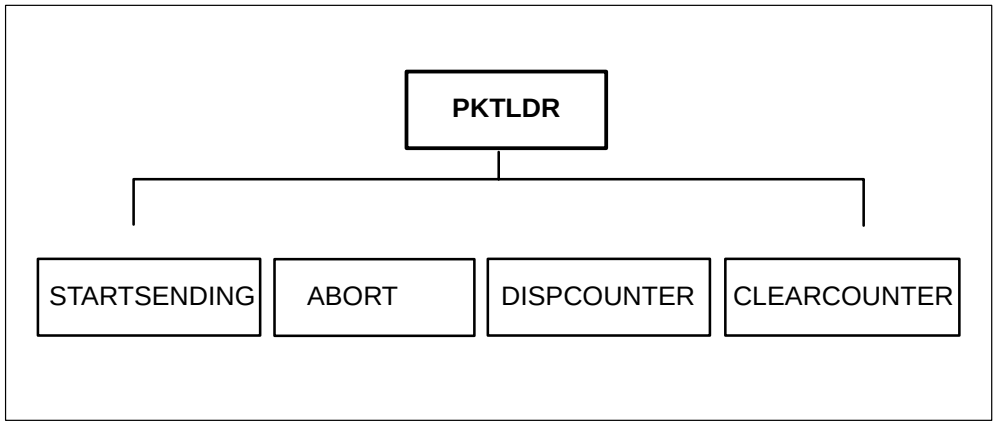
PKTLDR level

The PKTLDR level is accessed when the PKTLDR command is processed from the LTCMP level.

The PKTLDR level contains commands for debugging the functionality of packet messaging. Packets are used to transport long messages.

Figure 4-39 on page 4-163 shows the commands in the PKTLDR level.

Figure 4-39
Commands in the PKTLDR level



Following are the commands available in the PKTLDR level:

ABORT command

The ABORT command stops messages from being transported.

ABORT	

CLEARCOUNTER command

The CLEARCOUNTER command sets the error counter, lost messages counter, and messages received counter to zero. This command cannot be processed after the STARTSENDING command is entered. Message transportation should be halted with the ABORT command before the CLEARCOUNTER command is entered.

CLEARCOUNTER	

DISPCOUNTER command

The DISPCOUNTER command displays a summary of run time, messages lost, and messages received.

DISPCOUNTER	

STARTSENDING command

The STARTSENDING command allows you to enter parameters for long messages and to start transporting the messages.

STARTSENDING	

The CP level

The CP level of the LTC monitor displays the internal data structures associated with call processing, such as the active terminal header block (ATHB), the extended data block (XDB), the terminal variable area (TVA), the call header block (CHB), and the function block (FNB).

Active terminal header block (ATHB)

When a terminal becomes involved with a call, an ATHB is associated with the terminal. An ATHB is allocated for every active terminal involved with a particular call. The ATHB is released once the terminal becomes inactive.

In the case of digital trunks, an ATHB is associated with the trunk when the trunk is initialized. The ATHB is not released until the trunk is reinitialized.

The ATHB contains such information as:

chb_idx index of the CHB

aud_athb_free audit information

aud_athb_marked audit information

aud_athb_suspect audit information

buzzer_on buzzer on indicator

active_set which set (main or extension) is active

dtsr for controlling DTSR OM

ps_pe P-side path end value

sdb_idx index to the set data block

ps_pe_count number of logical P-side path end

lcm_time_stamp time stamp of the origination

active_addr address of the active extension byte

ltype type of line

wait_cc_response wait for CC message

queued_in_idle_q idle queue indicator

queued_msg_idx IPC index of queued message

start_pt index into LCM bundling buffer

kset_ring_count number of DNs ringing

disp_ctl_block control block for display set

trml_tone current P-side tone

XDB

XDB is used to hold additional information about a call. The XDB can be any of the following:

- set data block (SDB)
- function block (FNB)
- A/B trunk data block (ABDB)

Information common to all XDBs is as follows:

next_xdb index to the next extension block

aud_xdb_free audit information

aud_xdb_mark audit information

aud_xdb_suspect audit information

format the format for the extension block, such as FNB or ABDB

Terminal variable area (TVA)

When a subscriber goes off-hook, the CC instructs the peripheral to associate a channel with the active terminal. A TVA is assigned to the active terminal. One TVA is associated for every Call Header Block (CHB).

The TVA is a 64-byte buffer that holds information about a call in progress. The CC accesses the TVA by sending a string of primitives to the terminal.

The primitives contain instructions to select specific information and send the information to the CC.

The TVA contains information such as the digits outpulsed or received, tone information (such as the timing specifications and on/off patterns), and the state of the terminal.

The contents of the TVA change throughout the course of the call.

The TVA stores data for the following functions:

- supervision
- reflexing
- integrity
- time-stamping

The TVA uses 16 bytes for reflexing. The first two bytes are the message type, and the remaining 14 bytes are the reflex buffer, which contains a sequence of primitives.

Each report that passes through the terminal is compared with the message type in the reflex buffer, if a reflex has been requested. If the two message types match, the current process running on the terminal is completed, and the series of primitives in the reflex buffer is processed.

Two types of reflex messages are used. A nontransparent reflex causes the message to the CC to be intercepted and discarded. A transparent reflex sends the message to the CC. A transparent reflex is specified by a NOOP primitive as the first available byte in the reflex buffer.

The reflex message reduces the message traffic between the CC and the PM without requiring special execs.

The reflex buffer is also used as a data area. Once the reflex buffer is full, data in the buffer is overwritten.

A special primitive reads data from the reflex buffer and puts it on the stack. This application of the reflex buffer is useful for data that is read frequently but seldom changed.

Table 4-1 on page 4-168 contains the major areas and offsets of the TVA and background information on each.

Table 4-1
Terminal variable area offset

Area	Offset	Background Information
PP_MINOR_A_VARS	0	This five byte area is used by terminal process A. The specification of each terminal process defines how the locations are used. Any locations that are not required by the particular terminal process can be used as scratch store for any application.
PP_MINOR_B_VARS	5	This five byte area is designed to be used by terminal process B. The specification of each terminal process defines how the locations are used. Any locations that are not required by the particular terminal process can be used as scratch store for any application.
PP_LARGE_AREA	10	This area has two specific uses: ▪ During digit reception or outpulsing, the large area is used as a digit register and storage for other pertinent information. ▪ When periodic signals (audible ringing, reorder tone) are transmitted, the tones, timing specification, and on off pattern are stored in the large area. If the large area is not required for either of these applications, it can be used as scratch store for any application.
PP_SMALL_AREA	20	This three byte area serves as scratch store for any application that needs it.
TRMNL_STATE_INDIC	23	This offset is used to store the current terminal state indicator (TSI), which is used to help describe the current state of the PM terminal. The TSI is used by the execs, by the PM audit, and by the trunk audit to check for processing-state discrepancies and possible neglected PM terminals.

—continued—

Table 4-1
Terminal variable area offset (continued)

Area	Offset	Background Information
PP_AUDIT_EXEC	26	This offset is used to store the exec-id of the exec to be invoked when the PM audit locates a terminal requiring recovery action.
PP_CHANNEL	24	This offset is used to store the ID of the channel currently associated with the terminal.
PP_INTEG_EXEC, PP_INTEG_TO_EXEC, EXPECT_INTEG_VALUE	25, 29, 28	The offsets are used by the integrity background process.
PP_TIMER_EXEC	30	This offset is used to store the exec-ID that is invoked when a background timer expires. The time-out action of a running timer can be changed by replacing the exec-ID in this location.
PP_FLAG_BYTE	27	This offset is used to store pertinent flag bits. The use of these bits is as follows: 7 This offset contains the network plane from which pcm is currently being received. 6 This offset contains the parity-fail flag. If an integrity loss report is actually a parity error, this flag is 1. 5 This offset indicates whether integrity is found. This flag is cleared when integrity is searched for and is set when integrity is found. This information is useful for debugging purposes. 4 This offset indicates overlapped outpulsing. It enables overlapped DP outpulsing on trunks. 3 This offset is unused.

—continued—

Table 4-1
Terminal variable area offset (continued)

Area	Offset	Background Information
PP_FLAG_BYTE	27	2 When the interval timer is started, this bit is set to 1. Once the interval timer has expired or a kill-timer has been processed, this bit is set to 0.
		1 This offset contains the physical–answer flag. This bit is set when the called party goes off-hook. It is set in AMA_ANSWER_TIME_ BUILDER and reset in TRUNK_IDLE_ BUILDER.
		0 This offset contains the DDO progress mark. This flag is used to record progress through two-stage outpulsing on direct-dialed, overseas calls.

End

Terminal reflex area layout

Table 4-2 on page 4-166 contains a list of the standard offset names within the terminal reflex area.

Table 4-2
Terminal reflex area layout

Standard offset names	Byte	Number
PP_HKMON_ON_EXEC		
PP_HKMON_OFF_EXEC	1	
PP_ON_HOOK_EXEC	2	
PP_OFF_HOOK_EXEC	3	
PP_IDLE_EXEC	4	%% LS NIBBLE
PP_START_EXEC	4	%% MS NIBBLE
PP_SEIZ_ACTION	5	%% LS NIBBLE
PP_OUTPULSE_EXEC	5	%% MS NIBBLE

—continued—

Table 4-2
Terminal reflex area layout (continued)

Standard offset names	Byte	Number
PP_STOP_LIMIT	6	%% LS NIBBLE
PP_CONTROL_OFC	6	%% MS NIBBLE
PP_ID_TIME	7	
PP_MFOTP_EXEC	8	
PP_DPOTP_EXEC	8	
PP_DPREC_EXEC	9	
PP_MIN_MAX	10	
PP_INTERMED_TIME	11	
PP_INIT_TIME	12	
PP_GUARD_TIME	13	
End		

Call header block

A CHB is associated with each call in progress. A CHB is obtained for each terminal that originates a call and each terminal that is terminated by a call. Several CHBs may be linked to one ATHB for a business set. The CHB is released when the call disconnects.

In the case of digital trunks, the CHB is associated with the trunk when the trunk is initialized and is not released until the trunk is reinitialized.

Function block

A FNB is associated with each feature or function in progress, such as call origination, digit collection, or ring again.

Several FNBs can be associated with one CHB.

The FNB contains such information as:

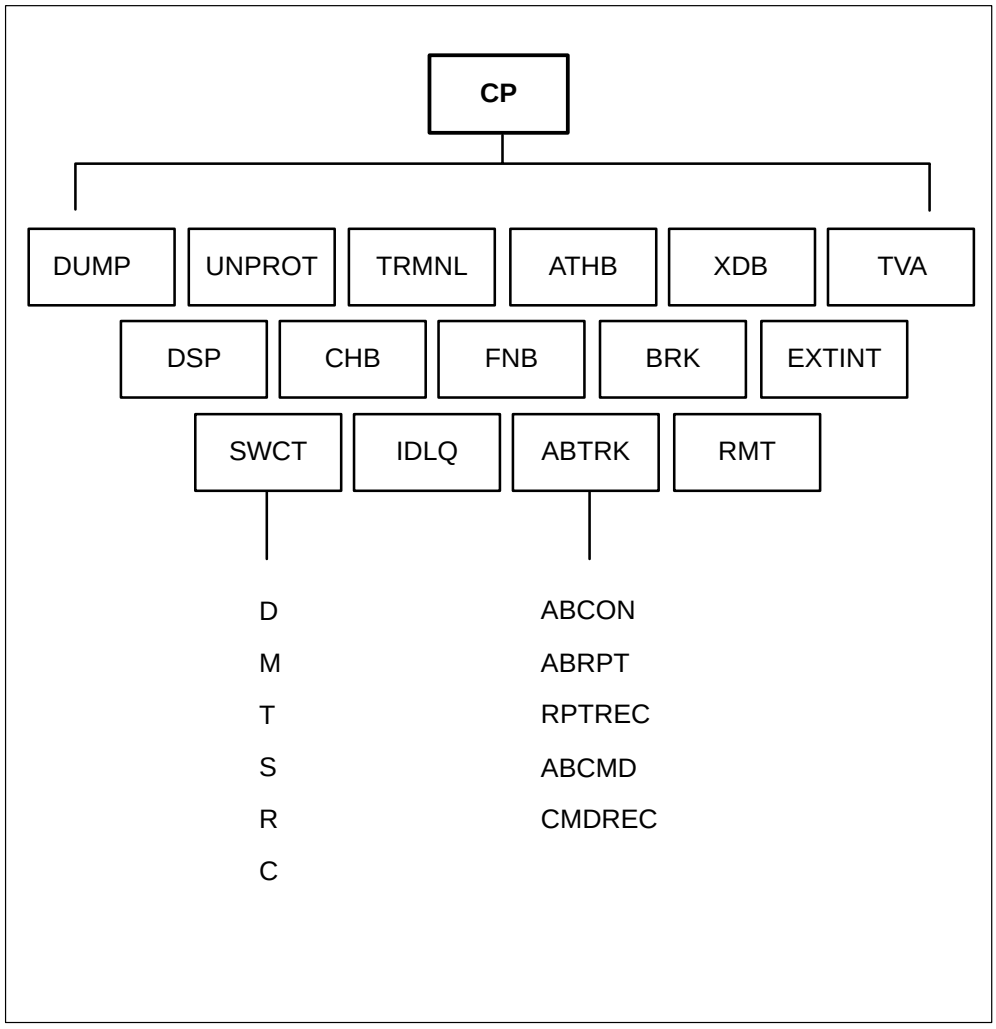
- the index into the next most recent FNB
- the feature associated with the FNB
- audit information
- data specific to each feature
- the state of the feature

Commands at the CP level

The CP level allows you to view and modify the internal data blocks. Either raw or symbolic data can be changed.

Figure 4-40 in page 4-172 contains the commands available at the CP level.

Figure 4-40
Commands in the CP level



The following commands are in the CP level:

ABTRK command

The ABTRK command displays the AB trunk data blocks.

ABTRK	

Following are the commands associated with the ABTRK level:

ABCON command

The ABCON command displays the AB control Block (ABCB).

ABCON	

ABRPT command

The ABRPT command displays the AB report buffer queue.

ABRPT	

RPTREC command

The RPTREC command displays the AB report buffer contents.

RPTREC	

ABCMD command

The ABCMD command displays the AB command buffer queue.

ABCMD	

CMDREC command

The CMDREC command displays the AB command buffer contents.

CMDREC	

ATHB command

The ATHB command prints the contents of the active terminal header block (ATHB).

ATHB	athb number

Where:

athb number is the number of the ATHB to be dumped. The ATHB number is obtained with the TRMNL command. Refer to “TRMNL” command on page 4-186.

Example:

```
Dump,Unprot,Trmnl,Athb,Xdb,tVa,Chb,dsP,Fnb,Mod,Brk,Extint,
Swct,Idlq,abtrK,Rmt,*
MP:CP>

>a 43

AUD_FREE = TRUE
AUD_MARKED = FALSE
AUD_SUSPECT = FALSE
DTSR = TRUE
TDB_IDX = 0
CHB_IDX = 0
BUZZER_ON = TRUE
ACTIVE_ADDR = FF
ACTIVE_SET = MAIN_SET
PS_PE:
  NIL_PATHEND
PS_PE_COUNT = 0

Wait_CC_Response = FALSE
Queued_In_Idle_Queue = FALSE
Queued_Msg_Idx = 0
start_pt = 0
kset_ring_count = 0
terminal tone = no tone
display control blk:
  active display key id =FF
  controlling = FALSE
  data_are_sent = FALSE
  allow_to_send_data = TRUE
  top_line_has_been_updated = FALSE
  bottom_line_has_been_updated = FALSE
  display timer index = 0

Dump,Unprot,Trmnl,Athb,Xdb,tVa,Chb,dsP,Fnb,Mod,Brk,Extint,
Swct,Idlq,abtrK,Rmt,*
MP:CP>
```

BRK command

The BRK command breaks the dynamically allocated data blocks linked to a terminal; therefore, call data blocks associated with the call are released. This command causes a call to drop.

BRK	

CHB command

The CHB command prints the contents of the call header block.

CHB	chb number

Where:

chb number is the number of the CHB to be dumped. The CHB number is obtained with the TRMNL command. Refer to “TRMNL command” on page 4-186.

Example:

```
Dump,Unprot,Trmnl,Athb,Xdb,tVa,Chb,dsP,Fnb,Mod,Brk,Extint,
Swct,Idlq,abtrK,Rmt,*
MP:CP>

>c 43

NEXT_CHB = 0   SR_IDX = 0   FNB_IDX = 0   CDB_IDX = 0
AUD_CHB_FREE = TRUE   MARKED = FALSE  SUSPECT = FALSE
INTTID:  NODENO = 2   TERMNO = 268   EXTBYTE = 0
TFA:    IMPL_CID    FCN_SEQ      FCN_CLASS   FCN_ACTIVE   TF_GP_DATA
a        000008C        2            pset         FALSE        38 32 00 00
b        00000FF        0            nil          FALSE        00 00 00 00
ltime    0000093        1            exec         FALSE        00 00 00 00
timer    00000FF        0            nil          FALSE        00 00 00 00
ntg      000008C        2            scan         FALSE        00 00 00 00

CS_PE:  NIL_PATHEND
RING_TMR = 0   RING_CHAR = 0  REV_RING_CHAR = 0
  SwActTED = FALSE  REFLEX_USED = TRUE
 REMOTE SwAct STATE = sync_tfs      INT_MAPSV = FALSE
FLASH_ALLOWED = FALSE    ASSOCIATED_KEY = 0
 PAD_VALUE = 6

CUR_TONE = audible ring
IA_CHB_IDX = 0   IA_CHN_BLK = FALSE  IA_NTG_FND = FALSE
NTG_FCN_SEQ = 0    NTG_FCN_TYPE = 0
CDB_FCN_SEQ = 0    CDB_FCN_TYPE = 0
ia_ps = FALSE  aud_tfa = FALSE  aud_tfb = FALSE
swab_inc    = 0   swab_out    = 0
IA_NTG_TMR = 0 DDB_IDX = 0  IA_LS = 0  MDN_RING = FALSE
                                                  warble = FALSE

Dump,Unprot,Trmnl,Athb,Xdb,tVa,Chb,dsP,Fnb,Mod,Brk,Extint,
Swct,Idlq,abtrK,Rmt,*
```

```
MP:CP>
```

CNDDB command

The CNDDB command displays the contents of the calling number delivery (CND) data block, which is a refinement of the extended data block.

CNDDB	intterm

Where:

intterm is the internal terminal number on the PM

DSP command

The DSP command displays the display data block (DDB).

DSP	intterm

Where:

intterm is the internal terminal number

Example:

```
Dump,Unprot,Trmnl,Athb,Xdb,tVa,Chb,dsP,Fnb,Mod,Brk,Extint,
Swct,Idlq,abtrK,Rmt,*
MP:CP>

>p 43

 DDB
NEXT_DDB = 0
AUD_FREE = TRUE
AUD_MARKED = FALSE
AUD_SUSPECT = FALSE
CALLING INFO:
  VALID_DSP = FALSE
CALLED INFO:
  VALID_DSP = FALSE

Dump,Unprot,Trmnl,Athb,Xdb,tVa,Chb,dsP,Fnb,Mod,Brk,Extint,
Swct,Idlq,abtrK,Rmt,*
MP:CP>
```

DUMP command

The DUMP command displays all the data blocks associated with a call.

DUMP	intterm

Where:

intterm is the internal terminal number

Example:

```
Dump,Unprot,Trmnl,Athb,Xdb,tVa,Chb,dsP,Fnb,Mod,Brk,Extint,
Swct,Idlq,abtrK,Rmt,*
MP:CP>

>d 43
*** Data blocks for terminal#: 43   002B
ATHB:463
XDB:85
EXT_BYTE:0  CHB:193   DDB:21   FNB:260
ATHB_IDX = 463
TERM_STATE = CPB
MAINSET_OFHK = FALSE
EXTSET_OFHK = FALSE
DTR = TRUE
SW_SYNC_DATA = FALSE
terminal type = keyset
loop config = 1
kset_display_not_equipped = FALSE

*** ATHB: 463
AUD_FREE = FALSE
AUD_MARKED = FALSE
AUD_SUSPECT = FALSE
DTSR = FALSE
TDB_IDX = 85
CHB_IDX = 193
BUZZER_ON = TRUE
ACTIVE_ADDR = 0
ACTIVE_SET = MAIN_SET

PS_PE:
  PORT = 3   CHNL = 14   DIR = TWO_WAY  PTYPE = DYNAMIC
PS_PE_COUNT = 1
Wait_CC_Response = FALSE
Queued_In_Idle_Queue = FALSE
Queued_Msg_Idx = 0
start_pt = 0
kset_ring_count = 0
terminal tone = pcm
```

```
display control blk:
  active display key id =0
  controlling = FALSE
  data_are_sent = FALSE
  allow_to_send_data = TRUE
  top_line_has_been_updated = FALSE
  bottom_line_has_been_updated = FALSE
  display timer index = 0

*** XDB (linked to ATHB):85
 XDB
NEXT_XDB = 0
AUD_FREE = FALSE
AUD_MARKED = FALSE
AUD_SUSPECT = FALSE
FORMAT = SDB
TERM_NO = 43
KSET SDB DATA:
CONFIG      RING_OPT     KEY_DATA
------      --------     --------
TRUE        TRUE         21
FALSE       FALSE        20
FALSE       FALSE        20
FALSE       FALSE        20
FALSE       FALSE        20
FALSE       FALSE        20
FALSE       FALSE        20
FALSE       FALSE        20
            FALSE        20

NO_OF_ADDONS = 0
UNIT_TYPE = 2
EXTENSION = FALSE
PROG_MAP = NIL_KEY_ID
TWC_IN_EFFECT = FALSE
CONF30_IN_EFFECT = FALSE

*** EXT_BYTE: 0  CHB: 193
NEXT_CHB = 0   SR_IDX = 0   FNB_IDX = 260   CDB_IDX = 0
AUD_CHB_FREE = FALSE  MARKED = FALSE  SUSPECT = FALSE
INTTID:  NODENO = 2   TERMNO = 43   EXTBYTE = 0
```

```
TFA:    IMPL_CID    FCN_SEQ      FCN_CLASS  FCN_ACTIVE  TF_GP_DATA
a       00000A3          2           pset       TRUE        20 81 06 00
b       00000FF          0            nil      FALSE        00 00 00 00
ltime   0000093          1           exec      FALSE        00 00 00 00
timer   00000FF          0            nil      FALSE        00 00 00 00
ntg     000008C          2           pset       TRUE        80 80 80 87
CS_PE:  PORT = 9   CHNL = 13   DIR = TWO_WAY  PTYPE = DYNAMIC
RING_TMR = 0   RING_CHAR = 0  REV_RING_CHAR = 0
  SwActTED = FALSE  REFLEX_USED = FALSE
  REMOTE SwAct STATE = sync_tfs      INT_MAPSV = FALSE
FLASH_ALLOWED = TRUE     ASSOCIATED_KEY = 0
  PAD_VALUE = 66
CUR_TONE = pcm
IA_CHB_IDX = 0   IA_CHN_BLK = FALSE  IA_NTG_FND = FALSE
NTG_FCN_SEQ = 0    NTG_FCN_TYPE = 0
CDB_FCN_SEQ = 0    CDB_FCN_TYPE = 0
ia_ps = FALSE  aud_tfa = FALSE  aud_tfb = FALSE
swab_inc    = 0   swab_out    = 0
IA_NTG_TMR = 0 DDB_IDX = 21  IA_LS = 0  MDN_RING = FALSE
                                                    warble = FALSE
*** TVA : 193
A8  AA  A9  C5  A6  CE 00  CD CD 01  21  43  00  00  00  00
00  00  40  33  00  00  00  00  00  B5  00  A0  87  00  08  00
F4  6F  02  00  00  00  00  00  00  01  00  00  00  00  00  00
00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00

*** DDB: 21
 DDB
NEXT_DDB = 0
AUD_FREE = FALSE
AUD_MARKED = FALSE
AUD_SUSPECT = FALSE
CALLING INFO:
  VALID_DSP = FALSE
CALLED INFO:
  VALID_DSP = TRUE
  DSP_TYPE = hex_dsp
  DIGIT_COUNT = 8
  DATA: 94841234

*** FNB: 260
AUD_FREE = FALSE
AUD_MARKED = FALSE
AUD_SUSPECT = FALSE
NEXT_FNB = 0
FN_TMR_IDX = 0
STATE = DN_ACTIVE
```

```
FTR = DN
RING_SET = TRUE
RING_EXT = FALSE
LOGICAL HOOK STATE = OFFHOOK
DN_STATE = DK_ACT_SUP
DN_REPORT = FALSE
HLD_REPORT = FALSE
PRLS_REPORT = FALSE
PRV_REPORT = FALSE
CPK_LAMP_ON = FALSE
DCPK_LAMP_ON = FALSE
PREV_DN_REPORT = FALSE
PREV_HLD_REPORT = FALSE
CC_DATA_ORIG_SENT = FALSE
DU_STATE = NO_TMR
DU_CNVRT_TIP_RING = FALSE
MWQRY_ON = FALSE
KSMOH_RPT_HLD_DN = FALSE

Dump,Unprot,Trmnl,Athb,Xdb,tVa,Chb,dsP,Fnb,Mod,Brk,Extint,
Swct,Idlq,abtrK,Rmt,*
MP:CP>
```

EXTINT command

The EXTINT command converts external node and terminal numbers to internal node and terminal numbers.

EXTINT	extnn exttn

Where:

extnn is the external node number

exttn is the external node number

Example:

```
Dump,Unprot,Trmnl,Athb,Xdb,tVa,Chb,dsP,Fnb,Mod,Brk,Extint,
Swct,Idlq,abtrK,Rmt,*
MP:CP>

>e 41 41

INTERNAL NODE    NUMBER = 2
INTERNAL TERMINAL NUMBER = 43

Dump,Unprot,Trmnl,Athb,Xdb,tVa,Chb,dsP,Fnb,Mod,Brk,Extint,
Swct,Idlq,abtrK,Rmt,*
MP:CP>
```

FNB command

The FNB command prints the contents of the Function Block (FNB).

FNB	fnb number

Where:

fnb number is the number of the FNB to be dumped. The FNB number is obtained with the TRMNL command. Refer to “TRMNL command” on page 4-186.

Example:

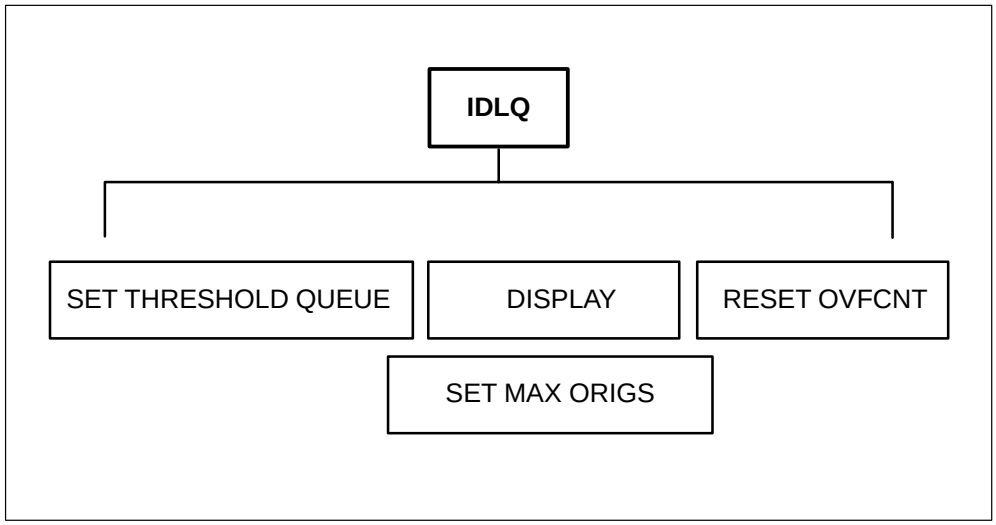
```
Dump,Unprot,Trmnl,Athb,Xdb,tVa,Chb,dsP,Fnb,Mod,Brk,Extint,
Swct,Idlq,abtrK,Rmt,*
MP:CP>

>f 43

AUD_FREE = TRUE
AUD_MARKED = FALSE
AUD_SUSPECT = FALSE
NEXT_FNB = 0
FN_TMR_IDX = 0
STATE = DCOL_DIALTONE

FTR = DIGIT_COLLECT
```

```
FNX_BODY:
AUD_FNX_FREE = TRUE
AUD_FNX_MARKED = FALSE
AUD_FNX_SUSPECT = FALSE
  DC_PARMS:  0  4  0  0  0  0  0  0  0
  DC_REG:  0  0  0  0  0  0  0  0  0
  TRANS_TABLE_IDX = 228
  DC_THRES = 0
  DCCRITI = 0
  DC_DIAL_TONE = TRUE
  INT_DIG_TIMING = TRUE
  LST_DGT_REPORT = FALSE
FN_INT_STATE = 1
  DC_ACT_CODE = 0
  DC_KEY_NO = 0

Dump,Unprot,Trmnl,Athb,Xdb,tVa,Chb,dsP,Fnb,Mod,Brk,Extint,
Swct,Idlq,abtrK,Rmt,*
MP:CP>

Dump,Unprot,Trmnl,Athb,Xdb,tVa,Chb,dsP,Fnb,Mod,Brk,Extint,
Swct,Idlq,abtrK,Rmt,*
MP:CP>

>f 43

AUD_FREE = TRUE
AUD_MARKED = FALSE
AUD_SUSPECT = FALSE
NEXT_FNB = 0
FN_TMR_IDX = 0
STATE = DCOL_DIALTONE

FTR = DIGIT_COLLECT
```

IDLQ command

The IDCQ command allows you to display and change data associated with the idle queue. Figure 4-41 on page 4-183 contains the commands available in the IDLQ level.

Figure 4-41
Commands in the IDLQ level



Following are descriptions of the commands available in the IDLQ level:

SET THRESHOLD QUEUE command

The SET THRESHOLD QUEUE command sets the idle-queue threshold.

SET	

DISPLAY command

The DISPLAY command displays the idle-queue variables.

DISPLAY	

RESET OVFCNT command

The RESET OVFCNT command resets the overflow counter.

RESET OVFCNT	

SET MAX ORIGS command

The SET MAX ORIGS command sets the maximum number of originations.

SET MAX ORIGS	

Example:

```
Dump,Unprot,Trmnl,Athb,Xdb,tVa,Chb,dsP,Fnb,Mod,Brk,Extint,
Swct,Idlq,abtrK,Rmt,*
MP:CP>

>i 43

Set queue threshold, Display, Reset ovfcnt, set Max origs,* quit
MP:CP>

>d
queue threshold              : 10
messages lost from queue     : 0
queue length                 : 0
origination threshold        : 3

Set queue threshold, Display, Reset ovfcnt, set Max origs,* quit
MP:CP>

>m

max originations :
MP:CP>

>5

Set queue threshold, Display, Reset ovfcnt, set Max origs,* quit
MP:CP>

>d

queue threshold              : 10
messages lost from queue     : 0
queue length                 : 0
origination threshold        : 5
Set queue threshold, Display, Reset ovfcnt, set Max origs,* quit
MP:CP>
```

RMT command

The RMT command displays the data associated with a remote cluster controller (RCC).

RMT	

SWCT command

The SWCT command provides access to commands that display and modify booleans associated with SwAct.

SWCT	

Following are the commands associated with the SWCT level:

D displays the SwAct booleans and the associated data

M modifies the SwAct booleans and the associated data

T enables time-data capture

S modifies the sync_disabled boolean

R modifies the sync_table_request boolean

C modifies the cp_data_synced boolean

Example:

```
Dump,Unprot,Trmnl,Athb,Xdb,tVa,Chb,dsP,Fnb,Mod,Brk,Extint,
Swct,Idlq,abtrK,Rmt,*
MP:CP>

>s

SwAct: (D)isplay data, (M)odify data, (T)ime data capture *
MP:CP>

>d

Currently sync_disabled   = FALSE
          sync_tbl_request = FALSE
         cp_data_synced   = TRUE
      last_checked_trmnl = 0 00000000 sync_request_bit=FALSE
       xfr_node_no       = 3
CP_TBL timeout peg count      = 0
SW_TBL timeout peg count      = 0
IMC down timeout peg count    = 0

SwAct: (D)isplay data, (M)odify data, (T)ime data capture *
MP:CP>

>m

Display, Sync_disabled, sync_tbl_Request, Cp_data_sync, *
MP:CP>
```

```
>s

Currently sync_disabled = FALSE
Set bool to (T)rue, (F)alse, anything else aborts.
MP:CP>

Bool left unchanged.
Display, Sync_disabled, sync_tbl_Request, Cp_data_sync, *
MP:CP>

>r

Currently sync_tbl_request = FALSE

Set bool to (T)rue, (F)alse, anything else aborts.
MP:CP>

Bool left unchanged.
Display, Sync_disabled, sync_tbl_Request, Cp_data_sync, *
MP:CP>

>c

Currently cp_data_synced = TRUE
Set bool to (T)rue, (F)alse, anything else aborts.
MP:CP>

Bool left unchanged.
Display, Sync_disabled, sync_tbl_Request, Cp_data_sync, *
MP:CP>

*

SwAct: (D)isplay data, (M)odify data, (T)ime data capture *
MP:CP>
```

TRMNL command

The TRMNL command prints the map of all dynamically allocated data blocks linked to a terminal. The data blocks include the ATHB, the active terminal data block (ATDB), the set data block (SDB), the CHB, and the FNB.

TRMNL	**terminal number**

Where:

terminal number is the terminal number on the PM

Note 1: The TRMNL command lists the ATHB, ATDB, SDB, CHB, and FNB numbers.

Note 2: The TRMNL command should not be used while messages are being sent to the terminal.

Example:

```
Dump,Unprot,Trmnl,Athb,Xdb,tVa,Chb,dsP,Fnb,Mod,Brk,Extint,
Swct,Idlq,abtrK,Rmt,*
MP:CP>

>t 43

ATHB:463
XDB:85
EXT_BYTE:0  CHB:193   DDB:21   FNB:260

Dump,Unprot,Trmnl,Athb,Xdb,tVa,Chb,dsP,Fnb,Mod,Brk,Extint,
Swct,Idlq,abtrK,Rmt,*
MP:CP>
```

TVA command

The TVA command prints the terminal variable area.

TVA	terminal number

Where:

terminal number is the terminal number on the PM

Example:

```
Dump,Unprot,Trmnl,Athb,Xdb,tVa,Chb,dsP,Fnb,Mod,Brk,Extint,
Swct,Idlq,abtrK,Rmt,*
MP:CP>

>v 43

B8 C0 C5 C5 A7 00 00 00 00 00 00 00 00 F0 0A A7
00 00 00 1F 00 00 00 00 00 02 00 20 2D 03 00 00
00 00 00 00 00 00 00 00 00 00 19 00 28 93 0C 0F
6D 01 06 28 93 0C 0F 6D 01 00 00 00 00 00 00 00

Dump,Unprot,Trmnl,Athb,Xdb,tVa,Chb,dsP,Fnb,Mod,Brk,Extint,
Swct,Idlq,abtrK,Rmt,*
MP:CP>
```

UNPROT command

The UNPROT command prints unprotected terminal data.

UNPROT	terminal number

Where:

terminal number is the terminal number on the PM.

Example:

```
MP:CP>
U 43
ATHB_IDX = 463
TERM_STATE = CPB
MAINSET_OFHK = FALSE
EXTSET_OFHK = FALSE
DTR = TRUE
SW_SYNC_DATA = FALSE
terminal type = keyset
loop config = 1
kset_display_not_equipped = FALSE

Dump,Unprot,Trmnl,Athb,Xdb,tVa,Chb,dsP,Fnb,Mod,Brk,Extint,
Swct,Idlq,abtrK,Rmt,*
MP:CP>
```

XDB command

The extension data block (XDB) command displays the XDB.

XDB	terminal number

Where:

terminal number is the terminal number on the PM

Example:

```
Dump,Unprot,Trmnl,Athb,Xdb,tVa,Chb,dsP,Fnb,Mod,Brk,Extint,
Swct,Idlq,abtrK,Rmt,*
MP:CP>

>x 43

 XDB
NEXT_XDB = 0
AUD_FREE = TRUE
AUD_MARKED = FALSE
AUD_SUSPECT = FALSE
FORMAT = NDB

Dump,Unprot,Trmnl,Athb,Xdb,tVa,Chb,dsP,Fnb,Mod,Brk,Extint,
Swct,Idlq,abtrK,Rmt,*
MP:CP>
```

SERVER level

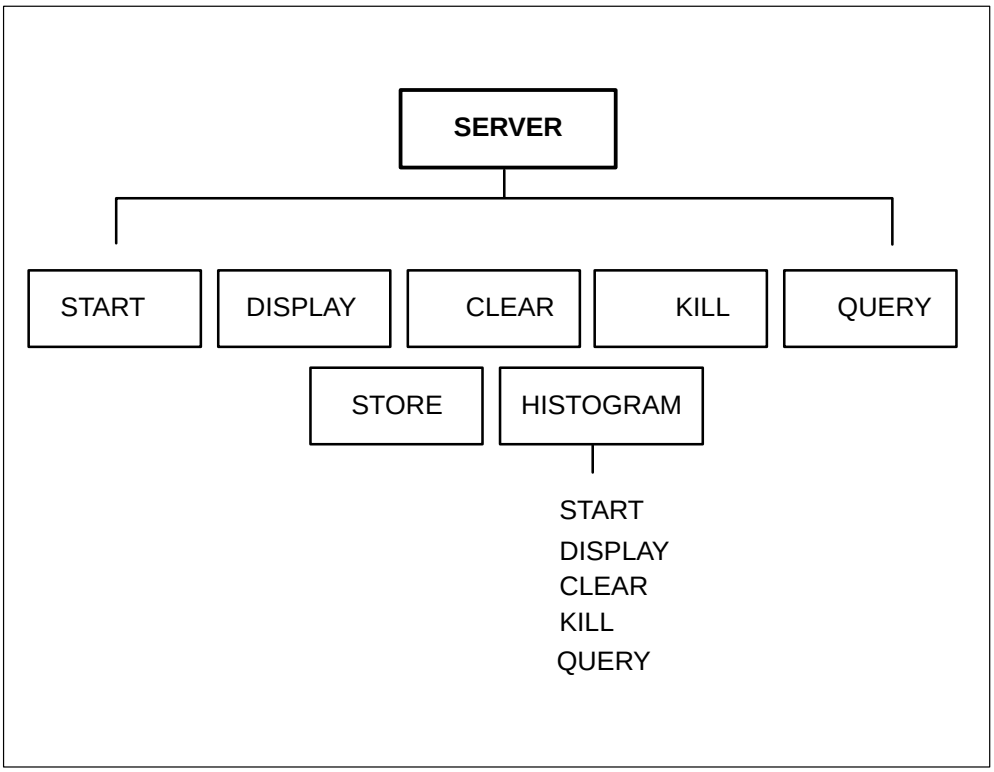
The SERVER level is accessed when the SERVER command is entered at the LTCMP level.

The SERVER level contains commands for tracing the primitives and execs sent to and from an XPM during a call.

Refer to Figure 7-3 on page 7-12 for an example of using the commands at the SERVER level.

Figure 4-42 on page 4-190 shows commands in the SERVER level.

Figure 4-42
Commands in the SERVER level



Primitives and execs

Each PM defines a set of basic operations or primitive instructions that can be processed on that PM. The operating system of the PM decodes and processes each primitive instruction in messages it receives or execs it is running.

An exec is a sequence of primitives that define specific PP instructions.

Execs are defined by software in the CC and are referred to by an identifier in the range of 1 to 254. Exec identifier 0 is reserved by the PM as the nil exec (NIL_EXEC_ID), which does nothing. Exec identifier 255 is reserved as the invalid exec, which generates a command protocol violation report.

Execs are called as an instruction sequence or are referenced indirectly in response to an event.

When the end of an exec is reached (ENDEXEC primitive), control returns to the processing level from the exec that called it.

Execs are the building blocks for maintenance and call-processing software that require processing in the PM. The PM responds to external events by calling an exec. Messages from the CC are required to initialize exec identifier locations and initiate supervisory or timing functions. Once processing has begun, CC intervention is limited only by the complexity of the execs and the processing resources available in the PM.

Following is a list of basic functions that execs help provide in call processing:

- call origination
- incoming proceed-to-send signaling
- dial tone
- digit collection
- call supervision
- outgoing-seizure signaling
- digit outpulsing
- call-duration timing
- tone cadence
- voice-path connection
- called-party disconnect
- calling-party disconnect

Following are the commands available in the SERVER level:

CLEAR command

The CLEAR command clears the exec and primitive trace buffers and the histogram buffers.

CLEAR	

DISPLAY command

The DISPLAY command displays the primitive and exec trace buffers. You are prompted to enter **EXEC** to display the exec trace buffers, **PRIM** to display the primitive trace buffers, or **ALL** to display both the exec and primitive trace buffers.

DISPLAY	

Following is an example of the output produced for an exec trace:

```
EXEC trace of :  NODENO = 3  TERMNO = 1256   EXTBYTE = 0
CB  B4  B3  A8     CE  00  00  00     00  00  00  00     00  00  00  00
00  00  00  00     00  00  00  00     00  00  00  00     00  00  00  00
00  00  00  00     00  00  00  00     00  00  00  00     00  00  00  00
00  00  00  00     00  00  00  00     00  00  00  00     00  00  00  00
00  00  00  00     00  00  00  00     00  00  00  00     00  00  00  00
00  00  00  00     00  00  00  00     00  00  00  00     00  00  00  00
00  00  00  00     00  00  00  00     00  00  00  00     00  00  00  00
00  00  00  00     00  00  00  00     00  00  00  00     00  00  00  00
00  00  00  00     00  00  00  00     00  00  00  00     00  00  00  00
00  00  00  00     00  00  00  00     00  00  00  00     00  00  00  00
00  00  00  00     00  00  00  00     00  00  00  00     00  00  00  00
00  00  00  00     00  00  00  00     00  00  00  00     00  00  00  00
00  00  00  00     00  00  00  00     00  00  00  00     00  00  00  00
00  00  00  00     00  00  00  00     00  00  00  00     00  00  00  00
00  00  00  00     00  00  00  00     00  00  00  00     00  00  00  00
00  00  00  00     00  00  00  00     00  00  00  00     00  00  00  04
(input index) trmnl 0, 1 or 2, *
```

The last byte of the exec trace in the previous example is 04. This byte is the index of the last valid entry in the exec trace buffer. This index can be used to determine whether the exec trace buffer has wrapped.

To use the index in determining the last byte written to in the buffer, number the columns of the exec trace from left to right, starting with 0 and ending with 15. Number the rows from top to bottom, starting with 0 and ending with 15. Following is an example:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	CB	B4	B3	A8	CE	00	00	00	00	00	00	00	00	00	00	00
1	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
2	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
3	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
4	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
5	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
6	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
7	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
8	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
9	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
10	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
11	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
12	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
13	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
14	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
15	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	04

In the previous example the index is 04. Therefore, the last byte written to in this buffer is in row 0, column 4.

KILL command

The KILL command disables an exec or primitive trace.

KILL	

QUERY command

The QUERY command displays which terminals are being traced.

QUERY	

START command

The START command starts an exec and primitive trace. You are prompted for the internal node number, the internal terminal number, and the extension byte of the terminals to be traced.

START	

STORE command

The STORE command displays the exec storage status. This command displays the amount of store allocated and the amount of store remaining.

STORE	

HISTOGRAM level

The HISTOGRAM command displays a histogram history of how often a specific primitive has been processed.

HISTOGRAM	

Following are the commands available in the HISTOGRAM level:

START command The START command starts the histogram trace. You are prompted for the node number, terminal number, and extension byte of the terminals to be traced.

DISPLAY command The DISPLAY command displays the histogram trace buffers.

CLEAR command The CLEAR command clears the histogram trace buffers.

KILL command The KILL command disables the histogram trace.

QUERY command The QUERY command displays information about the histogram trace.

