



DEFINITY[®]
Proc Mode Operation



585-222-201
Issue 1

DEFINITY[®] Proc Mode Operation

©1992 AT&T
All Rights Reserved
Printed in USA

TRADEMARK NOTICE

DIMENSION is a registered trademark of AT&T.
MS-DOS is a registered trademark of Microsoft Corporation.
INFORMIX is a registered trademark of Informix Software, Inc.
UNIX is a registered trademark of UNIX Systems Laboratories, Inc.

NOTICE

While reasonable efforts were made to ensure that the information in this document was complete and accurate at the time of printing, AT&T can assume no responsibility for any errors. Changes or corrections to the information contained in this document may be incorporated into future reissues.

TO ORDER ADDITIONAL COPIES OF THIS DOCUMENT

Contact: Your AT&T Account Team

or

AT&T Customer Information Center, Commercial Sales
P.O. Box 19901
Indianapolis, Indiana 46219
1-800-432-6600

Prepared by
AT&T Technical Publications Department
Lincroft, NJ

Contents

1 What's In This Guide?

What's In This Guide?	1-1
Who May Use Proc Mode	1-1
Intended Audience	1-3
How This Document Is Organized	1-4

2 Getting Started With Proc Mode

Getting Started With Proc Mode	2-1
--------------------------------	-----

3 Enhanced Mode

Enhanced Mode	3-1
Accessing Enhanced Mode Procedures	3-3
Leaving Enhanced Mode	3-4
Enhanced Mode Examples	3-5

4 Basic Mode

Basic Mode	4-1
Accessing Basic Mode Procedures	4-3
Leaving Basic Mode	4-5
Basic Mode Examples	4-6

5 Enhanced Mode and Basic Mode Commands and Keys

Enhanced Mode and Basic Mode Commands and Keys	5-1
Command Syntax	5-1
Command Reference	5-2
Function Key Reference	5-29
Special Key Reference	5-32

6 Run Files

Run Files	6-1
New Terms	6-1
Executing Run Files	6-3
Creating Proc Activity Logs	6-5
Executing Run Files from within a Manager III Script	6-8
Scheduling Run Files Using Manager III	6-9

7 Proc Mode Programming

Proc Mode Programming	7-1
Writing Programs	7-2
Executing Programs	7-6
Exiting the Program	7-8
How To Use Some Typical Programming Commands	7-9
Converting VMAAP Child Processes to Proc Mode Programs	7-11
Programming Commands	7-13

A Proc Mode Quick Reference

Proc Mode Quick Reference	A-1
Enhanced and Basic Mode Commands	A-2
Enhanced and Basic Mode Function/Control Keys	A-4

B Programming Examples

Programming Examples	B-1
----------------------	-----

Index

List of Figures

Figure		Page
2-1	Procedure Mode Screen (Enhanced Mode)	2-4
2-2	Procedure Mode Screen (Basic Mode)	2-6
5-1	Switch Mode Screen	5-13

List of Screens

Screen		Page
B-1	program <i>wait</i>	B-3
B-2	program <i>getmodes</i>	B-4
B-3	program <i>getmodes (continued)</i>	B-5
B-4	program <i>scrn_cntl</i>	B-5
B-5	program <i>scrn_cntl (continued)</i>	B-7
B-6	program <i>scrn_cntl (continued)</i>	B-8
B-7	program <i>scrn_cntl (continued)</i>	B-8
B-8	program <i>getfunctest</i>	B-9
B-9	program <i>getfunctest (continued)</i>	B-10
B-10	program <i>getfunctest (continued)</i>	B-12
B-11	program <i>getfunctest (continued)</i>	B-13
B-12	program <i>getfunctest (continued)</i>	B-14
B-13	program <i>getfunctest (continued)</i>	B-14
B-14	program <i>getfunctest (continued)</i>	B-15

List of Tables

Table		Page
1-1	Reference Documents	1-6
1-2	Table of Conventions	1-7
5-1	Wildcard Character Specifications	5-9
5-2	Example path.ssb files located in the directory /mgr/ssb/r2v5	5-22
6-1	Run File Commands	6-2
7-1	C and Shell Programs Using Commands with One Option Specified	7-9
7-2	VMAAP-to-Proc Mode Cross-Reference List	7-12
7-3	Math Operators	7-24
A-1	Proc Mode Commands	A-2
A-2	Proc Mode Function/Control Keys	A-4
A-3	Programming Commands Which Do Return Data	A-5
A-4	Useful Programming Commands Which Do Not Return Data	A-6

1 What's In This Guide?

What's In This Guide?	1-1
Who May Use Proc Mode	1-1
Proc Mode Command Availability	1-1
Proc Restriction	1-2
Intended Audience	1-3
How This Document Is Organized	1-4
Recommended Training	1-5
Reference Materials	1-5
Conventions Used in this Manual	1-7

What's In This Guide?

Proc Mode is the application that allows you to administer and maintain DEFINITY Generic 2 and pre-Generic 2 switches by interactively issuing Enhanced and Basic Mode commands. Proc Mode also provides a programming capability that allows you to write programs using some of these commands. When your program is executed, the commands interact with Proc Mode to display, change, or remove switch administration data. This Proc Mode programming feature is designed to help automate administration and maintenance of Generic 2 and pre-Generic 2 switches.

You can also issue most Proc Mode commands from a run file. However, run files provide a more limited programming capability. When a run file is executed, the commands access switch administration data in the same sequence as they are contained in the run file. You may use this feature to string together complex or frequently used procedure commands.

This document describes how to use Proc Mode commands to administer and maintain Generic 2, System 85 and DIMENSION® switches. This includes a detailed explanation of the programming feature. To correctly and efficiently utilize all Proc Mode features, you need a technical background and programming training. It is assumed that you are already familiar with the administration and maintenance features and capabilities of your communications system.