An example of the HISTOGRAM command follows:

```
Histogram trace of :  NODENO = 3  TERMNO = 1256   EXTBYTE = 0

01 05 00 00    00 02 00 00    02 00 00 00    00 00 00 00
02 01 00 00    00 00 01 00    00 00 00 00    00 00 00 00
00 01 00 00    00 05 02 00    00 01 01 00    00 01 01 00
00 00 00 00    00 02 00 00    03 00 02 00    01 00 00 00
00 00 00 00    00 00 00 01    03 00 01 00    01 00 00 00
00 00 01 00    00 00 00 00    00 00 00 00    00 00 00 00
00 00 00 00    01 01 00 00    00 00 00 01    00 00 00 00
00 00 00 00    00 00 00 00    00 00 00 00    00 00 00 00
04 04 02 03    00 01 00 00    00 00 00 00    00 00 00 00
07 00 00 03    00 00 00 00    00 00 00 00    00 00 00 00
00 01 00 04    00 00 00 00    00 00 00 00    00 00 00 00
00 00 00 01    01 00 00 00    00 00 00 00    00 00 00 00
00 00 00 00    00 00 00 00    00 00 00 00    00 00 00 00
00 00 00 00    00 00 00 00    00 00 00 00    00 00 01 00
04 00 00 03    02 00 00 00    00 00 00 00    00 00 00 00
00 00 00 00    00 00 00 00    00 00 00 00    00 00 00 00
```

Each byte in the previous histogram is related to an opcode. Therefore, byte 0 (the first byte in the upper left corner) corresponds to opcode 00. Byte 1 (the byte to the right of byte 0) corresponds to opcode 01. Byte 2 (the byte to the right of byte 1) corresponds to opcode 02, and so on.

Therefore, in the previous example, opcode 00 was posted one time. Opcode 01 was posted five times. Opcode 02 was not posted during this call.

A list of opcodes can be found in module PPCD.

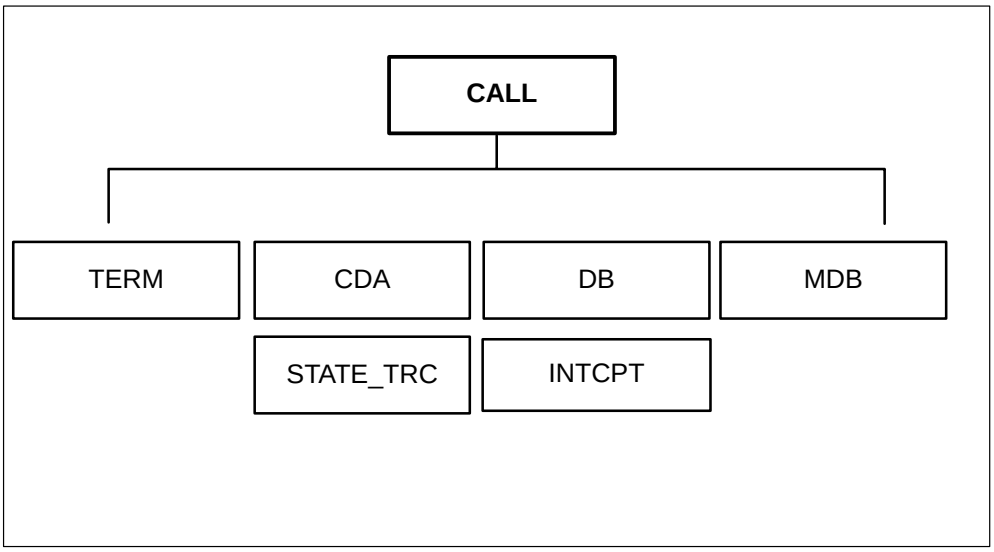
CALL level

The CALL level is accessed when the CALL command is entered at the LTCMP level.

The CALL level contains commands for examining the state of the call machine, the trunk state, and the ISUP and call data blocks. This level is used for debugging problems with new supervision.

Figure 4-43 on page 4-196 contains the commnds available at the CALL level.

Figure 4-43
Commands at the CALL level



Following are descriptions of the commands available at the CALL level:

TERM command

The TERM command displays the status of a specific terminal. You are prompted for the internal terminal number.

Information displayed by the TERM command includes the ISUP data block index, the terminal state, and the terminal type.

TERM	

CDA command

The CDA command displays the call data area. You are prompted for the data block index number.

CDA	

Note: The data block index number can be determined by issuing the TERM command. The index number is provided in the ISUP DB IDX field of the TERM display.

DB command

The DB command displays the ISUP data block. You are prompted for the data block index number.

DB	

Note: The data block index number can be determined by issuing the TERM command. The index number is provided in the ISUP DB IDX field of the TERM display.

MDB command

The MDB command displays the message data block for the last ISUP transaction.

MDB	

STATE_TRC command

The STATE_TRC command enables and disables the generation of an XPM SWERR each time the internal call machine state changes and each time a new tone is applied.

If the STATE_TRC command has never been issued, the first time the command is entered, the SWERR generation is enabled. The next time the command is entered, the SWERR generation is disabled.

This command is not recommended for use on a peripheral with more than one active trunk.

STATE_TRC	

Following is a model of the SWERR generated when the STATE_TRC command enables SWERR generation:

TASK: ISUP 'state'
DATA: aa aa bb bb cc cc dd dd xx xx
CALLED FROM: <five stacked procedure calls>

The data contained in the SWERR is defined as follows:

aa aa the previous call machine state

bb bb the new call machine state

cc cc the index of the call data area and the ISUP data block

dd dd where 00 00 indicates ongoing, 11 11 indicates incoming, FF FF indicates other

INTCPT command

CAUTION

The INTCPT command should *not* be used on a live switch. Users of this command should have a thorough knowledge of XPM and ISUP messaging. Incorrect use of this command results in service interruption.

The INTCPT command enables interception of ISUP messages. You are prompted for an internal terminal number and for the byte msg (message number) to intercept. If the command is entered a second time, the interception of messages is disabled.

INTCPT	

ISOM level

The ISOM level is accessed when the ISOM command is issued at the LTCMP level.

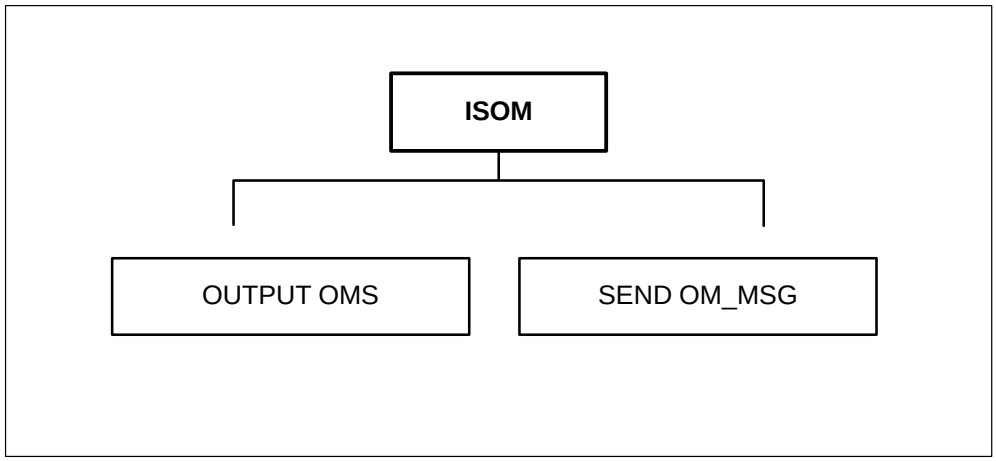
The ISOM level is used to test the OM group ISUPUSAG. At any time the current OM message counts can be sent up to the CC from the DTC or MSB7 and added to the current, active OM counts in the CC. Once sent, the counts are zeroed out.

By using the OUTPUT OMS command, the current OM counts can be displayed.

The OM message counts are sent to the CC automatically every 15 minutes, as well as when any count is about to overflow.

Figure 4-44 on page 4-199 contains the commands available at the ISOM level.

Figure 4-44
Commands available at the ISOM level



Following are descriptions of the commands available at the ISOM level:

OUTPUT OMS command

The OUTPUT OMS command displays the OM counts.

OUTPUT OMS	

SEND OM_MSG command

The SEND OM_MSG command sends the OM message counts to the CC.

SEND OM_MSG	

RCVRMON level

The RCVRMON level is accessed when the RCVRMON command is entered from the LTCMP level.

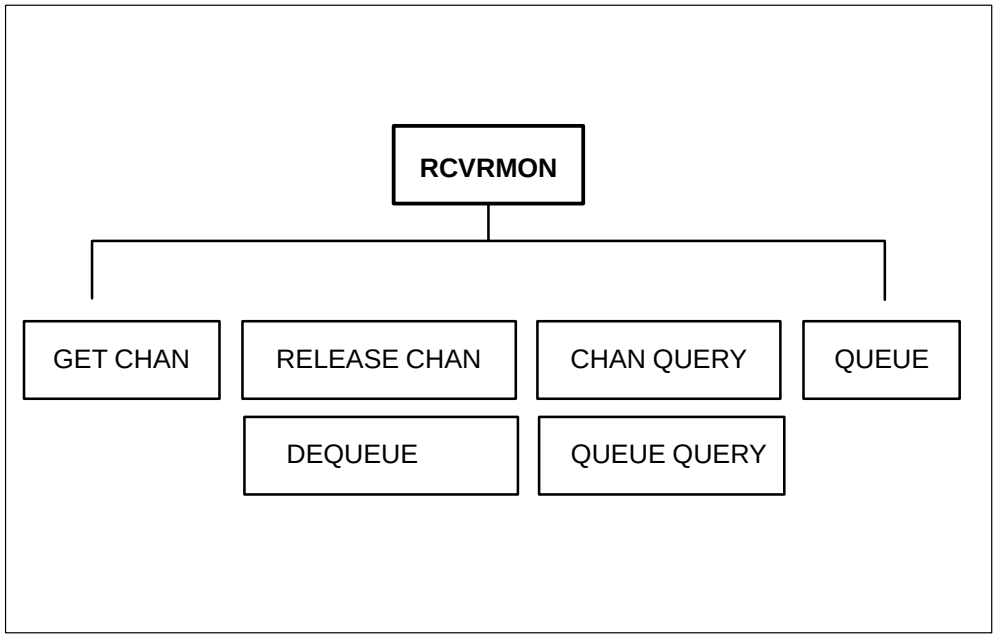
The RCVRMON level allows you to monitor the RCVRUTIL system, which manages the universal tone receiver (UTR) channels.

The number of channels for each UTR card may vary depending on the application. For domestic application, each UTR card uses 30 channels.

UTR channels are requested through the RCVRUTIL system. If all UTR channels are in use and a UTR channel request cannot be granted, the request is queued. Once a UTR channel is available, the queued request is processed. A maximum of 64 requests can be queued, regardless of the number of UTR cards.

Figure 4-45 on page 4-200 contains the commands available at the RCVRMON level.

Figure 4-45
Commands in the RCVRMON level



Following are the commands available in the RCVRMON level:

GET CHAN command

The GET CHAN command allows you to request a specific number of UTR channels. This command allocates the requested number of channels or the minimum number of free UTR channels.

This command allows you to decrease the number of UTR channels available for call processing.

You are prompted for the card number and the number of channels to be obtained.

GET CHAN	

RELEASE CHAN command

The RELEASE CHAN command releases one or more of the UTR channels allocated by the GET CHAN command.

You are prompted for the card number and the number of channels to free.

RELEASE CHAN	

Note: Channels allocated by any means other than the GET CHAN command cannot be deallocated with the RELEASE CHAN command.

CHAN QUERY command

The CHAN QUERY command allows you to display the status of the UTR channels.

CHAN QUERY	

The following information is displayed for each UTR card:

- number of UTR channels free
- number of UTR channels used by call processing
- number of channels allocated with the GET CHAN command

QUEUE command

The QUEUE command requests a specific number of items that are on the RCVRUTIL queue. This command allocates the number of requested queue entries or the minimum number of queue entries.

This command allows you to decrease the number of queue entries available for call processing.

You are prompted for the number of receiver (RCVR) elements to queue.

QUEUE	

DEQUEUE command

The DEQUEUE command releases one or more queue items allocated with the QUEUE command.

You are prompted for the number of receiver (RCVR) elements to dequeue.

DEQUEUE	

Note: Queued items allocated by any means other than the QUEUE command cannot be deallocated with the DEQUEUE command.

QUEUE QUERY command

The QUEUE QUERY command displays the status of the RCVRUTIL queue.

QUEUE QUERY	

The following information is displayed:

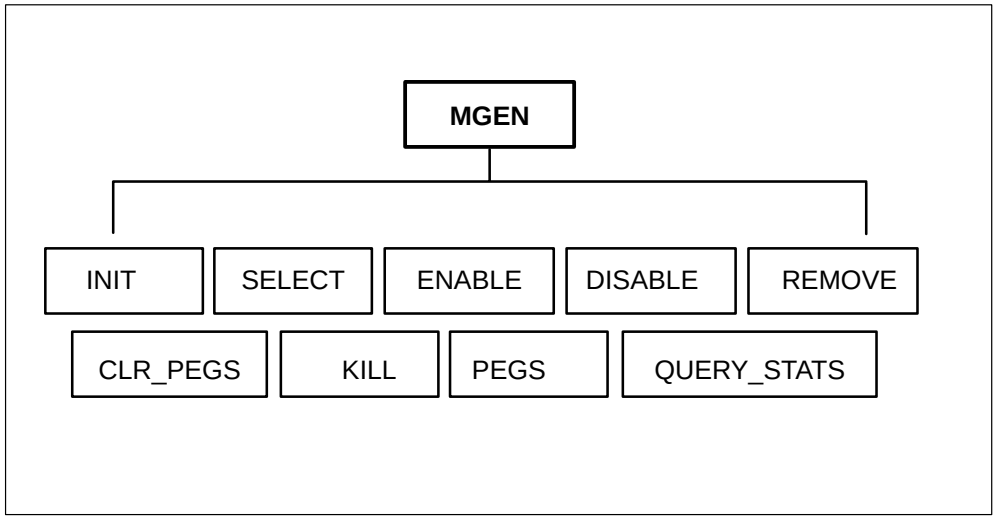
- queue size
- number of free queue entries
- number of queue entries used by call processing
- number of queue entries allocated with the QUEUE command.

MGEN level

The message generator (MGEN) level is accessed when the MGEN command is processed from the LTCMP level.

The MGEN level allows you to test and debug the 6X69 messaging card as well as the 6X43 messaging card. Figure 4-46 on page 4-203 contains the commands available in the MGEN level.

Figure 4-46
Commands in the MGEN level



Following are descriptions of the commands available in the MGEN level:

INIT command

The INIT command initializes the MGEN data structures and obtains storage for the data to be gathered by the ENABLE command. Refer to “ENABLE” command on page 4-205.

INIT	

SELECT command

The SELECT command constructs a message for the destination node. The INIT command must be issued before the SELECT command can be

processed. If the ENABLE command has been processed, the DISABLE command must be entered before the SELECT command can be issued.

SELECT	extnode	chnls	schnl	length	loadbal	rate

Where:

extnode is the external node number of the destination node

chnls is the number of channels or messaging links used for sending messages to the destination node

schnl is the start channel number. This channel is the first channel that routes the message. If more than one channel exists, the other channels are numbered consecutively from the start channel.

length is the length of the IPC buffer, including the routing header. The length must be greater than the size of the data body used by MGEN.

loadbal are two fields indicating the ratio of messages sent to messages returned. This parameter allows you to unbalance the links; for instance, the ratio can be 1 to 1 (balanced), many to 1, or 1 to many.

rate is the number of messages per second in the outgoing direction

Example:

```
>s

For selection number : 0
Enter external node number of destination node
MP:MGen>

>20

How many channels will that be?
MP:MGen>

>2

Starting with which channel number?
MP:MGen>
```

```
>0

Enter message length in bytes (include protocol header)
MP:MGen>

>30

Select load directional balance factor
MP:MGen>

>4

One to many or many to one?
MP:MGen>

>0

Input desired outgoing message rate (msg/sec)
MP:MGen>

>10

Sel Node #chnl startchnl msg rate msglen balfctr balancedir prcl
0   20   2     0        10     30     4     one to many DMSX
```

ENABLE command

The ENABLE command begins generating traffic messages on all selected channels. The INIT and SELECT commands must be processed before the ENABLE command can be issued.

ENABLE	

DISABLE command

The DISABLE command stops the generation of traffic messages.

DISABLE	

REMOVE command

The REMOVE command allows you to selectively remove links from the link table. This command can be processed only after messaging has been disabled with the DISABLE command.

Removing a link selection causes a change in the numbering of the remaining link selections.

REMOVE	**number** **ALL**

Where:

number is the number of the link to be removed

ALL indicates all links will be removed

CLR_PEGS command

The CLR_PEGS command clears the peg counters for all the link selections. The INIT command must be processed before the CLR_PEGS command can be processed.

CLR_PEGS	

KILL command

The KILL command disables and reinitializes the MGEN tool. The data store is returned to the storage manager.

KILL	

PEGS command

The PEGS command prints the peg counts and out-of-sequence labels for any one link selection.

PEGS	**number**

Where:

number is the link selection number

QUERY_STATS command

The QUERY_STATS command displays the user-input data that describes the link selection.

QUERY_STATS	**number** **ALL**

Where:

number is the number of the link to be displayed

ALL indicates all links will be displayed

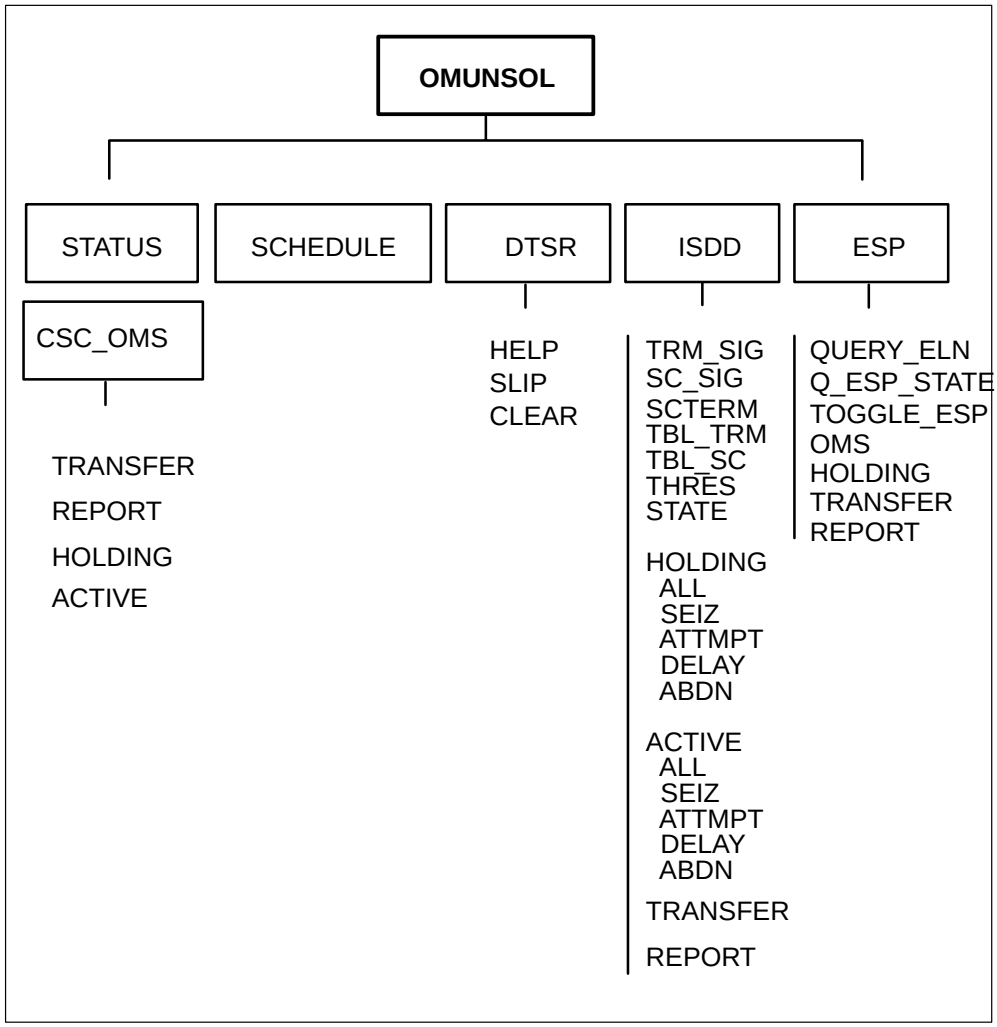
OMUNSOL level

The OMUNSOL level is accessed when the OMUNSOL command is entered from the LTCMP level.

The OMUNSOL level displays information associated with unsolicited OMs.

Figure 4-47 on page 4-208 shows the commands in the OMUNSOL level.

Figure 4-47
Commands in the OMUNSOL level



Following are descriptions of the commands available at the OMUNSOL level:

STATUS command

The STATUS command displays the OM state for each OM, such as not active, active, not loaded, or invalid.

STATUS	

Example:

```
STatus,SChedule.
MP:OMUnsol>

>>>status
Om_id:    0 , not loaded
Om_id:    1 , not loaded
Om_id:    2 , not loaded
Om_id:    3 , not loaded
Om_id:    4 , not loaded
Om_id:    5 , not loaded
Om_id:    6 , not loaded
Om_id:    7 , not loaded
Om_id:    8 , not loaded
Om_id:    9 , not loaded
Om_id:   10 , not loaded
Om_id:   11 , not loaded
Om_id:   12 , not loaded
Om_id:   13 , not loaded
Om_id:   14 , not loaded
Om_id:   15 , not loaded

MP:OMUnsol>
```

SCHEDULE command

The SCHEDULE command displays the hour schedule for each active OM.

SCHEDULE	

DTSR level

The DTSR command provides access to commands that display how long the line concentrating device (LCD) takes to respond to a DTSR request for statistics.

An XPM calculates the time taken between sending a statistics request and receiving the data. This information is inserted into each DTSR statistics report. If the calculated time is greater than five seconds, the data is included in the next DTSR report. This event is called a slip.

Following are descriptions of the commands available at the DTSR level:

SLIP command

The SLIP command displays the number of slips for an internal node number of an LCD. The number of slips is recorded starting with initialization or starting after the CLEAR command has been processed.

SLIP	int_node_num

Where:

int_node_num is the internal node number

CLEAR command

The CLEAR command sets the slip counter to zero.

CLEAR	

HELP command

The HELP command displays the command syntax of the CLEAR and SLIP commands.

HELP	

Example:

```
MP:OMUnsol>
STatus,SChedule,Esp,Dtsr,Isdd
MP:OMUnsol>

>sc

MP:OMUnsol>

>dt

Slip,Clear,Help.
MP:Dtsr>

>h
Slip <internal node number>
Clear

MP:Dtsr>
```

ISDD level

The ISDD command will access the ISDD (incoming Start to DIAL Delay) level of OMUNSOL. ISDD commands display ISDD OM information. The two password commands of ISDD alter ISDD OM performance.

ISDD	

Example:

```
LTCMP>
OMU
STatus,SChedule,Esp,Dtsr,Isdd
MP:OMUnsol>
Isdd
Trm_sig,Sc_sig,SCTerm,TBl_trm,TBL_Sc,THres,STate,Holding,
Active,TRAnsfer,Report
MP:Isdd>
```

Following are command descriptions at the ISDD sublevel of OMUNSOL:

TRM_SIG command

The TRM_SIG command displays the XPMs view of a particular internal terminal's trunk signaling type and start-to-dial type. Signal types are MF (multi-frequency), DP (dial pulse), DT (dial tone) or nil. Start-to-dial types are WK (wink), DD (dial delay), IM (immediate) or nil.

TRM_SIG	internal_tid

Where:

internal_tid is the internal terminal identifier number

Example:

```
MP:Isdd>
trm_sig 1
TID : 1
start  type : NIL
signal type : NIL
```

SC_SIG command

The SC_SIG command displays the signaling type and the start-to-dial type, as stored against the scan terminal number of the internal TID. The scan terminal is the trunk's C-side port * 20, plus C-side channel number. The scan terminal number has meaning only in the XPM.

SC_SIG	internal_tid

Where:

internal_tid is the internal terminal identifier number.

Example:

```
MP:Isdd>
sc_sig
TID: 6
start type = NIL
signal type : NIL
```

SCTERM command

The SCTERM command gives the scan terminal number of a trunk for an input internal tid.

SCTERM	internal_tid

Where:

internal_tid is the internal terminal number.

Example:

```
MP:Isdd>
Scterm 1
Scan Terminal : 1
```

TBL_TRM command

The TBL_TRM command dumps all entries of the TRM_SIG table.

TBL_TRM	

Example:

```
MP:Isdd>
tbl_trm
TID         START   SIGNAL
===         ==============
0           NIL     NIL
1           IM      MF
2           DD      DP
°           °       °
649         NIL     NIL
650         NIL     NIL
```

TBL_SC command

The TBL_SC command dumps all entries of the SC_SIG table.

TBL_SC	

Example:

```
MP:Isdd>
tbl_sc
SCTID       START   SIGNAL
=====       ==============
0           NIL     NIL
1           DD      DP
2           IM      MF
°           °       °
649         NIL     NIL
650         NIL     NIL
```

THRES command

The THRES command displays the ISDD OM threshold value in units of 10 ms. The ISDD OM threshold is used by ISDD to determine when to peg a delayed start-to-dial.

THRES	

Example:

```
MP:Isdd>
thres
ISDD threshold : 32767
```

STATE command

The STATE command dumps the internal ISDD state of an internal TID. The ISDD states are ISDD_IDLE, ORIG_TMR, PRE_TRAILING and ISDD_ACTIVE. The descriptions of the ISDD states follow:

Isdd_idle Trunks in the idle state are not currently active on a call.

Orig_Tmr An origination filter timer is running for the trunk.

Pre_Trailing Trunks that are non-IM start-type reside in this state when the filtering on a seizure is complete. This state does not apply to IM start-types.

Isdd_Active Trunks in the ISDD_ACTIVE state have progressed past the start-to-dial phase of the call.

STATE	internal_tid

Where:

internal_tid is the internal terminal identifier number.

Example:

```
MP:Isdd>
state 1
ISDD state =ORIG_TMR
```

HOLDING command

The HOLDING command is used to access the sublevel of ISDD that displays the holding registers of ISDD OM. The following five commands are in the HOLDING sublevel of ISDD:

HOLDING	

ALL command

The ALL command used in the sublevel HOLDING of ISDD displays all MP and SP holding peg registers.

ALL	

Example:

```
MP:Holding>
all
Peg Data For Holding MP seizures
--------------------------------
              WK        IM        DD              OTHER
MF            0         0         0               0
DP            0         0         0               0
DT            0         0         0               0
OT            0         0         0               0
Peg Data For Holding SP seizures
--------------------------------
              WK        IM        DD              OTHER
MF            0         0         0               0
DP            0         0         0               0
DT            0         0         0               0
OT            0         0         0               0
Peg Data For Holding MP attempts
--------------------------------
              WK        IM        DD              OTHER
MF            0         0         0               0
DP            0         0         0               0
DT            0         0         0               0
OT            0         0         0               0
Peg Data For Holding MP delays
------------------------------
              WK        IM        DD              OTHER
MF            0         0         0               0
DP            0         0         0               0
DT            0         0         0               0
OT            0         0         0               0
Peg Data For Holding SP delays
------------------------------
              WK        IM        DD              OTHER
MF            0         0         0               0
DP            0         0         0               0
DT            0         0         0               0
OT            0         0         0               0
Peg Data For Holding MP abandons
--------------------------------
              WK        IM        DD              OTHER
MF            0         0         0               0
DP            0         0         0               0
DT            0         0         0               0
```

```
OT          0          0          0          0
Peg Data For Holding SP abandons
---------------------------------
            WK         IM         DD         OTHER
MF          0          0          0          0
DP          0          0          0          0
DT          0          0          0          0
OT          0          0          0          0
```

SEIZ command

The SEIZ command used in the sublevel HOLDING of ISDD displays all MP and SP holding seizure registers.

SEIZ	

Example:

```
MP:Holding>
seiz
Peg Data For Holding MP seizures
---------------------------------
            WK         IM         DD         OTHER
MF          0          0          0          0
DP          0          0          0          0
DT          0          0          0          0
OT          0          0          0          0
Peg DAta For Holding SP seizures
---------------------------------
        WK       IM       DD       OTHER
MF          0          0          0          0
DP          0          0          0          0
DT          0          0          0          0
OT          0          0          0          0
```

ATTMPT command

The command ATTMPT used in the sublevel HOLDING of ISDD displays all MP and SP holding attempt registers.

ATTMPT	

Example:

```
MP:Holding>
attmpt
Peg Data For Holding MP attemps
-------------------------------
          WK        IM        DD        OTHER
MF        0         0         0         0
DP        0         0         0         0
DT        0         0         0         0
OT        0         0         0         0
```

DELAY command

The command DELAY used in the sublevel HOLDING of ISDD displays all MP and SP holding delay registers.

DELAY	

Example:

```
MP:Holding>
delay
Peg Data For Holding MP delays
-------------------------------
          WK        IM        DD        OTHER
MF        0         0         0         0
DP        0         0         0         0
DT        0         0         0         0
OT        0         0         0         0
Peg Data For Holding SP delays
-------------------------------
          WK        IM        DD        OTHER
MF        0         0         0         0
DP        0         0         0         0
DT        0         0         0         0
OT        0         0         0         0
```

ABDN command

The command ABDN used in the sublevel HOLDING of ISDD displays all MP and SP holding abandon registers.

ABDN	

Example:

```
MP:Holding>
abdn
Peg Data For Holding MP abandons
--------------------------------
              WK        IM        DD             OTHER
MF            0         0         0              0
DP            0         0         0              0
DT            0         0         0              0
OT            0         0         0              0
Peg Data For Holding SP abandons
--------------------------------
      WK      IM      DD      OTHER
MF            0         0         0              0
DP            0         0         0              0
DT            0         0         0              0
OT            0         0         0              0
```

ACTIVE command

The ACTIVE command is used to access the sublevel of ISDD that displays active registers of the ISDD OM. The following five commands (ALL, SEIZ, ATTMPT, DELAY, and ABDN) are in the ACTIVE sublevel of ISDD.

ALL command

The ALL command used in the sublevel ACTIVE of ISDD displays all MP and SP active peg registers.

ALL	

Example:

```
MP:Active>
all
Peg Data For Active MP seizures
--------------------------------
              WK        IM        DD             OTHER
MF            0         0         0              0
DP            0         0         0              0
DT            0         0         0              0
OT            0         0         0              0
Peg Data For Active SP seizures
--------------------------------
              WK        IM        DD             OTHER
MF            0         0         0              0
DP            0         0         0              0
```

```
DT          0          0          0          0
OT          0          0          0          0
Peg Data For Active MP attempts
-------------------------------
            WK         IM         DD         OTHER
MF          0          0          0          0
DP          0          0          0          0
DT          0          0          0          0
OT          0          0          0          0
Peg Data For Active MP delays
-----------------------------
            WK         IM         DD         OTHER
MF          0          0          0          0
DP          0          0          0          0
DT          0          0          0          0
OT          0          0          0          0
Peg Data For Active SP delays
-----------------------------
            WK         IM         DD         OTHER
MF          0          0          0          0
DP          0          0          0          0
DT          0          0          0          0
OT          0          0          0          0
Peg Data For Active MP abandons
-------------------------------
            WK         IM         DD         OTHER
MF          0          0          0          0
DP          0          0          0          0
DT          0          0          0          0
OT          0          0          0          0
Peg Data For Active SP abandons
-------------------------------
            WK         IM         DD         OTHER
MF          0          0          0          0
DP          0          0          0          0
DT          0          0          0          0
OT          0          0          0          0
```

SEIZ command

The SEIZ command in the sublevel ACTIVE of ISDD displays all MP and SP active seizure registers.

SEIZ	

Example:

```
MP:Active>
seiz
Peg Data For Active MP seizures
--------------------------------
          WK        IM        DD        OTHER
MF        0         0         0         0
DP        0         0         0         0
DT        0         0         0         0
OT        0         0         0         0
Peg Data For Active SP seizures
--------------------------------
          WK        IM        DD        OTHER
MF        0         0         0         0
DP        0         0         0         0
DT        0         0         0         0
OT        0         0         0         0
```

ATTMPT command

The ATTMPT command in the sublevel ACTIVE of ISDD displays all MP and SP active attempt registers.

ATTMPT	

Example:

```
MP:Active>
attmpt
Peg Data For Active MP attempts
--------------------------------
          WK        IM        DD        OTHER
MF        0         0         0         0
DP        0         0         0         0
DT        0         0         0         0
OT        0         0         0         0
```

DELAY command

The DELAY command in the sublevel ACTIVE of ISDD displays all MP and SP active delay registers.

DELAY	

Example:

```
MP:Active>
delay
Peg Data For Active MP delays
--------------------------------
            WK        IM        DD          OTHER
MF          0         0         0           0
DP          0         0         0           0
DT          0         0         0           0
OT          0         0         0           0
Peg Data For Active SP delays
--------------------------------
            WK        IM        DD          OTHER
MF          0         0         0           0
DP          0         0         0           0
DT          0         0         0           0
OT          0         0         0           0
```

ABDN command

The ABDN command in the sublevel ACTIVE of ISDD displays all MP and SP active abandon registers.

ABDN	

Example:

```
MP:Active>
abdn
Peg Date For Active MP abandons
--------------------------------
            WK        IM        DD          OTHER
MF          0         0         0           0
DP          0         0         0           0
DT          0         0         0           0
OT          0         0         0           0
Peg Data For Active SP abandons
--------------------------------
            WK        IM        DD          OTHER
MF          0         0         0           0
DP          0         0         0           0
DT          0         0         0           0
OT          0         0         0           0
```

TRANSFER command (PASSWORD PROTECTED)

The TRANSFER command of ISDD cannot be used unless the correct password is entered following the input of the TRANSFER command. The

TRANSFER command transfers ISDD data from the active registers to the holding registers of the XPM. This command can cause a loss of data being sent to the CC. When the data transfer occurs during the OM interval, another transfer takes place, and the data currently in the holding registers are not sent to the CC.

TRANSFER	

Example:

```
MP:Isdd>
transfer
Do Transfer
MP:TRAnsfer>
                    (Password is Entered Here)
MP:Isdd>
```

REPORT command (PASSWORD PROTECTED)

The REPORT command of ISDD can not be utilized unless the correct password is entered following the input of the REPORT command. The REPORT and of ISDD constructs an ISDD OM report from data in XPM holding registers and sends the report to the CC.

REPORT	

Example:

```
MP:Isdd>
report
Do Report
MP:Report>
                    (Password is Entered Here)
MP:Isdd>
```

ESP level

The essential service protection sublevel of OMUNSOL displays and manipulates ESP data. The essential service protection sublevel is accessed when the ESP command is entered from the OMUNSOL level of LTCMP. The commands that manipulate ESP are password protected.

Example:

```
MP:OMUnsol>
ESP
MP:ESP>
```

Following are descriptions of the commands and sublevels in the ESP sublevel of OMUNSOL:

QUERY_ELN command

The QUERY_ELN command displays the state of the ELN option for an individual line. The ELN state can be true or false.

QUERY_ELN	tid

Where:

tid is the terminal ID number. The valid range is 1 through 7055.

Example:

```
MP:ESP>
Q 22
ELN state for line 22 is : FALSE
```

QESP_STATE command

The QESP_STATE command displays the global state of ESP in the XPM. The global state of ESP is active, inactive or unknown.

QESP_STATE	

Example:

```
MP:ESP>
QE
Global ESP state is: ACTIVE
```

TOGGLE_ESP command (PASSWORD PROTECTED)

The TOGGLE_ESP command changes the global state of ESP in the XPM. The change is from active to inactive or inactive to active. The TOGGLE_ESP command cannot be used unless the correct password is entered following the input of the TOGGLE_ESP command.

TOGGLE_ESP	

Example:

MP:ESP>
T
Change global ESP state
MP:Toggle_esp>
(Password is Entered Here)
<ESP state : ACTIVEºINACTIVE>
MP:Toggle_esp>
active

OMS level

The OMS command accesses the operational measurement level of ESP.

OMS	

Example:

MP:ESP>
OMS
Active,Holding,Transfer,Report.

Following are descriptions of the commands in the OMS sublevel of OMUNSOL:

ACTIVE command

The ACTIVE command of the OMS sublevel displays the OM active data registers.

ACTIVE	

Example:

MP:OMS>
active
Peg Data For ELN Active Register:

NODE #	Origs	Steals	SP Deny	MP Deny	CC Origs
======	-----	------	-------	-------	--------
19	0	0	0	0	0
18	0	0	0	0	0

MP:OMS>

HOLDING command

The HOLDING command of the OMS sublevel displays the OM holding data registers.

HOLDING	

Example:

```
MP:OMS>
holding
Peg Data For ELN Holding Register:
_____________________________________
```

NODE #	Origs	Steals	SP Deny	MP Deny	CC Origs
======	-----	------	-------	-------	--------
19	0	0	0	0	0
18	0	0	0	0	0

```
MP:OMS>
```

TRANSFER command (PASSWORD PROTECTED)

The TRANSFER command of the OMS sublevel initializes an ELN OM transfer.

TRANSFER	

Example:

```
MP:OMS>
tra
Do Transfer
MP:Transfer>
                    (Password is Entered Here)
Transfer complete
```

REPORT command (PASSWORD PROTECTED)

The REPORT command of the OMS sublevel initializes an ELN OM report.

REPORT	

Example:

```
MP:OMS>
report
Do Report
MP:Report>
                    (Password is Entered Here)
Report requested
```

CSC_OMS level

You can display and change information in the cell site controller operational measurements information (CSC_OMS) level in the following classes:

- call processing (CP)
- maintenance (MTC)
- rssi measurement (LCR)
- handoff (HO).

To access the CSC_OMS level, access the OMUNSOL level while in the MP by entering OMU, pressing return, and then entering C.

Csc_oms	

Transfer command (PASSWORD PROTECTED)

The transfer command transfers OMs in the active register into the holding register.

Transfer	

Report command (PASSWORD PROTECTED)

The report command sends OMS in the holding register to the MTX.

> ***Note:*** By using the transfer and report selections, you can send OMs to the MTX at any time without having to wait for the MTX to request the OMs from the CSC.

Example:

```
MP:Csc_oms>
Transfer,Report,Holding,Active

MP:Csc_oms> T
Enter the Password for TRANSFER:

MP:Transfer>
TRANSFER completed. (Act --> Hold)
```

Report	

Holding command

The holding command views or changes the holding OMs for all partitions of the cell (sectors, tiers, or sector-tiers).

Example:

```
MP:Csc_oms>
Transfer,Report,Holding,Active

MP:Csc_oms> R
Enter the Password for REPORT

MP:Report>
REPORT completed. (to the CC)

MP:Csc_oms
```

Holding	**All_holding**	**Clear** **Increment_all** **Peg_one_om** ****
	Om_holding	**Clear** **Increment_all** **Peg_one_om** ****

where

All_holding You can specify all sectors with this command.

OM_holding You can specify a partition with this command.

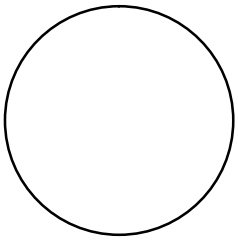
Clear resets all OM counters to zero.

Note: The following example and all examples given for Csc_oms are for an OMNI cell. You will be prompted to select the desired sector and tier number for the desired partition within the valid range. See the example on page 4-229 (ACT_NUM_Sector - 1: 0). The range will vary depending on the type of cell. Following, on page 4-228 are some examples of cells and the appropriate valid ranges.

Figure 4-48
Sectored and tiered cells

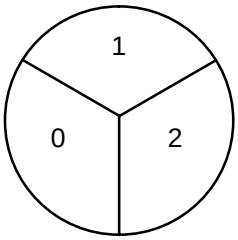
OMNI cell

(ACT_NUM_sector) – 1 = 0
Valid range = 0 → 0
(ACT_NUM_TIER) – 1 = 0
Valid range = 0 → 0



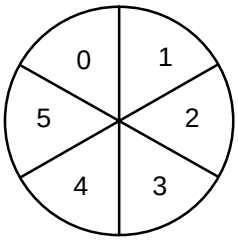
120° sectored cell

(ACT_NUM_sector) – 1 = 2
Valid range = 0 → 2
(ACT_NUM_TIER) – 1 = 0
Valid range = 0 → 0



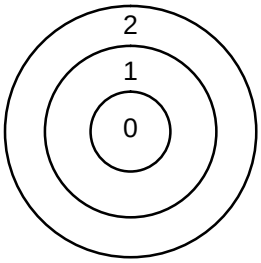
60° sectored cell

(ACT_NUM_sector) – 1 = 5
Valid range = 0 → 5
(ACT_NUM_TIER) – 1 = 0
Valid range = 0 → 0



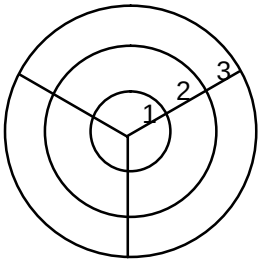
Tiered cell

(ACT_NUM_sector) – 1 = 0
Valid range = 0 → 0
(ACT_NUM_TIER) – 1 = 2
Valid range = 0 → 2



Sector_tiered cell

(ACT_NUM_sector) – 1 = 2 } identical to 120° sectored cell
Valid range = 0 → 2
(ACT_NUM_TIER) – 1 = 2 } identical to tiered cell
Valid range = 0 → 2



Example:

```
MP>
TIme,TAsk,Load,Xprompt,Debug,Swerr,Ipc,Queues,Uspace,Patches,Msgtr,Chnls,
MTc, OM, Rcu,CP,Ho,DIagnose,Audit,MAtediag,OMUnsol.
MP> OMU
STatus,SChedule,Csc_oms.

MP:OMUnsol>  C
Transfer,Report,Holding,Active.

MP:Csc_oms>  H
All_holding,Om_holding.

MP:Holding>  O

OM Data For Holding class
---------------------
ACT_NUM_Sector - 1  :       0
Enter Sector number (0 - (act_num_sector-1)  ):
MP:Om_holding>  0

Active_Num_Tier - 1  :
Enter Tier number (0 - (Active_Num_Tier-1) ):
MP:Om_holding> 0

OM Data For Holding class
---------------------
Call_proc, Handoff, Maint, Lcr, or *
MP:Om_holding> C

          Call-Processing Operational Measurements
            ---------------------------------
Page requests              :     100   Page responses          :     101
Page requests (NBP)        :     108   Page responses (NBP)    :     115
Originations               :     102   Intercepts              :     103
Reorders                   :     104   Completed originations  :     105
Load balancing reorders    :     106   Sat timeouts            :     112
Invalid SAT                :     113   Mobile registrations    :     114
Mobile terms completed     :     117   RSSI sent to MTX        :     107
MSGs CSC recvd. over CCH:

For Call-processing OMs: (Clear, Increment_all, Peg_one_om, or *)
MP:Om_holding> C

For Call-processing OMs: (Clear, Increment_all, Peg_one_om, or *)
MP:Om_holding> *
```

```
Continue_CP, or "*" for END/HandOff
MP:Om_holding> *

     Call-Processing Operational Measurements
       ----------------------------------
Page requests              :    0    Page responses          :    0
Page requests (NBP)        :    0    Page responses (NBP)    :    0
Originations               :    0    Intercepts              :    0
Reorders                   :    0    Completed originations  :    0
Load balancing reorders    :    0    Sat timeouts            :    0
Invalid SAT                :    0    Mobile registrations    :    0
Mobile terms completed     :    0    RSSI sent to MTX        :    0
MSGs CSC recvd. over CCH:

For Call-processing OMs: (Clear, Increment_all, Peg_one_om, or *)
MP:Om_holding> *

OM Data For Holding class
---------------------
Call_proc, Handoff, Maint, Lcr, or *
MP:Om_holding> *

MP:Holding>
```

Increment_all

You can increment all OMs with this command. You will be prompted for a base number that has a range from 0 to 255. The base number is added to an index which represents the OM. The base number is added to an OM index so the OM counts are differentiated for debugging activity.

Example:

```
MP:All_holding> I
Enter Base amount (0-255):

MP:All_holding> 100

For Call-processing OMs: (Clear, Increment_all, Peg_one_om, or *)
MP:Om_holding> *

Continue_CP, or "*" for END/HandOff

MP:Om_holding> C

          Call-Processing Operational Measurements
            ---------------------------------
Page requests              :   100   Page responses          :   101
Page requests (NBP)        :   108   Page responses (NBP)    :   115
Originations               :   102   Intercepts              :   103
Reorders                   :   104   Completed originations  :   105
Load balancing reorders    :   106   Sat timeouts            :   112
Invalid SAT                :   113   Mobile registrations    :   114
Mobile terms completed     :   117   RSSI sent to MTX        :   107
MSGs CSC recvd. over CCH:    109

For Call-processing OMs: (Clear, Increment_all, Peg_one_om, or *)
MP:Om_holding> *
```

Peg_one_om This command is the same as Increment_all except that only one OM is incremented by the base number.

> ***Note:*** The peg_one_om command is similiar in the holding and active levels. See the example of the peg_one_om command in the active level on page 4-232.

***** The * returns you to the previous level.

Active command

The active command views or changes the active OMs for all partitions of the cell (sectors, tiers, or sector-tier).

Active	**All_active**	**Clear** **Increment_all** **Peg_one_om** ****
	Om_active	**Clear** **Increment_all** **Peg_one_om** ****

where

All_active You can specify all sectors with this command.

The active command views or changes the holding OMs for all sectors or a partition.

OM_active You can specify a partition with this command.

Clear resets all OM counters to zero.

> ***Note:*** The clear command is similiar in the holding and active levels. See the example of the clear command in the holding level on page 4-227 .

Increment_all

You can increment all OMs with this command. You will be prompted for a base number that has a range from 0 to 255. The base number is added to an index which represents the OM. The base number is added to an OM index so the OM counts are differentiated for debugging activity.

> ***Note:*** The increment_all command is similiar in the holding and active levels. See the example of the increment_all command in the holding level on page 4-231.

Peg_one_om This command is the same as Increment_all except that only one OM is incremented by the base number.

Example:

```
MP> OMU
STatus,SChedule,Csc_oms

MP:OMUnsol> C
Transfer,Report,Holding,Active

MP:Csc_oms> A
All_active,Om_active
```

```
MP:Active> O

OM Data For Active class
________________________

ACT_NUM_Sector - 1 :      2
Enter Sector number (0 - (act_num_sector-1) ):
MP:Om_active> 0
Active_Num_Tier - 1 :        1
Enter Tier number (0 - (Active_Num_Tier-1) ):
MP:Om_active> 0

OM Data For Active class
________________________

Call_proc, Handoff, Maint, Lcr, or *
MP:Om_active> H

    Handoff Operational Measurements
    ________________________________

Handoff failures         :  0     Handoff requests       :   0
Handoff completions      :  0     Handoff Retry          :   0
Directed retry           :  0     Directed handoff       :   0
Normal status            :  0     Directed retry status  :   0
Directed handoff status  :  0     Dir. ho/retry status   :   0

For Handoff OMs: (Clear, Increment_all, Peg_one_om, or *)
MP:Om_active> P

Max_HO_len:  12
Enter OM index (0 - Max_HO_len):
MP:Om_active> 3

Enter Increment amount (0-255):
MP:Om_active> 100

Continue(next OM field),  or *
MP:Om_active> *

Continue_HO, or “*” for END/Maintenance
MP:Om_active> C

    Handoff Operational Measurements
    ________________________________

Handoff failures         :  0     Handoff requests       :   0
Handoff completions      :  0     Handoff Retry          :   0
Directed retry           :  100   Directed handoff       :   0
Normal status            :  0     Directed retry status  :   0
```

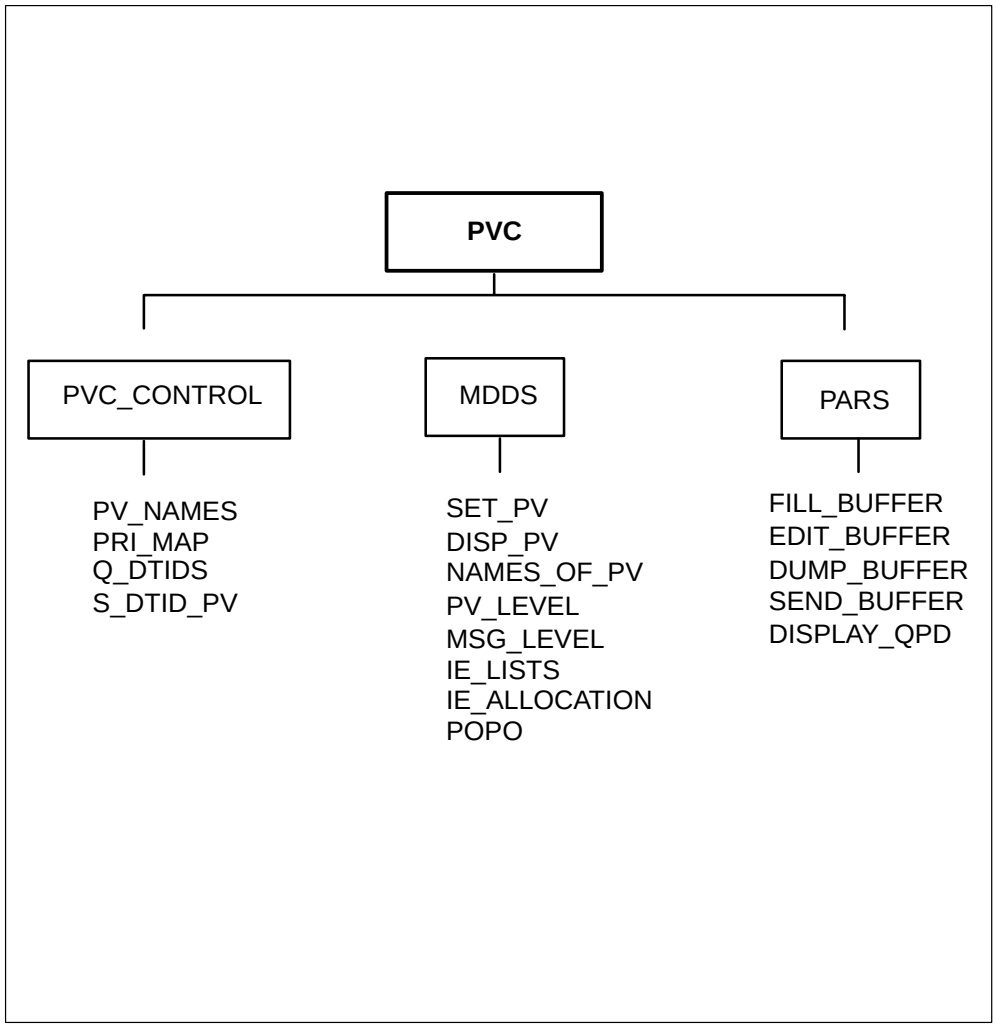
```
Directed handoff status      :     0      Dir. ho/retry status          :     0

For Handoff OMs: (Clear, Increment_all, Peg_one_om, or *)
MP:Om_active> *

Continue_HO, or "*" for END/Maintenance
MP:Om_active> *

OM Data For Active class
_____________________
Call_proc, Handoff, Maint, Lcr, or *
MP:Om_active> *

All_active,Om_active
MP:Active> *

MP:Csc_oms>
Transfer,Report,Holding,Active

MP:Csc_oms> *
MP:OMUnsol>

STatus,SChedule,Csc_oms
MP:OMUnsol> *

MP>
TIme,TAsk,Load,Xprompt,Debug,Swerr,Ipc,Queues,Uspace,Patches,Msgtr,Chnls,
MTc, OM, Rcu,CP,Ho,DIagnose,Audit,MAtediag,OMUnsol.
```

***** The * returns you to the previous level.

PVC level

The protocol variant control (PVC) level contains subcommands for the monitoring and control of protocol variants, which is a method of controlling variation in the Q931 protocol and functions for different customers/markets.

Figure 4-49
Commands in the PVC level



The following commands are available in the PVC level:

PVC_CONTROL level

Displays and controls information related with PVC. This level displays the protocols that are available and those that are used for the interfaces on the peripheral. This level is used by ISDN PRI.

PVC_CONTROL contains the following subcommands:

PV_NAMES command

This command displays all PVC values and the names of associated protocols.

PV_NAMES	

PRI_MAP

This command displays all PVC values and the names of associated protocols and the mapping to the restricted range of PVCs used by PRI.

PRI_MAP	

Q_DTIDS

This command displays the PVC value, PVC issue, and the name of the PVC for each D-channel datafilled on the peripheral.

Q_DTIDS	

S_DTID_PV

You can change the PVC value for each D-channel displayed by Q_DTIDS with this command.

MDDS level

You can print the message definition data structure (MDDS) for a given protocol variant. This structure defines how Q931 messages are handled for parsing and generation at various levels.

MDDS contains the following subcommands:

SET_PV

This command sets the protocol variant for the other MDDS commands.

SET_PV	

DISP_PV

This command displays the currently set PV.

DISP_PV	

NAMES_OF_PV

This command displays the PV range.

NAMES_OF_PV	

PV_LEVEL
You can determine if MMDS is enabled for the PV for the set PV. This command also displays the following information on how the PV handles Q931 messages at the fundamental format level:

- which procedures are used
- which coding rules are used
- which code sets are allowed
- which options are allowed.

PV_LEVEL	

MSG_LEVEL
This command displays all messages that the PV can support for each call reference type for the set PV.

MSG_LEVEL	

IE_LISTS
This command displays all messages that the PV can support for each call reference type along with every information element allowed within each message for the set PV.

IE_LISTS	

IE_ALLOCATION
This command displays the number of information element description elements used for the PV. With this command you can get an idea of the data storage used.

IE_ALLOCATION	

POPO
You can enable or disable the MDDS for the PV.

> ***Note:*** If MDDS is disabled the pre-MMDS method of message handling is used.

POPO	

PARS level

You can construct a Q931 message, invoke the Q931 parser and examine the resulting data structure (QP directory).

FILL_BUFFER

You can enter hex values into a message buffer with this command.

FILL_BUFFER	

EDIT_BUFFER

You can change hex values that were entered in FILL_BUFFER.

EDIT_BUFFER	

DUMP_BUFFER

You can display the contents of the message buffer.

DUMP_BUFFER	

SEND_BUFFER

This command sends the buffer to the Q931 parser, and displays the contents of the QP directory resulting from the parsing process. This displays the internal representation of the dismantled message.

SEND_BUFFER	

DISPLAY_QPD

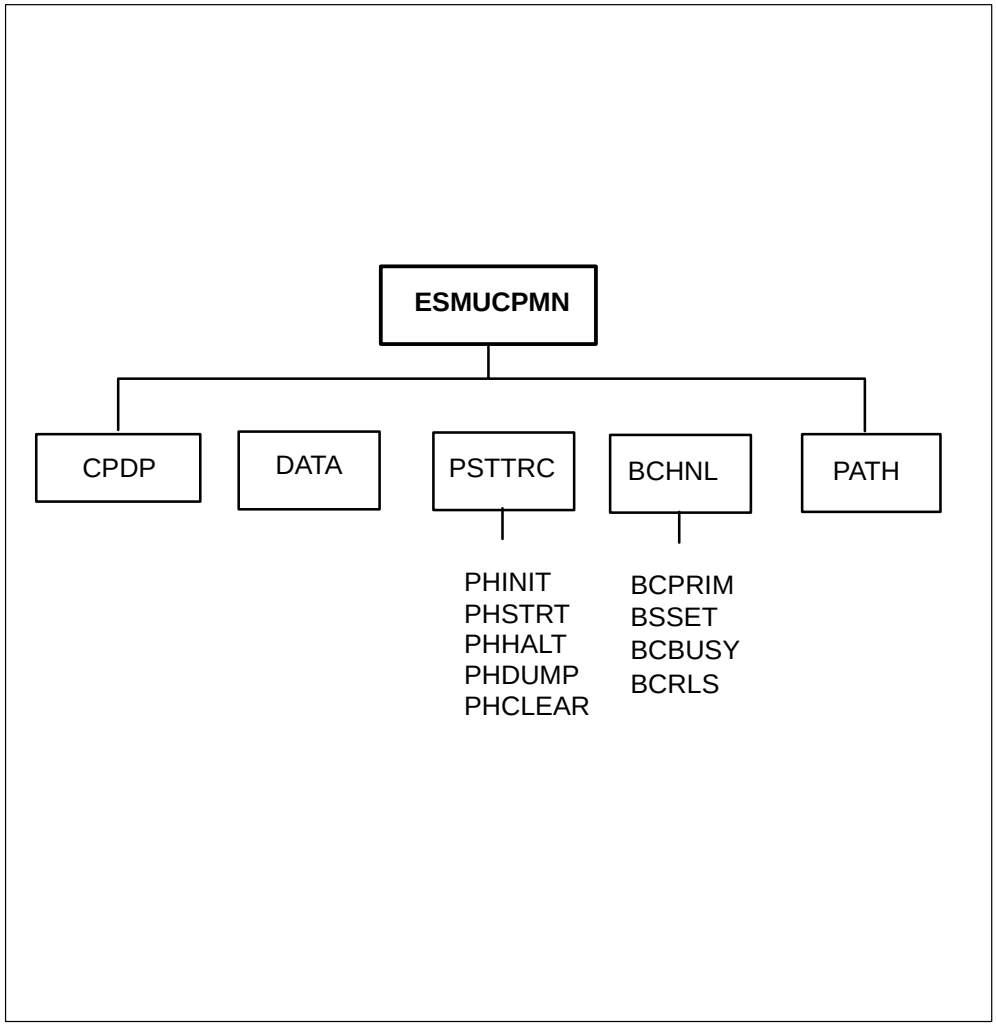
This command displays the contents of the QP directory.

DISPLAY_QPD	

ESMUCPMN level

The ESMUCPMN level is a monitor that affects the enhanced SMU peripheral. To enter this level type ECP.

Figure 4-50
Commands in the ESMUCPMN level



The following commands are available in the ESMUCPMN level:

CPDP command

The call processing display (CPDP) command displays call processing specific data structures for a particular line on the enhanced SMU. The parameters are RCU number (0..9) and line number (0..543). Both of these values are meaningful only to the SMU. It displays some fields of a call processing record that the SMU keeps for each line which varies depending

on which of the six line varieties it is. CPDP also displays the assigned port and channel if there is one.

Cpdp	

DATA command

The TPT data display (DATA) command displays selective fields of the ATHB, CHB, and FNB for a particular terminal on the SMU. The parameter for this command is internal terminal number (0..7022).

Data	

PSTTRC command

The P-phone state trace (PSTTRC) command contains options that allow the tracing of the channel state machine for a p-phone hanging of the SMU.

Psttrc	

The command PSTTRC has the following subcommands:

PHINIT

The PHINIT subcommand initializes traces, including selecting the line.

Phinit	

PHSTRT

The PHSTRT subcommand starts the trace used after PHINIT.

PHStrt	

PHHALT

The PHHALT subcommand halts tracing after test events have been completed.

PHHalt	

PHDUMP

The PHDUMP subcommand dumps all state transitions between PHSTRT and PHHALT.

PHDump	

PHCLEAR

The PHCLEAR subcommand clears out the captured state transitions. This subcommand can be used while tracing is active to erase unsignificant state transitions that would clutter the output.

PHClear	

BCHNL level

This level, busy channels, (BCHNL) contains options that allow simulating of All Channels Busy condition on the enhanced SMU. It simulates the ACB by filling in the pside busy/idle map as specified for the RCU being tested. It can be used only for one RCU at a time because of a desire to not use too much sotrage. It can simulate every channel being used. It can also simulate the scenario in which all but a few channels are being used. It remembers which channels it has marked busy so that after testing, just those channels can be returned to the idle state, and no other channels that might really be busy.

Bchnl	

The BCHNL level has the following subcommands:

BCPRIM

The BCPRIM subcommand selects the RCU. This subcommand has to be done once after XPM initialization. After initialization, the busy channel simulater will always act on the selected RCU until another RCU is explicitely selected via this command.

Bcprim	

BSSET

The BSSET subcommand sets the number of channels to leave idle. This subcommand does not busy any channels. After using this command to specify the number of idle channels per digroup, the BCBUSY command will busy all but that number per digroup. This command displays the current counts before and after the specified change is made.

BSset	digroup channels

where

digroup is the digroup number (0..7).

channels is the number of channels (0..23).

BCBUSY

The BCBUSY level busies all channels on the RCU selected by the BCPRIM command except for a small number of channels (default 0) per digroup specified using the BSSET command. This command keeps track of what it changes so BCRLs can change it back.

BCBusy	

BCRLS

The BCRLS command returns all channels to idle that were set to busy by the BCBUSY command. Channels that are actually busy, are not returned to idle. This command does nothing unless BCBUSY has been executed to busy out some channels.

BCRls	

PATH level

The display path_to_tid table (PATH) command displays the path_to_tid table for the ds1 port specified as a parameter. There are 32 entries displayed for each port, one for each channel. The tid displayed for each channel shows a node number, terminal number, and extension byte.

PAth	

PMDEBUG commands in the signaling processor

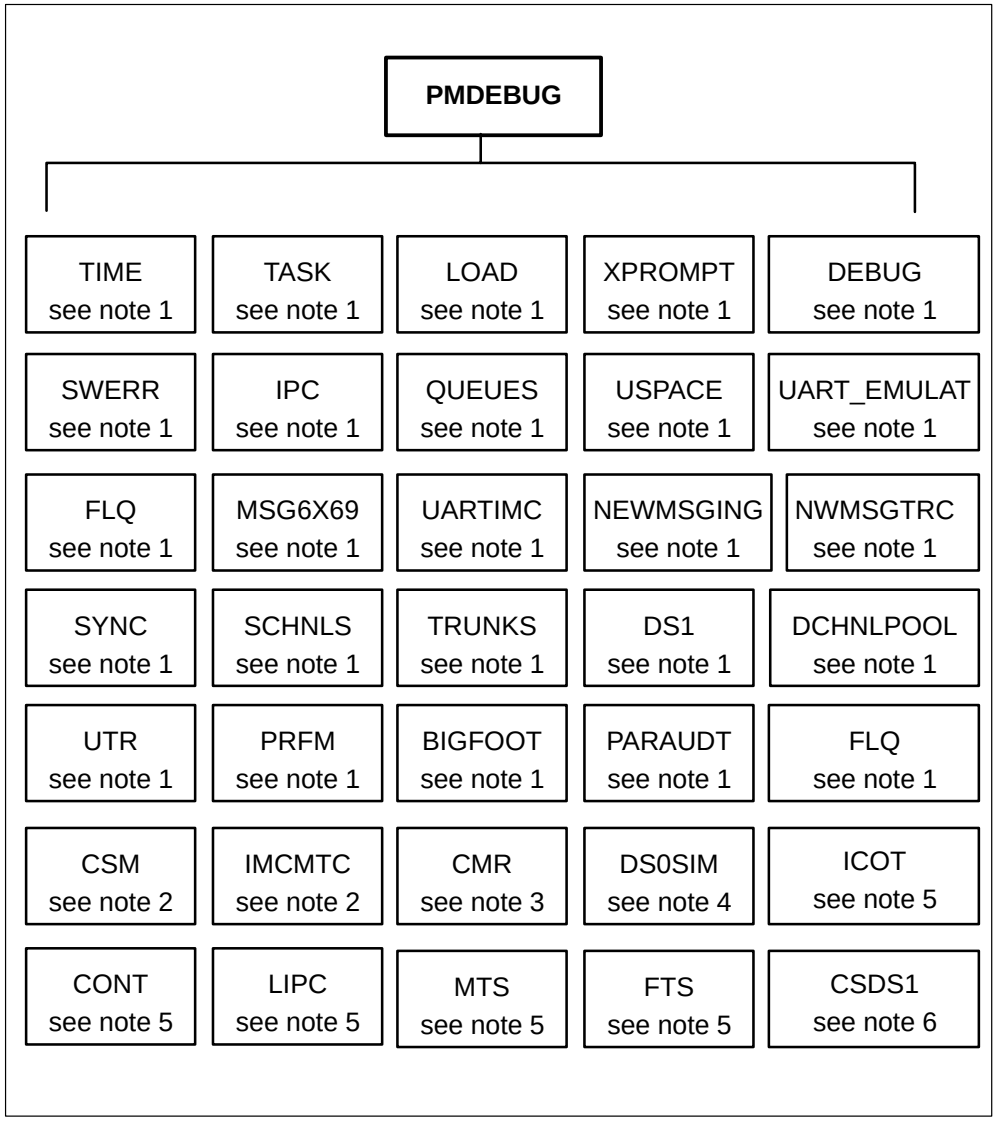
This section describes the commands that are available only in the signaling processor (SP). Commands that are common to the SP and the master processor (MP) are described in Chapter 4, “PMDEBUG commands in the master processor”.

You must access the PMDEBUG subsystem in order to use the PMDEBUG subcommands. See “Accessing PMDEBUG” beginning on page 7-1.

> ***Note:*** The top level of PMDEBUG varies depending on the peripheral being accessed. For simplicity in this document, the top level of PMDEBUG in the SP is referred to as the LTCSP level.

Figure 5-1 on page 5-2 contains the commands available at the LTCSP level of a DTC.

Figure 5-1
Commands at the SP level



Note: 1 This command is used in the standard DTC, CCS7 DTC, RCC, and LTC.

Note: 2 This command is used in the standard DTC and LTC.

Note: 3 This command is used in the LTC and RCC.

Note: 4 This command is only used in the standard DTC.

Note: 5 This command is only used in the CCS7 DTC.

Note: 6 This command is only used in the RCC.

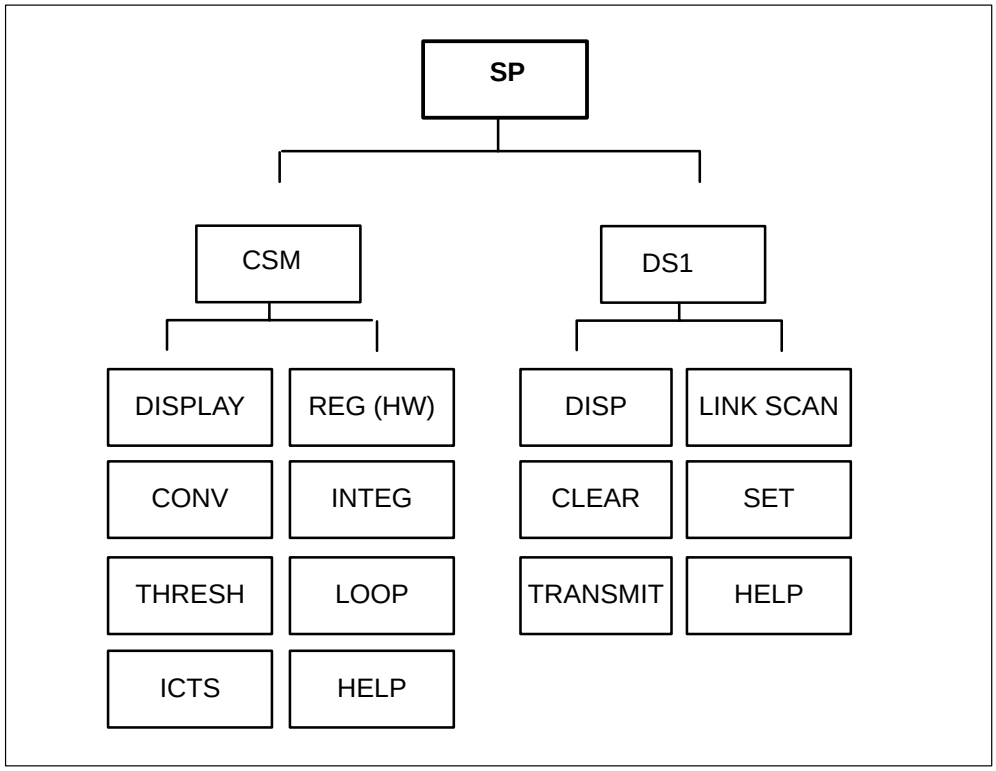
SP level

The signaling processor (SP) level is accessed when the SP command is processed from the LTCSP level.

The SP level provides access to the CSM level and the DS1 level.

Figure 5-2 on page 5-3 contains the commands available at the SP level.

Figure 5-2
Commands in the FACM level



Following are descriptions of the commands available in the SP level:

CSM level

You can view the channel supervision message (CSM) data structure values for a particular channel by using the CSM level. You can also view and set the hardware registers.

CSM	

Following are descriptions of the commands available in the CSM level:

CONV command

The CONV command converts external C-side channel and port formats to the internal channel format that is used by the CSM monitor commands. The internal channel number is displayed in both hex and decimal.

CONV	chnlno portno

Where:

chnlno is the C-side channel number to be converted

portno is the C-side port number to be converted

Note: The CONV command uses the following formula: external channel * 16 + port = internal channel number

Example:

```
CSM: Display, Reg(HW), Conv,iNteg,Thresh,Loop,Icts,
Help,*
SP:CSm>

>conv 25 2

converts channel/port format -> internal chnl
internal channel = Decimal 402
internal channel = hex     192

CSM: Display, Reg(HW), Conv,iNteg,Thresh,Loop,Icts,
Help,*
SP:CSm>
```

DISPLAY command

The DISPLAY command displays the current value of the CSM data structures for the specified C-side internal channel number.

DISPLAY	chnlno

Where:

chnlno is the C-side internal channel number to be displayed

Example:

```
CSM: Display, Reg(HW), Conv,iNteg,Thresh,Loop,Icts,
Help,*
SP:CSm>

>d #192

init_val = # 80
mask =  # 80
filter = # 4
ret_cid = # 8D
fs = # 2
tfcn = tfn_a
cdb_mode = cdb_scn
SV mapping flag = FALSE
timer index  =  # 0
look check flag = FALSE
scnd_pl flag = FALSE
pause: Do you wish to continue? y/n

SP:CSm>

>y

Extension block data...
  ..  no ext.  block
pause: Do you wish to continue? y/n

SP:CSm>

>y


ref term num = # 2B
ref node num = # 2
ref ext byte = # 0
ntg idx = # 0
ntg ret = # 8D
ntg fcn seq = # 2
ntg_mode = ntg_chkntg
ntg_ch = TRUE
mfcn = tfcn_all
cdb_masks =  # 80
cdb_llm =  # 80
ntg masks are: ACTIVE
```

```
CSM: Display, Reg(HW), Conv, iNteg, Thresh, Loop, Icts,
Help, *
SP:CSm>


init_val = # 80
mask =  # 80
filter = # 4
ret_cid = # 8D
fs = # 2
tfcn = tfn_a
cdb_mode = cdb_scn
SV mapping flag = FALSE
timer index  =  # 0
look check flag = FALSE
scnd_pl flag = FALSE
pause: Do you wish to continue? y/n

SP:CSm>

>y

Extension block data...
  ..  no ext.  block
pause: Do you wish to continue? y/n

SP:CSm>

>y
```

REG(HW) command

The REG command displays the current values of the hardware registers associated with the specified internal channel number.

REG	chnlno

Where:

chnlno is the channel number to be displayed.

Example:

```
CSM: Display, Reg(HW), Conv,iNteg,Thresh,Loop,Icts,
Help,*
SP:CSm>

>r #192

channel = 402
xmit_cdb = # 80
xmit_ntg = # FA
rcvd_cdb = # 80
xpec_ntg = # FA
ntg match bit = MATCH
plane select = 1
csm loop around = DISABLED
ntwk loop = DISABLED
parity injection = DISABLED

CSM: Display, Reg(HW), Conv,iNteg,Thresh,Loop,Icts,
Help,*
SP:CSm>
```

INTEG command

The INTEG command displays or clears the integrity hits counters. A counter is pegged when an integrity hit is detected on a specific port, but integrity is reestablished before the failure filter time has expired. A high count indicates a hardware problem on the port and channel.

One counter exists for each port on each plane.

INTEG	

Following are descriptions of the commands available in the INTEG level:

DISPLAY The DISPLAY command displays the integrity hit counters.

Example:

```
CSM: Display, Reg(HW), Conv,iNteg,Thresh,Loop,Icts,
Help,*
SP:CSm>

>n

Display, Clear, *
SP:CSm>

>d
```

```
plane\port                hitcount
     0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15
0 | 0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
1 | 0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0

Display, Clear, *
SP:CSm>

>*

CSM: Display, Reg(HW), Conv,iNteg,Thresh,Loop,Icts,
Help,*
SP:CSm>
```

CLEAR The CLEAR command clears the integrity hit counters.

THRESH command

The THRESH command displays the current parity threshold and integrity filter values.

THRESH	

Note: The parity threshold and integrity filter values are set from the Network (NET) level of the MAP.

Example:

```
CSM: Display, Reg(HW), Conv, iNteg, Thresh, Loop, Icts,
Help, *
SP:CSm>

>>>t

Parity = 20
Integrity = 12

CSM: Display, Reg(HW), Conv, iNteg, Thresh, Loop, Icts,
Help, *
SP:CSm>
```

LOOP command

The LOOP command displays the active internal channels for the specified internal terminal number.

LOOP	termno

Where:

termno is the internal terminal number

Example:

```
CSM: Display, Reg(HW), Conv,iNteg,Thresh,Loop,Icts,
Help,*
SP:CSm>

>I 43

Found an active channel    - 217
Found an active channel    - 402
```

ICTS command

The ICTS command displays the C-side port channels performing integrity check traffic simulation (ICTS).

ICTS	

Note: This command is time-consuming and should be used with caution.

HELP command

The HELP command displays the parameter syntax for the commands in the CSM level of the FACM level.

HELP	

Example:

```
CSM: Display, Reg(HW), Conv,iNteg,Thresh,Loop,Icts,
Help,*
SP:CSm>

>h

CSM monitor command syntax:
  Display  'channel number'
  Registers  'channel number'
  Convert  'channel number'  'port number'
  Integrity

CSM: Display, Reg(HW), Conv,iNteg,Thresh,Loop,Icts,
Help,*
SP:CSm>
```

DS1 level

The DS1 level contains commands that display information associated with the DS1 links.

DS1	

Following are descriptions of the commands available in the DS1 level:

DISP command

The DISP command displays the data values associated with a particular link.

DISP	**link**

Where:

link is the link number to be displayed.

Example:

```
DS1: Disp, Link scan, Clear, Set, Transmit, Help, *
SP:DS1>

>>>d 12

location : P_side       link_number: 12
carrier type index: 1   cardtype    :  nt6x50aa  DS1
counts:
  frameloss:  0          slip:         0

  ES:         0          SES:        0

  UAS:        0
port status/cntrl  (s/w view)   : 1 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0
port status/cntrl2  (s/w view) : 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
|frame|r ind |card |a mtce|f mtce|bpv m|bpv o|l ala|r ala|uninit
   X            X      X       X
|l-loop|r-loop |rlbenb |hw ind|hw alm|diag|a ind|a ala|r lb ind
status reg 1 :  0 0 0 1 1 1 0 1         control register 1: 0000
status reg 3 :  0 0 0 0 0 0 0 0         control register 2: 0060
                                        control register 3: 0043
  ber_count:  13
  slip:       no
  frame loss: no
  rcga:       no

NO remote alarm reported via software.

DS1: Disp, Link scan, Clear, Set, Transmit, Help, *
SP:DS1>
```

LINK SCAN command

The LINK SCAN command scans for the status of a single link or for all links.

LINK SCAN	[symbols] [linkno] [loop interval]

Where:

symbols prints the meaning of the link status symbols

linkno is the link number

loop continuously scans the specified links until you press Enter.

interval is the time interval in seconds. Consecutive loops will take place every n seconds, where n can be from 0 to 30

Example:

```
DS1: Disp, Link scan, Clear, Set, Transmit, Help, *
SP:DS1>

>>>l

Location: P_side
  0 :    'c'     'c'     'c'     'c'     'c'
  5 :    '!'     '!'     '!'     'c'     'c'
 10 :    'c'     'c'     'c'     'c'     'c'
 15 :    'c'     'c'     'c'     'c'     'c'

DS1: Disp, Link scan, Clear, Set, Transmit, Help, *
SP:DS1>
```

The possible values for the link status are as follows:

n indicates uninitialized

c indicates that the card is present and maintenance is not active on the link

! indicates that the link is stable and full maintenance is active on the link

x indicates that the card is not present

L indicates local alarm state

R indicates remote alarm state

CLEAR command

The CLEAR command clears the bits in the software image of the link status/cntr.

CLEAR	field	link1 ... linkn

Where:

field is one of the following integers:

0 frame sync

1 remote alarm received

2 card present

3 active maintenance

4 full maintenance

5 bipolar violation (BPV) maintenance

6 BPV out-of-service (OOS)

7 local alarm

8 remote alarm

9 restart uninitialized

10 under diagnosis

11 automatic intercept system (AIS) alarm received

12 local loop bit

13 remote loop bit

14 AIS alarm

15 unavailable second

link1 ... linkn
are the link numbers

SET command

You can set bits in the software image of the link status by using the SET command.

CAUTION

This command should *not* be used on a live switch.

SET	field link1 ... linkn

Note: For a description of the parameters associated with the SET command, refer to the parameter descriptions under "CLEAR command" on page 5-12.

TRANSMIT command

The TRANSMIT command changes the control register transmit for a link.

CAUTION

This command should *not* be used on a live switch.

TRANSMIT	register	data	ber	m_tuples	link1 ... linkn

Where:

register is the control register number

data is the hex byte representing the bit pattern that will be output to the control register

ber the ber information is the link_number

m_tuples the m_tuples information is the index number

link1 ... linkn are the link numbers.

HELP command

The HELP command displays the parameter syntax for the commands available in the DS1 level of the SP level.

HELP	

Example:

```
DS1: Disp, Link scan, Clear, Set, Transmit, Help, *
SP:DS1>

>>>h

DS1 maintenance monitor command syntax:
  Display        'link_number;
  Link Scan      <Symbols> <'link_number'> <Loop<'interval'>>
  Clear field    'field' 'link_1' ..  'link_n'
  Set field      'field' 'link_1' ..  'link_n'
  Transmit    'register #' 'data'  'link_1' ..  'link_n'
  Mon tuples     'index  #'
   BER info      'link_number'

For R, C, S, T, M a null line terminates input

DS1: Disp, Link scan, Clear, Set, Transmit, Ber, Mon tupels, Help, *
SP:DS1>
```

MSG6X69 level

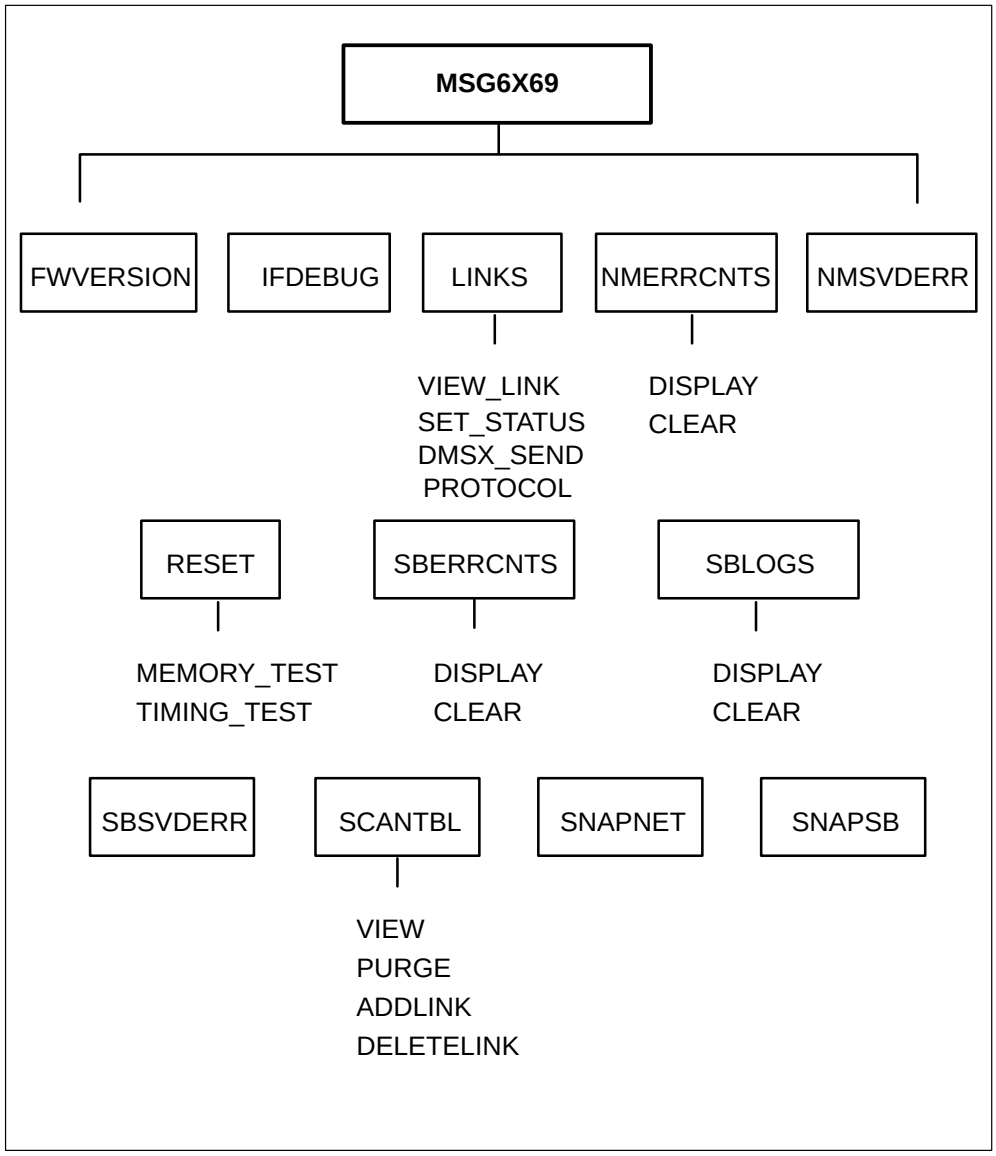
The MSG6X69 level is accessed when the MSG6X69 command is entered at the LTCSP level.

The MSG6X69 level contains commands for displaying and editing NT6X69 buffers and displaying error counts, logs, and link status.

Figure 5-3 on page 5-17 shows the commands available at the MSG6X69 level.

CAUTION
This monitor level should not be used by a customer, unless specifically directed to and guided by a BNR or NT representative.

Figure 5-3
Commands in the MSG6X69 level



Following are descriptions of the commands available at the MSG6X69 level:

FWVERSION command

The FWVERSION command displays the current MSG6X69 firmware version.

FWVERSION	

IFDEBUG level

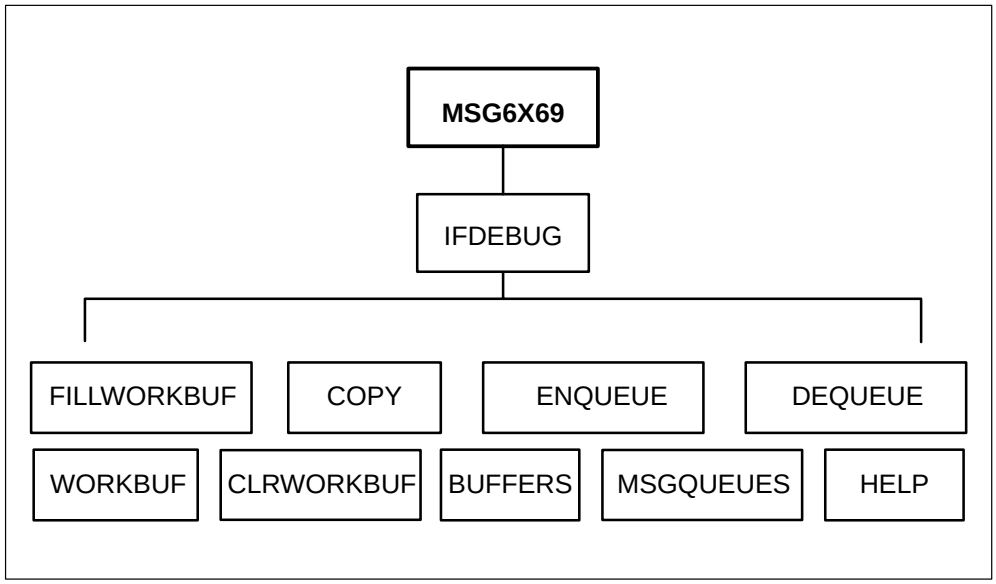
The IFDEBUG command accesses the IFDEBUG level, which contains buffer and queue editing commands.

Editing is not done directly to a NT6X69 buffer. The buffer is copied to a 256-byte array in the SP RAM to form an image of that buffer. This image is called the work buffer. When editing on the work buffer is complete, the buffer is copied back to the actual NT6X69 buffer.

Ifdebug	

Figure 5-4 on page 5-18 contains the commands available at the IFDEBUG level.

Figure 5-4
Commands in the IFDEBUG level



Following are the commands available at the IFDEBUG level:

FILLWORKBUF command

The FILLWORKBUF command fills the work buffer with the byte data. You can specify which byte of the work buffer should be filled.

Fillworkbuf	

Note: The format for filling the bytes of the work buffer is as follows:

<first byte to fill> <data> <data> ...

For example, to fill the 10th, 11th, 12th, and 13th byte of the work buffer, the 10th byte would be the first byte to fill. Therefore, the format would be as follows:

```
9 byte1 byte2 byte3 byte4
```

The 9 indicates the 10th byte in the buffer (when numbering from 0, 9 is the 10th byte.) Byte1, byte2, byte3, and byte4 are the data bytes to be inserted.

If no number precedes the data bytes, the default is 0.

COPY command

The COPY command copies the work buffer to the currently dequeued buffer and copies the currently dequeued buffer to the work buffer.

Copy	

ENQUEUE command

The ENQUEUE command puts the currently dequeued buffer on a specific queue.

Enqueue	

DEQUEUE command

The DEQUEUE command removes a buffer from a specified queue.

Dequeue	**Sbmogq SBMIcq SBRejq SBFogq SBFIcq Nmmicq NMMOgq NMRejq NMFicq NMFOgq Umogq UMIcq URejq UFogq UFIcq**

where

Sbmogq dequeues the speech bus message outgoing queue.

SBMIcq dequeues the speech bus message incoming queue.

SBRejq dequeues the speech bus reject queue.

SBFogq dequeues the speech bus free outgoing queue.

SBFIcq dequeues the speech bus free incoming queue.

Nmmicq dequeues the network module message incoming queue.

NMMOgq dequeues the network module message outgoing queue.

NMRejq dequeues the network module reject queue.

NMFicq dequeues the network module free incoming queue.

NMFOgq dequeues the network module free outgoing queue.

Umogq dequeues uart imc link message outgoing queue index.

UMIcq dequeues uart imc link message incoming queue index.

URejq dequeues uart imc link message reject queue index.

UFogq dequeues uart imc link message free outgoing queue index.

UFIcq dequeues uart imc link message free incoming queue index.

WORKBUF command

The WORKBUF command displays the contents of the work buffer.

Workbuf	

CLRWORKBUFF command

The CLRWORKBUFF command clears the working buffer.

CLrworkbuff	

BUFFERS command

The BUFFERS command displays a selected I/F buffer.

Buffers	

MSGQUEUES command

The MSGQUEUES command displays a selected I/F queue.

Msgqueues	**Sbmogq SBMIcq SBRejq SBFogq SBFIcq Nmmicq NMMOgq NMRejq NMFicq NMFOg Umogq UMIcq URejq UFogq UFIcq**

where

Sbmogq displays the speech bus message outgoing queue.

SBMIcq displays the speech bus message incoming queue.

SBRejq displays the speech bus reject queue.

SBFogq displays the speech bus free outgoing queue.

SBFIcq displays the speech bus free incoming queue.

Nmmicq displays the network module message incoming queue.

NMMOgq displays the network module message outgoing queue.

NMRejq displays the network module reject queue.

NMFicq displays the network module free incoming queue.

NMFOgq displays the network module free outgoing queue.

Umogq displays uart imc link message outgoing queue index.

UMIcq displays uart imc link message incoming queue index.

URejq displays uart imc link message reject queue index.

UFogq displays uart imc link message free outgoing queue index.

UFIcq displays uart imc link message free incoming queue index.

The valid queues are as follows:

- speech bus message outgoing queue (sb msg o/g)
- speech bus message incoming queue (sb msg i/c)
- speech bus reject queue (sb reject)
- speech bus free outgoing queue (sb free o/g)
- speech bus free incoming queue (sb free i/c)
- network module message outgoing queue (nm msg o/g)
- network module message incoming queue (nm msg i/c)
- network module reject queue (nm reject)
- network module free outgoing queue (nm free o/g)
- network module free incoming queue (nm free i/c)

HELP command

The HELP command displays instructions for editing a buffer.