Who May Use Proc Mode

Proc Mode is available through Manager III, IV, and Trouble Tracker. Each of these applications provides you with a different type of access determined by a user class assigned to you. In addition, procedure restrictions may further limit your access.

Proc Mode Command Availability

Proc Mode supports several classes of user. Each class defines what procedures and features the user may access. Depending on your class of user, you may have full or limited access the Proc Mode features and commands.

- Regular User**
- May use all Proc Mode programming commands except the *Services Commands* (**fm**, **getsym**, **nio**, **rio**, **rm**, **wio**, **wm**, **symbols**) and may access public symbols via the **math** command. The **math** command is available for any type of user.
 - May access any procedure, except 276, 490, 495, 496, 497, and 999.

Dial User ■ May access any procedure, except 276, 490, 495, 496, 497, and 999.

System Administrator In addition to all capabilities of the Regular User,
■ May access any procedure, except 276, 490, 495, and 999.

Limited Super User In addition to all capabilities of the System Administrator,
■ May access any procedure.
■ May enable or disable switch features.

Super User ■ May access any procedure.
■ May enable or disable switch features.
■ May access the *Services Commands* (**fm, getsym, nio, rio, rm, wio, wm, symbols**). The Super User can access *Public Symbols* via the **math** command.

Note: *Services Commands* (**fm, getsym, nio, rio, symbols, rm, wm**) can be used by a Super User only. These commands are described in detail in the TSC document *Services Commands*.

Proc Restriction

There are restricted procedures that a user cannot access, such as 276, 277, 490, 495, 496, 497, and 999. Manager IV users are allowed to access the 600 series procs only.

When starting up a Proc Mode session, your application identifies you to Proc Mode by indicating your user class.

Proc Mode uses this information to determine the type of access available to a particular application. This is how Manager III, IV, and Trouble Tracker may be identified to Proc Mode:

Manager III *Regular, Dial User, System Administrator, Limited Super User, Super User.*

Proc Restrictions limit you to the **FM** or **TCM** applications only; you may also be allowed to use both of them.

Manager IV *Regular User.*

Proc Restrictions restrict you to using maintenance procedures only.

Trouble Tracker *Regular User.*

No additional *Proc Restrictions* are specified.

The Manager III, IV and Trouble Tracker applications run on an AT&T 3B2/600 and an AT&T 6386.

DEFINITY Manager II, another administration and maintenance application, is similar to Proc Mode. Manager II executes under the MS-DOS® operating system. Refer to the *DEFINITY® Manager II Operation* manual for further details.

Intended Audience

This guide is intended for three types of users:

- **Switch Administrator or Communications Analyst (Regular User)**
the person(s) responsible for day-to-day administration of the switch, including station moves and changes, ongoing maintenance, and report generation. In some companies, the System Administrator may handle all of these duties.
- **System Administrator**
the person with overall responsibility for telecommunications in your company. This person manages station moves and changes, equipment allocation, and maintains associated systems like Manager III, IV.
- **Switch Technician (Super User)**
the person responsible for testing and troubleshooting the switch using the switch maintenance procedures.

TABLE 1-1
Reference Documents

Document Name	Select Code	Description
<i>DEFINITY® Generic 2 Administration Procedures</i>	555-104-506	Instructions for using DEFINITY administration procedures.
<i>DEFINITY® Generic 2 Maintenance Procedures</i>	555-104-117	Instructions for using DEFINITY maintenance procedures.
<i>DEFINITY® Manager III Operations</i>	585-222-701	Instructions for using Manager III's features and capabilities to maintain your communications system.
<i>DEFINITY® Manager III Installation and Initialization</i>	585-222-100	Instructions for installing the hardware and software needed to use Manager III.
<i>DEFINITY® Manager II Operation</i>	555-104-505	Instructions for using Manager II's features and capabilities to maintain your communications system.
<i>AT&T 3B2 Computer UNIX™ System V Release 3.0 Programmer Reference Manual</i>	307-013	Description of the programming features of the UNIX system.
<i>DEFINITY® Manager IV System Administration</i>	585-223-700	Instructions for using Manager IV's administration procedures.
<i>VMAAP Switch Administration and Maintenance with a UNIX® PC</i>	585-206-701	Instructions for using the VMAAP switch administration and maintenance procedures.
<i>Trouble Tracker R1V2 Operations Manual</i>	585-225-407	Instructions for using Trouble Tracker to monitor performance of networks.

Conventions Used in this Manual

TABLE 1-2
Table of Conventions

Convention	Meaning
add station	This typeface is used for references to Manager III commands, or to Manager III UNIX® directories.
<i>add station</i>	This typeface indicates a command or entry to be typed in by the user.
message	This typeface indicates a computer message.
<i>modelname</i>	This typeface indicates a variable name to be filled in by the user.
[option]	Square brackets indicate an optional entry.
Return	Rounded corners on a key indicate a key on your standard keyboard.
F1 OR Cancel	Square corners on a key indicate a function key. Function keys are the keys to the left of or above the standard keys on your keyboard. Labels for these keys appear at the bottom of your screen.
Ctrl + U	A plus sign (+) between keys indicates both keys should be pressed simultaneously, then released.

2 Getting Started With Proc Mode

Getting Started With Proc Mode	2-1
Procedure: Accessing Proc Mode from Manager III	2-1
Procedure: Accessing Proc Mode from Manager IV	2-2
Procedure: Accessing Proc Mode from Trouble Tracker	2-3
Procedure Mode Screens	2-4

Getting Started With Proc Mode

You may use Proc Mode to access a Generic 2, System 85 or DIMENSION switch from Manager III, IV, or Trouble Tracker. To start up a Proc Mode session, you must perform a set of prerequisites which are specific for each of the applications.