Help	

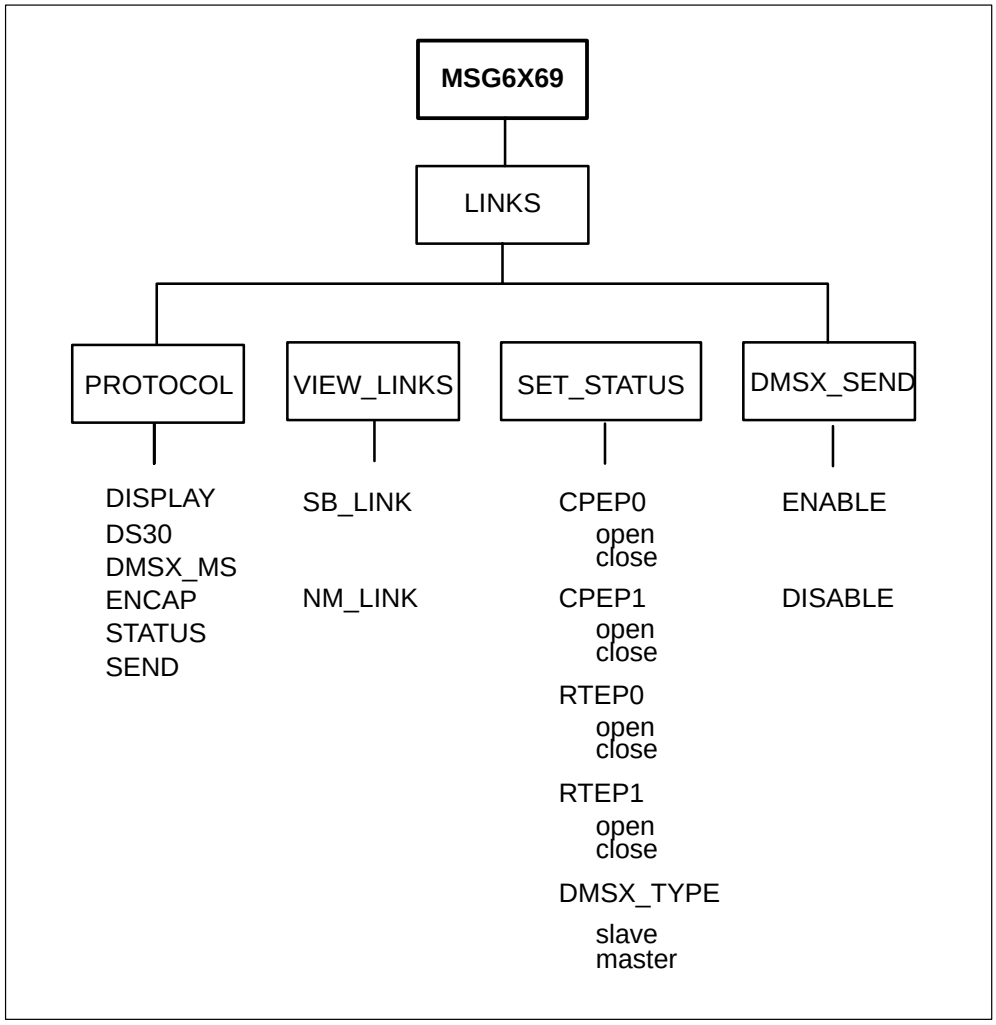
LINKS level

You can view speech bus and net links, set the send/receive status of network and speech bus links, and enable/disable the send or receive state of the DMSX link (speech bus time slot) by using the LINKS levels.

LINKS	

Figure 5-5 on page 5-22 contains the commands available at the LINKS level.

Figure 5-5
Commands in the LINKS level



Following are descriptions of the commands available at the LINKS level:

PROTOCOL command
The PROTOCOL level

Protocol	

Following are the commands available at the PROTOCOL level:

DISPLAY

The DISPLAY command

DIsplay	

DS30

The DS30 command

DS30	

DMSX_MS

The DMSX_MS command

DMSX_ms	

ENCAP

The ENCAP command

Encap	

STATUS

The STATUS command

STatus	

SEND

The SEND command

SEnd	

VIEW_LINK command

The VIEW_LINK level displays the status of the speech bus links and the network module links.

VIEW_LINK	

Following are the commands available at the VIEW_LINK level:

SB_LINK displays the SB links status. You are prompted for speech bus time slot. The link status can be one of:

- link active/inactive
- send enable/disable
- master/slave link
- process number currently handling that link. (Processes in the 6X69 card range from 0 to 6.)

NM_LINK displays the NM link status.

SET_STATUS command

You can change the network and speech bus links status with the SET_STATUS command. The network side can be set to open or closed, and the speech bus side can be set to master or slave.

Set_status	

Following are the options available at the SET_STATUS level:

CPEP0 represents the Call Processing Enabled (CPE) plane 0.

Note: The CPE and RTE status bytes store the scan and transmit status of each plane. The RTE byte is used to determine whether a message can be transmitted on the link. The CPE byte is used to determine whether a C-side link should be scanned for messages. If the CPE byte for a particular plane is open, the link is scanned for a May I Send (MIS) code.

Following are the options available at the CPEP0 level:

- open
- close

CPEP1 represents the Call Processing Enabled (CPE) plane 1.

Following are the options available at the CPEP1 level:

- open

- close

RTEP0 represents the Routing Enabled (RTE) plane 0.

Note: The CPE and RTE status bytes store the scan and transmit status of each plane. The RTE byte is used to determine whether a message can be transmitted on the link.

Following are the options available at the RTEP0 level:

- open
- close

RTEP1 represents the Routing Enabled (RTE) plane 1.

Following are the options available at the RTEP1 level:

- open
- close

DMSX_TYPE you can change the master/slave status of the specific P-side link with the DMSX_TYPE command. The DMSX_TYPE takes the timeslot number as a parameter.

Following are the commands available at the DMSX_TYPE level:

- master
- slave

DMSX_SEND
You can send a message on a specified speech bus link with the DMSX_SEND command.

DMSX_SEND	

Following are the commands available at the DMSX_SEND level:

enable you can enable sending on a specified speech bus link.

disable you can disable sending on a specified speech bus link.

NMERRCNTS level

The NMERRCNTS command displays or clears the current network module error counts.

Nmerrcnts	

Following are the commands available at the NMERRCNTS level:

DISPLAY command

The DISPLAY command displays the error count.

DISPLAY	

CLEAR command

The CLEAR command clears the error counters.

CLEAR	

NMSVDERR command

The NMSVDERR command displays the saved network module error counts.

NMSVDERR	

RESET level

The RESET command resets the MSG6X69 card. You are prompted to indicate whether the pretimeslice diagnostic should be run during the reset.

RESET	

Following are the commands available at the RESET level:

> ***Note:*** The MEMORY_TEST and TIMING_TEST commands are available in the RESET level after you access the Pretimeslice diagnostic.

MEMORY_TEST command

The MEMORY_TEST command performs a memory test while the 6X69 card is resetting.

MEMORY_TEST	

TIMING_TEST command

The TIMING_TEST command performs a timing test while the 6X69 card is resetting.

TIMING_TEST	

SBERRCNTS level

The SBERRCNTS command displays or clears the current speech bus error counts.

SBERRCNTS	

Following are the commands available at the SBERRCNTS level:

DISPLAY command

The DISPLAY command displays the speech bus error counts.

DISPLAY	

CLEAR command

The CLEAR command clears the speech bus error counters.

CLEAR	

SBLOGS level

The SBLOGS command displays or clears the speech bus logs.

SBLOGS	

Following are the commands available at the SBLOGS level:

DISPLAY command

The DISPLAY command displays the speech bus logs.

DISPLAY	

CLEAR command

The CLEAR command clears the speech bus logs.

CLEAR	

SBSVDERR command

The SBSVDERR command displays the speech bus error counts.

SBSVDERR	

SCANTBL level

The 6X69 card firmware can service a maximum of 640 lines. A scan table stores the time slot value associated with the lines. This table serves as a queue for lines that are currently recognized by the 6X69 firmware and/or are currently active.

A line is added to the scan table when any processing is required, such as sending data, receiving data, status changes, and so on.

The SCANTBL level views or clears the scan table, adds a speech bus link to it, or compacts the scan table.

SCANTBL	

Following are the commands available at the SCANTBL level:

VIEW command

The VIEW command displays the scan table.

VIEW	

PURGE command

The PURGE command clears the scan table.

PURGE	

ADDLINK command

The ADDLINK command adds a SB link to the scan table. You are prompted for the time slot number.

ADDLINK	

DELETELINK command

The DELETELINK command removes a SB link from the scan table. You are prompted for the time slot number.

DELETELINK	

SNAPNET command

The SNAPNET command displays a snapshot of the network module error counts.

SNAPNET	

SNAPSB command

The SNAPSB command displays a snapshot of the speech bus error counts.

SNAPSB	

Reset,Links,Scantbl,Nmerrcnts,SBerrcnts,NMSvderr,SBSvderr,
SBlogs,SNapnet,SNAPSb,Fwversion,Ifdebug.
SP:Msg6x69>

```
>>>l

View_links,Set_status,Dmsx_send.
SP:Links>

>>>v

Sb_link,Nm_link.
SP:View_link>

>>>s

Enter SPeech Bus timeslot
SP:Sb_link>

>>>18

Link Inactive
Send Disabled
Slave Link
process number0
SP:Sb_link>

>>>n

plane 0 cpe open
plane 1 cpe open
plane 0 rte open
plane 1 rte open
SP:View_link>

>>>*

SP:Links>

View_links,Set_status,Dmsx_send.
SP:Links>

>>>s

Cpep0,CPEP1,Rtep0,RTEP1,Dmsx_type.
SP:Set_status>

>>>c

Open,Close.
```

```
SP:Cpep0>

>>>o

SP:Cpep0>

>>>*

Cpep0,CPEP1,Rtep0,RTEP1,Dmsx_type.
SP:Set_status>

>>>cpep

Open,Close.
SP:CPEP1>

>>>c

SP:CPEP1>

>>>*

SP:Set_status>

>>>*

SP:Links>

View_links,Set_status,Dmsx_send.
SP:Links>

>>>d

Enable,Disable.
SP:Dmsx_send>

>>>e

Enter Speech Bus timeslot
SP:Enable>

>>>18

SP:Dmsx_send>

>>>d
```

Enter Speech Bus timeslot
SP:Disable>

>>>18

SP:Dmsx_send>

>>>*

SP:Links>

NEWMSGING level

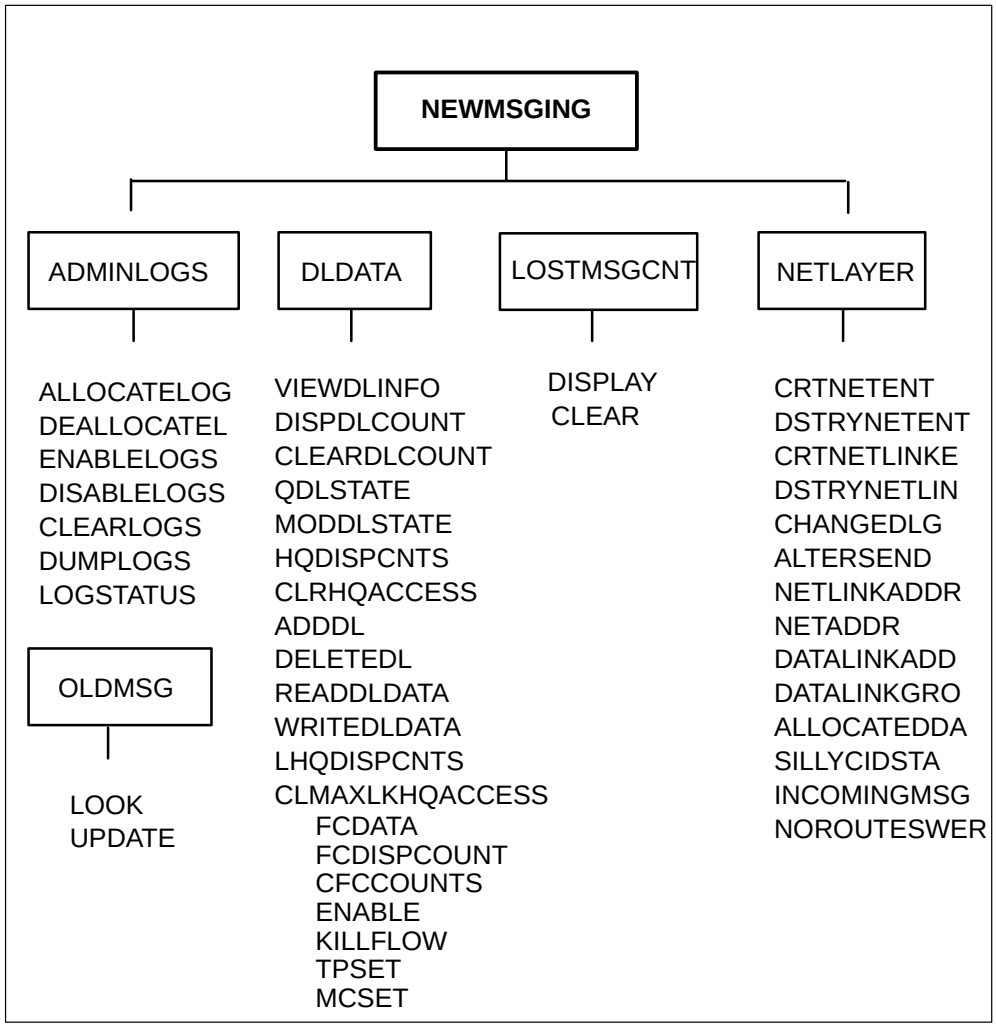
The NEWMSGING level is accessed when the NEWMSGING command is entered at the LTCSP level.

The NEWMSGING level contains commands for accessing the new messaging system. Figure 5-6 on page 5-33 contains the commands available at the NEWMSGING level.

CAUTION

This monitor level should not be used by a customer, unless specifically directed to and guided by a BNR or NT representative.
This command can cause service degradation to the PM.

Figure 5-6
Commands in the NEWMSGING level



ADMINLOGS level

The ADMINLOGS level contains commands for manipulating log information.

ADMINLOGS	

Following are descriptions of the commands available at the ADMINLOGS level:

ALLOCATELOG command

The ALLOCATELOG command allocates temporary storage for log entries.

ALLOCATELOG	

DEALLOCATEL command

The DEALLOCATEL command deallocates temporary storage for log entries.

DEALLOCATEL	

ENABLELOGS command

The ENABLELOGS command starts the log reports.

ENABLELOGS	

DISABLELOGS command

The DISABLELOGS command stops the log reports.

DISABLELOGS	

CLEARLOGS command

The CLEARLOGS command clears the log reports.

CLEARLOGS	

DUMPLOGS command

The DUMPLOGS command displays the log report information.

DUMPLOGS	

LOGSTATUS command

The LOGSTATUS command queries the status of the logs.

LOGSTATUS	

Example:

```
Adminlogs,Dldata,Netlayer,Oldmsg.
SP:Newmsging>

>>>a

Allocatedlog,Deallocatel,Enablelogs,DIsablelogs,Clearlogs,
DUmplogs,Logstatus.
SP:Adminlogs>

>>>a
```

```
LOGS HAVE BEEN DISABLED
LOGS HAVE BEEN CLEARED
LOG SPACE ALLOCATED, 100 LOG ENTRIES AVAILABLE

SP:Adminlogs>

>>>e

LOGS HAVE BEEN ENABLED

SP:Adminlogs>

>>>l

LOG SPACE ALLOCATED, LOGS ARE ENABLED, LOGS ARE NOT FULL

SP:Adminlogs>

>>>du

Log index -> 0
Netlayer Event -> open network link
Event Address ->
  Internal Node # : 1
  Unit Id : 0
  Link # : 0
Event time -> 00:00:20:49.81

Log Index -> 1
Netlayer Event -> open network link
Event Address ->
  Internal Node # : 1
  Unit Id : 3
  Link # : 0
Event Time -> 00:00:20:49.01
log index 1
link #2
action -> dl_send_enable
time: 00:00:20:49.81

log Index 2
link# 2
action -> dl_send_enable
time : 00:00:20:49.81
log index 3
link #2
```

```
action -> dl_rcv_enable
time: 00:00:20:49.81

Log Index -> 4
Netlayer Event -> open network link
Event Address ->
  Internal Node # : 1
  Unit Id : 0
  Link # : 0
Event Time -> 00:00:21:09.81
Log index -> 5
Netlayer Event -> open network link
Event Address ->
  Internal Node # : 1
  Unit Id : 0
  Link # : 0
action -> dl_send_enable
Event time -> 00:00:21:09.81
 .
 .
 .
log index 11
link # 2
action -> dl_rcv_enable
time : 00:00:21:29.81
Continue? ( Y/N )
SP:DUmplogs>

>>>n

SP:Adminlogs>

Allocatelog,Deallocatel,Enablelogs,DIsablelogs,Clearlogs,
DUmplogs,Logstatus.
SP:Adminlogs>

>>>di

LOGS HAVE BEEN DISABLED

SP:Adminlogs>

>>>d

LOGS HAVE BEEN DISABLED

SP:Adminlogs>
```

```
>>>*
```

DLDATA level

The DLDATA level provides access to the data link layer.

DLDATA	

Following are the commands available at the DLDATA level:

VIEWDLINFO command

The VIEWDLINFO command displays the link number, interface identifier type, format identifier, and physical address for a specified data link. You are prompted for the data link address.

VIEWDLINFO	

DISPDLCOUNT command

The DISPDLCOUNT command displays the number of messages sent, messages received, messages rebounded, and messages lost while transmitting and receiving. You are prompted for the data link address.

DISPDLCOUNT	

CLEARDLCOUNT command

The CLEARDLCOUNT command clears the counters that peg the number of messages sent, messages received, messages rebounded, messages lost while transmitting and receiving. You are prompted for the data link address.

CLEARDLCOUNT	

QDLSTATE command

The QDLSTATE command queries the send and receive state of a data link. You are prompted for the data link address.

QDLSTATE	

MODDLSTATE command

The MODDLSTATE command modifies the send and receive state of a data link. You are prompted for the data link address.

MODDL- STATE	

The valid send and receive states are as follows:

O open

C closed

HQDISPCNTS command

The HQDISPCNTS command displays the number of times the holding queue has been accessed and the number of items currently in the holding queue. You are prompted for the interface identifier type.

HQDISPCNTS	

You are prompted to enter one of the following formats:

0-INTERN internal interface

1-NET network (DS30) interface

2-SPCHBUS speech bus (DMSX) interface

3-CSC CSC message card

4-CSL CSL message card

5-IMC intermate interface

6-NIL no interface identifier

CLRHQACCESS command

The CLRHQACCESS command clears the counter that pegs the number of times the hold queue was accessed. You are prompted for the interface identifier type.

CLRHQACCESS	

You are prompted to enter one of the following formats:

0-INTERN internal interface

1-NET network (DS30) interface

2-SPCHBUS speech bus (DMSX) interface

3-CSC CSC message card

4-CSL CSL message card

5-IMC intermate interface

6-NIL no interface identifier

ADDDL command

You can add a dummy data link with the ADDDL command.

ADDDL	

DELETEDL command

The DELETEDL command deletes a data link. You are prompted for the data link address.

DELETEDL	

READDLDATA command

The READDLDATA command reads a byte of data from a data link. When the internal queue identifier is NET, the data is read from the MSG6X69 interface memory. When the internal queue identifier is SPCHBUS, the data is read from the time slot. You are prompted for the data link address.

READDLDA-TA	

WRITEDLDATA command

The WRITEDLDATA command writes a byte of data to a data link. When the internal queue identifier is NET, the data is read from the MSG6X69 interface memory. When the internal queue identifier is SPCHBUS, the data is read from the time slot. You are prompted for the data link address.

WRITEDLDATA	

LHQDISPCNTS command

The LHQDISPCNTS command displays the link number associated with the holding queue in use and the maximum number of holding queues in use.

LHQDISPCNTS	

You are prompted to enter one of the following formats:

0-INTERN internal interface

1-NET network (DS30) interface

2-SPCHBUS speech bus (DMSX) interface

3-CSC CSC message card

4-CSL CSL message card

5-IMC intermate interface

6-NIL no interface identifier

CLMAXLKHQACCESS command

The CLMAXLKHQACCESS command clears the counter that pegs the number of holding queues in use.

CLMAXLKHQACCESS	

You are prompted to enter one of the following formats:

0-INTERN internal interface

1-NET network (DS30) interface

2-SPCHBUS speech bus (DMSX) interface

3-CSC CSC message card

4-CSL CSL message card

5-IMC intermate interface

6-NIL no interface identifier

FCDATA level

The flow control data sublevel of the DLDATA level is password protected. The FCDATA sublevel displays the flow control values and allowable message counts for the assigned time period. Commands in the FCDATA level also display current flow control variable settings.

FCDATA command

FCDATA	

Example:

```
SP:Dldata>
fcdata
OGMFC level          (Outgoing Message Flow Control)
LTCSP>
                     (PASSWORD is Entered Here)
walk softly but with a big stick (Indicates successful entry)
Fcdispcount,Cfccounts,Enableflow,Killflow,Tpset,Mcset.
SP:Fcdata>
```

Following are command descriptions at the FCDATA sublevel of DLDATA:

FCDISPCOUNT command

The FCDISPCOUNT displays the flow control counters and states if flow control is installed, enabled, or started. FCDISPCOUNT displays the number of messages sent, number of accesses to link hold queue, and other flow control data.

FCDISPCOUNT	starting_address ending_address

Where:

starting_address is the starting data link number

ending_address is the ending data link number

Example:

```
SP:FCDATA>
F 0 1
link #  0
assigned time period 0
time period start timer 0
allowable msgs per time period 0
# of msgs sent in time period 0
# of accesses to link hold q
# of msgs remaining in link hold q 0
# flow ctrl activate 0
link #  1
assigned time period 5
time period start timer 5442
allowable msgs per time period 2
# of msgs per time period 1
# of accesses to link hold q 57
# of msgs remaining in link hold q 0
# flow control activate 45
flow control installed
flow control enabled
```

CFCCOUNTS command

The CFCCOUNTS command clears the number of times flow control has been activated.

CFCCOUNTS	starting_address ending_address

Where:

starting_address is the starting data link number

ending_address is the ending data link number

Example:

```
SP:Fcdata>
c 4  5
SP:Fcdata>
```

ENABLE command

The ENABLE command activates flow control on selected data links.

ENABLE	starting_address	ending_address

Where:

starting_address is starting data link number

ending_address is ending data link number

Example:

```
SP:Fcdata>
E 4 5
SP:Fcdata>
```

KILLFLOW command

The KILLFLOW command deactivates flow control on selected data links.

KILLFLOW	starting_address	ending_address

Where:

starting_address is starting data link number

ending_address is ending data link number

Example:

```
SP:Fcdata>
k 4 5
```

TPSET

The TPSET command sets the time period on selected data links.

TPSET	starting_address	ending_address

Where:

starting address is the starting data link number

ending address is the ending data link number

Example:

```
SP:Fcdata>
t 3 4
Enter the time period (1-255) for link :3

SP:Tpset>
200
Enter the time period (1-255) for link :4

SP:Tpset>
12
SP:Fcdata>
```

MCSET command

The MCSET command sets the throttle count threshold to start flow control.

MCSET	starting_address ending_address

Where:

starting_address is the starting data link number.

ending_address is the ending data link number.

Example:

```
SP:Fcdata>
m 3 4
Enter the msg count number (1-255) for link :3
SP:Mcset>
12
Enter the msg count number (1-255) for link :4
13
SP:Fcdata>
```

LOSTMSGCNT level

The LOSTMSGCNT level keeps track of which kind of messaging errors have occurred.

Lostmsgcnt	

Following are descriptions of the commands available at the LOSTMSGCNT level:

DISPLAY command

The DISPLAY command displays the number of times various messaging errors occurred when trying to transmit/receive messages.

Display	

Example:

```
DISPLAY>
REASON FOR ERROR      OCCURRENCES (dec)

Link is corrupt              0
Message send fail            0
Cant get buffer              7
Queue overflow               0
Set msg while inact          0
Cant resolve address         3
Mtc link closed              0
Data link closed             0
No route found               10
Bad rebounded msg            1
```

CLEAR command

The CLEAR command clears the sum totals of all messaging errors to zero.

CLEAR	

Note: When you select the CLEAR option, all sums are cleared, but nothing is displayed on the screen.

NETLAYER level

The NETLAYER level provides access to the network layer.

NETLAYER	

Following are descriptions of the commands available at the NETLAYER level:

CRTNETENT command

You can create a new network entity by using the CRTNETENT command. You are prompted for the necessary information.

CRTNETENT	

Note: A network link entity is a term representing the combination of internal node number, unit number, and data link number used to form the unique network address of a network member. Each peripheral off an XPM is identified by these three elements.

For example, an LGC has at least 3 nodes identified:

0 - CC
1 - itself
2 - mate unit.

Additional nodes are added sequentially to the list, up to a maximum of 200. The definition of an entity also depends on the type of node being addressed.

The unit number can be from 0 through 4, depending on whether the node is a single unit node, a hot standby node, or a load-sharing node.

DSTRYNETENT command

You can destroy a network entity with the DSTRYNETENT command.

DSTRYNETENT	

CRTNETLINKE command

You can create a new network link entity with the CRTNETLINKE command. You are prompted for the necessary information.

CRTNETLINKE	

DSTRYNETLIN command

You can destroy a network link entity with the DSTRYNETLIN command.

DSTRYNETLIN	

CHANGEDLG command

You can change the network entity data link group with the CHANGEDLG command.

CHANGEDLG	

ALTERSEND command

You can change the send status of a network link with the ALTERSEND command. You are prompted for the necessary information.

ALTERSEND	

NETLINKADDR command

The NETLINKADDR command displays the data link address, data link group, link status, and whether the silly CIDs table is used for a particular link number. You are prompted for the necessary information.

NETLINKADDR	

NETADDR command

The NETADDR command displays the data link information for a particular node and unit number.

NETADDR	

data linkADD command

The data linkADD command displays the data link group to which a specified data link belongs. You are prompted for a data link address.

data linkADD	

data linkGRO command

The data linkGRO command displays the open data links for a particular data link group. You are prompted for a data link group number.

data linkGRO	

ALLOCATEDDA command

The ALLOCATEDDA command displays which data link groups have been allocated.

ALLOCATEDDA	

SILLYCIDSTA command

The SILLYCIDSTA command displays the CIDs that are used in the silly CIDs table. These CIDs are used to route a message internally.

SILLYCIDSTA	

INCOMINGMSG command

The INCOMINGMSG command has not been implemented.

INCOMINGMSG	

NOROUTESWER command

The NOROUTESWER command enables or disables the output of the no route SWERR. The default is disabled.

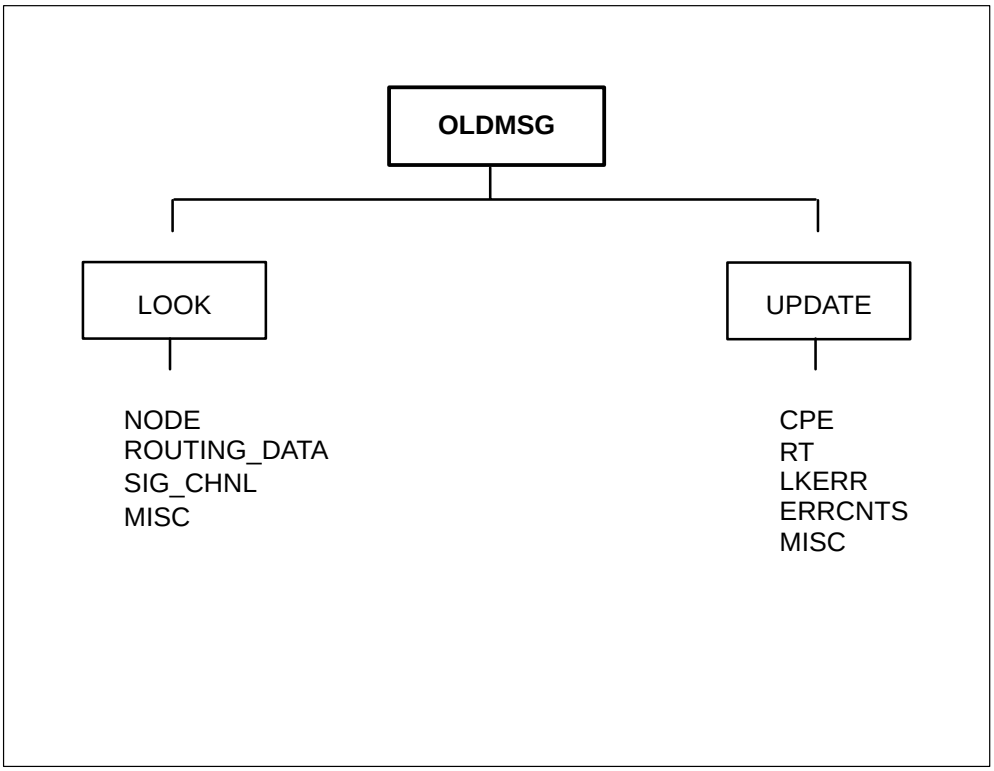
NOROUTESWERR	

OLDMSG level

The OLDMSG level contains two commands for the old messaging system. Figure 5-7 on page 5-48 contains the commands available at the OLDMSG level.

OLDMSG	

Figure 5-7
Commands in the OLDMSG level



Following are descriptions of the commands available at the OLDMSG level:

LOOK command

The LOOK command provides access to old messaging system information.

LOOK	

Following are the commands available at the LOOK level:

NODE displays the node data, such as the node number, number of terminals, start terminal, node type, and message channel index. You are prompted for the internal node number.

ROUTING_DATA displays the routing data. You are prompted for the internal node number.

SIG_CHNL is not implemented.

MISC is not implemented.

UPDATE command

You can change old messaging information using the UPDATE command.

UPDATE	

Following are the commands available at the UPDATE level:

CPE updates CPE information. You are prompted for the internal node number.

RT updates the routing data. You are prompted for the internal node number.

LKERR NONE

ERRCNTS NONE

MISC NONE

Example:

```
Adminlogs,Dldata,Netlayer,Oldmsg.
SP:Newmsging>

>>>o

Look,Update.
SP:Oldmsg>
```

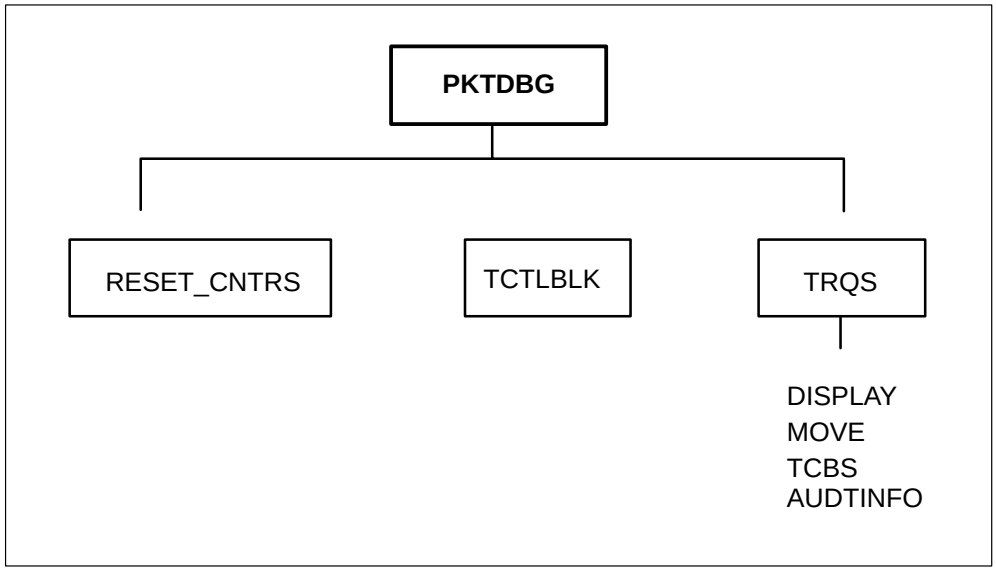
```
>>>l

Node,Routing_data,Sig_chnl,Misc *
SP:Look>

>>>n

Int, node (0-node_total)?
SP:Look>

>>>0

CPE = 0
Rt  = 0
Lke = 3

Node,Routing_data,Sig_chnl,Misc *
SP:Look>

>>>*

Look,Update.
SP:Oldmsg>

>>>u

Cpe, Rt,Lkerr,Errcnts, Misc, *
SP:Update>

>>>r

int_node (1-node_total)?
SP:Update>

>>>1

Rt (0_3)?
SP:Update>

>>>2

O.K.

Cpe, Rt,Lkerr,Errcnts, Misc, *
SP:Update>
```

The PKTDBG level

The PKTDBG level is accessed when the PKTDBG command is entered at the LTCSP level. The PKTDBG level contains commands for manipulating the packet messaging control blocks.

Figure 5-8 on page 5-51 contains the commands available at the PKTDBG level.

Figure 5-8
Commands in the PKTDBG level



Following are descriptions of the commands available at the PKTDBG level:

RESET_CNTRS command

The RESET_CNTRS command clears the audit information counters.

RESET_CNTRS	

TCTLBLK command

The TCTLBLK command displays a specific transaction control block entry. You are prompted for the transaction control block number.

TCTLBLK	

TRQS command

The TRQS level contains commands for displaying the queues for the transaction control blocks.

TRQS	

Following are the commands available at the TRQS level:

DISPLAY

The DISPLAY command displays all items on a specific queue. You are prompted for the queue number.

DISPLAY	

MOVE

The MOVE command moves item from one queue to another. You are prompted for the relevant information.

MOVE	

TCBS

The TCBS command dumps all queue information. You are prompted for the queue number.

TCBS	

AUDTINFO command

The AUDTINFO command displays audit information, including the number of messages sent and received, as well as the number of transactions audited.

AUDTINFO	

SYNC level

The synchronization (SYNC) level is accessed when the SYNC command is entered at the LTCSP level. The SYNC level contains commands to query synchronization control data, to resynchronize the XPM unit, and to force the unit to synchronize to a specific synchronization source.

CAUTION
The SYNC level is not recommended for use on a live switch.
Users of this level should have a thorough knowledge of XPM synchronization. Improper use of the commands at the SYNC level could result in service interruption.

Synchronization

Every node in the DMS must synchronize to the network clock to provide reliable communication in the DMS system. Therefore, transmitting messages and speech pcm between the network and an XPM depends on the integrity of the 125 µs frame pulse that the network and the XPM are using.

The XPM formatter card (6X41) (or 6X72 for RCC) provides the 125 µs internal pulse for the XPM. This pulse may not coincide with the framing pulse generated by the network. Therefore, the XPM must align the internal pulse with the network framing pulse.

The XPM has phase comparators that detect any differences between the internal pulse and the network framing pulse. The values detected by the phase comparators are provided to the digital-to-analog (D/A) converter, which modifies the XPM clock source to coincide with the network framing pulse.

An XPM has four phase comparators. In a host XPM (such as an LGC, DTC or SCM-100) the phase comparators are used as follows:

- one for each of the network planes
- one mate comparator, to synchronize the inactive unit frame pulse with the mate

- one mate superframe comparator, to align the superframe frame pulses between the inactive and active units of a peripheral

For remote peripherals, such as an RCC which synchronizes from a host peripheral, only one phase comparator is used to synchronize to the network. In a dual RCC configuration, a second network phase comparator is used to synchronize from one RCC to the other.

Types of synchronization

Three types of synchronization exist:

- link frame synchronization
- mate synchronization
- spouse synchronization (for RCC)

  Note: The spouse RCC is the second RCC in a dual RCC configuration. Each RCC is the spouse of the other RCC.

- superframe synchronization

Link frame synchronization

Link frame synchronization, also called frame synchronization or network synchronization, involves the alignment of the XPM pulse to the network framing pulse. All XPM units that are CP out-of-service, with the exception of RCCs, should be in or attempting to achieve link frame synchronization. RCCs can run free.

Mate synchronization

Peripherals, including trunks, downline from an XPM synchronize themselves to the pulse of the XPM. Differences in network ports and cable length cause the XPM to detect a lag between the network framing pulses.

To prevent the downline peripherals from detecting the lag after a SwAct, inactive in-service XPM units run in sync with the active side (mate) view of the network. After a SwAct, the newly active unit temporarily drops sync to the mate unit and try to gain synchronization from the network. This causes the synchronization transition to take place over a number of synchronization intervals, rather than one large lag. The newly inactive unit then uses the mate phase comparator to synchronize off the newly active unit.

Superframe synchronization

Peripherals downline from the XPM expect a constant framing pulse and a superframe pulse every 240 frames. The inactive unit uses the mate superframe phase comparator to compare the superframe it generates with the superframe generated by the active unit.

As a result, mate synchronization involves the following:

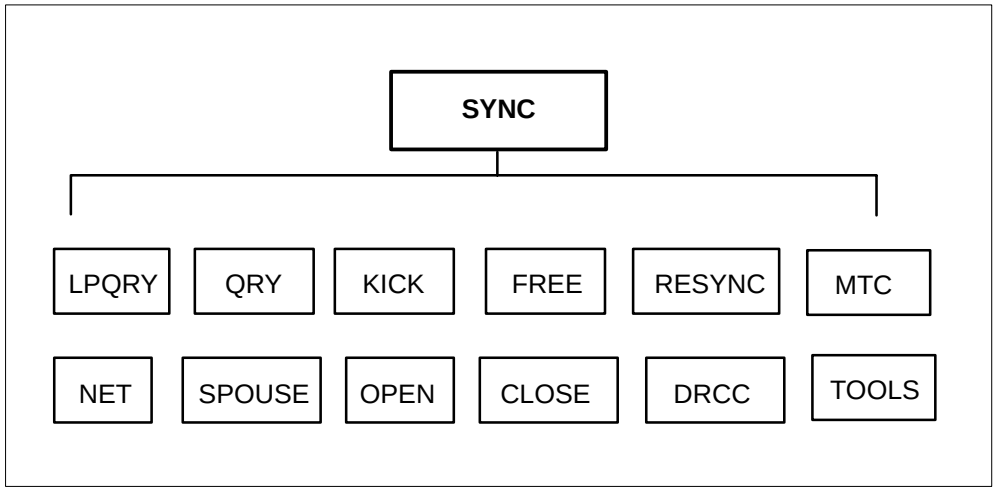
- shifting the inactive unit clock until the superframe of the active and inactive units are aligned in one frame
- aligning the 125 µs frame pulse of the inactive unit to the frame pulse of the active mate

Mate and superframe synchronization are required for the inactive in-service units only. The active XPM unit synchronizes to one of the network planes.

Commands at the SYNC level

Figure 5-9 on page 5-55 contains the commands available at the SYNC level.

Figure 5-9
Commands in the SYNC level



Following are descriptions of the commands available at the SYNC level:

CLOSE command

The CLOSE command causes a synchronization source to be unavailable for synchronizing. You are prompted for a link number, from 0 to 3.

CLOSE	

DRCC command

The DRCC command causes an RCC to operate as though it is part of a dual RCC configuration. This command is only effective on an RCC.

DRCC	

FREE command

The FREE command causes the unit to enter free running mode. Free running mode indicates that the unit ignores information provided by the phase comparators and uses only information prior to free running mode. This mode is useful on peripherals with Emergency Stand Alone (ESA) capability.

Only peripherals with ESA capability are allowed to run free without the help of the XPM monitor.

FREE	

Example:

```
L)pqry,Q)ry,K)ick,R)esync,M)tcsync,N)et,S)pouse,O)pen,
C)lose,F)ree,D(rcc, *
SP:SYnc>

>>>f

Entering free running mode.
```

KICK command

The KICK command causes a change in the D/A converter output to initiate a change in the phase comparators.

KICK	

LQRY command

The LQRY command displays synchronization control data every half second. Every 160ms synchronization change is not displayed.

Data continues to be displayed until you press Enter.

LQRY	

Example:

```
L)pqry,Q)ry,K)ick,R)esync,M)tcsync,N)et,S)pouse,O)pen,
C)lose,F)ree,D(rcc, *
SP:SYnc>

>>>l
```

```
MODE  FLG   PC-0   PC-1   PC-M   PC_S   INS   TIMR    d2A  ACC
net0    T   04E4   04E4   0002   FF10   OOI   0000   0C57  00004AFF 1E2
net0    T   04E4   04E5   0002   FF10   OOI   0000   0C58  00004B00 1ED
net0    T   04E4   04E4   0002   FF10   OOI   0000   0C57  00004AFF 1F9
net0    T   04E4   04E5   0001   FF10   OOI   0000   0C58  00004B00 204
net0    T   04E4   04E4   0002   FF10   OOI   0000   0C57  00004AFF 210
net0    T   04E4   04E5   0002   FF10   OOI   0000   0C58  00004B00 21B
net0    T   04E4   04E4   0001   FF10   OOI   0000   0C57  00004AFF 227
net0    T   04E4   04E5   0002   FF10   OOI   0000   0C57  00004AFF 232
net0    T   04E4   04E4   0001   FF10   OOI   0000   0C58  00004AFF 23E
net0    T   04E4   04E4   0002   FF10   OOI   0000   0C57  00004AFF 249
net0    T   04E4   04E4   0001   FF10   OOI   0000   0C58  00004B00 255
net0    T   04E4   04E5   0002   FF10   OOI   0000   0C57  00004AFF 261

L)pqry,Q)ry,K)ick,R)esync,M)tcsync,N)et,S)pouse,O)pen,
C)lose,F)ree,D(rcc, *
SP:SYnc>
```

MTC command

You can force the unit to resynchronize to a specific phase comparator with the MTCRSYNC command. You are prompted for the mode type.

If one of the non-maintenance modes is chosen, the software exits the non-maintenance mode if the link goes insane. If a maintenance mode is chosen, the software stays in the maintenance mode until you instruct the software to exit the mode.

MTC	

The mode types are as follows:

0 synchronize to network plane 0

1 synchronize to network plane 1

2 synchronize to the mate

3 synchronize to the spouse

4 run free

5 maintenance -- synchronize to network 0

6 maintenance -- synchronize to network 1

7 maintenance -- synchronize to the mate

8 maintenance -- synchronize to the spouse

Example:

```
L)pqry,Q)ry,K)ick,R)esync,M)tcrsync,N)et,S)pouse,O)pen,
C)lose,F)ree,D(rcc, *
SP:SYnc>

>>>m

Resyncing to a mode or your choice.
0=net0,1-net1,2=mate,3=spouse,4=free,5=mtc0,6=mtc1,7=mtcmate,
8=mtcspouse
ENTER MODE ( 0..8 )
SP:SYnc>

>>>2

L)pqry,Q)ry,K)ick,R)esync,M)tcrsync,N)et,S)pouse,O)pen,
C)lose,F)ree,D(rcc, *
SP:SYnc>
```

NET command

The NET command allows synchronizing to the network. This command clears the link indicators and sets the synchronization mode back to the preferred link.

NET	

SPOUSE command

The SPOUSE command synchronizes the RCC to the spouse RCC. The spouse RCC can be synchronizing to the network or free-running. This command is only effective on an RCC.

SPOUSE	

OPEN command

The OPEN command causes a synchronization source to be available for synchronizing. You are prompted for the link number, from 0 to 3.

OPEN	

QRY command

The QRY command displays the synchronization control data.

QRY	

The QRY display is as follows:

```
MODE  FLG   PC-0  PC-1  PC-M  PC_S  INS   TIMR  d2A  ACC
net0   T    04E4  04E4  0002  FF10  OOI   0000  0C58 00004B001C2
```

The fields are as follows:

MODE the synchronization source, which can be one of the following:

- net0
- net1
- mate
- free
- mtc0
- mtc1
- mtc_mate

FLG indicates whether the unit is in sync. Values are as follows:

- T - true
- F - false

PC-0 phase comparator 0

PC-1 phase comparator 1

SPSE for dual RCC only; displays the four synchronization source statuses

PC-M mate phase comparator

PC-S mate superframe phase comparator

IND is the link status indicator, which can be one of the following:

- O - open and operational
- H - hit occurred in last reading

- I - link insane
- S - phase comparator stuck at an insane or sane value
- C - link closed

TIMR timer for measuring the amount of time (in 160 ms units) a unit takes to sync to the present source. The value is 0 when synchronization is achieved.

D2A the value given to the D/A converter

ACC the value in the software accumulator. This value is used in an algorithm to calculate the value that will be written to the D/A converter

(untitled column)
displays the amount of time (in 160 ms units) since the software kick was applied to the digital-to-analog converter. The kick is applied at 0. After #320 readings (160 ms), the kick is applied again.

RESYNC command

The RESYNC command causes synchronization to return to the original mode.

RESYNC	

Example:

```
L)pqry,Q)ry,K)ick,R)esync,M)tcrsync,N)et,S)pouse,O)pen,
C)lose,F)ree,D(rcc, *
SP:SYnc>

>>>n

resyncing to network
```

TOOLS level

The TOOLS command

Tools	

The following commands are available in the TOOLS level:

ACC

The ACC command

Acc	

STICK

The STICK command

Stick	

INSANE

The INSANE command

Insane	

LPSIZE

The LPSIZE command

Lpsize	

VALUE

The VALUE command

Value	

KICK

The KICK command

Kick	

INIT

The INIT command

INit	

DELAY

The DELAY command

Delay	

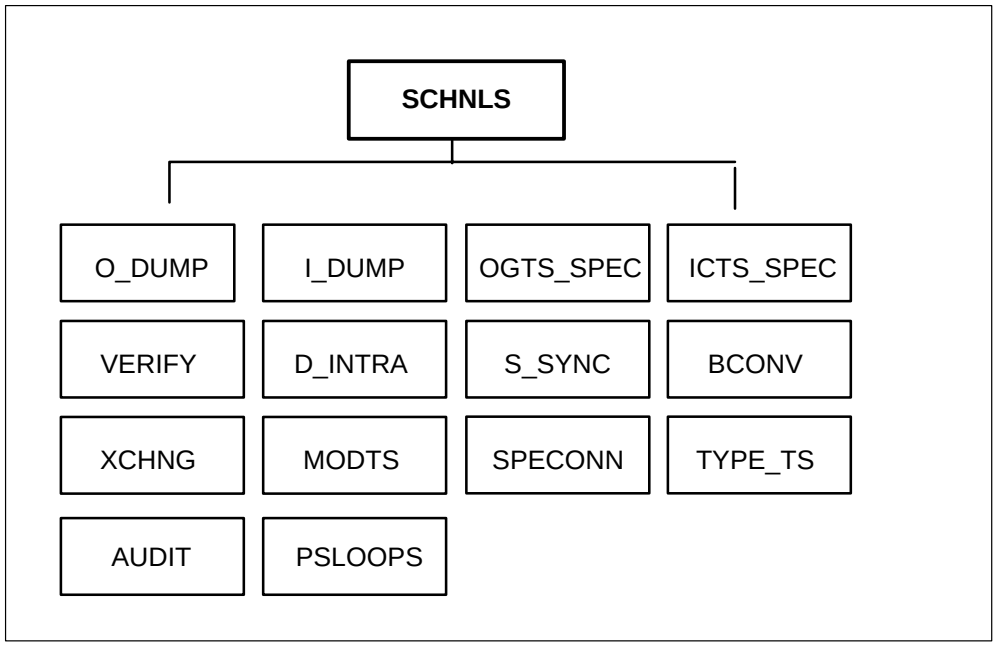
SCHNLS level

The signaling processor channels (SCHNLS) level is accessed when the SCHNLS command is processed from the LTCSP level.

This level contains commands that allow you to display information about channels associated with the XPM.

Figure 5-10 on page 5-62 contains the commands available at the SCHNLS level.

Figure 5-10
Commands in the SCHNLS level



Following are descriptions of the commands available at the SCHNLS level:

S_SYNC command

The S_SYNC command indicates whether the XPM is in superframe sync and whether the unit being tested is inactive.

S_SYNC	

Example:

```
O_dump,I_dump,oGts_spec,iCts_spec,Verify,D_intra,S_sync,
Bconv,Xchng,modTs,*
SP:SChnls>
s
module_activity NOT inactive
NOT in super_frame sync

O_dump,I_dump,oGts_spec,iCts_spec,Verify,D_intra,S_sync,
Bconv,Xchng,modTs,*
SP:SChnls>
```

BCONV command

The BCONV command displays the A to MU law conversion.

BCONV	

O_DUMP command

The O_DUMP command dumps outgoing (toward the P-side of the XPM) time switch connection memory. The display indicates how the outgoing time slot is connected to the incoming time slot in the time switch.

O_DUMP	

Example:

```
O_dump,I_dump,oGts_spec,iCts_spec,Verify,D_intra,S_sync,
Bconv,Xchng,modTs,*
SP:SChnls>

>o

OUT GOING TIME SWITCH CONNECTION MEMORY
Index(hex)  conn_mem(hex)
000     0000  0001  0002  0003  0004  0005  0006  0007  0008  0009
00A     000A  000B  000C  000D  000E  000F  0010  0011  0012  0013
014     0026  003A  0024  0024  004E  0062  0024  0024  0024  0024
01E     0024  0024  0024  0024  0024  0024  0024  0024  0024  0024
028     0024  0024  0024  0024  0024  0024  0024  0024  0024  0024
032     0024  0024  0024  0024  0024  0024  0024  0024  0024  0024
03C     0024  0024  0024  0024  0024  0024  0024  0024  0024  0024
046     0024  0024  0024  0024  0024  0024  0024  0024  0024  0024
050     0024  0024  0024  0024  0024  0024  0024  0024  0024  0024
```

05A	0024	0024	0024	0024	0024	0024	0024	0024	0024	0024
064	0024	0024	0024	0024	0024	0024	0024	0024	0024	0024
06E	0024	0024	0024	0024	0024	0024	0024	0024	0024	0024
078	0024	0024	0024	0024	0024	0024	0024	0024	0024	0024
082	0024	0024	0024	0024	0024	0024	0024	0024	0024	0024
08C	0024	0024	0024	0024	0024	0024	0024	0024	0024	0024
096	0024	0024	0024	0024	0024	0024	0024	0024	0024	0024
0A0	0024	0024	0024	0024	0024	0024	0024	0024	0024	0024
0AA	0024	0024	0024	0024	0024	0024	0024	0024	0024	0024
0B4	0024	0024	0024	0024	0024	0024	0024	0024	0024	0024
0BE	0024	0024	0024	0024	0024	0024	0024	0024	0024	0024
0C8	0024	0024	0024	0024	0024	0024	0024	0024	0024	0024
0D2	0024	0024	0024	0024	0024	0024	0024	0024	0024	0024
0DC	0024	0024	0024	0024	0024	0024	0024	0024	0024	0024
0E6	0024	0024	0024	0024	0024	0024	0024	0024	0024	0024
0F0	0024	0024	0024	0024	0024	0024	0024	0024	0024	0024
0FA	0024	0024	0024	0024	0024	0024	0024	0024	0024	0024
104	0024	0024	0024	0024	0024	0024	0024	0024	0024	0024
10E	0024	0024	0024	0024	0024	0024	0024	0024	0024	0024
118	0024	0024	0024	0024	0024	0024	0024	0024	0024	0024
122	0024	0024	0024	0024	0024	0024	0024	0024	0024	0024
12C	0024	0024	0024	0024	0024	0024	0024	0024	0024	0024
136	0024	0024	0024	0024	0024	0024	0024	0024	0024	0024
140	0027	0027	0027	0027	0027	0027	0027	0027	0024	0024
14A	0024	0024	0024	0024	0024	0024	0024	0024	0024	0024
154	0024	0024	0024	0024	0024	0024	0024	0024	0024	0024
15E	0024	0024	0024	0024	0024	0024	0024	0024	0024	0024
168	0024	0024	0024	0024	0024	0024	0024	0024	0024	0024
172	0024	0024	0024	0024	0024	0024	0024	0024	0024	0024
17C	0024	0024	0024	0024	0024	0024	0024	0024	0024	0024
186	0024	0024	0024	0024	0024	0024	0024	0024	0024	0024
190	0024	0024	0024	0024	0024	0024	0024	0024	0024	0024
19A	0024	0024	0024	0024	0024	0024	0024	0024	0024	0024
1A4	0024	0024	0024	0024	0024	0024	0024	0024	0024	0024
1AE	0024	0024	0024	0024	0024	0024	0024	0024	0024	0024
1B8	0024	0024	0024	0024	0024	0024	0024	0024	0024	0024
1C2	0024	0024	0024	0024	0024	0024	0024	0024	0024	0024
1CC	0024	0024	0024	0024	0024	0024	0024	0024	0024	0024
1D6	0024	0024	0024	0024	0024	0024	0024	0024	0024	0024
1E0	0024	0024	0024	0024	0024	0024	0024	0024	0024	0024
1EA	0024	0024	0024	0024	0024	0024	0024	0024	0024	0024
1F4	0024	0024	0024	0024	0024	0024	0024	0024	0024	0024
1FE	0024	0024	0024	0024	0024	0024	0024	0024	0024	0024

```
208     0024  0024  0024  0024  0024  0024  0024  0024  0024  0024
212     0024  0024  0024  0024  0024  0024  0024  0024  0024  0024
21C     0024  0024  0024  0024  0024  0024  0024  0024  0024  0024

226     0024  0024  0024  0024  0024  0024  0024  0024  0024  0024
230     0024  0024  0024  0024  0024  0024  0024  0024  0024  0024
23A     0024  0024  0024  0024  0024  0024  0024  0024  0024  0024
244     0024  0024  0024  0024  0024  0024  0024  0024  0024  0024
24E     0024  0024  0024  0024  0024  0024  0024  0024  0024  0024
258     0024  0024  0024  0024  0024  0024  0024  0024  0024  0024
262     0024  0024  0024  0024  0024  0024  0024  0024  0024  0024
26C     0024  0024  0024  0024  0024  0024  0024  0024  0024  0024
276     0024  0024  0024  0024  0024  0024  0024  0024  0024  0024

O_dump,I_dump,oGts_spec,iCts_spec,Verify,D_intra,S_sync,Bconv,
Xchng,modTs,*
```

I_DUMP command

The I_DUMP command dumps incoming (toward the C-side of the XPM) time switch connection memory. The display indicates how the incoming time slot is connected to the outgoing time slot in the time switch.

I_DUMP	

Example:

```
O_dump,I_dump,oGts_spec,iCts_spec,Verify,D_intra,S_sync,
Bconv,Xchng,modTs,*
SP:SChnls>

>i

IN COMING TIME SWITCH CONNECTION MEMORY
Index(hex)  conn_mem(hex)
000     0268  0269  026A  026B  026C  026D  026E  026F  0270  0271
00A     0272  0273  0274  0275  0276  0277  0278  0279  0024  0010
014     0028  002A  002C  002E  0030  0010  0034  0014  0038  0015
01E     0015  0015  0040  0010  0044  0046  027A  0119  0022  004E
028     0026  0052  0054  0056  0027  0025  0022  005E  0026  0027
032     0027  0024  0027  006A  006C  006E  027B  0072  0074  0038
03C     0078  003B  007C  003D  0080  0038  0084  0038  003D  0038
046     0036  003D  0090  003D  0094  0096  027E  009A  0126  0126
050     004F  0126  0126  0126  0126  0126  0126  0126  0126  0126
05A     0126  0126  0126  0126  00BC  00BE  027F  00C2  00C4  0065
```

064	00C8	0060	005F	0065	0063	005F	0065	0060	005E	0060
06E	00DC	0063	00E0	0065	00E4	00E6	00E8	00EA	0077	00EE
078	0072	0075	0076	00F6	0079	00FA	0077	00FE	0072	0102
082	0072	0106	0078	010A	010C	010E	0110	0112	0114	008D
08C	0086	0088	011C	008D	0120	008D	0124	0089	0128	0088
096	012C	0088	0130	008B	0134	0136	0138	013A	009A	009D
0A0	009F	0126	009F	0126	009C	0126	009F	0126	009B	0126
0AA	009A	0126	009A	00A1	015C	015E	0160	0162	0126	0126
0B4	0126	0126	0126	0126	0126	0126	0126	00B3	0126	0126
0BE	0126	0126	0126	0126	0184	0186	0188	018A	00C2	018E
0C8	00C2	0192	00C7	0196	00C2	019A	00C6	019E	00C7	01A2
0D2	00C5	00C5	00C2	00C5	01AC	01AE	01B0	01B2	00D7	00D8
0DC	00D7	00D8	01BC	00DD	01C0	00D8	01C4	00DD	01C8	00DD
0E6	01CC	01CE	00D6	00D8	01D4	01D6	01D8	01DA	00EB	00ED
0F0	00EA	0126	00EF	0126	00EA	0126	00EA	0126	00EF	00ED
0FA	00EE	0126	00EB	0126	01FC	01FE	0200	0202	0204	0101
104	0208	0100	020C	0105	0210	0104	0214	0100	0103	0105
10E	00FF	0100	0220	0105	0224	0226	0228	022A	0126	0126
118	0126	0126	0126	0126	0126	0126	0126	0126	0126	0126
122	0126	0126	0116	0126	024C	024E	0250	0252	013B	0141
12C	0258	025A	025C	013C	0260	0141	0264	013E	0268	0141
136	013F	013C	0270	0140	0274	0276	0278	027A	0126	0126
140	0126	0126	0126	0126	0126	0126	0126	0126	0126	0126
14A	0126	0126	0126	0126	0099	0099	0099	0099	0152	0099
154	0153	0099	0151	0099	0153	0099	0099	0099	0153	0099
15E	0153	0150	014E	0150	0099	0099	0099	0099	0099	0164
168	0099	0162	0099	0099	0099	0164	0163	0166	0099	0169
172	0099	0169	0163	0168	0099	0099	0099	0099	0176	0126
17C	0176	0126	0126	0126	0126	0126	0126	0126	0126	0126
186	0126	0126	0126	0126	0099	0099	0099	0099	0126	018C
190	0126	0190	018B	0190	018F	018C	018A	0190	018F	018C
19A	018B	018C	0126	0191	0099	0099	0099	0099	01A3	0099
1A4	019E	0099	019E	0099	019E	01A0	019E	01A5	019E	01A1
1AE	01A2	0099	01A3	0099	0099	0099	0099	0099	0099	01B9
1B8	01B3	01B8	0099	01B4	01B2	01B8	0099	01B4	01B2	01B8
1C2	0099	01B8	0099	01B9	0099	0099	0099	0099	01CB	0099
1CC	01CB	0099	01C6	0099	01C6	01CB	01CA	01CD	01C6	0099
1D6	01CB	0099	01CA	0099	0099	0099	0099	0099	0126	0126
1E0	0126	0126	0126	0126	0126	01DC	0126	0126	0126	0126
1EA	0126	0126	0126	0126	0099	0099	0099	0099	01F2	0099
1F4	01F3	01F1	01F3	0099	01F3	01F1	01F3	0099	01F3	0099
1FE	01EF	0099	01F3	0002	0004	0006	0008	000A	000C	0209
208	0202	0209	0209	0209	0203	0209	001C	0202	0020	0202
212	0024	0205	0028	0209	002C	002E	0030	0032	0216	0219

```
21C   021C 003A 021B 0219 021B 0042 021B 0046 021B 004A
226   021B 004E 021B 0052 0054 0056 0058 005A 022B 022C
230   0231 0126 022E 022C 0126 0231 0126 0231 0126 022C
23A   0126 0230 0126 022B 007C 007E 0080 0082 0126 0126
244   0126 0126 0126 0126 0126 0126 0126 0126 0126 0126
24E   023E 0126 0126 0126 00A4 00A6 00A8 00AA 00AC 0254
258   0252 00B2 00B4 0254 00B8 0258 00BC 0259 00C0 0255
262   00C4 0254 0252 0259 00CC 00CE 00D0 00D2 0010 027C
26C   00D8 00DA 027C 00DE 000E 00E2 000E 00E6 0013 00EA
276   0012 027D 0013 00F2 00F4 00F6 00F8 00FA 0266 0267

O_dump,I_dump,oGts_spec,iCts_spec,Verify,D_intra,S_sync,Bconv,
Xchng,modTs,*
SP:SChnls>
```

OGTS_SPEC command

The OGTS_SPEC command displays the corresponding time switch in the incoming time slot for the specified outgoing time switch.

OGTS_SPEC	**ps_port** **ps_chnl**

Where:

ps_port is the P-side port number, from 0 to 19

ps_chnl is the P-side channel number, from 0 to 31

Example:

```
O_dump,I_dump,oGts_spec,iCts_spec,Verify,D_intra,S_sync,
Bconv,Xchng,modTs,*
SP:SChnls>

>g 2 31

For the OUTGOING time switch:
Given pside port =  2  and pside chnl = 31
The corresponding csport = 16  and cside chnl =  1

O_dump,I_dump,oGts_spec,iCts_spec,Verify,D_intra,S_sync,
Bconv,Xchng,modTs,*
SP:SChnls>
```

ICTS_SPEC command

The ICTS_SPEC command displays the corresponding time switch in the outgoing time slot for the specified incoming time switch.

ICTS_SPEC	cs_port cs_chnl

Where:

cs_port is the C-side port number, from 0 to 19

cs_chnl is the C-side channel number, from 0 to 31

Example:

```
SP:SChnls>

>c 16 1

For the INCOMING time switch:
Given cside port = 16  and cside chnl =  1
The corresponding psport = 14  and pside chnl =  4

O_dump,I_dump,oGts_spec,iCts_spec,Verify,D_intra,S_sync,
Bconv,Xchng,modTs,*
SP:SChnls>
```

D_INTRA command

The D_INTRA command indicates whether a channel is looped for intraswitching.

D_INTRA	chnl

Where:

chnl is the channel number.

Example:

```
O_dump,I_dump,oGts_spec,iCts_spec,Verify,D_intra,S_sync,
Bconv,Xchng,modTs,*
SP:SChnls>

>d 2

chnl NOT looped for intra_switching

O_dump,I_dump,oGts_spec,iCts_spec,Verify,D_intra,S_sync,
Bconv,Xchng,modTs,*
SP:SChnls>
```

VERIFY command

The VERIFY command verifies the time switch connection in both directions of the time switch. If the connection is two-way, then both the actual and the expected values in the time switch should be identical.

VERIFY	**cs_port** **cs_chnl** **ps_port** **ps_chnl**

Where:

cs_port is the C-side port number, from 0 to 19

cs_chnl is the C-side channel number, from 0 to 31

ps_port is the P-side port number, from 0 to 19

ps_chnl is the P-side channel number, from 0 to 31

Example:

Note: Due to size restrictions, the following example has been modified.

```
SP:SChnls>

>v 16 1 14 4
Values may not correspond unless connection are two_way
FOR CSIDE PORT AND CHNL  16   1
                AND FOR PSIDE PORT AND CHNL  14   4
EXPECTED VALUE(INCOMING(hex)) = 0044
                ACTUAL VALUE(INCOMING(hex)) = 0044
EXPECTED VALUE(OUTGOING(hex)) = 0024
                ACTUAL VALUE(OUTGOING(hex)) = 0024
O_dump,I_dump,oGts_spec,iCts_spec,Verify,D_intra,S_sync,
Bconv,Xchng,modTs,*
SP:SChnls>
```

XCHNG command

CAUTION
This command is *not* recommended for use in the field.

The XCHNG command sets the conversion on the signaling channels, such as A to mu law or mu to A law.

XCHNG	

MODTS command

CAUTION

This command should not be used by a customer, unless specifically directed to and guided by a BNR or NT representative.
This command should *not* be used on a live switch.

The MODTS command modifies the outgoing time switch.

MODTS	

Following are the commands available at the MODTS level:

ICTS modifies the incoming time switch. You are prompted for C-side port and channel numbers.

OGTS modifies the outgoing time switch. You are prompted for the P-side port and channel numbers.

SPECONN level

The SPECONN level displays information related to the special connections on the time switch and the 6X69 speech bus connection memory.

SPECONN	

Following are the commands available at the SPECONN level:

VER_SPEC verifies special connections for the P-side port(s) specified by you.

DISP_SPCH_BUS displays information on the 6x69 speech bus.

REMAKE_CON remakes all the special connections on a given port.

TYPE_TS command

The TYPE_TS command displays the peripheral's type of time switch.

TYPE_TS	

AUDIT command

The AUDIT command

Audit	

PSLOOPS command

The PSLOOPS command

Psloops	

UTR level

The universal tone receiver (UTR) level contains commands for displaying information about the UTR card(s) present and the contents of the UTR-SP buffers.

UTR function

One or two UTR cards may reside in the peripheral.

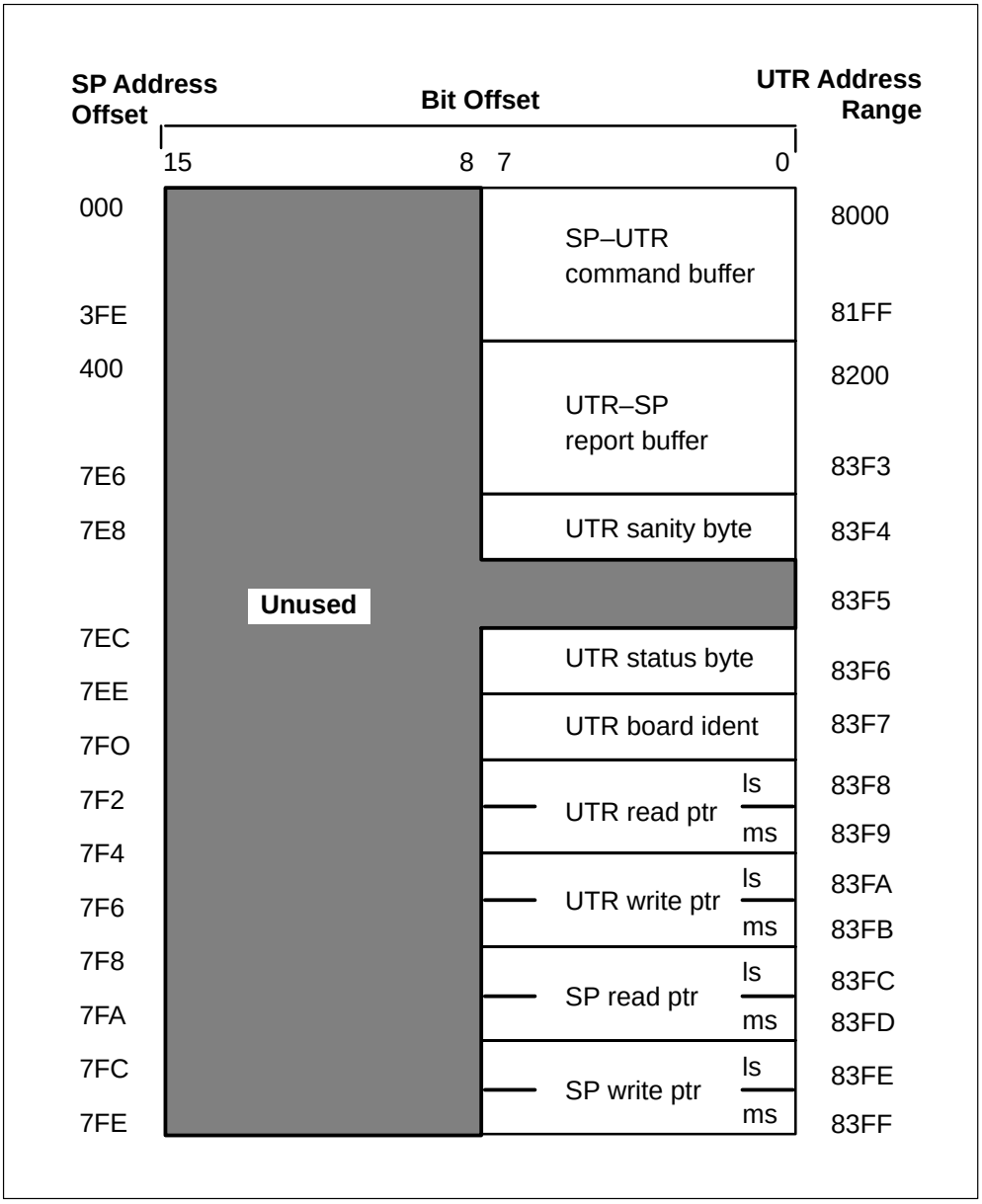
The UTR has a maximum of 32 channels available for digit collection. International XPMs use all 32 channels for digit collection. In North American peripherals, channels 0 and 16 are reserved for messaging and cannot be used for tone detection. Therefore, only 30 channels are available for digit collection.

Channels requiring tone detection are switched by the SP onto the DS30A port occupied by the UTR(s). The UTR samples all the channels on the port. The tones from the channels are sorted and evaluated according to a given set of parameters. Errors and valid tones are reported to the SP through the UTR-SP report buffer.

The SP issues commands to the UTR card through the SP-UTR command buffer. Commands to begin monitoring (specifying the signaling set and the UTR channel), to stop monitoring (specifying the UTR channel), and to request a sanity check are all written to the SP-UTR command buffer.

The SP-UTR interface also contains a status byte, a sanity byte, the firmware version of the card, and the read and write pointers for the command and report buffer. Refer to Figure 5-11 on page 5-73 for a diagram of the UTR-SP interface buffer. The addresses depend on which spare slot the UTR occupies.

Figure 5-11
UTR-SP interface buffer

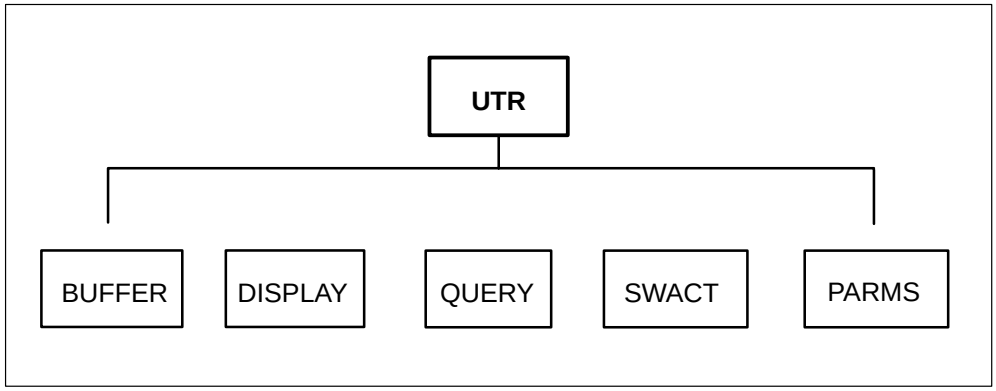


UTR monitor

The UTR level is accessed when the UTR command is entered at the LTCSP level.

You can display information about the UTR-SP interface, such as the command buffer, the report buffer, and the values of the read and write pointers used in the UTR-SP interface with the UTR level.

Figure 5-12
Commands in the UTR level



Following are descriptions of the commands available at the UTR level:

QUERY command

The QUERY command displays the UTR cards present in the unit, the SP card address, and the digit reporting mechanism (each of the card(s) used.

QUERY	

BUFFER command

The BUFFER command prints the specified range of contents of the command or report buffer.

BUFFER	

The contents of the report buffer displayed by the BUFFER command include the following:

Address
the SP address offset, which is calculated from the base UTR address. The base UTR address can be obtained with the QUERY command.

Chan the UTR channel on which the digit was detected (0 to 31 for UTR card 0 and 32 to 63 for UTR card 1)

Sig id contains either the digit reported to the SP for that channel or FF, depending on the message type

Status 1 identifies the error message being reported. Refer to Table 5-1 on page 5-75.

Status 2 is the interdigit timing (IDT) of valid tone on and tone off periods.

> ***Note:*** IDT = Interdigit Timing. Coded in 12 ms intervals, the interdigit timer records a maximum of approximately 3 s. Interdigit timing is reported during a valid tone-on or tone-off message, and the timer is reset upon the report of a valid tone-on. Error messages do not affect IDT.

Table 5-1 on page 5-75 contains the format of the various error messages to the SP and the relationship of the messages to the signal identifier, status 1 and status 2.

Table 5-1
Error message code formats

Message	Signal id	Status 1	Status 2
Tone on	Digit	80	IDT
Tone off	FF	00	IDT
No interdigit pause	Digit	04	IDT
Cycle time error	Digit	05	IDT
Receiver noise high	Digit	06	IDT
Large twist	Digit	07	IDT
CCIS tone off long	FF	02	IDT
Command error	FF	10	00

The contents of the command buffer displayed by the BUFFER command include the following:

Address is the SP address offset calculated from the base UTR address. The base UTR address can be obtained with the QUERY command.

Byte is the number of bytes in the message (command) to be read

Cmnd is the command to be processed by the UTR card, one of the following:

05 start monitor

07 stop channel monitor

09 software reset

0D default parameter change

A0 sanity check

Sig Set is the signal set code, one of the following:

00 DTMF

Note: Dual-Tone Multifrequency (DTMF) is a signaling method employing set combinations of two specific voice-band frequencies, one of which is selected from a group of four low frequencies, and the other from a group of three or four relatively high frequencies.

10 MF

Note: Multifrequency (MF) is a signaling method that uses pairs of standard tones to transmit signaling codes, digit pulsing, and coin-control signals.

20 MF SOCOTEL

Note: SOCOTEL is a signaling system designed for the Spanish telephone system.

30 CMF forward

Note: Compelled Multifrequency (CMF) forward applies to R2 signaling system. CMF forward frequencies are sent by the originating office.

40 CMF backward

Note: Compelled Multifrequency (CMF) backward applies to R2 signaling system. CMF backward frequencies are sent by the terminating office.

50 CCIS

Note: Common Channel Interoffice Signaling (CCIS) is a system for exchanging information between processor-equipped switching systems over a network of switching links. Information is encoded and transmitted over a separate channel using time-division digital techniques.

60 SIT

Note: A machine detectable tone preceding a recorded announcement is called a Special Information Tone (SIT). Such tones alert customers that a machine generated announcement will follow. Thirty-two SIT tones exist, each lasting approximately one second. A decoder detects these tones and generates statistics concerning the number of call connected to a particular announcement.

Chan is the UTR channel number to be monitored (0 to 31 for UTR card 0 and 32 to 63 for UTR card 1).

Parm id is the parameter identifier.

Parm val
is the value associated with the parameter identifier.

Table 5-2 on page 5-77 is a list of the default parameter values for the various signal sets.

Table 5-2
Default parameter values for the command buffer

Parameter	DTMF	MF	MF_SOC	CMF_F	CMF_B	CCIS	SIT
ERR_CNT	00	00	00	00	00	00	06
MIN_POW	78	71	77	9B	9B	78	78
MAX_TWIST	28	20	20	24	24	-	-
P23 VALUE	28	28	28	28	28	28	28
SIG_ON	02	01	01	05	05	02	-
SIG_OFF	01	00	00	00	00	02	-
KP_SPECIAL	-	03	03	-	-	-	-
INT_D_TIM	07	00	00	00	00	00	00
TP OFFSET	FE	40	40	40	40	FE	FE
FRAC TPOW	30	30	30	30	30	30	30

Table 5-2
Default parameter values for the command buffer

Parameter	DTMF	MF	MF_SOC	CMF_F	CMF_B	CCIS	SIT
SIG_L_MIN	-	-	-	-	-	-	14
SIG_H_MIN	-	-	-	-	-	-	1D
SIG_L_MAX	-	-	-	-	-	-	17
SIG_H_MAX	-	-	-	-	-	-	21

DISPLAY command

The DISPLAY command displays the UTR-SP command and report buffer read and write pointers. The command buffer is referenced by a UTR read pointer and an SP write pointer. The report buffer is referenced by an SP read and a UTR write pointer. This command also displays the current value of the status byte.

DISPLAY	

The status byte output may have several values, including:

00
no errors

20
dual port memory full

40
synchronization error

The read and write pointers are displayed with the SP address offset and the UTR addess.

SWACT command

The SWACT command causes a SwAct of the peripheral by changing the status byte to an invalid value.

SWACT	

PARMS command

The PARMS command

Parms	

Example:

```
LTCSP>

>ut

UTR Monitor - Display, Query, Buffer, Swact, *
SP:UTr>

>d

Info For UTR Card Number 0
UTR status byte is #00
```

Pointer	UTR	SP
UTR Read Pointer is	#8190	#0320
UTR Write Pointer is	#83EC	#07D8
SP Red Pointer is	#83EC	#07D8
sp Write Pointer is	#8190	#0320

```
UTR Monitor - Display, Query, Buffer, Swact, *
SP:UTr>

>q

UTR Card 0  is present
 SP Card Address = #00160000
 UTR Reports Tone OFF conditions

UTR Monitor - Display, Query, Buffer, Swact, *
SP:UTr>

>b

Which buffer - Report or Command
SP:UTr>

>r

Card number (0 or 1)
SP:UTr>

>0
```

Number of reports wanted before read pointer (0-60)
SP:UTr>

>0

Number of reports wanted after read pointer (0-60)
SP:UTr>

>2

* Contents of UTR-SP report buffer; Addresses relative to SP *

Address	Chan	Sig Id	Stat 1	Stat 2
->07D8	01	2F	00	0D
07E0	01	11	00	FF
0400	01	12	00	0D

ISDN additions to PMDEBUG

This section describes the ISDN (integrated services digital network) commands available with PMDEBUG.

With the introduction of ISDN to the DMS family, several new levels have been added to PMDEBUG. The ISDN peripheral, ISDN access controller (IAC), is the only XPM that will have these new levels. The IAC also uses existing levels found in the LTC and the MSB.

The new levels that pertain to the IAC are as follows: TIME, TASK, LOAD, XPROMPT, DEBUG, SWERR, IPC, QUEUES, LIPC, USPACE, PATCHES, MSGTR, CHNLS, GDB, RAMFILE, STIF, ISDN, BDMGR, STMT, STTRC, MTC, GDT, DIAGNOSE, AUDIT, CAUDIT, CP, SE, BCM, and ISDNCP.

The GDB and RAMFILE commands will be added to the rest of the XPM peripheral types and the STTRC command may be added to the MSB7 peripheral. These areas will be described in depth in subsequent sections. The last section will describe how to run a DCH continuity test from within PMDEBUG, by way of DIAGNOSE level and the STMT level, in particular how to connect through to any ISDN signalling terminal (DCH or PHI).

GDB (GENDEBUG) LEVEL

You can examine registered variables and send pre-defined message sequences and templates with GDB level. A third function supports the interception of specified CIDs. GDB has two levels: VAR, MSG.

VAR level

The VAR level can be used to inspect important variables, options, or peg counts. These variables are actual variables defined in source code. The VAR level has three subcommands: VARLIST, READ, and EDIT.

VARLIST command

This command displays all variables that are registered with GENDEBUG.

Varlist	

READ command

This command displays the current value of the variable var_name.

Read	**<var_name>**

Where:

var_name the name of a variable that is registered with GENDEBUG. Var_name is a string of a maximum size of 8.

EDIT command

You can change the value of the variable var_name with the EDIT command.

EDIT	**<var_name>**

Where:

var_name is the name of a variable that is registered with GENDEBUG. Var_name is a string of a maximum size of 8.

MSG level

The MSG level may be used to utilize some application specific message templates, some application specific message sequences, or to intercept certain CIDs, which is very useful for workstation testing. The MSG level has the following subcommands:

MSGLIST command

This command displays all message templates and sequences that are registered with GENDEBUG.

Msglist	

SEND command

This command is used to build and send a message template or sequence that is registered with GENDEBUG. You may be prompted for more information (for example, actual message bytes), if the code designed to handle the template or sequence requires it.

Send	**<msg_id>**

Where:

msg_id is the name of a message template or sequence that is registered with GENDEBUG. Msg_id is a string.

VERBOSE command

This command is used to enable/disable verbose dumping of the messaging.

Verbose	

INTERCEPT command

This command is used to enable intercepting of messages with destination equal to com_id.

Intercept	**<com_id>**

Where:

com_id is the value of an IPC communication identifier. Com_id is a byte.

RESTORE command

This command is used to release com_id from intercept mode.

REStore	**<com_id>**

Where:

com_id the value of an IPC communication identifier. Com_id is a BYTE.

RECEIVE command

This command is used to look at intercepted messages.

Receive	

RAMFILE Level

The RAMFILE level is used to set up a wraparound text file in the XPM memory. The file can be used to write debug statements from the code. The file can also be used to capture message trace or primitive/exec trace data. The contents of a ramfile can be cleared, dumped, or browsed by using XPM monitor commands. The RAMFILE monitor level contains the following subcommands:

ALLOCATE command

This command is used to allocate a temporary/permanent RAMFILE of size10xK. A permanent RAMFILE can survive restarts while a temporary one cannot. You may allocate only one RAMFILE at a time.

Allocate	**<file_type>**

Where:

file_type is the kind of ramfile. File_type is of typeTemp or Perm.

DEALLOCATE command

This command is used to deallocate the ramfile.

Deallocate	

ENABLE command

This command is used to enable data capture in the ramfile.

Enable	

DISABLE command

This command is used to disable data capture in the ramfile.

DISable	

TYPE command

This command is used to dump statements written to ramfile by user_id. A nil user-id dumps all messages written to the ramfile.

Type	**<user_id>**

Where:

user_id is the hexadecimal value of a ramfile user. A **<cr>** specifies a nil user.

BROWSE command

This command is used to browse through statements written to ramfile by userid. A nil user-id browses the entire file.

Browse	**<user_id>**

Where:

user_id is the hexadecimal value of a ramfile user. A **<cr>** specifies a nil user.

There are five subcommands within the BROWSE level:

TOP The TOP command positions ramfile at the top.

BOTTOM The BOTTOM command positions ramfile to the bottom.

UP The UP command moves ramfile up one page.

PAGE The PAGE command moves ramfile down one page.

FIND <STR_TO_FIND> The FIND <STR_TO_FIND> command scans ramfile for str_to_find.

LEVEL command

This command is used to restrict data capture to messages that have a level of filter-level and greater.

Level	**<filter_level>**

Where:

filter_level is an integer value used to mask out messages

DUMP command

This command is used to display a hex dump of ramfile from start_offset to end_offset.

DUmp	**<start_offset> <end_offset>**

Where:

start_offset is the start point for dumping. Start_offset must be greater than 0.

end_offset the stop point for dumping. End_offset must be greater than start_offset and less than 32,767.

TEST command

This command is used to write a test pattern to ramfile with userid equal to #0000.

TEst	

ISDN level

Note: In this level and for some other levels, references are made to local_st_id. To determine this value, it is necessary to post the selected ST on the MAP, in the STC level. The MAP displays the ST number (CC view) as well as the STCM and CTRL numbers. The STCM is the STC group ID and the CTRL is the card in the group.

To find the local st_id for PMDEBUG purposes, use this formula:

$$local\_st\_id = (8 \times STCM) + CTRL$$

Local_phi_id is a local_st_id in the range 0 to 7

Local_dch_id is a local_st_id in the range 8 to 79

The ISdn level has three sub_levels: STatmux, Bdmgr, Llm.

STATMUX level

Note: Most of the subcommands in this level must have full syntax at the command input (that is, you are not prompted for missing parameters.)

There are three subcommands in the STatmux level: DIt, PTc, SQf.

DIT command

This command is used to display the IAC ST_table starting at the entry for local_st_id and for num_records of entries. The information that is displayed is the local_st_id and its index into the DCH/PHI statmux table.

You can display ISDN statmux tables, display and clear traffic pegs using the DIT command. There are 5 subcommands in this level:

ST-TBL command

Stbl	**<local_st_id> <num_records>**

Where:

local_st_id is a number 0-79

num_records is a number 0-8

PHI_TBL command

This command is used to display the PHI/DCH statmux table starting at row start_idx and displaying num_rows of entries. Three rows of information are displayed by this command; the triple is owned by a PHI. The top row is a list of LTIDs that uses SAPI16 services through the PHI of interest. The

second row is the DCH that owns the LTID above it. The third row is the logical link on the DCH that belongs to the LTID. To Determine which PHI is represented by the entry, you must use the STTBL command described earlier, and enter the local_phi_id of the PHI as the first parameter. You should then use the index (P-idx) displayed as the start_idx parameter for this command.

Phitbl	**<start_idx> <num_rows>**

Where:

start_idx is a number 0 thru 7

num_rows is a number 0 thru 8

HARDCODE command

CAUTION

This command should *not* be used in a live office situation.

This command replaces the PHI/DCH statmux table with hard-coded testing tables.

Hardcode	

TRAF_PEGS command

This command displays traffic peg counts for SAPI16 services.

Traf_pegs	

CLEAR_PEGS command

This command clears the peg counts for SAPI16 services.

Clear_pegs	

PTC command

CAUTION

This command should *not* be used in a live office situation.

You can make changes to the PHI/DCH statmux table with the PTC command.

PTc	<table_opr> <local_phi_id> <ltid_num> <local_dch_id> <ll>

Where:

table_opr is a number 0-3

local_phi_id is a number 0-7

ltid_num is a number 0-1024

local_dch_id is a number 8-79

ll is a number 0-15

This command performs table_opr on the PHI/DCH statmux table, where:

0 = a NULL operation - no action to table

1 = add a LTID/DCH/LL triple to the PHI

2 = delete a LTID/DCH/LL triple from the PHI

3 = delete all triples from the PHI

SQF command

CAUTION

This command should *not* be used in a live office situation.

This command is used to send Q.921 frames to either the packet handler (by way of the PHI) or the DCH.

SQf	<st_func_id> <local_st_id> <dist_id> <msg_type_or_ltid> <n>

Where:

st_func_id is a number 0-2.

local_st_id is a number 0-7 for st_func_id = 2, and 8-79 for st_func_id = 1.

dist_id is a number 4-5 for st_func_id = 2, and 5 for st_func_id = 1.

msg_type_or_ltid is a number 0..#FF for st_func_id = 2, and 0 for st_func_id = 1.

n is a number 0..#FF for st_func_id = 2, and 0..#15 for st_func_id = 1.

This command performs st_func_id where:

0 = no function

1 = DCH function frame is sent to DCH local_st_id, with a distributor id of dist_id, for LTID msg_type_or_ltid, on logical link 11.

2 = PHI function frame is sent to PHI local_st_id, with a distributor id of dist_id, a message type of msg_type_or_ltid, on logical link 11.

After the command syntax is entered, you are prompted for the data to be the body of the message.

BDMGR level

You can busy, rts, and set internal or external loop-backs on a BD-channel associated with any PHI through the BDMGR level. The level has 4 subcommands as follows.

BDBSY command

You can busy the BD-channel on PHI local_phi_id with the BDBSY command.

bdBsy	<local_phi_id>

Where:

local_phi_id is a number 0-7

BDRTS command

You can RTS the BD-channel on PHI local_phi_id with the BDRTS command. You may wait for a reply from the BD-channel manager or not wait. The wait time is five min.

bdRts	**<local_phi_id> [wait_nowait]**

Where:

local_phi_id is a number 0-7.

wait_nowait wait or nowait the default is wait.

BDINTLP

CAUTION

This command should *not* be used in a live office situation.

Ps_port and chnl are the actual time switch port and channel for details refer to DIS doc FD6X50AA. You can set a loop-back toward the DMS on the DS1 card (NT6X50AB) with the BDINTLP command. The ps_port should be either 16 or 17, and the chnl is the DS1 channel that the bd-channel is nailed to (refer to table SPECCONN in the CC for this information).

bdIntlp	**<local_phi_id> <ps_port> <chnl>**

Where:

local_phi_id is a number 0-7

ps_port is a number 16-17

chnl is a number 0-31

BDEXTLP

CAUTION

This command should *not* be used in a live office situation.

Ps_port and chnl are the actual TIMESWITCH port and channel (for details refer to DIS doc FD6X50AA). You can set a loopback toward the DPN on the DS1 card (6X50AB) with the BDEXTLP command. The ps_port should be either 16 or 17, and the chnl is the DS1 channel the bd-channel is nailed to (refer to the DPN datafill for this info).

bdExtlp	**<local_phi_id> <ps_port> <chnl>**

Where:

local_phi_id is a number 0-7

ps_port is a number 16-17

chnl is a number 0-31

LIM level

You can perform actions on the logical link area of the IAC with the LM level. These actions include link manipulation (establishing and releasing links), display link status, TEI check and restore, display and update of the DLCI table and layer 2 configuration parameters. There are 12 subcommands in this level:

DISPLAY command

You can display the following with the DISPLAY command:

- Lm - the link management state of the ST local_dch_id
- lSc - the state of SAPI 0 logical links on ST local_dch_id
- All - both sets of information.

Display	**<local_dch_id> <set_op>**

Where:

local_dch_id is a number 0-79/12790

disp_opt is Lm, lSc, <logical_link>, and All

logical_link is a number 0-1 Ìor A for all links

SET command

CAUTION

This command should *not* be used in a live office situation.

Set	**<local_dch_id> <set_op>**

Where:

local_dch_id is a number 0-79/1279

set_opt is Lm <new_state> <new_req_state> lSc <ll> <new_ll_state> <fl_code> <mtc>

new_state is Oos, Rst, Cfg, Trf, Dlc, Ins

new_req_state is Avl, Strt, Upd, Est, Rls, Chk, resT

ll a number 0..11

new_ll_state is Und, Rls, Dat, Aes, Est, rdY, arL, Tei

fl_code is Ok, Flt, Trn, Dwn

Mtc is Yes, No

This command is used to change the states of Lm and lSc.

Lm the link management state of the ST local_dch_id. New_state will be the updated state of the DCH, the possible inputs are:

- Oos - out of service
- Rst - reset sent
- Cfg - configuration parameters sent
- Trf - enter traffic request sent
- Dlc - DLCI data sent
- Ins - in-service

New_req_state will be the updated request state of the DCH; the possible inputs are as follows:

- Avl - DCH is available to receive requests

- Strt - Rts request sent
- Upd - data update request sent
- Est - link establish request sent
- Rls - link release request sent
- Chk - TEI check request sent
- resT - TEI restore request sent

lSc changes the state of SAPI 0 logical link <ll> on the ST local_dch_id. New_ll_state is the updated state of the logical link, the possible inputs are as follows:

- Und - undefined link
- Rls - link is released
- Dat - link is datafilled
- Aes - link is awaiting establishment
- Est - link is established
- rdY - link is ready for data transfer
- arL - link is awaiting release
- Tei - TEI has been removed

Fl_code will be the updated reason code for link failure; the possible inputs are as follows:

- Ok - no failure
- Flt - link fail received from DCH
- Trn - SABME received from DCH (transient failure)
- Dwn - hard failure of logical link

Note: Mtc will be the new condition of the maintenance flag for the logical link. If the flag is on the logical link audit will not try to reestablish a link that is down. The possible inputs are: YES to enable the flag or NO to disable the flag.

CLEAR command

You can initialize the following using the CLEAR command:

- Lm - the link management state for ST local_dch_id
- lSc - the logical link records for SAPI 0 logical links on ST local_dch_id
- All - both sets of records

Clear	**<local_dch_id> <clear_op> <logical_link>**

Where:

local_dch_id is a number 0-79/1279

clear_opt is Lm, lSc <logical_link>, and All

logical_link a number 0-1, or A for all links.