Following are the procedures to be used by Manager III, IV, and Trouble Tracker users to invoke Proc Mode. Depending on the application available to you, select the appropriate procedure and follow the steps as described. With all applications, access to Proc Mode features is provided via the Procedure Mode screen. The screen description presented in "Procedure Mode Screens" applies to all applications.

Procedure: Accessing Proc Mode from Manager III

To access Proc Mode from Manager III, follow these steps:

- 1 Log on as a Manager III user. At the UNIX prompt, enter *CAFE*. The AT&T SWITCH ADJUNCT APPLICATION INTERFACE MENU will be displayed.
- 2 Select the Manager III application.
- 3 Select a switch (the same as the target) name.
- 4 At the **Enter command:** prompt, type *run proc*. This command will establish connection to the switch and log all user and switch activities in the log file. The screen for Procedure Mode will appear.
- 5 Select a switch mode: *Administration, Maintenance, Disk/Tape Subsystem*, or any combination of these.
- 6 At the **Enter command:** prompt, type an Enhanced or Basic Mode command. Further details are provided in "Procedure Mode Screens."

The Procedure Mode section of the *Generic 2 Administration Procedures* contains information about Procedure Mode.

Run Proc—Log All Activities

Use **run proc** to log all user and switch activities. The following Proc Mode activities may be written to a log file:

- Command lines prefixed by <
- Switch procedure images prefixed by >
- Error messages prefixed by !
- Data lines prefixed by ^
- Status messages prefixed by ~

The **run proc** command uses the syntax given below.

```
run proc runfile log logfile errornumber
```

For further details, refer to the **run proc** description in Chapter 6, "Run Files."

Procedure: Accessing Proc Mode from Manager IV

You may access Proc Mode from Manager IV.

- 1 Log on as a Manager IV user. At the UNIX prompt, enter *CAFE*.
The AT&T SWITCH ADJUNCT APPLICATION INTERFACE MENU will be displayed.
- 2 At the **Enter application:** prompt, type *maintenance*
- 3 At the **Enter target:** prompt, type a target name.

Note: The switch name must have been previously defined using the **product add** transaction.

- 4 At the **Enter object:** prompt, type *proc*
- 5 At the **Enter verb:** prompt, type *info* or *run*. You have a choice of viewing the Proc Mode Help Facility via the **proc info** command or connecting to the switch by entering the **proc run** command to access the selected maintenance proc.

Proc Mode supports the on-line Help Facility in *enhanced* mode only if the Switch Support Base (SSB) software is installed. The Help Facility contains information necessary to administer and maintain Generic 2 in Proc Mode. Using the Help Facility, you can display information about the procedures on your system. You may even display the procs themselves by using the 6 Field function key.

You are allowed to access the on-line help before connecting to the switch by using the **proc info** command. To invoke the on-line help from within a Test screen (when connected to the switch), enter *h* or *help* at the command line or press 5 Help.

For a detailed description of the Help Facility, refer to *DEFINITY Communications System Generic 2 Maintenance Procedures*.

Connect to the Switch

Proc Mode allows you to connect directly to all switch types via the **proc run** command, thus minimizing the load on Manager IV. When you enter the **proc run** command, Proc Mode will try to establish a connection to the specified switch. If a connection cannot be established, the appropriate error codes will be returned.

- 1 Enter the **proc run** command to establish connection to the specified switch.
 - The **proc run** transaction works only for a dial out connection. If the connection is defined as *dedicated*, the **proc run** transaction fails.

WARNING:

Before the connection is established, the warning message will be generated: **THE MANAGER IV DATABASE WILL NOT REFLECT CHANGES DONE WITH THIS TRANSACTION.** This is a reminder that any administration done via Proc Mode will directly affect the switch but not the Manager IV database which may result in an "out-of-sync" condition. Therefore, if you modify the switch configuration in Proc Mode, you must either update the Manager IV database to keep it "in sync" with the switch, or report any switch updates to the System Administrator.

- With the connection established, and depending on the switch (Generic 2 or earlier release) selected and the SSB software, the Enhanced or Basic Mode screen will be displayed. For a screen description, refer to "Procedure Mode Screens."

DEFINITY® *Manager IV System Administration*, "Appendix E: Proc Mode Application" contains detailed information about accessing Proc Mode from Manager IV.

Procedure: Accessing Proc Mode from Trouble Tracker

To invoke Proc Mode from Trouble Tracker, follow these steps:

- 1 Log on as a Trouble Tracker user. The Trouble Tracker Main Menu is displayed.
- 2 Select the "Utilities Menu" (selection # 4) in the Select application field.
- 3 Select "Cut-Through" (selection # 2) from the Utilities Menu.
- 4 At the **Enter Product Name** prompt, type the switch name. Trouble Tracker will invoke Proc Mode to establish connection to the switch. If the switch name, telephone number and security code are verified, the screen for Procedure Mode will be displayed. For further details, see "Procedure Mode Screens."

Procedure Mode Screens

Proc Mode is a screen-based application that facilitates your interactive session with the switch via prompts, error and help messages.