ESTLL command

You can establish SAPI0 logical links on ST local_dch_id using the ESTLL command. Selecting All causes attempts to be made on all links on that ST or you may specify a link number.

EstlI	**<local_dch_id> <logical_link>**

Where:

local_dch_id a number 0-79/1279

logical link a number 0-1, or A or all links

RLSLL command

You can release SAPI0 logical links on ST local_dch_id with the RLSLL command. Selecting All will cause all logical links to be released, or you may select a link number.

RlsII	**<local_dch_id> <logical_link>**

Where:

local_dch_id a number 0-79/1279.

logical_link a number 0-1, or A for all links.

Note: The links being released will have a maintenance flag raised against them and the link audit will NOT try to reestablish them. Also, the ISDN terminal on this logical link will not be able to establish the link.

TEIRES command

You can attempt to restore the TEI tei_val on the ST local_dch_id with the TEIRES command.

Teires	**<local_dch_id> <tei_val>**

Where:

local_dch_id is a number 0-79/1279

tei_val a number 1-63

TEICHK command

You can perform TEI check on the ST local_dch_id with the TEICHK command.

TEIChk	<local_dch_id>

Where:

local_dch_id is a number 0-79/1279

BSY command

CAUTION

This command should *not* be used in a live office situation.

You can busy out the specified DCH_D-channel with the BSY command. It subsequently releases any in-service logical links. This command applies to non-IACs and DCHs only.

Bsy	<local_dch_id>

Where:

local_dch_id is a number 0-1279

RTS command

CAUTION

This command should *not* be used in a live office situation.

You can return to service the specified DCH D-Channel with the RTS command. It subsequently establishes any datafilled logical links. This command applies to non-IAC-DCH only.

RTs	<local_dch_id>

Where:

local_dch_id is a number 0-1279

DCNFG command

You can display the DCH configuration data for layer 2 with the DCNFG command. It applies to IAC-DCH only.

DCnfg	

SCNFG command

You can update the DCH configuration for layer 2 with the SCNFG command. It applies to IAC-DCH only.

SCnfg	<tn1><tn2><tn3><tn4><n200><SO_n201><S16_n201><k_val

Where:

n1 is a number 50-250

n2 is a number 50-250

n3 is a number 50-250

n4 is a number 500-3000

n200 is a number 1-7

s0_n201 is a number 10-54

s16_n201 is a number 10-260

k_val is a number 1-3

The ranges and meanings of the fields are shown in Table 6-1 on page 6-16 as follows:

Table 6-1
SCNFG fields

Field	Meaning	Range	Units
tn1	timer for frame re-Xmit	50...250	20ms
tn2	timer for TEI check	50...250	20ms
tn3	timer for TEI request (not used by network)	50...250	20ms

Table 6-1
SCNFG fields

Field	Meaning	Range	Units
tn4	link sanity timer	500...3000	20ms
n200	maximum re-Xmissions	1...7	frames
s0_n201	size of I-field in SAPI 0 frame	10...54	bytes
s16_n201	size of I-field in SAPI 16 frame	10...260	bytes
k_val	outstanding Xmit frame count	1...3	frames

BDMGR level

You can set the state of a Bd-channel, display the state of a Bd-channel and its PHI, and set a debug flag with the BDMGR level. The following are three subcommands in this level:

SETDATA command

CAUTION

This command should *not* be used in a live office situation.

You can set the Bd-channel on PHI local_phi_id to either a busy or an inservice state with the SETDATA command.

Setdata	**<local_phi_id> <state>**

Where:

local_phi_id is a number 0-7

state is the state of command (Busy or Insv)

DISPLAY command

This command displays the state of PHI local_phi_id as well as the Bd-channel state for that PHI.

Display	**<local_phi_id>**

Where:

local_phi_id is a number 0-7

DEBUG command

CAUTION

This command should *not* be used in a live office situation.

This command toggles a debug flag. This reduces the wait time for RTSing a bd-channel from five min to one min.

deBug	<on_or_off>

Where:

on_or_off is the state of debug (on or off)

STTRC level

This level allows tracing or interception of messages that are exchanged from the IAC and the STs. The following four subcommands are in this level.

ENABLE command

This command turns on the tracing facility.

Enable	

DISABLE command

This command turns off the tracing facility.

Disable	

SET command

This command is used to set the criteria for tracing.

Set	<mode> <dir> <local_st_id> <dist_id> <msg_type> <ll_num>

Where:

mode is {Mon,Int}, and is used to tell the facility what to do with the messages. Mon is used to monitor the messages; Int is used to intercept messages.

dir is {In,Out,Both}, and is used for deciding what messages to trace. If In is selected, then only messages coming from the ST are traced. If Out is selected, then only messages going to the ST are traced. If Both is selected, then all messages are traced.

local_st_id is a number 0-79, or X for all STs, and is used to decide which ST is to be traced. Enter a number for a particular ST or **X** for all ST's.

dist_id is a number 1-8, or X for all distributors, and is used to restrict message capture to a selected distributor. Enter dist_id for a particular distributor or **X** for all. Following is the list of distributors:

DISTRIBUTOR IDs COMMON TO DCH AND PHI
1 = Any maintenance function on the ST (bsy,rts,...)
2 = Firmware loader messages
3 = ST monitor messages
DISTRIBUTOR IDs FOR THE DCH ONLY
4 = layer 3 Q.931 SAPI 0 messages
5 = all SAPI 16 messages
6 = messages that affect logical link states
7 = link management messages
8 = TEI management messages
DISTRIBUTOR IDs FOR PHI ONLY
4 = Valid SAPI 16 frames to send to packet handler
5 = SAPI 16 frames that are too long
6 = Bd-channel maintenance

msg_type is a number 0-255, or X for all message types, and is used to select specific message types for message capture. Enter the message type value or **X** for all message types.

rtd ll_num is a number 0-15, and is used to restrict message capture to a particular logical link (hence terminal). Enter a link number (0-#FF) or **X** for all logical links. To find out which terminal is on which logical link, use the DDlCI command in the Llm sub-level of the ISdn level, and match the TEI to the logical link.

QUERY command

You can display the tracing setup with the QUERY command.

Query	

BCM evel

You can perform operations on the B-channels with the BCM level. Each subcommand has at least 3 parameters, the first two are the internal terminal and node numbers of the loop, and the third is the b-channel 0 or 1 (some commands offer 3 for either/both). The following subcommands are in this level:

ALLOCATE command

This command will assigns a B-channel on the loop indicated by the internal node and terminal to the user_id.

Allocate	**<int_term_num> <int_node_num> <b_channel> <user_id>**

Where:

int_term_num is a number 0-6399

int_node_num is 1-200

b_channel is a number 0-1

user_id is a number 0-255

DEALLOCATE command

This command is used to free the B-channel that was assigned to user_id. This command must be used twice to actually return the B-channel to idle, since the B-channel manager uses the first time to set the B-channel to a release-pending state.

Deallocate	**<int_term_num> <int_node_num> <b_channel> <user_id>**

Where:

int_term_num is a number 0-6399

int_node_num is a number 1-200

b_channel is a number 0-1

user_id is a number 0-255

DELOAD command

This command changes the state of the B-channel from CPB to CPD, or from idle to busy.

deLoad	<int_term_num> <int_node_num> <b_channel>

Where:

int_term_num is a number 0-6399

int_node_num is a number 1-200

b_channel is a number 0-1

CANCEL command

If a DELOAD command was issued successfully to the B-channel, this command will cancel the command (that is, change the state from CPD back to CPB or from busy back to idle).

Cancel	<int_term_num> <int_node_num> <b_channel>

Where:

int_term_num is a number 0-6399

int_node_num is 1-200

b_channel is a number 0-1

FORCERIS command

This command changes B-channel from any state to busy.

Forceris	<int_term_num> <int_node_num> <b_channel>

Where:

int_term_num is a number 0-6399

int_node_num is 1-200

b_channel is a number 0-1

BUSY command

This puts the B-channel into the busy state, if it was in an idle state.

Busy	<int_term_num> <int_node_num> <b_channel>

Where:

int_term_num is a number 0-6399

int_node_num is 1-200

b_channel is a number 0-1

RTS command

This puts the B-channel into the idle state, if it was in the busy state.

Rts	**<int_term_num> <int_node_num> <b_channel>**

Where:

int_term_num is a number 0-6399

int_node_num is a number 1-200

b_channel is a number 0-1

BSYSCON command

This command is used to put the B-channel into the busy state, if it is in the special connection state SCON.

bsYscon	**<int_term_num> <int_node_num> <b_channel>**

Where:

int_term_num is a number 0-6399

int_node_num is a number 1-200

b_channel is a number 0-1

MONITOR command

This command puts the B-channel into monitor mode, which means a message is sent to the CC every time the B-channel changes state.

Monitor	**<int_term_num> <int_node_num> <b_channel>**

Where:

int_term_num is a number 0-6399

int_node_num is a number 1-200

b_channel is a number 0-1

STOPMON command

This command removes the B-channel from the monitor mode.

Stopmon	**<int_term_num> <int_node_num> <b_channel>**

Where:

int_term_num is a number 0-6399

int_node_num is a number 1-200

b_channel is a number 0-1

CONNECT command

This command places the B-channel into the special connection state SCON, if the B-channel is in the idle state.

cOnnect	**<int_term_num> <int_node_num> <b_channel>**

Where:

int_term_num is a number 0-6399

int_node_num is a number 1-200

b_channel is a number 0-1

DISCONNECT command

This command makes the B-channel idle, if it is in the SCON state.

dIsconnect	**<int_term_num> <int_node_num> <b_channel>**

Where:

int_term_num is a number 0-6399

int_node_num is a number 1-200

b_channel is a number 0-1

RECONNECT command

If B-channel was previously connected and disconnected, then this command will return the B-channel to the SCON state.

rEconnect	<int_term_num> <int_node_num> <b_channel>

Where:

int_term_num is a number 0-6399

int_node_num is a number 1-200

b_channel is a number 0-1

QUERY command

This command displays the current state of the B-channel.

Query	<int_term_num> <int_node_num> <b_channel>

Where:

int_term_num is a number 0-6399

int_node_num is a number 1-200

b_channel is a number 0-2

REINIT command

This command initializes the B-channel to idle regardless of its current state.

reiNit	<int_term_num> <int_node_num> <b_channel>

Where:

int_term_num is a number 0-6399

int_node_num is a number 1-200

b_channel is a number 0-1

ISDNCP level

You can trace Q.931 messages, simulate Q.931 messages, and perform some maintenance on ISDN sets and loops with the ISDNCP level. ISDNCP has 2 sub-levels: LLMSIM and ISLOOP.

LLMSIM Level

This level offers the tracing and simulation of Q.931 messaging. There are two sub-levels and five commands at this level: FUNC, OPTIONS_LIM, and FA, KP, BCC, INI, and HEX.

Sub-level Options_llm is used to set up the message trace, sub-level FUNC is used to generate functional Q.931 messages, and the other commands at this level generate stimulus Q.931 messages.

The message-generation commands prompt for the terminal that works with the message (set TID) unless a working terminal has been established using the Options sub-level. Set up a working terminal before proceeding to generate messages. These messages are sent to the terminal processing task (TPT) where they are processed as if an ISDN terminal sent them. The stimulus generation commands are described as follows:

FA command

This command generates a Q.931 feature activator message and sends it to the TPT.

Fa	**[set_tid_id] <fa_value>**

Where:

set_tid_id is a number 0-7055. This parameter is considered optional, since the command searches for a working TID first.

fa_value is a number 1-64

Kp command

This command generates a Q.931 keypad message and sends it to the TPT.

Kp	**[set_tid_id] <kp_digits>**

Where:

set_tid_id is a number 0-7055. This parameter is considered optional, since the command searches for a working TID first.

kp_digits is a string of keypad digits

BCC command

This command generates a Q.931 B-channel control message and sends it to the TPT. The following four values are valid for bcc_value:

- 0 = release B-channel 1

- 1 = release B-channel 2
- 2 = connect B-channel 1
- 3 = connect B-channel 2

Bcc	**[set_tid_id] <bcc_value>**

Where:

set_tid_id is a number 0-7055. This parameter is considered optional since the command searches for a working TID first.

bcc_value is a number 0-3

INI command

This command generates a Q.931 initialize message and sends it to the TPT.

Ini	**[set_tid_id]**

Where:

set_tid_id is a number 0-7055. This parameter is considered optional, since the command searches for a working TID first.

HEX command

This command generates a message in Q.931 format and sends it to the TPT. This is useful for testing error paths caused by bogus messages from ISDN terminals.

Hex	**[set_tid_id] <hel_val1> [<hel_val2>..<hex_valN>]**

Where:

set_tid_id is a number 0-7055. This parameter is considered optional, since the command searches for a working TID first.

hex_val is a number #00-#FF (in hex format)

OPTIONS level

The options sub-level contains the facilities for tracing Q.931 messages. This level contains the following commands:

WSTID command

This command makes the terminal referenced by the ISDN set_tid_id the working terminal that is used by the Q.931 message generation commands described earlier.

Wstid	**<set_tid_id>**

Where:

set_tid_id is a number 0-7055

WLTID

This command will make the terminal on loop int_tid_id with TEI equal to TEI_value the WORKING terminal which is used by the Q.931 message generation commands described earlier.

WLtid	**<int_tid_id> <tei_value>**

Where:

int_tid_id is a number 0-6399

tei_value is number 1-63

WETID command

This command makes the terminal reference by the external TID information and TEI equal to TEI_value the working terminal that is used by the Q.931 message-generation commands described earlier.

WEtid	**<ext_node_no> <ext_term_no> <tei_value>**

Where:

ext_node_no is a number 0-4095

ext_term_no is a number 1-20

tei_value is a number 1-63

WDUMP

This command displays the current WORKING terminal in all formats.

WDump	

WRESET command

This command clears the working terminal.

WReset	

ININT command

CAUTION

This tool should *not* be used in a live office situation.

This command turns the incoming message intercept on.

Inint	

INMON command

This command turns incoming message monitoring on.

INMon	

INOFF command

This command turns incoming message tracing off.

INOff	

OUTINT command

CAUTION

This tool should never be used in a live office situation.

This command turns the outgoing message intercept on.

Outint	

OUTMON command

This command turns the outgoing message monitoring on.

OUTMon	

OUTOFF command

This command turns the outgoing message tracing off.

OUTOff	

ALLOFF command

This command turns all tracing off.

Alloff	

HEX command

This command toggles the hex display of message tracing on/off.

Hex	

VERBOSE command

This command toggles Verbose display of message trace ON/OFF.

Verbose	

SYNC command

This command toggles the sync/async display of tracing.

> ***Note:*** Sync display *will* slow down TPT processing. It is strongly recommended that Sync display not be used in the field.

Sync	

STID command

This command toggles the display of the SET TID in message tracing on/off.

STid	

LTID command

This command toggles the display of the internal TID and TEI in message tracing on/off.

Ltid	

ETID command

This command toggles display of external TID and TEI in message tracing on/off.

Etid	

RAMFILE command

This commands toggles trace dumping to RAMFILE on/off.

Ramfile	

SCREEN command

This command toggles trace dumping to SCREEN on/off.

SCreen	

OPTDUMP command

This command displays the current tracing options.

OPtdump	

SELECT command

SElect	**<set_tid1> [<set_tid2>..<set_tid5>]**

Where:

set_tid is a number 0-7055. This allows selective tracing (up to five terminals at once) of Q.931 messages.

REMOVE command

This command removes terminal(s) from the selective tracing list.

REmove	**<set_tid1> [<set_tid2>..<set_tid5>]**

Where:

set_tid is a number 0-7055

FLEX command

This command will toggle the edit mode (used in the functional message-generation command “Edit” described on page 6-33 between flexible and regular.

Flex	

WCR command

This command sets the working call reference to be used by the functional message-generation command “Edit” described on page 6-33. When set, you are not prompted during message entry or edit.

WCr	<Dummy | Global | Hex | Field>

Where:

Dummy sets the working CR to the dummy call reference (0 length)

Global sets the working CR to the 2-octet global call reference (00 00)

Hex sets the working CR to the hex octets entered. The call reference flag is the most significant bit of the octets entered.

Field sets the working CR to the value entered in response to the prompts you are prompted for (CR flag, 15-bit CR values)

WCRDUMP command

This command displays the current working CR.

WCRDump	

WCRRESET command

This command clears the working CR.

WCRReset	

FUNC level

This level provides the Q.931 functional message-generation facility. This level was added by feature AC0332 and is fully documented in the FN and MM sections for that feature. Following is a summary of the commands in the FUNC level.:

LIST command

This command lists the saved messages, displaying the message number and type.

List	

VIEW command

This command displays the contents of a message in verbose format.

View	**<msg_no>**

Where:

msg_no is a number 1-8

SEND command

This command sends a message to the TPT. If the working terminal is not set, you are prompted for a TID (set TID, loop TID + TEI, or external TID + TEI).

Send	**<msg_no>**

Where:

msg_no is a number 1-8

DELETE command

This command deletes a message.

Delete	**<msg_no>**

Where:

msg_no is a number 1-8

EDIT command

This command is used to enter or edit a Q.931 functional message. You are prompted for the contents of the information elements in the message. In flexible edit mode, you choose which information elements to include. In regular edit mode, only those information elements valid for the particular message type are included. At the end of message entry, the message can be viewed, saved, or sent.

Edit	**<msg_no | msg_type>**

Where:

msg_no is a number 1-8 used to specify a saved message to edit

msg_type is used to specify the type of a new message to be entered. Message types are as follows: ALERT, CALL_PROC, CONN, CONN_ACK, DISC, INFO, PROG, REL, REL_COMP, REST, REST_ACK, SETUP, SETUP_ACK, STATUS, STAT_ENQ.

ISLOOP Level

You can BUSY and RTS all the logical terminals on the loop as well as displaying some information about them with the ISLOOP level. There are 3 subcommands:

LOOPINFO command

This command will display information about the loop referenced by int_term_num.

Loopinfo	**<int_term_id> <disp_set>**

Where:

int_term_id is a number 0-6399

disp_set is Yes or No

Note: A response of yes for disp_set results in the information about each logical terminal on the loop.

RTSLTS command

This command will RTS all logical terminals on the loop referenced by int_term_num.

Rtslts	<int_term_id>

Where:

int_term_id is a number 0-6399.

BSYLTS command

This command will BUSY all logical terminals on the loop referenced bsy int_term_num.

Bsylts	<int_term_id>

Where:

int_term_id is a number 0-6399

DCH continuity test

To invoke the DCH continuity test from PMDEBUG, go to the DIagnose level and type in the following:

```
TSTCONT <(r=local_st_id>
```

Where:

local_st_id is a number 0-79

Note: Ensure that a loop-back point is set somewhere on the loop, or this command will report failure.

Executing ISDN ST monitor commands

To invoke the ST monitor commands, go to the STMt level of PMDEBUG and type in the following:

sTdebug	**<local_st_id>**

Where:

local_st_id is a number 0-79

Common PMDEBUG procedures

This section contains several common PMDEBUG procedures.

Accessing PMDEBUG

To access PMDEBUG type the following and press Enter:

PMDEBUG <PM type> <PM number> <subunit>

Refer to "Command syntax" on page 3-3 to determine the command syntax for the XPM under analysis.

The PMDEBUG command string accesses the master processor (MP) level. Press the Enter key to verify that the MP level has been accessed. The LTCMP> prompt should appear. If the LTCSP> prompt appears, indicating that the signaling processor (SP) level has been accessed, type **MP** to access the MP level.

If the prompt indicates that you are in a level other than the LTCMP or the LTCSP level, enter an asterisk (*) until the LTCMP> prompt appears. Then enter **MP**. Refer to "Master processor" on page 1-2 for more information on the MP level.

Once in PMDEBUG, only the PMDEBUG subcommands are valid and MAP level commands are not accepted.

PP load in the MP

To determine the peripheral processor (PP) load in the MP, type **L**.

Displaying MP modules

To display the list of the MP module names:

1 Access the debug mode by entering **D**.

2 Access the information mode by entering **I**.

3 Access the environment mode by entering **E**.

4 To return to the top of the MP level, type *** * ***.

Accessing the SP level

To access the SP while in PMDEBUG, enter **SP**.

Displaying SP modules

To display the list of the SP module names:

1 Access the Debug mode by entering `D`.

2 Access the Information mode by entering `I`.

3 Access the Environment mode by entering `E`.

4 To return to the top of the SP level, type `* * *`.

SWERR dump

SWERR are sent to the CC and printed as log reports, but SWERRs are also stored in the XPM.

How to dump SWERR

To dump SWERR:

1 Access PMDEBUG. For information on accessing PMDEBUG, refer to "Accessing PMDEBUG" on page 7-1.

2 Access the SWERR level by entering **S(werr)**.

3 Display SWERR information by performing any of the following steps:

Note: To stop a SWERR dump, press Enter.

- Dump all the SWERR stored in the XPM by entering **D(ump)**.
- Dump the last SWERR stored by entering **L(ist) <n> .**

  For example, to dump the last three SWERR stored, enter **L3**.

- Dump the last SWERR stored with the procedure traceback showing where the SWERR occurred by entering **T(race) <n>**.

  For example, to dump the last three SWERR with the procedure traceback, enter **T3**.

4 Clear the SWERRS in the MP by entering **C(lear)**.

If a trace is needed for a specific SWERR, perform the following steps:

1 Access the SWERR level by entering **S(werr)**.

2 Obtain a list of the SWERR by entering **L(ist)**.

3 Trace a particular SWERR by entering **T(race) M(ark) <swerr number>**.

For more information on commands in the SWERR level, refer to "SWERR level" on page 4-54.

Figure 7-1 on page 7-4 contains an example of a SWERR dump. Refer to the example under “TRACE command” on page 4-57 for more examples of displaying SWERRs.

Figure 7-1
Example of a SWERR dump (Part 1 of 2)

```
pmdebug lgc 0

PMDEBUG MODE - CONNECTING TO PM
WARNING:  You now have access to the PM monitor...Proceed
          with caution
LTCMP>

TIme,TAsk,Load,Xprompt,Debug,Swerr,Ipc,Queues,Uspace,Patches,
Msgtr,Chnls,Opf,MTc,Trmtrc,Gdt,DIagnose,Audit,CAudit,CP,SE,
Rcvrmon,MGen,PRfm,OMUnsol.
LTCMP>

>s

Dump,Notime_dump,List,Trace,Clear,Prev_clear,COntswrs.
MP:Swerr>

>l

SWERR Sequence #    17
 Task : MPAUDTK    FATHB
 CC Time: 01:14:08:20.36
 PP Time: 00:01:04:17.98     current  load
 Data: 02 8D 05 46  41 54 48 42  00 01 00
SWERR Sequence #    16
 Task : MPAUDTK    au
 CC Time: 01:14:07:11.63
 PP Time: 00:01:03:09.25     current  load
 Data: 04 00 02 01  00 00 01 00  00 00 00 00  00 00
SWERR Sequence #    15
 Task : MPAUDTK    au
 CC Time: 01:14:07:10.94
 PP Time: 00:01:03:08.56     current  load
 Data: 02 00 01 E8  02 8D 03 00  00 00 00 00  02 01
SWERR Sequence #    14
 Task : MPAUDTK    au
 CC Time: 01:14:07:10.64
 PP Time: 00:01:03:08.26     current  load
 Data: 01 00 01 52  01 E8 00 00  00 00 00 00  00 00
SWERR Sequence #    13
 Task : MPAUDTK    FATHB
```

Figure 7-1
Example of a SWERR dump (Part 2 of 2)

```
 CC Time: 01:13:48:01.35
 PP Time: 00:00:43:58.97     current  load
 Data: 02 8D 05 46  41 54 48 42  00 01 00
SWERR Sequence #    12
 Task : MPAUDTK    au
 CC Time: 01:13:46:50.98
 PP Time: 00:00:42:48.60     current  load
 Data: 04 00 01 5C  00 00 01 00  00 00 00 00  00 00
SWERR Sequence #    11
 Task : MPAUDTK    au
 CC Time: 01:13:46:50.32
 PP Time: 00:00:42:47.94     current  load
 Data: 02 00 01 4A  02 8D 03 00  00 00 00 00  01 5C
SWERR Sequence #    10
 Task : MPAUDTK    au
 CC Time: 01:13:46:50.09
 PP Time: 00:00:42:47.71     current  load
 Data: 01 00 01 BC  01 4A 00 00  00 00 00 00  00 00

Cont? (y,<cr>)
MP:List>

>*

MP:Swerr>

>*

LTCMP>
```

Dumping trap information

Traps occur when errors are detected by firmware or by the software operating system (SOS). An example of a trap is an attempt to divide by zero or an attempt to access the fifth item of a four-item table.

During a trap, the call process is killed, and a death message is sent to its daddy process. The daddy process then recreates the call process. When the call process begins to execute, it cleans up the call in which the trap occurred.

Following are examples of traps:

- data port write protect: caused by an attempt to write to a protected data port without disabling protection

- store stack overflow: occurs at the beginning of a procedure if there is insufficient room for the new stack frame on the procedure call stack
- bad procedure variable: occurs when an uninitialized or perprocess variable is called during a procedure.

Some traps are recoverable and do not cause a unit to drop activity. However, if several recoverable traps occur in a short period of time, the unit drops activity.

Trap log reports

The CC subsystem generates PM185 log reports when the firmware, hardware, or software detects an error condition that causes a trap interrupt.

Trap log reports have three main sections.

- The first section summarizes the reason for the trap and shows what part of the program was being processed when the trap occurred.
- The second section provides a traceback of the procedures that were completed prior to the trap. A traceback is a record of the procedures processed by the software, showing the sequence in which the instructions were processed.
- The third section provides information that varies according to the trap reason.

For more information on PM185, refer to 297-1001-510.

How to display TRAPS

1 Access PMDEBUG. For information on accessing PMDEBUG, refer to "Accessing PMDEBUG" on page 7-1.

2 To access the DEBUG level enter **D(ebug)**.

3 To access the trap information mode, enter **T(rapinfo)**.

4 Display a summary of the traps by entering **S(ummary)**.

This command will not give a traceback. The SUMMARY command provides the sequence numbers and a brief description of each trap.

5 Display a trap traceback for a specific trap sequence number by entering **<sequence number>**.

6 Display the last trap that occurred by entering **L(ast)**.

7 Return to the debug level by entering *****.

8 Clear all the traps in the XPM load by entering **C(leartrap) ALL**.

To clear only those traps that occurred in the previous load or before the last RTS, enter **C(leartrap) PREV.**

For more information on commands in the TRAPINFO level, refer to “TRAPINFO level” on page 4-37.

Figure 7-2 on page 7-8 contains an example of dumping trap information.

Figure 7-2
Dumping trap information (Part 1 of 2)

```
LTCMP>

>>>d

DM,FIndproc,SM,SW,Trapinfo,STopontrap,Cleartrap,Zerodiv,Rom,Info,
TRCpoints,CAlltrace,Availtime,COverage,TEmpstore.
MP:Debug>

>>>t
2 error traps have been saved
Enter <trap identifier>, Last, Summary, All, or *
MP:Trapinfo>

>>>s
Trap id  Seq #  Load      Error
   2      2   Current  Addr err accessing loc 0000000F
   1      1   Current  Parity:SR=EFE0 Addr range=000000-020000

Enter <trap identifier>, Last, Summary, All, or *
MP:Trapinfo>

>>>2

Ram Load Name = NDT24AQ
MP rom name = XPMRFA10, SP rom name = XPMRFA10
trap in MP  :  Addr err accessing loc 0000000F
Task: TIMER   0006 0006   Trap sequence #: 2 Current load
PP TIme : 00:00:00:29.87
Occurred at:  001DFCEC  TIMETASK 14   TIME      5   Offset:  #2
Called from:            UTIL      9   CALLCRIT 19   Offset: #??
              001DFC0E  TIMETASK 14   TIMEIMP   4   Offset: #70
              001DFCFA  TIMETASK 14   TIME      5   Offset: #10
              001DFDF8  TIMETASK 14   XTIMETAS  5   Offset: #4A

PC:00003416  SR=2104  US=00017FAC  SS=001E33DE  TCB=000177C0
D0 =FFFFFFFF D1 =00000132 D2 =00BA0470 D3 =00010082
D4 =000707C0 D5 =000177C2 D6 =04700470 D7 =001D0001
A0 =0005DB38 A1 =0000000F A2 =00017FB0 A3 =001DFCEC
A4 =00112F42 A5 =00017FB4 A6 =001614D8 A7 =001E33D4
Stack:0000 0308 2114 0000 3B0E 0000 3AA0 0000 41B2 0104
                                                0016 14E8
```

Figure 7-2
Dumping trap information (Part 2 of 2)

```
Enter <trap identifier>, Last, Summary, All, or *
MP:Trapinfo>

>>>1

Ram Load Name = NDT24AQ
MP rom name = XPMRFA10, SP rom name = XPMRFA10

trap in MP  :  Parity: SR = EFE0 Addr range = 000000 - 020000
Task: TIMER   0006 0006   Trap sequence #: 1 Current load
PP TIme : 00:00:00:29.80
Occurred at :00161D06  UTILMOD   8  assembler routine 40
                                                 offset: #476
Called from:            UTIL      9   CALLCRIT 19   Offset: #??
            001DFC0E    TIMETASK 14   TIMEIMP   4   Offset: #70
            001DFCFA    TIMETASK 14   TIME      5   Offset: #10
            001DFDF8    TIMETASK 14   XTIMETAS  5   Offset: #4A

PC:00161D06  SR=2104  US=00017FAC  SS=001E33DE  TCB=000177C0
D0 =FFFFFFFF D1 =00000132 D2 =00BA0470 D3 =00010082
D4 =000707C4 D5 =000177C0 D6 =04700470 D7 =001D0001
A0 =0005DB38 A1 =0000000F A2 =00017FB0 A3 =001DFCEC
A4 =00112F42 A5 =00017FB4 A6 =001614D8 A7 =001E33CE
Stack: 0000 0308 B095 0000 000F B091 2104 0000 3416 2114
                                                 0000 3B0E

Enter <trap identifier>, Last, Summary, All, or *
MP:Trapinfo>

>>>* *

LTCMP>
```

Exec and primitive tracing

This section contains information on tracing primitives and execs.

Execs

An exec is a sequence of primitives that define specific PP instructions.

Execs are defined by software in the central control (CC) and are referred to by an identifier in the range of 1 to 254. Exec identifier 0 is reserved by the peripheral module (PM) as the nil exec (NIL_EXEC_ID), which does

nothing. Exec identifier 255 is reserved as an invalid exec that generates a command protocol violation report.

Execs are called as an instruction sequence or are referenced indirectly in response to an event.

When the end of an exec is reached (ENDEXEC primitive), control returns to the processing level from which the exec was called.

Execs are the building blocks for maintenance and call-processing (CP) software that requires processing in the PM. The PM responds to external events by calling an exec. Messages from the CC are required to initialize exec identifier locations and initiate supervisory or timing functions. Once processing has begun, CC intervention is limited only by the complexity of the execs and the processing resources available in the PM.

Following is a list of basic functions that execs provide in CP:

- call origination
- incoming proceed-to-send signaling
- dial tone
- digit collection
- call supervision
- outgoing seizure signaling
- digit outpulsing
- call duration timing
- tone cadence
- voice path connection
- called party disconnect
- calling party disconnect

How to display a list of primitives and execs

1 Access PMDEBUG. For information on accessing PMDEBUG, refer to "Accessing PMDEBUG" on page 7-1.

2 Calculate the internal terminal identifier for the terminal you need to trace. Refer to "Calculating internal terminal identifiers" on page 2-6.

3 Access the SERVER level by entering **SE(rver)**.

4 Start the exec and primitive trace by entering **S(tart)**.

5 Enter the internal node number, internal terminal number, and the extension byte of the terminal to be traced: **nodenum termnum ext byte**.

For example: **1 3 0**.

Note: The extension byte is the unit number. For business sets, the extension byte is equal to the key number; for nonbusiness sets, the extension byte is equal to zero.

Up to three terminals can be traced simultaneously. When all the terminal numbers have been entered, return to the SERVER level by typing *****.

6 Make the call.

7 Display the exec trace by entering **D(isplay)**.

The DISPLAY command displays the terminal index numbers for each terminal being traced. Note the terminal index number that corresponds with each terminal being traced.

8 Indicate which trace back will be displayed by doing one of the following:

- Display the exec trace for a particular terminal by entering **E(xec)**.

  Indicate which exec trace to display by entering **<terminal index number>**.

- Display the primitive trace for a particular terminal by entering **P(rimitive)**.

  Indicate which primitive trace to display by entering **<terminal index number>**.

- Display both the exec and primitive traces for all terminals being traced by entering **A(ll)**.

9 Stop the exec and primitive trace by entering **K(ill)**.

For more information on commands at the SERVER level, refer to “SERVER level” on page 4-190.

Refer to Figure 7-3 on page 7-12 for an example of an exec and primitive trace.

Figure 7-3
Sample exec and primitive trace (Part 1 of 5)

```
>pmdebug dtc 0 0

PMDEBUG MODE - CONNECTING TO PM
WARNING: You now have access to the PM monitor...Proceed
            with caution
LTCMP>

TIme,TAsk,Load,Xprompt,Debug,Swerr,LIpc,Ipc,Patches,Msgtr,
Uspace,Chnls,MTc,DIagnose,Audit,PKtldr,CP,SE,C7tu,PEg,Rcvrmon.
LTCMP>

>cp

Dump,Unprot,Trmnl,Athb,Xdb,tVa,Chb,Fnb,Mod,Brk,Extint,
Swct,Idlq,abtrK,Rmt,*
MP:CP>

>e 26 2

INTERNAL NODE     NUMBER = 1
INTERNAL TERMINAL NUMBER = 3
Dump,Unprot,Trmnl,Athb,Xdb,tVa,Chb,Fnb,Mod,Brk,Extint,
Swct,Idlq,abtrK,Rmt,*
MP:CP>

>e 26 66

INTERNAL NODE     NUMBER = 1
INTERNAL TERMINAL NUMBER = 67
Dump,Unprot,Trmnl,Athb,Xdb,tVa,Chb,Fnb,Mod,Brk,Extint,
Swct,Idlq,abtrK,Rmt,*
MP:CP>

>*

LTCMP>

>se

Start,Display,Clear,Kill,Query,sTore,Histogram,*
MP:SE>

>s
```

Figure 7-3
Sample exec and primitive trace (Part 2 of 5)

```
nodenum,termnum,ext byte,*
MP:SE>

>1 3 0

nodenum,termnum,ext byte,*
MP:SE>

>1 67 0

nodenum,termnum,ext byte,*
MP:SE>

>*

Prim/exec tracing started on 2 terminals
Start,Display,Clear,Kill,Query,sTore,Histogram,*
MP:SE>

>d

Terminal index             Terminal Id
       0       : NODENO = 1  TERMNO = 3     EXTBYTE = 0
       1       : NODENO = 1  TERMNO = 67     EXTBYTE = 0
       2       : not in use
Exec, Prim, All, *
MP:SE>

>e

(input index) trmnl 0, 1 or 2, *

MP:SE>

>0

EXEC Trace of :  NODENO = 1  TERMNO = 3  EXTBYTE = 0
5B 13 19 58 38 8C 39 99 03 00 9B 5A 98 9A 4D 00
52 5D 9D 0A 5E 00 F4 5E 00 36 00 49 5F 06 19 03
76 9C 1F 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Figure 7-3
Sample exec and primitive trace (Part 3 of 5)

```
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 .
 .
 .
00 00 00 00 00 00 00 00 00 00 00 00 00 00 22
(input index) trmnl 0, 1 or 2, *

MP:SE>

>1
EXEC Trace of :  NODENO = 1  TERMNO = 67  EXTBYTE = 0
9B 42 66 13 3D 98 3E 38 8C 3F 37 8D 45 90 29 47
16 06 19 09 03 76 3A 98 21 22 7B 38 8C 7A 10 13
14 1F 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 .
 .
 .
00 00 00 00 00 00 00 00 00 00 00 00 00 00 21
(input index) trmnl 0, 1 or 2, *

MP:SE>

>*

Exec, Prim, All, *
MP:SE>

>p

(input index) trmnl 0, 1 or 2, *
MP:SE>

>0
```

Figure 7-3
Sample exec and primitive trace (Part 4 of 5)

```
PRIM Trace of :  NODENO = 1  TERMNO = 3  EXTBYTE = 0
D0 80 10 01 E7 80 E0 80 E0 84 90 47 11 E0 2F 2F
81 E1 90 26 82 61 59 38 A3 E2 80 47 E0 25 81 55
87 38 83 90 47 E0 E0 E0 E0 22 25 E0 E0 58 81 55
47 10 84 1F 38 81 55 26 80 36 25 25 A0 81 A0 81
E2 E0 38 81 55 26 83 E0 90 56 25 25 0F 25 25 80
E0 01 38 90 82 90 48 E0 E0 E0 26 25 20 25 01 38
90 26 A0 90 81 E0 E0 0F 25 38 83 E0 90 56 25 25
01 38 90 26 0D 90 26 25 38 16 4B 83 E0 55 84 1F
87 82 59 E0 E2 E0 55 4c 55 26 82 E0 E1 89 5D 25
4B 4A 10 81 4A 4B 45 10 81 90 47 10 84 E0 58 25
01 38 82 90 47 10 03 25 01 38 90 26 25 4B 4A 10
38 90 26 25 16 38 80 59 90 26 38 04 25 83 90 E0
E2 E0 38 83 90 47 E0 E0 E0 E0 22 25 28 25 01 38
90 82 90 48 E0 E0 E0 26 90 26 81 3F 25 90 26 83
E0 90 58 25 25 20 25 01 80 E0 01 38 0D 90 26 03

(input index) trmnl 0, 1 or 2, *
MP:SE>

>1

PRIM Trace of :  NODENO = 1  TERMNO = 67  EXTBYTE = 0
38 90 26 38 80 E0 25 38 A3 E2 80 47 E0 25 39 83
81 55 47 10 81 55 84 1F A0 A0 E0 81 E0 0E 25 25
25 90 80 0E 25 01 38 38 81 55 26 80 37 25 25 38
87 E0 E0 E0 E2 55 26 5C 25 83 55 E1 E0 90 4D 4C
03 01 38 4A 4B 10 90 26 84 E0 E0 05 38 98 B8 4B
4C 25 08 03 25 25 01 81 2D 1A 01 84 90 47 11 E0
2F 2F 38 04 25 87 80 E0 E3 E3 38 83 90 47 E0 E0
E0 E0 22 25 39 01 87 E7 38 66 25 01 38 90 82 90
48 E0 E0 E0 26 90 26 81 55 26 83 E0 90 56 25 25
90 26 83 E0 90 58 25 25 20 25 01 38 0D 90 26 3B
B1 55 26 80 26 25 25 38 81 38 89 05 93 A3 35 90
90 91 A1 B9 08 38 A3 E2 80 47 E0 25 25 38 A3 E2
80 48 E0 25 25 25 28 38 83 E0 90 58 25 25 01 38
90 26 38 81 38 89 05 93 A3 35 90 90 91 A1 B9 08
38 A3 E2 80 47 E0 25 25 25 38 D0 80 10 01 00 FD

(input index) trmnl 0, 1 or 2, *
MP:SE>

>*
```

Figure 7-3
Sample exec and primitive trace (Part 5 of 5)

```
Exec, Prim, All, *
MP:SE>

> *
LTCMP>
```

Exec and primitive trace with a histogram

To display a list of the primitives and execs used during a particular call, and the number of times each primitive was posted, perform the following steps:

1 Access PMDEBUG. For information on accessing PMDEBUG, refer to "Accessing PMDEBUG" on page 7-1.

2 Calculate the internal terminal identifier for the terminal you need to trace. Refer to "Calculating internal terminal identifiers" on page 2-6.

3 Access the SERVER level by entering **SE(rver)**.

4 Start the histogram by entering **H(istogram)**.

5 Start the histogram by entering **S(tart)**.

6 Enter the internal node number, the internal terminal number, and the extension byte to be traced: **nodenum termnum ext byte**.

For example: **1 3 0**.

Only one terminal can be traced simultaneously.

7 Return to the SERVER level by typing *****.

8 Start the exec and primitive trace by entering **S(tart)**.

9 Enter the internal node number, internal terminal number, and the extension byte of the terminal to be traced: **nodenum termnum ext byte .**

For example: **1 3 0**.

Up to three terminals can be traced simultaneously. When all the terminal numbers have been entered, return to the SERVER level by typing *****.

10 Make the call.

11 Display the exec trace by entering **D(isplay)**.

The DISPLAY command displays the terminal index numbers for each terminal being traced. Note the terminal index number that corresponds with each terminal being traced.

12 Indicate which traceback will be displayed by doing one of the following:

- Display the exec trace for a particular terminal by entering **E(xec)**.

  Indicate which exec trace to display by entering **<terminal index number>**.

- Display the primitive trace for a particular terminal by entering **P(rimitive)**.

  Indicate which primitive trace to display by entering **<terminal index number>**.

- Display both the exec and primitive traces for all terminals being traced by entering **A(ll)**.

13 Return to the SERVER level by entering *****.

14 Access the histogram by entering **H(istogram**.

15 Display the histogram by entering **D(isplay)**.

16 Return to the SERVER level by entering *****.

17 Stop the exec and primitive trace by entering **K(ill) .**

For more information on commands at the SERVER level, refer to “SERVER level” on page 4-190.

Refer to Figure 7-4 on page 7-18 for an example of an exec and primitive trace with a histogram.

Figure 7-4
Exec and primitive trace with a histogram (Part 1 of 7)

```
>qlen 1 1 19 4
-------------------------------------------------------
LEN:     HOST 01 1 19 04
TYPE: SINGLE PARTY LINE
SNPA: 619
DIRECTORY NUMBER:     6757000
LINE CLASS CODE:  PSET (WITH DISPLAY)
CUSTGRP:              BNR  SUBGRP: 0  NCOS: 0  RING: Y
ADDONS: NONE   EXTENSION: N
CARDCODE:  6X21AC    GND: N  PADGRP: STDLN  BNV: NL MNO: Y
PM NODE NUMBER     :    44
PM TERMINAL NUMBER :    613
OPTIONS:
 NONE
   KEY       DN
   ---       --
      1      DN     6757000
   KEY     FEATURE
   ---     -------
      NONE
-------------------------------------------------------
>qlen 1 1 11 8
-------------------------------------------------------
LEN:     HOST 01 1 11 08
TYPE: SINGLE PARTY LINE
SNPA: 619
DIRECTORY NUMBER:     6757175
LINE CLASS CODE:  IBN
IBN TYPE: STATION
CUSTGRP:            BNR     SUBGRP: 0  NCOS: 0
SIGNALLING TYPE:  DIGITONE
CARDCODE:  6X17AC    GND: N  PADGRP: STDLN  BNV: NL MNO: N
PM NODE NUMBER     :    44
PM TERMINAL NUMBER :    361
OPTIONS:
CWT CWI CHD DGT
-------------------------------------------------------

>pmdebug lgc 0

PMDEBUG MODE - CONNECTING TO PM
WARNING: You now have access to the PM monitor...Proceed with
caution
```

Figure 7-4
Exec and primitive trace with a histogram (Part 2 of 7)

```
LTCMP>

>cp e 44 613

INTERNAL NODE     NUMBER = 3
INTERNAL TERMINAL NUMBER = 1256

Dump,Unprot,Trmnl,Athb,Xdb,tVa,Chb,dsP,Fnb,Mod,Brk,Extint,Swct,
Idlq,abtrK,Rmt,*
MP:CP>

>e 44 361

INTERNAL NODE     NUMBER = 3
INTERNAL TERMINAL NUMBER = 1004

Dump,Unprot,Trmnl,Athb,Xdb,tVa,Chb,dsP,Fnb,Mod,Brk,Extint,Swct,
Idlq,abtrK,Rmt,*
MP:CP>

>*
LTCMP>

>se

Start, Display, Clear, Kill, Query, sTore, Histogram, *
MP:SE>

>h

HIST: Start, Display, Clear, Kill, Query, *
MP:SE>

>s

nodenum,termnum,ext byte
MP:SE>

>3 1256 0

Histogram trace started
HIST: Start, Display, Clear, Kill, Query, *
MP:SE>
```

Figure 7-4
Exec and primitive trace with a histogram (Part 3 of 7)

```
>*

Start, Display, Clear, Kill, Query, sTore, Histogram, *
MP:SE>

>s

nodenum,termnum,ext byte,*
MP:SE>

>3 1256

ext byte
MP:SE>

>0

nodenum,termnum,ext byte,*
MP:SE>

>3 1004 0

nodenum,termnum,ext byte,*
MP:SE>

>*

Prim/exec tracing started on 2  terminals
Start, Display, Clear, Kill, Query, sTore, Histogram, *
MP:SE>

>d

Terminal index               Terminal Id
       0        : NODENO = 3  TERMNO = 1256   EXTBYTE = 0
       1        : NODENO = 3  TERMNO = 1004   EXTBYTE = 0
        2        : not in use

Exec, Prim, All, *
MP:SE>

>e 0
```

Figure 7-4
Exec and primitive trace with a histogram (Part 4 of 7)

```
EXEC trace of :  NODENO = 3  TERMNO = 1256   EXTBYTE = 0
CB B4 B3 A8  CE 00 00 00  00 00 00 00  00 00 00 00
00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00
00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00
00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00
00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00
00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00
00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00
00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00
00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00
00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00
00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00
00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00
00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00
00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00
00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00
00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 04
(input index) trmnl 0, 1 or 2, *

MP:SE>
>e 1
EXEC trace of :  NODENO = 3  TERMNO = 1004   EXTBYTE = 0
CC BA A9 DB  D1 A8 CE CD  00 00 00 00  00 00 00 00
00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00
00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00
00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00
00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00
00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00
00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00
00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00
00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00
00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00
00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00
00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00
00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00
00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00
00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00
00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 07
(input index) trmnl 0, 1 or 2, *

MP:SE>

>*
```

Figure 7-4
Exec and primitive trace with a histogram (Part 5 of 7)

```
Exec, Prim, All, *
MP:SE>

>p 0

PRIM trace of :  NODENO = 3  TERMNO = 1256   EXTBYTE = 0
00 DE 80 2D  65 83 E3 83  3C 01 82 E0  90 47 11 2A
83 6B 38 85  E4 A3 E4 25  80 E0 80 E0  01 38 A1 81
90 10 26 93  A3 35 81 4A  48 4C 90 10  81 05 B3 08
A3 E3 25 25  01 38 90 26  16 3A 21 29  80 90 48 E0
90 90 48 93  A3 35 E3 93  81 05 B4 08  25 25 01 3A
82 52 2E 64  01 00 00 00  00 00 00 00  00 00 00 00
00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00
00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00
00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00
00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00
00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00
00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00
00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00
00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00
00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00
00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 54
(input index) trmnl 0, 1 or 2, *

MP:SE>

>1

PRIM trace of :  NODENO = 3  TERMNO = 1004   EXTBYTE = 0
80 2D 65 82  E0 90 47 11  85 6C 38 85  A3 E4 E4 25
80 E0 80 E0  01 38 38 0D  21 29 90 26  80 E0 25 80
4B 10 80 E0  38 93 A3 35  E3 25 25 25  01 38 90 26
90 82 61 0D  48 4C 38 21  29 81 90 47  10 90 90 48
93 81 05 B4  08 25 25 25  01 3A 82 52  2E 64 01 00
00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00
00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00
00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00
00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00
00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00
00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00
00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00
00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00
00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00
```

Figure 7-4
Exec and primitive trace with a histogram (Part 6 of 7)

```
00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00
00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 4E
(input index) trmnl 0, 1 or 2, *

MP:SE>

>*

Exec, Prim, All, *
MP:SE>

>*

Start, Display, Clear, Kill, Query, sTore, Histogram, *
MP:SE>

>h

HIST: Start, Display, Clear, Kill, Query, *
MP:SE>

>d

Histogram trace of :  NODENO = 3  TERMNO = 1256   EXTBYTE = 0
01 05 00 00  00 02 00 00  02 00 00 00  00 00 00 00
02 01 00 00  00 00 01 00  00 00 00 00  00 00 00 00
00 01 00 00  00 05 02 00  00 01 01 00  00 01 01 00
00 00 00 00  00 02 00 00  03 00 02 00  01 00 00 00
00 00 00 00  00 00 00 01  03 00 01 00  01 00 00 00
00 00 01 00  00 00 00 00  00 00 00 00  00 00 00 00
00 00 00 00  01 01 00 00  00 00 00 01  00 00 00 00
00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00
04 04 02 03  00 01 00 00  00 00 00 00  00 00 00 00
07 00 00 03  00 00 00 00  00 00 00 00  00 00 00 00
00 01 00 04  00 00 00 00  00 00 00 00  00 00 00 00
00 00 00 01  01 00 00 00  00 00 00 00  00 00 00 00
00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00
00 00 00 00  00 00 00 00  00 00 00 00  00 00 01 00
04 00 00 03  02 00 00 00  00 00 00 00  00 00 00 00
00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00
HIST: Start, Display, Clear, Kill, Query, *
MP:SE>
```

Figure 7-4
Exec and primitive trace with a histogram (Part 7 of 7)

```
>q

Histogram trace being done on:NODENO=3 TERMNO=1256 EXTBYTE=0
HIST: Start, Display, Clear, Kill, Query, *
MP:SE>

*

Prim/exec tracing being done on :
 NODENO = 3  TERMNO = 1256   EXTBYTE = 0
 NODENO = 3  TERMNO = 1004   EXTBYTE = 0
Start, Display, Clear, Kill, Query, sTore, Histogram, *

MP:SE>

>k

MP:SE>

>*

LTCMP>

quit

NOTE: PMDEBUG will terminate when last request is complete
PMDEBUG TERMINATES
```

Dumping call-processing blocks

If a problem exists with a particular terminal and the terminal is active, you can dump the CP information associated with that terminal.

How to dump CP

18 Access PMDEBUG. For information on accessing PMDEBUG, refer to "Accessing PMDEBUG" on page 7-1.

19 Calculate the internal terminal identifier for the terminal you need to trace. Refer to "Calculating internal terminal identifiers" on page 2-6.

20 Access the CP level by entering **CP**.

21 Dump the CP information for the terminal by entering **D(ump) <internal terminal number>**.

If nothing is active on the terminal, the following is displayed: `No ATHB linked`

Note: When a CP dump is obtained, a dump of CSM information should also be obtained.

For more information on commands in the CP level, refer to “CP level” on page 4-165.

Figure 7-5 on page 7-26 contains an example of a CP dump.

Figure 7-5
CP dump example (Part 1 of 4)

```
PMDEBUG LGC 0
PMDEBUG MODE - CONNECTING TO PM
WARNING: You now have access to the PM monitor...Proceed
         with caution

TIme,TAsk,Load,Xprompt,Debug,Swerr,Ipc,Queues,Uspace,Patches,
Msgtr,Chnls,Opf,MTc,Trmtrc,Gdt,DIagnose,Audit,CAudit,CP,SE,
Rcvrmon,MGen,PRfm,OMUnsol.
LTCMP>

>cp

MP:CP>

Dump,Unprot,Trmnl,Athb,Xdb,tVa,Chb,dsP,Fnb,Mod,Brk,Extint,Swct,
Idlq,abtrK,Rmt,*
MP:CP>

>e 41 41

INTERNAL NODE     NUMBER = 2
INTERNAL TERMINAL NUMBER = 43
Dump,Unprot,Trmnl,Athb,Xdb,tVa,Chb,dsP,Fnb,Mod,Brk,Extint,Swct,
Idlq,abtrK,Rmt,*
MP:CP>

>d 43

*** Data blocks for terminal#: 43   002B
ATHB:463
XDB:85
EXT_BYTE:0  CHB:193   DDB:21   FNB:260
ATHB_IDX = 463
TERM_STATE = CPB
MAINSET_OFHK = FALSE
EXTSET_OFHK = FALSE
DTR = TRUE
SW_SYNC_DATA = FALSE
terminal type = keyset
loop config = 1
kset_display_not_equipped = FALSE
```

Figure 7-5
CP dump example (Part 2 of 4)

```
*** ATHB: 463
AUD_FREE = FALSE
AUD_MARKED = FALSE
AUD_SUSPECT = FALSE
DTSR = FALSE
TDB_IDX = 85
CHB_IDX = 193
BUZZER_ON = TRUE
ACTIVE_ADDR = 0
ACTIVE_SET = MAIN_SET
PS_PE:
  PORT = 3   CHNL = 14   DIR = TWO_WAY  PTYPE = DYNAMIC
PS_PE_COUNT = 1
Wait_CC_Response = FALSE
Queued_In_Idle_Queue = FALSE
Queued_Msg_Idx = 0
start_pt = 0
kset_ring_count = 0
terminal tone = pcm
display control blk:
  active display key id =0
  controlling = FALSE
  data_are_sent = FALSE
  allow_to_send_data = TRUE
  top_line_has_been_updated = FALSE
  bottom_line_has_been_updated = FALSE
  display timer index = 0

*** XDB (linked to ATHB):85
 XDB
NEXT_XDB = 0
AUD_FREE = FALSE
AUD_MARKED = FALSE
AUD_SUSPECT = FALSE
FORMAT = SDB
TERM_NO = 43
KSET SDB DATA:
CONFIG  RING_OPT  KEY_DATA
------  --------  --------
TRUE     TRUE       21
FALSE    FALSE      20
FALSE    FALSE      20
FALSE    FALSE      20
```

Figure 7-5
CP dump example (Part 3 of 4)

```
FALSE    FALSE      20
FALSE    FALSE      20
FALSE    FALSE      20
FALSE    FALSE      20
         FALSE      20
NO_OF_ADDONS = 0
UNIT_TYPE = 2
EXTENSION = FALSE
PROG_MAP = NIL_KEY_ID
TWC_IN_EFFECT = FALSE
CONF30_IN_EFFECT = FALSE

*** EXT_BYTE: 0  CHB: 193
NEXT_CHB = 0   SR_IDX = 0   FNB_IDX = 260   CDB_IDX = 0
AUD_CHB_FREE = FALSE  MARKED = FALSE  SUSPECT = FALSE
INTTID:  NODENO = 2   TERMNO = 43   EXTBYTE = 0
TFA:  IMPL_CID   FCN_SEQ   FCN_CLASS  FCN_ACTIVE   TF_GP_DATA
a       00000A3       2       pset      TRUE       20 81 06 00
b       00000FF       0       nil       FALSE      00 00 00 00
ltime   0000093       1       exec      FALSE      00 00 00 00
timer   00000FF       0       nil       FALSE      00 00 00 00
ntg     000008C       2       pset      TRUE       80 80 80 87
CS_PE:  PORT = 9   CHNL = 13   DIR = TWO_WAY  PTYPE = DYNAMIC
RING_TMR = 0   RING_CHAR = 0  REV_RING_CHAR = 0
   SWACTTED = FALSE  REFLEX_USED = FALSE
  REMOTE SWACT STATE = sync_tfs      INT_MAPSV = FALSE
FLASH_ALLOWED = TRUE     ASSOCIATED_KEY = 0
  PAD_VALUE = 66
CUR_TONE = pcm
IA_CHB_IDX = 0   IA_CHN_BLK = FALSE  IA_NTG_FND = FALSE
NTG_FCN_SEQ = 0    NTG_FCN_TYPE = 0
CDB_FCN_SEQ = 0    CDB_FCN_TYPE = 0
ia_ps = FALSE  aud_tfa = FALSE  aud_tfb = FALSE
swab_inc    = 0   swab_out    = 0
IA_NTG_TMR = 0 DDB_IDX = 21  IA_LS = 0  MDN_RING = FALSE
                                          warble = FALSE

*** TVA : 193
A8 AA A9 C5 A6 CE 00 CD CD 01 21 43 00 00 00 00
00 00 40 33 00 00 00 00 00 B5 00 A0 87 00 08 00
F4 6F 02 00 00 00 00 00 00 01 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Figure 7-5
CP dump example (Part 4 of 4)

```
*** DDB: 21
 DDB
NEXT_DDB = 0
AUD_FREE = FALSE
AUD_MARKED = FALSE
AUD_SUSPECT = FALSE
CALLING INFO:
  VALID_DSP = FALSE
CALLED INFO:
  VALID_DSP = TRUE
  DSP_TYPE = hex_dsp
  DIGIT_COUNT = 8
  DATA: 94841234

*** FNB: 260
AUD_FREE = FALSE
AUD_MARKED = FALSE
AUD_SUSPECT = FALSE
NEXT_FNB = 0
FN_TMR_IDX = 0
STATE = DN_ACTIVE
FTR = DN
RING_SET = TRUE
RING_EXT = FALSE
LOGICAL HOOK STATE = OFFHOOK
DN_STATE = DK_ACT_SUP
DN_REPORT = FALSE
HLD_REPORT = FALSE
PRLS_REPORT = FALSE
PRV_REPORT = FALSE
CPK_LAMP_ON = FALSE
DCPK_LAMP_ON = FALSE
PREV_DN_REPORT = FALSE
PREV_HLD_REPORT = FALSE
CC_DATA_ORIG_SENT = FALSE
DU_STATE = NO_TMR
DU_CNVRT_TIP_RING = FALSE
MWQRY_ON = FALSE
KSMOH_RPT_HLD_DN = FALSE

Dump,Unprot,Trmnl,Athb,Xdb,tVa,Chb,dsP,Fnb,Mod,Brk,Extint,
Swct,Idlq,abtrK,Rmt,*
MP:CP>
```

Trunk call-processing dump

This section describes how to dump CP information involving a trunk.

How to dump CP information involving a trunk

22 Determine the trunk number of the trunk to be inspected. The trunk number is in the form: **DTC <dtc number > <port> <channel>**.

23 Calculate the internal terminal number of the trunk as follows:

```
internal terminal number = (port * 32) + channel + 2
external terminal number = internal terminal number - 1
internal node number     = 1
```

24 Access PMDEBUG. For information on accessing PMDEBUG, refer to "Accessing PMDEBUG" on page 7-1.

25 Dump the CP information associated with the calculated internal terminal number. Refer to "Dumping call processing blocks" on page 7-24.

Find the fields labeled SCTERM and PS_TMSLOT. These values are used in the following steps:

26 Access the SP level by entering **SP**.

27 Access the TRUNKs level by entering **TRUNKS**.

28 Access the AB bit transmit level by entering **TX**.

29 Display the AB transmit memory by entering **DISP <ps_tmslot>**.

30 Reaccess the TRUNKs level by entering *****.

31 Access the AB RECEIVE level by entering **RX**.

32 Display the A bit by entering **DA <scterm>**.

33 Reaccess the TRUNKs level by entering *****.

34 Access the AB RECEIVE level by entering **RX**.

35 Display the B bit by entering **DB <scterm>**.

36 Reaccess the TRUNKS level by entering *****.

37 Display the scan block by entering **SCBLCK <scterm>**.

Channel supervision message dump

This section describes how to dump the CSM with PMDEBUG.

Channel supervision message

The speech/data channels of a DS30 carry eight bits of speech/data between PM. The digital voice on the subscriber line is sampled every 125 µs. This sample is called pulse code modulation (PCM). The PM hardware adds two additional bits to the 8 bit PCM sample before sending the data to the network:

- a channel parity bit
- a channel supervision bit from CSM.

The channel parity bit verifies the network hardware.

The CSM is a 40-bit frame that is transmitted serially over successive network samples. The 40 CSM bits are arranged as follows:

- 8 bits - channel data byte (CDB)
- 8 bits - integrity byte
- 24 bits - framing bits

The integrity byte is used to verify that a valid connection exists through the network. Before the CP sends a network connection control message, each PM is provided with the integrity value that should be received from the other PM and with the values it should transmit. The integrity byte is checked by the CSM task to ensure that a connection exists for the duration of the call.

The remaining 24 bits of the CSM are used for synchronization.

Whenever a dump of CP information is done, CSM information should also be dumped.

To dump CSM information

1 Access PMDEBUG. For information on accessing PMDEBUG, refer to "Accessing PMDEBUG" on page 7-1.

2 Calculate the internal terminal identifier for the terminal you need to trace. Refer to "Calculating internal terminal identifiers" on page 2-6.

3 Access the SP level by entering **SP**.

4 Access the FACM level by entering **FACM**.

5 Access the CSM monitor level by entering **CS(m)**.

6 Determine the internal channel number of the active channel by entering **L(oop) <internal terminal number>**.

7 Display the data structures by entering **DISPLAY <internal channel number>**.

8 Display the hardware registers by entering **REGISTERS <internal channel number>**.

9 Access the integrity level by entering **INTEG**.

Display the integrity hits counter by entering **DISPLAY.**

10 Return to the highest PMDEBUG level by entering *** ***.

For more information on commands at the CSM level, refer to CSM level on page 5-3.

Figure 7-6 on page 7-33 contains an example of a CSM dump.

Figure 7-6
Example of a CSM dump (Part 1 of 5)

```
>pmdebug lgc 0

PMDEBUG MODE - CONNECTING TO PM
WARNING: You now have access to the PM monitor...Proceed
         with caution
LTCMP>

TIme,TAsk,Load,Xprompt,Debug,Swerr,Ipc,Queues,Uspace,Patches,
Msgtr,Chnls,Opf,MTc,Trmtrc,Gdt,DIagnose,Audit,CAudit,CP,SE,
Rcvrmon,MGen,PRfm,OMUnsol.
LTCMP>

>sp

TIme,TAsk,Load,Xprompt,Debug,Swerr,Ipc,QueuTIme,TAsk,Load,
Xprompt,Debug,Swerr,Ipc,Queues,Uspace,Patches,Msgtr,Chnls,Opf,
es,Uspace,Facm,FLq,Msg6x69,Newmsging,
MTc,Trmtrc,Gdt,DIagnose,Audit,CAudit,CP,SE,NWmsgtrc,SYnc,
SChnls,UTr,XBert,Prfm.Rcvrmon,MGen,PRfm,OMUnsol.
LTCSP> LTCMP>

>f

CSm,DS1.
SP:Facm>

>cs

CSM: Display, Reg(HW), Conv,iNteg,Thresh,Loop,Icts,Help,*
SP:CSm>

>l 1256

Found an active channel     -  68
CSM: Display, Reg(HW), Conv,iNteg,Thresh,Loop,Icts,Help,*
SP:CSm>

>l 1004

Found an active channel     -  85

CSM: Display, Reg(HW), Conv,iNteg,Thresh,Loop,Icts,Help,*
SP:CSm>
```

Figure 7-6
Example of a CSM dump (Part 2 of 5)

```
>d 68

init_val = # 80
mask =  # 80
filter = # 4
ret_cid = # 8D
fs = # 2
tfcn = tfn_a
cdb_mode = cdb_scn
SV mapping flag = FALSE
timer index  =  # 0
look check flag = FALSE
scnd_pl flag = FALSE
pause: Do you wish to continue? y/n
SP:CSm>

>y

Extension block data...
  ..  no ext.  block
pause: Do you wish to continue? y/n
SP:CSm>

>y

ref term num = # 4E8
ref node num = # 3
ref ext byte = # 0
ntg idx = # 0
ntg ret = # 8D
ntg fcn seq = # 2
ntg_mode = ntg_chkntg
ntg_ch = TRUE
mfcn = tfcn_all
cdb_masks =  # 80
cdb_llm =  # 80
ntg masks are: ACTIVE
CSM: Display, Reg(HW), Conv,iNteg,Thresh,Loop,Icts,Help,*
SP:CSm>

>d 85
```

Figure 7-6
Example of a CSM dump (Part 3 of 5)

```
init_val = # 80
mask =  # 80
filter = # 4
ret_cid = # 8D
fs = # 1
tfcn = tfn_a
cdb_mode = cdb_scn
SV mapping flag = FALSE
timer index  =  # 0
look check flag = FALSE
scnd_pl flag = FALSE
pause: Do you wish to continue? y/n
SP:CSm>

>y

Extension block data...
  ..  no ext.  block
pause: Do you wish to continue? y/n
SP:CSm>

>y

ref term num = # 3EC
ref node num = # 3
ref ext byte = # 0
ntg idx = # 0
ntg ret = # 8D
ntg fcn seq = # 2
ntg_mode = ntg_chkntg
ntg_ch = TRUE
mfcn = tfcn_all
cdb_masks =  # 80
cdb_llm =  # 80
ntg masks are: ACTIVE

CSM: Display, Reg(HW), Conv,iNteg,Thresh,Loop,Icts,Help,*
SP:CSm>

>r 68

channel = 68
xmit_cdb = # 80
```

Figure 7-6
Example of a CSM dump (Part 4 of 5)

```
xmit_ntg = # F1
rcvd_cdb = # 80
xpec_ntg = # F1
ntg match bit = MATCH
plane select = 1
csm loop around = DISABLED
ntwk loop = DISABLED
parity injection = DISABLED

CSM: Display, Reg(HW), Conv,iNteg,Thresh,Loop,Icts,Help,*
SP:CSm>

>r 85

channel = 85
xmit_cdb = # 80
xmit_ntg = # F1
rcvd_cdb = # 80
xpec_ntg = # F1
ntg match bit = MATCH
plane select = 0
csm loop around = DISABLED
ntwk loop = DISABLED
parity injection = DISABLED

CSM: Display, Reg(HW), Conv,iNteg,Thresh,Loop,Icts,Help,*
SP:CSm>

>n

Display, Clear, *
SP:CSm>

>d

plane \ port                        hitcount
         0  1  2  3  4  5  6  7  8  9  10  11  12  13  14  15
  0   |  0  0  0  0  0  0  0  0  0  0   0   0   0   0   0   0
  1   |  0  0  0  0  0  0  0  0  0  0   0   0   0   0   0   0

Display, Clear, *
SP:CSm>
```

Figure 7-6
Example of a CSM dump (Part 5 of 5)

```
>*

CSM: Display, Reg(HW), Conv,iNteg,Thresh,Loop,Icts,Help,*
SP:CSm>

>*

SP:Facm>

>*

LTCSP>

>quit

NOTE: PMDEBUG will terminate when last request is complete
PMDEBUG TERMINATES
```

CSM threshold query

The CSM uses thresholds to determine when to report a parity or integrity loss. These thresholds can differ among peripherals.

How to query the CSM thresholds in a peripheral

1 Access PMDEBUG. For information on accessing PMDEBUG, refer to "Accessing PMDEBUG" on page 7-1.

2 Access the SP level by entering **SP**.

3 Access the FACM level by entering **F(acm)**.

4 Access the CSM monitor level by entering **CS(m)**.

5 Query the integrity and parity thresholds by entering **T(hresh)**.

6 Return to the highest PMDEBUG level by entering *****.

For more information on commands in the CSM level, refer to CSM Level on page 5-3.

Figure 7-7 on page 7-38 contains an example of querying the CSM threshold.

Figure 7-7
CSM threshold query

```
>pmdebug dtc 0 0

PMDEBUG MODE - CONNECTING TO PM WARNING: You now have access to
the PM monitor...Proceed            with caution LTCMP>

TIme,TAsk,Load,Xprompt,Debug,Swerr,LIpc,Ipc,Patches,Msgtr,
Uspace,Chnls,MTc,DIagnose,Audit,PKtldr,CP,SE,C7tu,PEg,Rcvrmon.
LTCMP>

>sp

TIme,Task,Load,Xprompt,Debug,Swerr,Ipc,Queues,Uspace,
Facm,FLq,Msg6x69,Newmsging,SYnc,SChnls,Trunks,UTr,STr,Prfm.
LTCSP>

>f

CSm,DS1.  SP:Facm>

>cs

CSM: Display,Reg(HW), Conv,iNteg,Thresh,Lop,Icts,Help,* SP:CSM>

>t

Parity = 20 Integrity = 12

CSM: Display,Reg(HW), Conv,iNteg,Thresh,Lop,Icts,Help,* SP:CSM>

> *

LTCMP>
```

Channel blockage dump

This section describes how to dump the channel information from a channel that may be blocked.

How to dump channel information

To dump channel information, perform the following steps:

1 Access PMDEBUG. For information on accessing PMDEBUG, refer to "Accessing PMDEBUG" on page 7-1.

2 Access the CHNLS level by entering **CHNLS**.

3 Access the protected data level by entering **PROT**.

4 Display the node table by entering **NODE**.

5 Display the port table by entering **PORT 0 80**.

6 Display the P-side port map by entering **PS_MAP**.

7 Exit the protected data level by entering *****.

8 Access the unprotected data level by entering **UNPROT**.

9 Display the node status by entering **NODE_STAT**.

10 Display the port status by entering **PORT_STAT**.

11 Access the bimap level by entering **BIMAP**.

12 Display the P-side bimap by entering **PDISPLAY**.

13 Display the IU bimap by entering **IDISPLAY**.

For more information on commands in the CHNLS level, refer to CHNLS Level on page 4-85.

> ***Note:*** Figure 7-8 on page 7-40 contains an example of a channel blockage dump. The output in Figure 7-8 on page 7-40 has been modified slightly due to size constraints.

Figure 7-8
Channel blockage dump (Part 1 of 5)

```
>pmdebug dtc 0 0

PMDEBUG MODE - CONNECTING TO PM
WARNING: You now have access to the PM monitor...Proceed
            with caution
LTCMP>

TIme,TAsk,Load,Xprompt,Debug,Swerr,LIpc,Ipc,Patches,Msgtr,Us-
pace,Chnls,MTc,DIagnose,Audit,PKtldr,CP,SE,C7tu,PEg,Rcvrmon.
LTCMP>

>c
Prot,Unprot,Rlcm_intra.
MP:Chnls>

>p

Node,Port,Spec,H_l_mux,Utr,NX,PRt_ps,Audt,Mn,Cs_map,PS_map,
Iu_map,Ring,ST_cd,RCc_ltc.
MP:Prot>

>n

NODE_TABLE
int extnode n_t type p_t msg_pcl relation n_p s_p e_p s_t
 1    26    641  11  19  pcl_ds30  slave  16   0  15   1
                                                    rct host
                                                    0    1
MP:Prot>

>port 0 80

PORT_TABLE

index p_host  side      spch    msg     msg_chnl type_of_port
   0      0  port_cside spch_cap msg_cap    0     prt_ds30
   1      1  port_cside spch_cap  no_msg    0     prt_ds30
   2      2  port_cside spch_cap msg_cap    0     prt_ds30
   3      3  port_cside spch_cap  no_msg    0     prt_ds30
   4      4  port_cside spch_cap  no_msg    0     prt_ds30
   5      5  port_cside spch_cap  no_msg    0     prt_ds30
   6      6  port_cside spch_cap  no_msg    0     prt_ds30
   7      7  port_cside spch_cap  no_msg    0     prt_ds30
```

Figure 7-8
Channel blockage dump (Part 2 of 5)

```
   8      8  port_cside spch_cap  no_msg    0     prt_ds30
   9      9  port_cside spch_cap  no_msg    0     prt_ds30
  10     10  port_cside spch_cap  no_msg    0     prt_ds30
  11     11  port_cside spch_cap  no_msg    0     prt_ds30
  12     12  port_cside spch_cap  no_msg    0     prt_ds30
  13     13  port_cside spch_cap  no_msg    0     prt_ds30
  14     14  port_cside spch_cap  no_msg    0     prt_ds30
  15     15  port_cside spch_cap  no_msg    0     prt_ds30
  16      0  port_cside no_spch   no_msg    0     prt_nil
  17      0  port_cside no_spch   no_msg    0     prt_nil
  18      0  port_cside no_spch   no_msg    0     prt_nil
  19      0  port_cside no_spch   no_msg    0     prt_nil
  20      0  port_cside no_spch   no_msg    0     prt_nil
  21      0  port_cside no_spch   no_msg    0     prt_nil
  22      0  port_cside no_spch   no_msg    0     prt_nil
  23      0  port_cside no_spch   no_msg    0     prt_nil
   .
   .
   .
  78      0  port_cside no_spch   no_msg    0     prt_nil
  79      0  port_cside no_spch   no_msg    0     prt_nil
  80      0  port_cside no_spch   no_msg    0     prt_nil

MP:Prot>

>*

MP:Chnls>

>u

Bimap,Node_stat,Port_stat,Ch_term,Audit,Iuconn,ASsoc,
Rcc_pcm,Swt.
MP:Unprot>

Bimap,Node_stat,Port_stat,Ch_term,Audit,Iuconn,ASsoc,
Rcc_pcm,Swt.
MP:Unprot>

>n
```

Figure 7-8
Channel blockage dump (Part 3 of 5)

```
NODE_STATUS (slave nodes only)
node        ns        ms[0]         ms[1]
  2          3       ls_busy         ls_busy
  3     --- HOST NODE IS RCC ---
  4     --- HOST NODE IS RCC ---
  5     --- HOST NODE IS RCC ---
  6     --- HOST NODE IS RCC ---
  7          2       ls_running      ls_running
  8          2       ls_running      ls_running

MP:Unprot>
Bimap,Node_stat,Port_stat,Ch_term,Audit,Iuconn,ASsoc,
Rcc_pcm,Swt.
MP:Unprot>

>p

     CS_PORT_STATUS           PS_PORT_STATUS
PORT PORT_STAT MSG_CHNL PORT PORT_STAT MSG_CHNL
  0     okay       1      0     okay       1
  1     busy       0      1     okay       0
  2     busy       1      2     okay       0
  3     busy       0      3     okay       0
  4     busy       0      4     okay       0
  5     busy       0      5     okay       0
  6     busy       0      6     busy       0
  7     busy       0      7     busy       0
  8     busy       0      8     uneq       0
  9     busy       0      9     uneq       0
 10     busy       0     10     busy       0
 11     busy       0     11     busy       0
 12     busy       0     12     uneq       0
 13     busy       0     13     uneq       0
 14     busy       0     14     busy       0
 15     busy       0     15     uneq       0
 16     busy       0     16     uneq       0
 17     uneq       0     17     uneq       0
 18     uneq       0     18     uneq       0
 19     uneq       0     19     uneq       0
                                        IU_PORT_STATUS
                                  PORT PORT_STAT MSG_CHNL
                                     0     uneq       0
                                     1     uneq       0
```

Figure 7-8
Channel blockage dump (Part 4 of 5)

```
                                          2     uneq       0
                                          3     uneq       0
                                          4     uneq       0
                                          5     uneq       0
                                          6     uneq       0
                                          7     uneq       0
                                          8     uneq       0
                                          9     uneq       0
                                         10     uneq       0
                                         11     uneq       0
                                         12     uneq       0
                                         13     uneq       0
                                         14     uneq       0
                                         15     uneq       0
                                         16     uneq       0
                                         17     uneq       0
                                         18     uneq       0
                                         19     uneq       0
MP:Unprot>

>b

CDisplay,PDisplay,IDisplay,CChange,PChange,IChange,*

MP:Bimap>

>pd

PORT          CHNLS (0-31)
 0      3333 3333 3333 3333 3333 3333 3000 3000
 1      3333 3333 3000 3000 3030 0003 0000 3000
 2      3333 3333 3333 3333 3333 3333 3000 3000
 3      3333 3333 3000 3000 3030 0003 0000 3000
 4      3030 0003 0000 3000 3030 0003 0000 3000
 5      3030 0003 0000 3000 3030 0003 0000 3000
 6      3030 0003 0000 3000 3030 0003 0000 3000
 7      3030 0003 0000 3000 3030 0003 0000 3000
 8      3333 3333 3333 3333 3333 3333 3333 3333
 9      3333 3333 3333 3333 3333 3333 3333 3333
10      3333 3333 3333 3333 3333 3333 3333 3333
11      3333 3333 3333 3333 3333 3333 3333 3333
12      3333 3333 3333 3333 3333 3333 3333 3333
13      3333 3333 3333 3333 3333 3333 3333 3333
```

Figure 7-8
Channel blockage dump (Part 5 of 5)

```
14      3333 3333 3333 3333 3333 3333 3333 3333
15      3333 3333 3333 3333 3333 3333 3333 3333
16      3333 3333 3333 3333 3333 3333 3333 3333
17      3333 3333 3333 3333 3333 3333 3333 3333
18      3333 3333 3333 3333 3333 3333 3333 3333
19      3333 3333 3333 3333 3333 3333 3333 3333
CDisplay,PDisplay,IDisplay,CChange,PChange,IChange,*

MP:Bimap>
CDisplay,PDisplay,IDisplay,CChange,PChange,IChange,*

MP:Bimap>

>id

PORT    CHNLS (0-31)
 0      3333 3333 3333 3333 3333 3333 3333 3333
 1      3333 3333 3333 3333 3333 3333 3333 3333
 2      3333 3333 3333 3333 3333 3333 3333 3333
 3      3333 3333 3333 3333 3333 3333 3333 3333
 4      3333 3333 3333 3333 3333 3333 3333 3333
 5      3333 3333 3333 3333 3333 3333 3333 3333
 6      3333 3333 3333 3333 3333 3333 3333 3333
 7      3333 3333 3333 3333 3333 3333 3333 3333
 8      3333 3333 3333 3333 3333 3333 3333 3333
 9      3333 3333 3333 3333 3333 3333 3333 3333
10      3333 3333 3333 3333 3333 3333 3333 3333
11      3333 3333 3333 3333 3333 3333 3333 3333
12      3333 3333 3333 3333 3333 3333 3333 3333
13      3333 3333 3333 3333 3333 3333 3333 3333
14      3333 3333 3333 3333 3333 3333 3333 3333
15      3333 3333 3333 3333 3333 3333 3333 3333
16      3333 3333 3333 3333 3333 3333 3333 3333
17      3333 3333 3333 3333 3333 3333 3333 3333
18      3333 3333 3333 3333 3333 3333 3333 3333
19      3333 3333 3333 3333 3333 3333 3333 3333
CDisplay,PDisplay,IDisplay,CChange,PChange,IChange,*

MP:Bimap>

> * * *

LTCMP>
```

One-way connection dump

This section describes how to determine if a connection is one-way or two-way.

How to determine if a connection is one-way or two-way

To determine if a connection is one-way or two-way, perform the following steps.

1 Access PMDEBUG. For information on accessing PMDEBUG, refer to "Accessing PMDEBUG" on page 7-1.

2 Determine the C-side path end that the terminal is currently using. The pathend is found by dumping CP information. Refer to "Dumping call processing blocks" on page 7-30. The path end is indicated in the call handler block (CHB) information of the CP dump.

3 Access the SP level by entering **SP**.

4 Access the CHNLS level by entering **SCHNLS**.

5 Display the incoming timeswitch by entering **ICTS_SPEC <C-side port> <C-side channel>**.

6 Display the outgoing times witch by entering **OGTS_SPEC <P-side port> <P-side channel>**.

7 Verify the two-way connection by entering **VERIFY <C-side port><C-side channel><P-side port><P-side channel**>.

For more information on the commands at the SCHNLS level, refer to "SCHNLS level" on page 5-62.

Figure 7-9 on page 7-46 contains an example of a one-way connection dump.

Note: The format of the output in Figure 7-9 on page 7-46 has been modified due to size constraints.

Figure 7-9
One-way connection dump (Part 1 of 2)

```
>pmdebug dtc 0 0

PMDEBUG MODE - CONNECTING TO PM
WARNING: You now have access to the PM monitor...Proceed
            with caution
LTCMP>

TIme,TAsk,Load,Xprompt,Debug,Swerr,LIpc,Ipc,Patches,Msgtr,
Uspace,Chnls,MTc,DIagnose,Audit,PKtldr,CP,SE,C7tu,PEg,
Rcvrmon.
LTCMP>

>sp

TIme,Task,Load,Xprompt,Debug,Swerr,Ipc,Queues,Uspace,
Facm,FLq,Msg6x69,Newmsging,SYnc,SChnls,Trunks,UTr,STr,Prfm.
LTCSP>

>sc

O_dump,I_dump,oGts_spec,iCts_spec,Verify,D_intra,S_sync,
Bconv,Xchng,*
SP:SChnls>

>c 2 1

For the INCOMING time switch:
Given cside port = 2 and cside chnl = 1
The corresponding psport = 3 and pside chnl = 1

O_dump,I_dump,oGts_spec,iCts_spec,Verify,D_intra,S_sync,
Bconv,Xchng,*
SP:SChnls>

>g 3 1

For the OUTGOING time switch:
Given pside port = 3 and pside chnl = 1
The corresponding csport = 16 and cside chnl = 1

O_dump,I_dump,oGts_spec,iCts_spec,Verify,D_intra,S_sync,
Bconv,Xchng,*
SP:SChnls>
```

Figure 7-9
One-way connection dump (Part 2 of 2)

```
>v 2 1 16 1

Values may not correspond unless connection are two_way
FOR CSIDE PORT AND CHNL  2  1  AND FOR PSIDE PORT AND CHNL 16 1
EXPECTED VALUE (INCOMING (hex))=00A
                        ACTUAL VALUE (INCOMING(hex))=27D
EXPECTED VALUE (OUTGOING (hex))=0016
                        ACTUAL VALUE (OUTGOING(hex))=24

O_dump,I_dump,oGts_spec,iCts_spec,Verify,D_intra,S_sync,
Bconv,Xchng,*
SP:SChnls>

> *

LTCSP>
```

P-side peripheral loading problems

This section describes the procedure for determining if loading problems are due to the P-side peripheral.

How to display P-side peripheral information

To display P-side peripheral information, perform the following steps:

1 Access PMDEBUG. For information on accessing PMDEBUG, refer to "Accessing PMDEBUG" on page 7-1.

2 Calculate the internal terminal identifier for the terminal to be traced. Refer to "Calculating internal terminal identifiers" on page 2-6.

3 Access the CHNLS level by entering **CHNLS**.

4 Access protected data store by entering **PROT**.

5 Display the node table by entering **NODE**.

The node table contains the internal node number, the external node number, the start port, and the end port for all peripherals on the P-side.

6 Display the port table by entering **PORT <start port> <end port>**.

7 Display the P-side port map by entering **PS_MAP**.

8 Display the special connections by entering **SPEC 20**.

9 Access the SP level by entering **SP**.

10 Access the message level by entering **NEWMSGING**.

11 Access the old message level by entering **OLDMSG**.

12 Access the look level by entering **LOOK**.

13 Display the information for the node by entering **NODE <internal node number>**.

The information obtained from these commands can be used to determine if the P-side peripheral is causing loading problems.

For more information on the commands at the CHNLS level, refer to “CHNLS level” on page 4-85.

For more information on the commands at the NEWMSGING level, refer to NEWMSGING Level on page 5-32.

Figure 7-10 on page 7-49 contains an example of the procedure for P-side peripheral loading problems.

Note: The format of the output in Figure 7-10 on page 7-49 has been modified due to size constraints.

Figure 7-10
P-side peripheral loading problems (Part 1 of 5)

```
>pmdebug dtc 0 0

PMDEBUG MODE - CONNECTING TO PM
WARNING: You now have access to the PM monitor...Proceed
            with caution
LTCMP>

TIme,TAsk,Load,Xprompt,Debug,Swerr,LIpc,Ipc,Patches,Msgtr,
Uspace,Chnls,MTc,DIagnose,Audit,PKtldr,CP,SE,C7tu,PEg,Rcvrmon.
LTCMP>

>c

Prot,Unprot,Rlcm_intra.
MP:Chnls>

>p

Node,Port,Spec,H_l_mux,Utr,NX,PRt_ps,Audt,Mn,Cs_map,PS_map,
Iu_map,Ring,ST_cd,RCc_ltc.
MP:Prot>

>n

NODE_TABLE
int extnode n_t type p_t msg_pcl relation n_p s_p e_p s_t
 1   26    641  11  19  pcl_ds30   slave  16   0  15   1
                                                    rct host
                                                     0    1
MP:Prot>

>port 0 15

PORT_TABLE

index p_host    side       spch    msg     msg_chnl type_of_port
   0       0  port_cside spch_cap msg_cap    0     prt_ds30
   1       1  port_cside spch_cap  no_msg    0     prt_ds30
   2       2  port_cside spch_cap msg_cap    0     prt_ds30
   3       3  port_cside spch_cap  no_msg    0     prt_ds30
   4       4  port_cside spch_cap  no_msg    0     prt_ds30
   5       5  port_cside spch_cap  no_msg    0     prt_ds30
   6       6  port_cside spch_cap  no_msg    0     prt_ds30
```

Figure 7-10
P-side peripheral loading problems (Part 2 of 5)

```
   7       7  port_cside spch_cap  no_msg    0     prt_ds30
   8       8  port_cside spch_cap  no_msg    0     prt_ds30
   9       9  port_cside spch_cap  no_msg    0     prt_ds30
  10      10  port_cside spch_cap  no_msg    0     prt_ds30
  11      11  port_cside spch_cap  no_msg    0     prt_ds30
  12      12  port_cside spch_cap  no_msg    0     prt_ds30
  13      13  port_cside spch_cap  no_msg    0     prt_ds30
  14      14  port_cside spch_cap  no_msg    0     prt_ds30
  15      15  port_cside spch_cap  no_msg    0     prt_ds30
MP:Prot>

>s

specify ps_port (ps_port = 20 for all ports)
MP:Spec>

>20

SPEC_CON_TABLE <entries with special conn will be displayed>
ps_port ps_chnl   cs_port cs_chnl  dir  path_type
     0     1         3      31  two_way rsvd_path
     0     2         0       2  two_way rsvd_path
     0     3         1       3  two_way rsvd_path
     0     4         2       4  two_way rsvd_path
     0     5         3       5  two_way rsvd_path
     0     6         0       6  two_way rsvd_path
     0     7         1       7  two_way rsvd_path
     0     8         2       8  two_way rsvd_path
     0     9         3      10  two_way rsvd_path
     0    10         0      12  two_way rsvd_path
     0    11         1      13  two_way rsvd_path
     0    12         2      14  two_way rsvd_path
     0    13         3      15  two_way rsvd_path
     0    14         0      18  two_way rsvd_path
     0    15         1      19  two_way rsvd_path
     0    16         2      20  two_way rsvd_path
     0    17         3      21  two_way rsvd_path
     0    18         0      23  two_way rsvd_path
     0    19         1      24  two_way rsvd_path
     0    20         2      25  two_way rsvd_path
     0    21         3      26  two_way rsvd_path
     0    22         0      28  two_way rsvd_path
     0    23         1      29  two_way rsvd_path
```

Figure 7-10
P-side peripheral loading problems (Part 3 of 5)

```
      0    24          2      30  two_way rsvd_path
      1     1          0       1  two_way rsvd_path
      1     2          1       2  two_way rsvd_path
      1     3          2       3  two_way rsvd_path
      1     4          3       4  two_way rsvd_path
      1     5          0       6  two_way rsvd_path
      1     6          1       7  two_way rsvd_path
      1     7          2       8  two_way rsvd_path
      1     8          3       9  two_way rsvd_path
      2     1          1       1  two_way rsvd_path
      2     2          2       2  two_way rsvd_path
      2     3          3       3  two_way rsvd_path
      2     4          0       5  two_way rsvd_path
      2     5          1       6  two_way rsvd_path
      2     6          2       7  two_way rsvd_path
      2     7          3       8  two_way rsvd_path
      2     8          0      10  two_way rsvd_path
      2     9          1      11  two_way rsvd_path
      2    10          2      12  two_way rsvd_path
      2    11          3      13  two_way rsvd_path
      2    12          0      15  two_way rsvd_path
      2    13          1      17  two_way rsvd_path
      2    14          2      18  two_way rsvd_path
      2    15          3      19  two_way rsvd_path
      2    16          0      21  two_way rsvd_path
      2    17          1      22  two_way rsvd_path
      2    18          2      23  two_way rsvd_path
      2    19          3      24  two_way rsvd_path
      2    20          0      26  two_way rsvd_path
      2    21          1      27  two_way rsvd_path
      2    22          2      28  two_way rsvd_path
      2    23          3      29  two_way rsvd_path
      2    24          0      31  two_way rsvd_path
      3     1          2       1  two_way rsvd_path
      3     2          3       2  two_way rsvd_path
      3     3          0       4  two_way rsvd_path
      3     4          1       5  two_way rsvd_path
      0     5          2       6  two_way rsvd_path
      0     6          3       7  two_way rsvd_path
      0     7          0       9  two_way rsvd_path
      0     8          1      10  two_way rsvd_path
```

Figure 7-10
P-side peripheral loading problems (Part 4 of 5)

```
Node,Port,Spec,H_l_mux,Utr,NX,PRt_ps,Audt,Mn,Cs_map,PS_map,
Iu_map,Ring,ST_cd,RCc_ltc.
MP:Prot>

>*

MP:Chnls>

>*

LTCMP>

>sp

TIme,Task,Load,Xprompt,Debug,Swerr,Ipc,Queues,Uspace,
Facm,FLq,Msg6x69,Newmsging,SYnc,SChnls,Trunks,UTr,STr,Prfm.
LTCSP>

>n

Adminlogs,Dldata,Netlayer,Oldmsg.
SP:Newmsging>

>o

Look,Update.

SP:Oldmsg>

>l

Node, Routing_data, Signchnl, Misc *
SP:Look>

> n 1

Node # = 26
# trmnls = 641
Starts Trmnl = 1
Node type   = 11
Msg Chnl Idx = 1
```

Figure 7-10
P-side peripheral loading problems (Part 5 of 5)

```
Node, Routing_data, Signchnl, Misc *
SP:Look>

>* * *

LTCSP>
```

Patch query

This section describes the procedure for displaying patch information.