If a Generic 2 switch has been specified as a target, and the SSB software is present, Proc Mode defaults to the *enhanced* mode, and the input Procedure Mode screen is as follows:

ENHANCED MODE – PROCEDURE: MODE			
SYSTEM MANAGEMENT ACCESS PORT STATUS			
CURRENT PORT		MODE CONTROLLER	
1. Administration:	<input type="checkbox"/> 0	11. Administration:	<input type="checkbox"/> Not Active
2. Maintenance:	<input type="checkbox"/> 0	12. Maintenance:	<input type="checkbox"/> Not Active
3. Disk/Tape System:	<input type="checkbox"/> 0	13. Disk Tape System:	<input type="checkbox"/> Not Active
AGENTS		14. RAMP:	<input type="checkbox"/> Not Active
4. TN492 Port 0:	<input type="checkbox"/>	15. SMAP:	<input type="checkbox"/> Not Active
5. TN492 Port 1:	<input type="checkbox"/>		
6. TN563 Port 0:	<input type="checkbox"/>		
7. TN563 Port 1:	<input type="checkbox"/>		
8. Pseudo Port 0:	<input type="checkbox"/>		
9. Pseudo Port 1:	<input type="checkbox"/>		
10. DCIU Port:	<input type="checkbox"/>		
Connected to CC0 ON-LINE * MAJOR MINOR RUN TAPE			
enter command:			
<input type="text"/>	<input type="text"/> 2 Repeat <input type="text"/>	<input type="text"/> 5 Help <input type="text"/>	<input type="text"/> 8 Cmds <input type="text"/>

* Asterisk (*) indicates active connection when flashing.

FIGURE 2-1
Procedure Mode Screen (Enhanced Mode)

Proc ID Line

The Proc ID Line occupies the top highlighted line of the screen. This line identifies *enhanced* or *basic* mode and the selected proc number.

Switch Modes

Fields 1–3 of the Proc Mode screen list the available switch modes, their numbers, names, and "Active" or "Not Active" state depending on the mode chosen. Depending on your class of user, select "Administration," "Maintenance," "Disk/Tape Subsystem," or any combination of these.

Administration mode is selected by entering *1* on the command line. To release the mode for other users, enter *1* again. Similarly, select Maintenance mode by entering *2* and Disk/Tape mode by entering *3*.

The Proc Fields Area is populated with the field descriptions and data. Fields 11–15 tell which port has control of a switch mode. Fields 4–10 display a code that tells which agent has control of the given port.

Status Line

The Status Line indicates the state of the switch and of your connection to the switch. Six status indicators are available: *MAJOR*, *MINOR*, *RUN TAPE*, *BUSY OUT*, *IN USE*, and *WAIT*. The indicators interpretation is covered in detail in *DEFINITY Manager II MS-DOS® Version Operation*.

Message/Help Line

The Message/Help Line is used for displaying error and help messages.

Command Line

The Command Line starts with the prompt `enter command: .` All Proc Mode commands are typed at this command line.

Function Keys (F1 through F8)

Labels for the active Function Keys are displayed in this area.

If a Generic 2 switch has been specified but the SSB software is not installed, or if a System 85 R2V3 or R2V4 switch has been requested, Proc Mode defaults to the *basic* mode, and the Procedure Mode screen looks as follows:

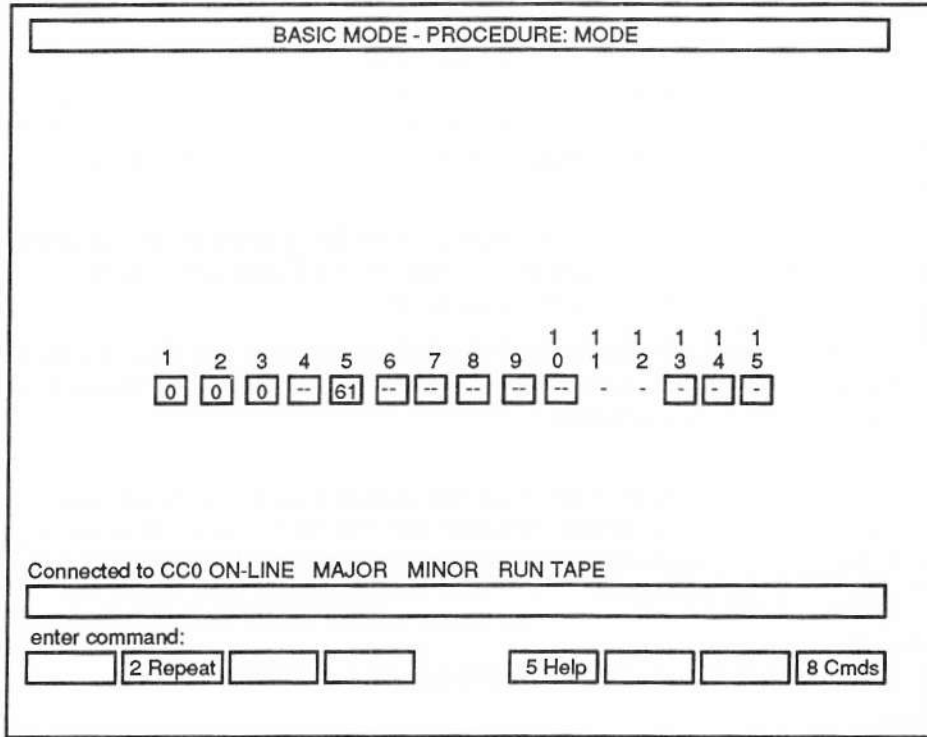


FIGURE 2-2
Procedure Mode Screen (Basic Mode)

The screen description for the Enhanced Mode screen applies to the Basic Mode screen as well. However, on these two screens, the same information is presented in a different manner; the field descriptions and the way the fields are organized are completely different.

- The proc fields are positioned side-by-side, horizontally.
- The field numbers appear above each field.

If a System 85 pre-R2V3 or DIMENSION switch has been selected, a Procedure Mode screen is *NOT* displayed because these switches do not support that procedure. Instead, you are prompted to select a procedure number (eq. 000).