How to query what patches reside in an XPM load

To query what patches reside in an XPM load, perform the following steps:

1 Access PMDEBUG. For information on accessing PMDEBUG, refer to "Accessing PMDEBUG" on page 7-1.

2 Display the patch information by entering **PATCHES**.

For more information on the PATCHES command, refer to "PATCHES level" on page 4-84.

Diagnostics

Diagnostics are run at RTS of the peripheral and by audits. Diagnostics can be run manually with PMDEBUG.

How to perform diagnostics

To perform diagnostics with PMDEBUG, do the following steps:

1 Determine the external node number of the line to be queried. Refer to "Calculating external terminal identifiers" on page 2-1.

2 Access PMDEBUG. For information on accessing PMDEBUG, refer to Accessing PMDEBUG on page 7-1.

3 Access the diagnostics level by entering **DIAGNOSTICS**.

4 Test all of the diagnostics by entering **T FACAUD**.

This command runs the diagnostics performed by the audits. To display a list of diagnostic sets, type **HELP SETS**.

To test a diagnostic that is not part of the audit system, enter **T <diagnostic name>**.

To obtain a list of diagnostics, type **HELP DIAGS**.

For more information on the commands available at the DIAGNOSE level, refer to DIAGNOSE Level on page 4-143.

Figure 7-11 on page 7-54 contains an example output from diagnostic testing.

Figure 7-11
Diagnostic testing

```
LTCMP>

>>>di

Current Mode : Active/Running T(est, CA(rds, CO(nfig, Q(ueues,
H(elp, * MP:DIagnose>

>>>t facaud

ABDIAG     Passed          Res: FF  Parms:  00 00 00 00 00 00 (R)
PSLDIAG    Passed          Res: FF  Parms:  00 00 00 00 00 00 (R)
MSG6X69    Passed          Res: FF  Parms:  00 00 00 00 00 00 (R)
SPCHDIAG   Passed          Res: FF  Parms:  00 00 00 00 00 00 (R)
TSDIAG     Passed          Res: FF  Parms:  00 00 00 00 00 00 (R)
TONEDIAG   Passed          Res: FF  Parms:  00 00 00 00 00 00 (R)
LFMTDIAG   Passed          Res: FF  Parms:  00 00 00 00 00 00 (R)
CSMDIAGG   Passed          Res: FF  Parms:  00 00 00 00 00 00 (R)
UTRDIAG    Passed          Res: FF  Parms:  00 00 00 00 00 00 (R)
STRDIAG    Not run         Res: FF  Parms:  00 00 00 00 00 00 (R)
PSLDIAG    Qual=14 Loc=0002 0000 Exp=00 Act=00
PSLDIAG    Qual=14 Loc=0002 0000 Exp=00 Act=00
PSLDIAG    Qual=14 Loc=0002 0000 Exp=00 Act=00
PSLDIAG    Qual=14 Loc=0002 0000 Exp=00 Act=00
Current Mode : Active/Running
T(est, CA(rds, CO(nfig, Q(ueues, H(elp, *
MP:DIagnose>

>>>*

LTCMP>
```

Message trace

Message trace is a utility program available in XPM peripherals.

How to perform a message trace

To perform a message trace, do the following:

1 Access PMDEBUG. For information on accessing PMDEBUG, refer to “Accessing PMDEBUG” on page 7-1.

2 Calculate the internal terminal identifier for the terminal to be traced. Refer to “Calculating internal terminal identifiers” on page 2-6.

3 Access the message tracing level by entering **M(sgtr)**.

4 Determine the CID name and number of the task messages that will be sent to and received. Obtain a list of valid CIDs by entering **V(alid)**.

5 Initialize the data structures for the selective trace by entering **I(nit)**.

6 Select the messages to be traced by entering **S(elect <source> <dest> <data1...data10>**.

For more information on the source command and on the data byte assignments, refer to the "SELECT command" on page 4-77.

7 Display the message trace criteria by entering **Q(uery**.

To remove an unneeded selected message enter **R(emove <n>** (where n is the entry number corresponding to the message selection criteria). For more information on the remove command, refer to the "REMOVE command" on page 4-76.

8 To display the message trace on the VDU screen, enter **O(utput ON**.

In the previous example, output from the terminal is displayed on the screen. To enable output to be sent to the screen with the Quick Look option, enter **O ON ON**.

9 To stop displaying the output on the VDU screen, enter **O OFF**.

10 Start the message trace for all the selected messages by entering **E(nable**.

11 Make the call. The message trace will begin.

12 Stop tracing for the selected messages by entering **D(isable**.

13 Clear the trace buffer by entering **C(lear**.

14 Remove the select message by entering **K(ill**.

15 Reinitialize the buffer and selects by entering **I(nit**.

16 To query the select messages, enter **Q(uery**.

For more information on commands in the MSGTR level, refer to "MSGTR level" on page 4-72.

Figure 7-12 on page 7-56 contains an example of a message trace. The first message trace displays the messages sent by the TPTP1 task. The second message trace displays all messages sent to and from terminal number 585 (internal terminal number 587, which is 24B in hex).

Figure 7-12
Message trace example (Part 1 of 6)

```
pmdebug lgc 0
PMDEBUG MODE - CONNECTING TO PM
WARNING: You now have access to the PM monitor...Proceed
with caution
LTCMP>

>m

Init,Select,Remove,Enable,Disable,Query,Output,Notime_dump,Clear,
Kill,Valid,DEbug.
MP:Msgtr>

>i

TRACE INITIALIZED --> 200  SNAPSHOTS AVAILABLE
MP:Msgtr>

>s a1 xx

#1  SRC:  MP TPTP1   A1  DEST:   XX
    DATA: XX XX XX XX XX XX XX XX XX XX

MP:Msgtr>

>o on

output turned ON
MP:Msgtr>

>e

MP:Msgtr>

Init,Select,Remove,Enable,Disable,Query,Output,Notime_dump,
Clear,Kill,Valid,DEbug.
MP:Msgtr>

>q

TRACE STATUS:  INITIALIZED, ENABLED, OUTPUT ON, QUICK LOOK OFF
#        SOURCE           DEST                  DATA
-----------------------------------------------------------------
1   MP TPTP1  A1          XX       XX XX XX XX XX XX XX XX XX XX
```

Figure 7-12
Message trace example (Part 2 of 6)

```
MP:Msgtr>

Init,Select,Remove,Enable,Disable,Query,Output,Notime_dump,
Clear,Kill,Valid,DEbug.
MP:Msgtr>

<0001>  SRC:  MP TPTP1   A1    DEST:  MP LCMMTC  08     BUF#: 007
GET: 00:00:45:13.29
 SEND: 00:06:11:.30 RELEASE: 00:00:45:13.30
22DATA:  000C 08A1 0002 0200 19FF 19FF FFFF FFFF
       FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF
       FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF
       FFFF FFFF FFFF FFFF FFFF FFFF 0022 0022
<0002>  SRC:  MP TPTP1   A1    DEST:  MP LCMMTC  08     BUF#: 009
GET: 00:00:45:13.29 SEND: 81:08:17:55.27 RELEASE: 00:00:45:13.30
DATA:  000C 08A1 0002 0201 19FF 19FF FFFF FFFF
       FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF
       FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF
       FFFF FFFF FFFF FFFF FFFF FFFF 0022 0022
<0003>  SRC:  MP TPTP1   A1    DEST:  MP LCMMTC  08     BUF#: 00B
GET: 00:00:45:13.29 SEND: 52:17:23:27.42 RELEASE: 00:00:45:13.30
DATA:  000C 08A1 0283 0300 19FF 19FF FFFF FFFF
       FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF
       FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF
       FFFF FFFF FFFF FFFF FFFF FFFF 0022 0022
<0004>  SRC:  MP TPTP1   A1    DEST:  MP LCMMTC  08     BUF#: 001
GET: 00:00:45:13.29 SEND: 70:00:38:03.86 RELEASE: 00:00:45:13.31
DATA:  000C 08A1 0283 0301 19FF 19FF FFFF FFFF
       FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF
       FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF
       FFFF FFFF FFFF FFFF FFFF FFFF 0022 0022
<0005>  SRC:  MP TPTP1   A1    DEST:  MP TPTP1   A1     BUF#: 011
GET: 00:00:45:13.30 SEND: 00:06:11:22.30 RELEASE: 00:00:45:18.30
DATA:  0007 0000 0000 0000 0E49 FFFF FFFF FFFF
       FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF
       FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF
       FF01 0000 0004 25D6 0032 0002 0000 0032
<0006>  SRC:  MP TPTP1   A1    DEST:  MP NETMSG  CC     BUF#: 009
GET: 00:00:45:18.30 SEND: 83:21:35:36.95 RELEASE: 00:00:45:18.31
DATA:  001B 0000 0001 0100 410B FF00 0401 2B00
       0000 0000 0000 0000 0100 0000 FFFF FFFF
       FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF
       FFFF FFFF FFFF FFFF FFFF FFFF 0022 0022
```

Figure 7-12
Message trace example (Part 3 of 6)

```
Init,Select,Remove,Enable,Disable,Query,Output,Notime_dump,
Clear,Kill,Valid,DEbug.
MP:Msgtr>

>d

MP:Msgtr>

>k

MP:Msgtr>

>*

LTCMP>

>cp e 43 585

INTERNAL NODE     NUMBER = 2
INTERNAL TERMINAL NUMBER = 587

Dump,Unprot,Trmnl,Athb,Xdb,tVa,Chb,dsP,Fnb,Mod,Brk,Extint,
Swct,Idlq,abtrK,Rmt,*
MP:CP>

>*

LTCMP>

>m

Init,Select,Remove,Enable,Disable,Query,Output,Notime_dump,
Clear,Kill,Valid,DEbug.
MP:Msgtr>

>i

TRACE INITIALIZED --> 200  SNAPSHOTS AVAILABLE
MP:Msgtr>

Init,Select,Remove,Enable,Disable,Query,Output,Notime_dump,
Clear,Kill,Valid,DEbug.
MP:Msgtr>
```

Figure 7-12
Message trace example (Part 4 of 6)

```
>s xx xx xx xx xx xx 02 4b

#1  SRC:   XX  DEST:   XX
    DATA: XX XX XX XX 02 4B XX XX XX XX

MP:Msgtr>

>q

TRACE STATUS: INITIALIZED,NOT ENABLED,OUTPUT ON,QUICK LOOK OFF
#        SOURCE           DEST                  DATA
-------------------------------------------------------------
1        XX               XX      XX XX XX XX 02 4B XX XX XX XX

>o on

output turned ON
MP:Msgtr>

>e

MP:Msgtr>
<0001>  SRC:  SP nilname 07    DEST:  SP FLWO    B7     BUF#: 007
GET: 00:00:52:14.07 SEND: 00:06:11:22.30 RELEASE: - - - - - - -
DATA:  0010 B707 024B 0200 0B00 0000 7BAD B729
       B1FF FFFF FFFF FFFF FFFF FFFF FFFF FFFF
       FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF
       FFFF FFFF FFFF FFFF FFFF FFFF 0008 0008
<0002>  SRC:  SP nilname 07    DEST:  MP TPTP3   A4     BUF#: 007
GET: - - - - - - -  SEND: 00:00:52:14.09 RELEASE: 00:00:52:14.09
DATA:  000D A407 024B 0200 0B00 0000 7BAD B729
       B1FF FFFF FFFF FFFF FFFF FFFF FFFF FFFF
       FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF
       FFFF FFFF FFFF FFFF FFFF FFFF 0008 0008
<0003>  SRC:  MP TPTP3   A4    DEST:  MP NETMSG  CC     BUF#: 00A
GET: 00:00:52:14.09 SEND: 52:17:23:27.42 RELEASE: 00:00:52:14.10
DATA:  000D 0000 024B 0200 9C00 00C0 0000 FFFF
       FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF
       FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF
       FFFF FFFF FFFF FFFF FFFF FFFF 0022 0022
<0004>  SRC:  SP NETMSG  CC    DEST:  SP FLWT    1F     BUF#: 00A
GET: 00:00:52:14.12 SEND: 70:00:38:03.86 RELEASE: - - - - - - -
DATA:  002C 8A04 024B 0200 00DE A1CC 0E08 0015
```

Figure 7-12
Message trace example (Part 5 of 6)

```
       1414 1414 1414 1414 0101 8080 652D 6583
       94E8 0EAA E306 8301 0099 693C 00FF FFFF
       FFFF FFFF FFFF FFFF FFFF FFFF 0008 0008
<0005>  SRC:  SP NETMSG  CC    DEST:  MP CIDSERV 83     BUF#: 00A
GET: - - - - - - -  SEND: 00:00:52:14.13 RELEASE: 00:00:52:14.17
DATA:  002C 8304 024B 0200 00DE A1CC 0E08 0015
       1414 1414 1414 1414 0101 8080 652D 6583
       94E8 0EAA E306 8301 0099 693C 00FF FFFF
       FFFF FFFF FFFF FFFF FFFF FFFF 0008 0008
<0006>  SRC:  MP MPCHNLS 8A    DEST:  SP CHNL    9B     BUF#: 00D
GET: 00:00:52:14.14 SEND: 00:00:52:14.15 RELEASE: 00:00:52:14.15
DATA:  0007 0000 024B 0200 C010 FFFF FFFF FFFF
       FFFF 3846 3846 FFFF FFFF FFFF FFFF FFFF
       FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF
       FFFF FFFF FFFF FFFF FFFF FFFF 0022 0022
<0007>  SRC:  MP MPCHNLS 8A    DEST:  SP CHNL    9B     BUF#: 00B
GET: 00:00:52:14.14 SEND: 00:00:52:14.15 RELEASE: 00:00:52:14.15
DATA:  0007 0000 024B 0200 C000 FFFF FFFF FFFF
       FFFF 2846 3865 FFFF FFFF FFFF FFFF FFFF
       FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF
       FFFF FFFF FFFF FFFF FFFF FFFF 0022 0022
<0008>  SRC:  MP MPCHNLS 8A    DEST:  SP CHNL    9B     BUF#: 00B
GET: 00:00:52:14.15 SEND: 00:00:52:14.16 RELEASE: 00:00:52:14.17
DATA:  0007 0000 024B 0200 C006 FFFF FFFF FFFF
       FFFF 3846 0000 FFFF FFFF FFFF FFFF FFFF
       FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF
       FFFF FFFF FFFF FFFF FFFF FFFF 0022 0022
<0009>  SRC:  MP TPTP3   A4    DEST:  MP MSGDMSX 84     BUF#: 00D
GET: 00:00:52:14.15 SEND: 00:06:11:22.30 RELEASE: 00:00:52:14.17
DATA:  0038 03A4 024B 0200 070D 0102 E607 E610
       070B FF0F AD0F 8D0F B50F 9C0F 9D0F 9B07
       0D01 02E6 07E6 0807 0BFF 0F60 0F6C 0B08
       17FF 00FF FF02 7BAD 01FF FFFF 0022 0022
<000A>  SRC:  MP CIDTEQ  8D    DEST:  SP CSM     8C     BUF#: 007
GET: 00:00:52:14.16 SEND: 00:00:52:14.17 RELEASE: 00:00:52:14.18
DATA:  002D 8C8D 024B 0200 22FF 0001 0000 0080
       0000 0000 0000 0000 0000 0000 0000 0000
       0000 0000 0000 3865 0000 0000 0000 FFFF
       FFFF FFFF FFFF FFFF FFFF FFFF 0022 0022
<000B>  SRC:  SP nilname 07    DEST:  SP FLWO    B7     BUF#: 007
GET: 00:00:52:17.50 SEND: 04:11:57:08.54 RELEASE: - - - - - - -
DATA:  0010 B707 024B 0200 0B03 000F D2AE FF29
       B1FF FFFF FFFF FFFF FFFF FFFF FFFF FFFF
```

Figure 7-12
Message trace example (Part 6 of 6)

```
       FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF
       FFFF FFFF FFFF FFFF FFFF FFFF 0008 0008

MP:Msgtr>

>d

MP:Msgtr>

>k

trace KILLED

Init,Select,Remove,Enable,Disable,Query,Output,Notime_dump,
Clear,Kill,Valid,DEbug.
MP:Msgtr>

>*

LTCMP>
```

Inserting messages

This section describes how to insert IPC messages into the XPM. This procedure is useful for inserting faults and testing the system responses.

CAUTION

The IPC level should *not* be used on a live switch. Users of this level should have a thorough knowledge of XPM tasks and messaging. Incorrect data provided in the message body can cause the peripheral to trap or to go down, causing severe service interruption.

How to insert messages into an XPM

To insert messages into an XPM, do the following:

1 Access the PMDEBUG subsystem by entering **PMDEBUG N <node number> <unit number>**.

2 Access the message tracing level by entering **M(sgtr)**.

3 Determine the CID name and number of the sending and receiving tasks messages. Obtain a list of valid CIDs by entering **V(alid)**.

4 Leave the MSGTR level by entering *****.

5 Access the IPC level by entering **I(pc)**.

6 Obtain an IPC buffer by entering **G(et**.

7 Fill the working buffer for the destination CID by entering **F(ill**.

At the prompt, define the destination CID by entering **D(est_cid**.

At the prompt, enter the CID.

At the prompt, input the contents of the buffer by typing **C(ontents)** and the contents. For information on the format of the message data byte assignments, refer to Figure 4-23 on page 4-79.

Leave the FILL level by entering *****.

8 Define the source CID by entering **F(ill**.

At the prompt, define the source CID by entering **S(rc_cid.**

At the prompt, enter the CID.

At the prompt, input the contents of the buffer. For information on the format of the message data byte assignments, refer to Figure 4-23 on page 4-79.

C(ontents

Leave the FILL level by entering *****.

9 Send the inserted message by entering **S**.

For more information on commands in the IPC level, refer to the “IPC Level” on page 4-61.

Figure 7-13 on page 7-63 contains an example of getting and sending an IPC buffer.

Figure 7-13
Example of getting and sending an IPC buffer (Part 1 of 2)

```
pmdebug lgc 0

PMDEBUG MODE - CONNECTING TO PM
WARNING: You now have access to the PM monitor...Proceed with
caution
LTCMP>

TIme,TAsk,Load,Xprompt,Debug,Swerr,Ipc,Queues,Uspace,Patches,
Msgtr,Chnls,Opf,MTc,Trmtrc,Gdt,DIagnose,Audit,CAudit,CP,SE,
Rcvrmon,MGen,PRfm,OMUnsol.
LTCMP>

>ipc

View,Fill,Get,Send,Release,Queue,Dequeue,Inventory,Copy,Test.
MP:Ipc>

>g

ipc buffer # = 3

MP:Ipc>

View,Fill,Get,Send,Release,Queue,Dequeue,Inventory,Copy,Test.
MP:Ipc>

>f

working, or buffer number

LTCMP>

>working
View,Dest_cid,Src_cid,Contents.
MP:Fill>

>d

Input dest cid
MP:Dest_cid>

>a1
```

Figure 7-13
Example of getting and sending an IPC buffer (Part 2 of 2)

```
MP:Fill>

>c

Input contents start & data
MP:Contents>

>00 00 00 00 02 4b 02 00 ff ff

MP:Fill>

>v

buffer 3  dest: A1 TPTP1  src: 0 LCMLOAD
00 00 00 02  4B 02 00 FF  FF FF FF FF  FF FF FF FF
FF FF FF FF  FF FF FF FF  FF FF FF FF  FF FF FF FF
FF FF FF FF  FF FF FF FF  FF FF FF FF  FF FF FF FF
FF FF FF FF  FF FF FF FF  FF FF FF FF  00 09 00 09
MP:Fill>

View,Dest_cid,Src_cid,Contents.
MP:Fill>

>*

MP:Ipc>

View,Fill,Get,Send,Release,Queue,Dequeue,Inventory,Copy,Test.
MP:Ipc>

>s

MP:Ipc>
```

Program execution trace (Call trace)

A program execution trace shows the names of all procedures called by a task, the module where the procedure is found, and timing information.

How to perform a call trace

To perform a program execution trace, enter the following:

1 Access PMDEBUG. For information on accessing PMDEBUG, refer to “Accessing PMDEBUG” on page 7-1.

2 Access the DEBUG level by entering **D(ebug**.

3 Access the CALL TRACE level by entering **CA(lltrace**.

4 Allocate buffers for storing the trace information by entering `A(lloc`.

Allocate the default number of buffers by entering **D**.

5 Select the specific terminals to trace by entering the following:

a. Access the SELECTRACE level by entering **SE(lectrace**.

b. Query to verify that the SELECTRACE tool is disabled by entering **Q(uery**.

c. Enable selective tracing by entering **ON**.

d. Enter the internal terminal number to be traced: **n** (where n is an internal terminal number).

e. Return to the CALLTRACE level by entering *** ***.

6 Start the call trace by entering **S(tart**.

At the prompt, enter the name of the task to trace. For call processing, enter **TPT**.

At the next prompt, enter the number of procedure calls to skip. If you want the trace to start immediately, enter **0**.

7 Make the call.

8 Query the status of the call trace by entering **Q(uery**.

9 Stop the call trace by entering **E(nd**.

10 Dump the contents of the buffer by entering **DU(mp**.

You are prompted for the output format. Choose the format of the output.

11 Deallocate the buffers by entering **DE(allocate**.

For more information on the commands at the CALLTRACE level, refer to “CALLTRACE level” on page 4-12.

Figure 7-14 on page 7-66 contains an example of a selective call trace.

Note: In this example, the DEPTH column has been removed.

Figure 7-14
Selective call trace example (Part 1 of 4)

```
PMDEBUG LGC 0
PMDEBUG MODE - CONNECTING TO PM
WARNING: You now have access to the PM monitor...Proceed
         with caution
LTCMP>

TIme,TAsk,Load,Xprompt,Debug,Swerr,Ipc,Queues,Uspace,Patches,
Msgtr,Chnls,Opf,MTc,Trmtrc,Gdt,DIagnose,Audit,CAudit,CP,SE,
Rcvrmon,MGen,PRfm,OMUnsol.
LTCMP>

>d

DM,FIndproc,SM,SW,Trapinfo,STopontrap,Cleartrap,Zerodiv,Rom,
Info,TRCpoints,CAlltrace,Availtime,COverage,TEmpstore.
MP:Debug>

>ca

Alloc,Start,End,Query,Verify,DUmp,Notime_dump,Reset,PBench,
AMsbench,DEalloc,SElectrace.
MP:CAlltrace>

>a

Mem available for 3875 bufs
# to allocate for proc calls?(”D” for default)
MP:Alloc>

>d

Space for 800  proc calls & 1648 table entries.
MP:CAlltrace>

>se

On,OFf,Query.
MP:SElectrace>

>q

Selectrace turned off
MP:SElectrace>
```

Figure 7-14
Selective call trace example (Part 2 of 4)

```
On,OFf,Query.
MP:SElectrace>

>on

select the terminal number ( 0-7055 )
MP:On>

>43

select the terminal number ( 0-7055 )
MP:On>

>*

MP:SElectrace>


On,OFf,Query.
MP:SElectrace>

>*

MP:CAlltrace>

>s tpt

Tracing: TPT
No of proc calls to ignore before starting call trace?
MP:Start>

>0

Selective
Tracing started...
MP:CAlltrace>

Alloc,Start,End,Query,Verify,DUmp,Notime_dump,Reset,PBench,
AMsbench,DEalloc,SElectrace.
MP:CAlltrace>

>e
```

Figure 7-14
Selective call trace example (Part 3 of 4)

```
Selective
Tracing stopped.
MP:CAlltrace>

>du

630 of 1648 table entries used.
Thinking
Output ==> Trace, Summary, Both, Vomit, *
MP:DUmp>

>t

#

#

#

#

#

#

#

#

#

#

#

#

#

Proc call trace for TID: 0022 0022 - TPT
Segment     Procedure Include Time Exclude Time  Time Stamp
123 TPTASK   23  TPTORIGC  0   0 509   0   0 509   02:32:40.34
123 TPTTASK  22  TPTHDLR   1 660 979   0   0 957   02:32:40.34
126 TPTLCCA  31  SOCTRAFF  0   0 325   0   0 126   02:32:40.34
126 TPTLCCA  29  SOCAUDHA  0   0 198   0   0  94   02:32:40.34
```

Figure 7-14
Selective call trace example (Part 4 of 4)

```
126 TPTLCCA  15  SOCXMITL  0   0 104   0   0 104   02:32:40.34
 25 DEBUGPER 23  DISABLEC  1 659 696   0   0 187   02:32:40.34
 25 DEBUGPER 15  TRACE     1 659 509   1 659 509   02:32:40.34
123 TPTTASK   5  FIATSTAR  0   5 621   0   0 202   02:32:42.00
 15 LNGIP|C   5  GETPTRPR  0   0 115   0   0 115   02:32:42.00
  .
  .
  .
123  TPTTASK  23  TPTORIGC 0   0 547     0   0 547   02:32:55.88
123  TPTTASK  22  TPTHDLR  6 241 163     0   0 858   02:32:55.88
126  TPTLCCA  31  SOCTRAFF 0   0 214     0   0 123   02:32:55.88
126  TPTLCAA  23  GDTRETXO 0   0  91     0   0  91   02:32:55.88
 25  DEBUGPER 23  DISABLEC 6 240  90     0   0 190   02:32:55.88
 25  DEBUGPER 15  TRACE    6 239 900     6 239 900   02:32:55.88
 80  TPTSUTL  35  SETMDBPT 0   0 509     0   0 343   02:33:02.12
 80  TPTSUTL  34  SETLINEC 0   0 166     0   0 166   02:33:02.12
126  TPTLCAA  27  TARGETGD 0   0 194     0   0  92   02:33:02.12
126  TPTLCAA  26  TRGTSOCR 0   0 102     0   0 102   02:33:02.12

MP:CAlltrace> Alloc,Start,End,Query,Verify,DUmp,Notime_dump,
Reset,PBench,AMsbench,DEalloc,SElectrace.
MP:CAlltrace>

>de

Bufs de-allocated.

MP:CAlltrace>

>quit

NOTE: PMDEBUG will terminate when last request is complete
PMDEBUG TERMINATES
```

Display call processing data blocks

This section describes the procedure for displaying CP data blocks.

How to display CP data blocks

To display XPM call processing data block contents, perform the following steps:

1 Determine the external node number of the line to be queried. Refer to “Calculating external terminal identifiers” on page 2-1.

2 Access PMDEBUG. For information on accessing PMDEBUG, refer to "Accessing PMDEBUG" on page 7-1.

3 Calculate the internal terminal identifier for the terminal to be traced. Refer to "Calculating internal terminal identifiers" on page 2-6.

4 Access the CP level by entering **CP**.

5 Dump all the call processing information for a live call by entering **D(ump**.

For more information on dumping the individual call processing block, refer to "Commands" at the CP level on page 4-172.

6 Dump the UNPROT data for the internal terminal number by entering **U <internal terminal number>**.

7 Dump the names and numbers of data blocks linked to the internal terminal number by entering **T <internal terminal number>**.

8 Dump the contents of the Active Terminal Header Block (ATHB) by entering **A <internal node number>**.

9 Dump the contents of the X Data Block (XDB) by entering **X <internal node number>**.

10 Dump the contents of the Terminal Variable Area (TVA) by entering **V <internal node number>**.

11 Dump the contents of the Call Header Block (CHB) by entering **C <internal node number>**.

12 Dump the contents of the Function Block (FNB) by entering **F <internal node number>**.

For more information on commands at the CP level, refer to CP Level on page 4-165.

Inserting and dumping a tracepoint

This section describes how to insert a tracepoint and dump the information gathered at the tracepoint.

CAUTION

Observe the following cautions:

- To prevent two different tools from inspecting the same instructions, the Tracepoint tool should not be used simultaneously with other tools that inspect code.
- Tracepoints can be set in any code other than the tracepoint code or code that is used by both the MP and the SP or both the MP and the FP.
- Tracepoints should not be placed in P-code.

How to insert and dump a tracepoint

To insert and dump a tracepoint, perform the following steps:

1 Access PMDEBUG. For information on accessing PMDEBUG, refer to "Accessing PMDEBUG" on page 7-1.

2 Access the DEBUG level by entering **D(ebug**.

3 Determine the location of the procedure to be traced by entering **FI(ndproc <procedure name>**.

4 Disassemble the segment found with the FINDPROC command to determine the address at which the tracepoint will be sent by entering **DM S <segment number> <procedure number>.**

5 Access the TRCPOINTS level by entering **TRC(points**.

6 Assign a new tracepoint by entering **A(ssign**.

Specify that the tracepoint will be new by entering **N(ew.**

You will be prompted for information concerning the tracepoint being defined. At the prompts enter **Y** if the information described by the prompt will be gathered at the tracepoint or **N** if the information is not gathered.

Quit from the ASSIGN level by entering *****.

7 Display the information associated with the tracepoint by entering **E(xamine n** (where n is the tracepoint number).

8 Insert the tracepoint into the code by entering **I(nstall n** (where n is the tracepoint number).

9 Make the call.

10 When the tracepoint examination is finished, remove the tracepoints from the code. Remove one tracepoint by entering **R)emove n** (where n is the tracepoint number).

If no other users have defined tracepoints, remove all the tracepoints by entering **P)urge_all**.

11 Display the information associated with the tracepoint by entering **D(mp n** (where n is the tracepoint number).

You will be prompted to either continue displaying information gathered at the tracepoint or to stop displaying information. Enter **Y** to continue displaying information or **N** to stop displaying information.

12 Verify that the tracepoints were removed by entering **E)xamine n** (where n is the tracepoint number).

For more information on commands at the TRCPOINTS level, refer to “TRCPOINTS level” on page 4-40.

Figure 7-15 on page 7-73 contains an example of setting and dumping a tracepoint.

Figure 7-15
Example of inserting and dumping a tracepoint (Part 1 of 6)

```
PMDEBUG LGC 0
PMDEBUG MODE - CONNECTING TO PM
WARNING: You now have access to the PM monitor...Proceed
         with caution
LTCMP>

TIme,TAsk,Load,Xprompt,Debug,Swerr,Ipc,Queues,Uspace,Patches,
Msgtr,Chnls,Opf,MTc,Trmtrc,Gdt,DIagnose,Audit,CAudit,CP,SE,
Rcvrmon,MGen,PRfm,OMUnsol.
LTCMP>

>d

DM,FIndproc,SM,SW,Trapinfo,STopontrap,Cleartrap,Zerodiv,Rom,
Info,TRCpoints,CAlltrace,Availtime,COverage,TEmpstore.
MP:Debug>

>fi tpthdlr

Seg : TPT       110       Proc : TPTHDLR  46

MP:Debug>

>dm s 110 46

Seg : TPT       110       Proc : TPTHDLR   46
1A6476 (000):  *EXIT   INTERP              F7D7
1A6478 (002):  MOVE    #0001,4534(A4)      397C 0001 4534
1A647E (008):  CLR     0270(A4)            426C 0270
1A6482 (00C):  BTST    #00,4535(A4)        082C 0000 4535
1A6488 (012):  BEQ     A6FCC               6700 0B42
1A648C (016):  CLR     3D0E(A4)            426C 3D0E
1A6490 (01A):  LEA     4244(A4),A1         43EC 4244
1A6494 (01E):  LEA     4108(A4),A2         45EC 4108

1A6498 (022):  MOVEQ   #43,D1              7243

Cont.?(Y,<cr>)
MP:DM>

MP:Debug>
```

Figure 7-15
Example of inserting and dumping a tracepoint (Part 2 of 6)

```
DM,FIndproc,SM,SW,Trapinfo,STopontrap,Cleartrap,Zerodiv,Rom,
Info,TRCpoints,CAlltrace,Availtime,COverage,TEmpstore.
MP:Debug>

>trc

Assign,Hildatps,Examine,Install,Clear,Dump,Remove,Purge_all,
Tracetest.
MP:TRCpoints>

>a

New,Copy,Modify.
MP:Assign>

>n

TP # (0-15)
MP:New>

>0

Enter absolute address or S seg(#,name) proc(#,name) offset
MP:New>

>1A6478

Select the options you want
- Traceback (y/n), and stack traceback (s)
MP:New>

>Y

- Reg dump (y/n)
MP:New>

>y

- Mem location display (y/n)
MP:New>

>y
- Do you want a (L)ongword or (B)yte
```

Figure 7-15
Example of inserting and dumping a tracepoint (Part 3 of 6)

```
MP:New>

>l

Enter absolute address, or G offset
MP:New>

>g

Enter offset
MP:New>

>329a

-Do you want a (L)ongword or (B)yte
MP:New>

>*

- Display longword on stack (y/n)
MP:New>

>n

offset (hexadecimal)
MP:New>

>10

- Conditional (y/n)
MP:New>

>n

- Survive restarts (y/n)
MP:New>

>n

MP:Assign>

New,Copy,Modify.
MP:Assign>
```

Figure 7-15
Example of inserting and dumping a tracepoint (Part 4 of 6)

```
>*

MP:TRCpoints>

Assign,Hildatps,Examine,Install,Clear,Dump,Remove,Purge_all,
Tracetest.
MP:TRCpoints>

>i 0

**DONE TP inserted
MP:TRCpoints>

>e 0

Trace point located at 1A6478
- Display longword off stack
     - offset - 10
- Reg dump
- Traceback
- Display of locations:
087F30 <L>
MP:TRCpoints>

>i 0

**Done TP inserted
MP:TRCpoints>

Assign,Hildatps,Examine,Install,Clear,Dump,Remove,Purge_all,
Tracetest.
MP:TRCpoints>

>r 0

TP removed.
MP:TRCpoints>

>d 0

Trace point :  0     Hit : 1     Address : 1A6478
Time : 00:01:12:25.61.3303
Procedure traceback
```

Figure 7-15
Example of inserting and dumping a tracepoint (Part 5 of 6)

```
called from : 1A7048  TPT      110   ACCEPTSU  45   offset: #16
              1A71D4  TPT      110   TPTTASK   29   offset: #176
Memory dump
087F30 --> 00160000
Register dump
PC  = 001A6478  SR  = 0104      US  = 00053DDC  SS  = 001DF3F0
D0  = 001A6478  D1  = 001A0001  D2  = 00050000  D3  = 0001003A
D4  = 00053E46  D5  = 000532C8  D6  = 00230023  D7  = 001A6FF0
A0  = 00002864  A1  = 001A6FF0  A2  = 00053DDC  A3  = 001A6478
A4  = 000D7FC8  A5  = 00053DE0  A6  = 000015BA  A7  = 00053DDC
Stack display
Offset : 10   Value --> 00002864
continue (y/n)
MP:Dump>

>y

Trace point :  0     Hit : 2     Address : 1A6478
Time : 00:01:12:26.61.3278
Procedure traceback
called from : 1A7048  TPT      110   ACCEPTSU  45   offset: #16
              1A71D4  TPT      110   TPTTASK   29   offset: #176
Memory dump
087F30 --> 00160000
Register dump
PC  = 001A6478  SR  = 0104      US  = 00053DDC  SS  = 001DF3F0
D0  = 001A6478  D1  = 001A0001  D2  = 00050000  D3  = 0001003A
D4  = 00053E46  D5  = 000532C8  D6  = 00230023  D7  = 001A6FF0
A0  = 00002864  A1  = 001A6FF0  A2  = 00053DDC  A3  = 001A6478
A4  = 000D7FC8  A5  = 00053DE0  A6  = 000015BA  A7  = 00053DDC
Stack display
Offset : 10   Value --> 001A6478
continue (y/n)
MP:Dump>

>y

Trace point :  0     Hit : 3     Address : 1A6478
Time : 00:01:12:27.61.3679
Procedure traceback
called from : 1A7048  TPT      110   ACCEPTSU  45   offset: #16
              1A71D4  TPT      110   TPTTASK   29   offset: #176
```

Figure 7-15
Example of inserting and dumping a tracepoint (Part 6 of 6)

```
Memory dump
087F30 --> 00160000
Register dump
PC  = 001A6478  SR  = 0104      US  = 00053DDC  SS  = 001DF3F0
D0  = 001A6478  D1  = 001A0001  D2  = 00050000  D3  = 0001003A
D4  = 00053E46  D5  = 000532C8  D6  = 00230023  D7  = 001A6FF0
A0  = 00002864  A1  = 001A6FF0  A2  = 00053DDC  A3  = 001A6478
A4  = 000D7FC8  A5  = 00053DE0  A6  = 000015BA  A7  = 00053DDC
Stack display
Offset : 10   Value --> 001A6478
continue (y/n)
MP:Dump>

>n

MP:TRCpoints>

Assign,Hildatps,Examine,Install,Clear,Dump,Remove,Purge_all,
Tracetest.
MP:TRCpoints>

>p

**DONE
MP:TRCpoints>

>e 0
TP not assigned
MP:TRCpoints>
QUIT
NOTE: PMDEBUG will terminate when last request is complete
PMDEBUG TERMINATES
```

List of terms

Absolute address

An address in a computer language that identifies a storage location or a device without the use of any intermediate reference; an address that is permanently assigned by the machine designer to a storage location. Synonymous with explicit address, machine address, specific address.

ATHB

Active terminal header block

Audit

A process that verifies the integrity of the system and may also fix errors when error conditions are detected.

Call handler block (CHB)

A data block that contains routing and translation information.

CC

Central control

CCS7

Common Channel Signaling No. 7

Central control (CC)

Comprises the data processing functions of the DMS-100 Family, with associated data store and program store.

CHB

Call handler block

Channel supervision message (CSM)

A 40-bit message that is received and transmitted once every 5 ms on each connected voice channel of a peripheral module. The CSM contains a connection data byte, which includes the channel supervision bit and an integrity byte, which ensures call path integrity.

Channel supervision bit (CSB)

The CSB is used to transmit supervision status on each connected voice channel of the peripheral modules. CSB forms part of the channel supervision message.

CI

Command interpreter

CID

Communication identifier

Command interpreter (CI)

A support operating system component that functions as the main interface between machine and user. Its principal roles are as follows:

- to read lines entered by a terminal user
- to break each line into recognizable units
- to analyze the units
- to recognize command input-numbers on the input lines
- to invoke these commands

Common Channel Signaling No. 7 (CCS7)

A Canadian version of signaling system 7. A version of signaling system #7, developed for North American use.

Communication identifier (CID)

A unique name that identifies a single task.

Continuity

A tone sent over a line by an originator and looped back by the terminator to verify that the tone received back by the originator is in an acceptable range.

CSB

Channel supervision bit

CSM

Channel supervision message

Dial pulse (DP)

Method of transmitting signaling information from a telephone set or trunk circuit. Dial pulses are generated by alternately opening and closing a Contrast with Dual-Tone Multifrequency Dialing.

Direct Memory Access (DMA)

A device for transferring blocks of continuous data to and from memory at a high rate.

Directory number (DN)

The full complement of digits required to designate a subscriber's station in one NPA - usually a three-digit central office code followed by a four-digit station number.

DMA

Direct memory access

DN

Directory number

DP

Dial pulse

DTMF

Dual-Tone Multifrequency

Dual-Tone Multifrequency (DTMF) Signaling

A signaling method employing set combinations of two specific voice-band frequencies, one of which is selected from a group of four low frequencies, and the other from a group of three or four relatively high frequencies.

Exclude time

Number of microseconds spent in a procedure, excluding procedure calls.

Exec

A string of primitives that define PP procedures.

Extract

A user removes specific messages from the I/O message stream.

FNB

Function block

Hook

1) An element added to old software to enable it to recognize new features that did not exist when the older software was written. 2) An element of a software module that accesses other optional modules. Responsible for evaluating a conditional dependence.

Include time

The number of microseconds spent in a procedure, including procedure calls.

Insert

A user puts a message into the I/O message stream.

Interprocessor communication (IPC)

Software that allows tasks in the SP or the MP to communicate with each other by sending and receiving buffers containing messages.

I/O

Incoming and/or outgoing

Intercept

1) A user records what messages are sent in the I/O message stream. 2) A user prevents messages from reaching a PM or the CC.

IPC

Interprocessor communication

LEN

Line equipment number

Line equipment number (LEN)

An identifier composed of the site, frame number, unit number, drawer number, and circuit number. For example, the LEN HOST 00 0 05 08 has the site HOST, frame number 00, unit number 0, drawer number 05, and circuit number 08.

LIPC

Long interprocessor communication

Long interprocessor communication (LIPC)

Intertask/intratask communication in an XPM using buffers longer than 64 bytes. Used for messaging in CCS7 and ISDN. See Interprocessor communication.

Master processor (MP)

In DMS, the processor that contains the instruction set that implements the tasks assigned by the central control software. The MP carries out all high-level tasks.

Message

The unit of information-transfer between nodes in the DMS system. A message is incoming if it is sent from a peripheral to the CC and outgoing if it is sent from the CC to a peripheral. Messages are never sent between peripherals.

I/O messages should not be confused with SOS messages, which are used for information transfer between processes in the CC.

Message format

Rules for the placement of portions of a message, such as message header, address, text, end-of-message indication, and error-detecting bits.

MF

Multifrequency

MMU

Memory management unit

MP

Master processor

Multifrequency (MF)

A method that makes use of pairs of standard tones to transmit signaling codes, digit pulsing, and coin-control signals.

Network module (NM)

The basic building-block of the DMS-100 Family switching network. The NM accepts incoming calls and, using connection instructions from the central control complex, connects them to the appropriate outgoing channels. Activities in the NM are controlled by the network message controllers.

NM

Network module

Node

Any unit that can accept or originate messages.

Node number

A system-assigned number unique to a node.

Opcode

The identifier of a specific operation to be performed.

Primitive

A string of opcodes for specific operations that the PP will perform.

PCM

Pulse code modulation

Peripheral module (PM)

A generic term referring to all hardware modules of the DMS-100 Family switches that provide interfaces with external line, trunk, or service

facilities. PMs contain peripheral processors, which perform local routines, thus relieving the load on the central processing unit.

Peripheral module intercept system test (PMIST)

A debugging tool that traces messages between the peripheral modules.

Peripheral processor (PP)

Hardware devices in the peripheral modules that perform local processing functions independent of the central processing unit. PPs are driven by read-only memory in the PM, thus releasing CPU run-time for higher level activities.

PM

Peripheral module

PMIST

Peripheral Module Intercept System Test

PP

Peripheral processor

Pulse code modulation (PCM)

Representation of an analog waveform by coding and quantizing periodic samples of the signal such that each element of information consists of a binary number representing the value of the sample.

Queue

A queue contains an ordered list of PROTEL structures, called queue items, and a queue header, which contains the locations of the items in the queue. Queues operate by adding items to them (enqueuing) or removing items from them (dequeuing).

Read-only memory (ROM)

A memory system in which data is stored in a permanent, non erasable form. Applications in DMS include driving the peripheral processors.

Reflex buffer

Contained in each terminal of a PM. Holds a 2-byte report type and 14 bytes of primitive instructions.

Reflex flag

Signals that CC instructions have been executed or an exec has been processed.

ROM

Read-only-memory

SFDEV

Store file device

Signaling processor (SP)

The interface between a master processor and the control circuits in the line-side of a line module. Through the SP, the line circuits, ringing multiplexers, programmable ringing generators, and the activity circuit are controlled, and their status reported.

SP

Signaling processor

Superframe sync

Peripherals downline from the XPM expect a constant framing pulse and a superframe every 240 frames. The inactive unit uses the mate superframe phase comparator to compare the superframe it generates with the superframe generated by the active unit.

SWERR

Software error

Task

An independent program that accomplishes a specific function, such as auditing, timing, or integrity-checking.

Terminal

1) Refers to both the interface circuit on a circuit board mounted in a PM unit and the device to which it is connected. Devices include telephone sets, trunk circuits, and data links. 2) An external connection to the DMS system.

Terminal identifier (TID)

The node number and the terminal number.

Terminal number

A number given to a specific terminal attached to a node.

Terminal process

Procedures usually set up to perform signaling and supervision tasks. The process performs its function until a predefined event, such as end of digits, occurs.

Terminal variable area (TVA)

In DMS, a 30-byte block of store associated with each terminal in a PM. It is used for the storage of information unique to the particular terminal.

TID

Terminal identifier

Tracepoint

An interruption in the flow of a software program to collect information. A tracepoint causes a program to branch to system code to collect information and then return to the original program to continue execution.

Trap

1) An unprogrammed conditional jump to a specified address that is automatically activated by hardware; a recording is made of the location from which the jump occurred. 2) An error condition detected by the firmware, software or hardware which causes a trap interrupt. Execution of the running process stops on the instruction at fault.

Trap interrupt

An interrupt which occurs when there is a hardware or software error.

TVA

Terminal variable area

User

A person, organization, or other group that uses the services of a DMS switch.

XDB

Extended data block

XMS-based peripheral module (XPM)

The generic term for XMS peripherals, which use the Motorola 68000 microprocessor. XPMs have two processors, the signaling processor and the master processor, in hot standby configuration.

XPM

XMS-based peripheral module

DMS-100 Family

PMDEBUG

Technical Assistance Manual

Publication number: TAM-1001-004
Product release: BCS36 and up
Document release: Standard 09.03
Date: April 1996

Printed in the United States of America