3 Enhanced Mode

Enhanced Mode	3-1
Accessing Enhanced Mode Procedures	3-3
Leaving Enhanced Mode	3-4
Enhanced Mode Examples	3-5
Example 1: Extension Class of Service (Command Line)	3-5
Example 2: Extension Class of Service (On Screen)	3-7
Example 3: Add a Call Appearance (Command Line)	3-7
Example 4: Add a Call Appearance (On Screen)	3-8

Enhanced Mode

This section describes in detail Enhanced Mode operation. All information, commands and examples are intended for a Manager III user only. For Manager IV and Trouble Tracker applications, the appropriate procedures are provided in Chapter 2, "Getting Started With Proc Mode."

Why? To access DEFINITY Generic 2 procedures from Manager III. Enhanced Mode screens show data as separated fields in a full screen layout. Each field has a descriptive label. Help information of several types is available: general, input requirements and specific field ranges.

When? To run administration and maintenance procedures that are not available through task mode, or for simple changes that don't require all the information on a task-oriented station or trunk screen.

Command `run proc runfile log logfile error x`

runfile is the name of a run file to be executed. This is used mainly for scheduling proc commands through scripts or the Work Organizer. Refer to Chapter 7 in *DEFINITY® Manager III Operations*.

logfile is the name of the file that will store all proc commands and output for this Enhanced Mode session. You can specify a full path name for *logfile*. If no path name is specified, the file will be created in **\$HOME/mgr3/logs**. When a run file is present and no log file name is specified, the output will be written to *runfile.log*.

All commands and system output run during a run proc session are logged in **\$\$SWITCH/logs/enh**. At the end of the proc session, they are copied to the **svrmmdd** log. If the log option is turned on, the output for the session is also copied to *logfile* at the end of the session.

x is the number of errors that can occur before the run file stops. The default is 1.

Examples `run proc` takes you to Enhanced Mode, interactive.

`run proc user2 log user2.log error 1` runs a run file called user2, creates the log **user2.log** in **\$HOME/mgr3/logs**, and stops on first error.

`run proc user2` runs a run file called user2, creates the default log file **user2.log**, and stops on first error by default.

`run proc log user.log` takes you to Enhanced Mode, interactive, and creates the log **user.log**.

Notes **Security:** Your System Administrator may have placed restrictions on your User ID. These restrictions can block your access to switch modes (administration, maintenance, or tape) and/or to certain procedures.

Time-Out Interval: When you connect to the switch through Enhanced Mode, the message **Note: <nn> minutes of inactivity will terminate this session** is displayed. If you do not submit a proc command to the switch within this time-out interval, you will be returned to Task Mode. This time-out prevents users from unnecessarily tying up the switch port.

The minimum interval is 10 minutes. The System Administrator can change the interval. See "System Administration" in Chapter 10 of *DEFINITY® Manager III Operations*.

Prerequisites The Switch Support Base (SSB) must be installed on the processor before you can use Enhanced Mode. See the *DEFINITY Manager III Installation and Initialization* manual. If the SSB is not installed, you will be placed in Basic Mode.

References *DEFINITY Generic 2 Administration Procedures (555-104-506)*
DEFINITY Generic 2 Maintenance Procedures (555-104-117)

A listing of all Enhanced Mode commands and function keys and a description of their use are provided in Chapter 5 of this document.

For a quick reference, see "Appendix A: Proc Mode Quick Reference."

Help Press **5 Help** or type *h* at the Enhanced Mode command line to access a menu of subjects related to the selected procedure, or a menu of general subjects if no procedure has been selected.

Press **8 Cmds** or type *hc* at the Enhanced Mode command line to see a list of commands and their meanings.

Accessing Enhanced Mode Procedures

You can access Enhanced Mode procedures from the Manager III, IV, or Trouble Tracker application.

- 1 Access Proc Mode through your application. For more details, refer to Chapter 2, "Getting Started With Proc Mode."
- 2 The screen for Procedure Mode is displayed. Fields 11–15 display values that tell which port has control of the given mode. Fields 4–10 display a code that tells which agent has control of the given port. Proc Mode screens are discussed in detail in Chapter 2, "Getting Started With Proc Mode." Also, additional information can be found in "Procedure Mode" in *Generic 2 Administration Procedures*.
- 3 Type the number of the mode(s) you want to use and press .
 - For example, type `1` to get administration mode, or `123` to get all modes.

Mode	Purpose	Type
Administration	To change the system configuration using Administration Procs (000-499)	1
Maintenance	To test and troubleshoot the system using Maintenance Procs (500-999)	2
Disk/Tape System	To write or read from the system's magnetic media	3

- Error code 76 displays if another agent already has control of the mode. An error message is displayed, if you are not allowed to access the mode you selected.
- 4 Type `p` followed by the number of the procedure to be run.
 - For example, to run Proc 10, type: `p 010` or `p 10` (leading zeroes are not necessary) or `p10` (spaces are not necessary).
 - 5 Most procedures have numbered sub-procedures. In Administration Procedures, these are called *words*; in Maintenance Procedures, they are called *tests*. When you select a procedure, word or test number 1 is automatically displayed.
 - To view a word or test other than word or test number 1, type the "word" command `w` or the "test" command `t`.
 For example, to run word 2, type `w 2`
 To run test 3, type `t 3`
 - You can select both proc and word from one command entry.
 For example, to run proc 12 word 2, type `p12 w2`
 - 6 When the proc screen is displayed you can enter data in two ways.
 - To enter data from the command line, just type the data on the command line. You can enter one command or field entry at a time, or enter a string leaving spaces between each entry. When you press the data will appear in the field(s) on the screen. See Examples 1 and 3 later in this chapter.

- To enter data in the active field (the field highlighted on the screen) use the keys and commands explained in "Entering Data On Enhanced Mode Screens" below. See Examples 2 and 4 later in this chapter.
- 7 When data has been entered correctly, use the correct Enhanced Mode command to carry out your request. For example, to display type *dx*; to add, type *ax*. For a complete list of commands, press **8 Cmds** for on-line help, or see "Enhanced and Basic Mode Commands" in Appendix A. Not all procedures will accept every command.

Errors from the switch will be displayed on the message line.

Consult the *Maintenance and Administration Procedures* document for details about inputting data and correcting errors.

Entering Data On Enhanced Mode Screens

To Do This	Press
Move cursor to active field	3 Form
Remove a character	Backspace
Blank the entire field	4 Clear
Type data in active field and move cursor to next field (Leaves next field intact)	Any of these: Return , Tab , → or ↓
Type data in active field and move cursor to preceding field (Blanks out the preceding field)	Any of these: Shift Tab , ← or ↑
Return to command line	3 Line When you exit the last field on the screen, the cursor automatically returns to the command line.

Leaving Enhanced Mode

To return to Task Mode, type *task*

To enter Basic Mode, type *bas*

To return to Enhanced Mode from Basic Mode, type *enh*

Enhanced Mode Examples

Example 1: Extension Class of Service (Command Line)

This example shows how to add features to a line class of service from the command line. The commands can be typed on a single line. We have broken them logically to make the example clearer.

Refer to Procedure 010 Word 1 in the *Administration Procedures* manual for more details on field entries.

Refer to "Enhanced and Basic Mode Commands" in Appendix A for a complete list of commands.

- 1 Select procedure 010 word 1 and do a display of class of service "1."

```
p10 1 dx
```

The screen displays class of service 1.

- 2 To enable Call Waiting (field 9), Conference 3 Party/Transfer (field 14), Touch-Tone Dialing (field 15) and Send All Calls (field 23), type:

```
cf9 1 cf14 1 1 cf23 1
```

Note: You do not need to enter *cf15* to change Field 15; since you are changing both Field 14 and 15, simply enter *cf14 1 1* since field 15 is the field that follows field 14.

- 3 Type *CX* to execute the change. The screen looks like the following when complete.

ENHANCED MODE – PROCEDURE: 010, WORD 1			
EXTENSION CLASS OF SERVICE – FEATURES			
1.	Class of Service: 1	20.	ACD Member: 0 Disabled
3.	Automatic Callback: 0 Disabled	21.	Stop Hunt: 0 Hunting
		22.	ACD Agent Override: 0 Disabled
		23.	Send All Calls: 1 Enabled
CALL FORWARDING			
4.	Busy and Don't Answer: 0 Disabled		
5.	Follow Me: 0 Disabled		
6.	Call Hold: 0 Disabled		
7.	Calling Number Display: 0 Disabled		
8.	Priority Calling: 0 Disabled		
9.	Call Waiting: 1 Enabled		
10.	Override: 0 Disabled		
13.	Priority Paging: 0 Disabled		
14.	Conference 3 Party/Transfer: 1 Enabled		
15.	Touch-tone Dialing: 1 Enabled		
16.	Timed Recall Exempt: 0 Disabled		
17.	Ring Ping Immediate: 0 Disabled		
19.	ACD Queue Display: 0 Disabled		
Connected to CC0 ON-LINE * MAJOR MINOR RUN TAPE BUSY OUT			
enter command:			
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
<input type="text"/>	2 Repeat	3 Form	<input type="text"/>
<input type="text"/>	5 Help	6 Field	7 Input
<input type="text"/>	<input type="text"/>	8 Cmds	<input type="text"/>

Example 2: Extension Class of Service (On Screen)

This example shows how to add features to a line class of service through the Enhanced Mode screen.

Refer to Procedure 010 Word 1 in the *Administration Procedures* manual for more details on field entries.

Refer to "Enhanced and Basic Mode Commands" in Appendix A for a complete list of commands.

- 1 Select procedure 010 word 1 and do a display of class of service "1."
p10 1 dx
- 2 Press to move the cursor to the active field.
Field 1 will be highlighted.
- 3 To enable Call Waiting (field 9), press until field 9 is highlighted.
Then type *1* and press .
- 4 To enable Conference 3 Party/Transfer (field 14), press until field 14 is highlighted.
Then type *1* and press .
- 5 Repeat for Touch Tone Dialing (field 15) and Send All Calls (field 23).
When you press after the last field entry on the screen, the cursor moves to the command line or press when you have completed entering data.
- 6 Type *cx* on the command line to execute the change.

Example 3: Add a Call Appearance (Command Line)

This example shows adding a call appearance for extension 46789 to extension 46665's set, from the command line.*

A multi-button set is identified by its ELL, since that is unique even if the extension has call appearances on other sets. If you do not know the ELL of extension 46665, the easiest thing to do is check your Extension Report to find it. This example shows how you can find the ELL using procs.

- 1 Select procedure 052 word 2 and display extension 46665:
p52 w2 46665 dx

Make a note of the set's ELL.
- 2 Select procedure 052 word 1 to display the ELL for this extension.
Type *w1* and enter the ELL, then *x* to execute the transaction.

* For Generic 2 switches, you will probably prefer to make this kind of change using the "station" or "button" command in task mode.

- 3 Type *nd* (next data) to display the buttons on the set.
- 4 When an unassigned button is displayed (the value in field 15 is zero), add the call appearance to the button:

```
cf8 46789 2 0 0 ax
```

This command line entry makes field 8 the active field, then it changes the button number to a call appearance of extension "46789," designates the line appearance as "2", the line type as "0" (no prime line) and the ringing type as "0" (no ringing), then executes the add.

Example 4: Add a Call Appearance (On Screen)

This example shows adding a call appearance for extension 46789 to extension 46665's set, through the Enhanced Mode screen.

A multi-button set is identified by its ELL, because the ELL is unique even if the extension has call appearances on other sets. If you do not know the ELL of extension 46665, the easiest thing to do is check your Extension Report to find it. This example shows how you can find the ELL using procs.

- 1 Select procedure 052 word 2 and display extension 46665:

```
p52 w2 46665 dx
```

Make a note of the set's ELL.

- 2 Select procedure 052 word 1 to display the ELL for this extension.
Type *w1* and enter the ELL, then *x* to execute the transaction.

- 3 Type *nd* (next data) to display the buttons on the set.

When an available button is displayed (field 15 is zero), add the call appearance to the button.

- 4 Press to move the cursor to the active field. Field 1 will be highlighted.

- 5 Press until field 8 is highlighted.

- 6 Type *46789* in field 8, and press .

- 7 Type *2* in field 9 (to make 46789 call appearance number 2) and press .

- 8 Type *0* in field 10 (Line Type) and press .
- The meaning of encode 0, **no prime line**, is displayed.

You can also use to display the valid choices for the field.

- 9 Type *0* in field 11 (Ringing Type) and press .
- The meaning of encode 0, **no ringing**, is displayed.

- 10 Press to get to the end of the screen, or press to move the cursor to the command line.

- 11 Type *ax* to execute the add.

The screen looks like the following when complete.

ENHANCED MODE – PROCEDURE: 052, WORD: 1	
MULTIAPPEARANCE TERMINAL/DATA MODULE – LINE APPEARANCE	
TERMINAL EQUIPMENT LOCATION	DEVICE ID
1. Module: 1	6. Device Type: 0 Basic Set
2. Cabinet: 0	7. Member Button: 8
3. Carrier: C	EXTENSION APPEARANCE ID
4. Slot: 7	8. Extension: 46789
5. Circuit: 2	9. Line Appearance: 2
10. Line Type: 0 No Prime Line	
11. Ringing Type: 0 No Ringing	
12. Home Terminal: 0 No	
13. Originating Only: -	
14. SAC Group: -	
DISPLAY ONLY	
15. Button Type: 1 Line appearance (052 w1)	
Connected to CC0 ON LINE	
<input type="text"/>	
enter command: ax	
<input type="text"/>	2 Repeat
<input type="text"/>	3 Form
<input type="text"/>	5 Help
<input type="text"/>	6 Field
<input type="text"/>	7 Input
<input type="text"/>	8 Cmds

4 Basic Mode

Basic Mode	4-1
Accessing Basic Mode Procedures	4-3
Leaving Basic Mode	4-5
Basic Mode Examples	4-6
Example 1: Extension Class of Service (Command Line)	4-6
Example 2: Extension Class of Service (On Screen)	4-7

Basic Mode

This section describes in detail Basic Mode operation. All information, commands and examples are intended for a Manager III user only. For Manager IV and Trouble Tracker applications, the appropriate procedures are provided in Chapter 2, "Getting Started With Proc Mode."

- Why?** To access the numbered administration or maintenance procedures in your System 85 or DIMENSION switch. *
- When?** Whenever you need to run a System 85 or DIMENSION maintenance or administration procedure.
- Notes** Basic Mode procedure fields are displayed side-by-side on a single line, much like the LED display on the SMT or VMAAP's screen display. However, field numbers appear above each field. The active field is highlighted, as in Enhanced Mode. Data can be entered from the command line or directly into the fields.

Basic Mode is similar to Enhanced Mode, but there are some important differences.

All commands and system output during a basic mode session are logged in **\$\$SWITCH/logs** in the **svrmmdd** log where *mm* is a month and *dd* is the day. For example, **svr0911** is for September 11. The logs will be kept for seven days and then cleaned up. If the logs exceed 4 MB or approximately 10,000 procs in a day, then data within the logs will be lost.

Security: Your System Administrator may have placed restrictions on your User ID. These restrictions can block your access to switch modes (administration, maintenance, or tape) and/or to certain procedures.

- Commands that depend on information in the Switch Support Base (SSB) to function properly are not available in Basic Mode. These include on-line help commands related to the procedures and the interpret string command that translates field encodes.
- Basic Mode processes only numerical data. Therefore, you must enter the numerical codes associated with the character data rather than the characters themselves (for example, in the "Name" procedure 012, you must enter numeric encodes instead of letters for the name).
- Basic Mode does not have field labels nor does it display encode meanings.

* If you are comfortable with Basic Mode, or if you have chosen not to load a Switch Support Base (SSB), you can use Basic Mode procedures with the DEFINITY Generic 2 switch.

- Scheduling of Basic Mode run files is done using the UNIX `at` command. See "Scheduling Pre-Generic 2 Run Files" in Chapter 5 of *DEFINITY® Manager III Operations*.
- Basic Mode does not accept universal or XE equipment location specifications. Use the `el` command to translate universal or XE equipment locations to the traditional equipment location numbers. Then use the traditional values when working with equipment locations in procedures.

References

To find information about procedures, use the documents listed in "What's In This Guide" earlier in this manual, or use the flip charts that accompany your switch's MAAP.

For more details on log files and scheduling run files in Basic Mode, refer to Chapter 3, "Enhanced Mode."

A listing of all Basic Mode commands and function keys and a description of their use are provided in Chapter 5 of this document.

For a quick reference, see "Appendix A: Proc Mode Quick Reference."

Help

Press `8 Cnds` or type `hc` to see a list of commands and their meanings.

Accessing Basic Mode Procedures

You can access Basic Mode procedures from the Manager III, IV, or Trouble Tracker application.

- 1 Access Proc Mode through your application. For more details, refer to Chapter 2, "Getting Started With Proc Mode."
- 2 For DEFINITY G2, System 85 R2V3 and System 85 R2V4, the screen for Procedure Mode is displayed. Fields 11–15 display encodes that tell which port has control of the given mode. Fields 4–10 display a code that tells which agent has control of the given port. See "Procedure Mode" in the *Administration Procedures* or flip charts for your switch.

For System 85 pre-R2V3 or DIMENSION switches, Procedure Mode is not displayed. You are prompted to enter a proc number (e.g. 000) which is then displayed.

- 3 For DEFINITY G2, System 85 R2V3 and System 85 R2V4, type the number(s) of the modes you want to use and press .
 - For example, type *1* to select administration mode, or *123* to select all three modes.

Mode	Purpose	Type
Administration	To change the system configuration using Administration Procs (000-499)	1
Maintenance	To test and troubleshoot the system using Maintenance Procs (500-999)	2
Disk/Tape System	To write or read from the system's magnetic media	3

- Error code 76 displays if another agent already has control of the mode. An error message is displayed if you are not allowed to access the mode you have selected.
- 4 For all switch releases, type *p* followed by the number of the procedure to be run.
 - For example, to run Proc 10, type: *p 010* or *p 10* (leading zeroes are not necessary) or *p10* (spaces are not necessary).
 - 5 Most procedures have numbered sub-procedures. In Administration Procedures, these are called *words*; in Maintenance Procedures, they are called *tests*. When you select a procedure, word or test number 1 is automatically displayed.
 - To view a word or test other than word or test 1, type the **word** command *w* or the **test** command *t*.
 For example, to run word 2, type *w 2*
 To run test 3, type *t 3*
 - You can select both proc and word from one command entry.
 For example, to run proc 12 word 2, type *p12 w2*

Basic Mode Examples

Example 1: Extension Class of Service (Command Line)

This example shows how to add features to a line class of service from the command line. The commands can be typed on a single line. We have broken them logically to make the example clearer.

Refer to Procedure 010 Word 1 in the *Administration Procedures* manual for more details on field entries.

Refer to "Basic and Enhanced Commands" in Appendix A for a complete list of commands.

- 1 Select procedure 010 word 1 and do a display of class of service "1."

```
p10 1 dx
```

The screen displays class of service 1.

- 2 To enable Call Waiting (field 9), Conference 3/Party Transfer (field 14), Touch Tone Dialing (field 15) and Send All Calls (field 23), type:

```
cf9 1 cf14 1 1 cf23 1
```

- 3 Type `CX` to execute the change. The screen changes to the following.

BASIC MODE – PROCEDURE: 010

											1	1	1	1	1	1	1	1	1	2	2	2	2
1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	
1	1	-	0	0	0	0	0	0	1	0	-	0	1	1	0	0	-	0	0	0	0	1	

Connected to CC0 ON-LINE * MAJOR MINOR RUN TAPE BUSY OUT

enter command:

	2 Repeat	3 Form		5 Help	6 Field	7 Input	8 Cmds
--	----------	--------	--	--------	---------	---------	--------

Asterisk (*) indicates active connection when flashing.

Example 2: Extension Class of Service (On Screen)

This example shows how to add features to a line class of service through the Basic Mode screen.

Refer to Procedure 010 Word 1 in the *Administration Procedures* manual for more details on field entries.

Refer to "Basic and Enhanced Commands" in Appendix A for a complete list of commands.

- 1 Select procedure 010 word 1 and do a display of class of service "1"

p10 w1 1 dx

- 2 Press **3 Form** to move the cursor to the active field. Field 3 will be highlighted.
- 3 To enable Call Waiting (field 9) press **Return** until field 9 is highlighted. Then type *1* and press **Return**.
- 4 Repeat for Touch Tone Dialing (field 15) and Send All Calls (field 23).

When you press **Return** after the last field entry on the screen, the cursor moves to the command line. You can also press **3 Line** to return to the command line.

5 Enhanced Mode and Basic Mode Commands and Keys

Enhanced Mode and Basic Mode Commands and Keys	5-1
Command Syntax	5-1
Command Reference	5-2
“string”—Interpret a String	5-2
@—One-Second Pause	5-2
;—Advance One Field	5-3
#—Comment	5-3
add—Add	5-3
alarms—Display Switch Alarm Status	5-3
ask—Display a Prompt and Wait for a Response	5-3
ax—Add Execute	5-4
basic—Basic Mode	5-4
bo—Busy Out	5-4
cdx—Clear Data Execute	5-4
ce—Clear Entry	5-4
cf—Change Field	5-4
chg—Change	5-5
continue	5-5
cp—Customer Procedures	5-5
cx—Change Execute	5-5
disconnect—Switch Disconnect	5-5
dsp—Display	5-6
dx—Display Execute	5-6
el—Equipment Location	5-6
enhanced—Enhanced Mode	5-6
fields—Display a Procedure Image	5-7
get—Get Support Files	5-7
hc or ?—Command Help	5-8
help or h—General Help	5-8
hf—Field Help	5-9
hi—Input Help	5-10
history—Display History	5-10
ignore	5-10
log—Display the Name of the Log File	5-12
math—Evaluate Math Expressions	5-12

m—Mode Procedure	5-12
nc—Next Circuit	5-13
nd—Next Data	5-14
nf—Next Fault	5-14
nt—Next Test	5-14
nu—Next Unit	5-14
note—Print a Message	5-14
p—Procedure	5-15
program—Initiate Execution of a Program	5-15
ptx—Park Tape Execute	5-15
quit—Exit Proc Mode	5-15
rb—Release from Busy	5-15
reload—Reload Common Control	5-16
rmv—Remove	5-16
repeat	5-16
rs—Reset	5-19
rtx—Run Tape Execute	5-19
run—Execute Run Files and Log Files	5-19
rx—Remove-Execute	5-19
s—Stop	5-19
scroll—Scroll Screen Format	5-20
silent—Silent Screen Format	5-20
ssb—Switch Support Base	5-21
status—Display Connection Status	5-25
sw—Switch (Change) Common Controllers	5-25
t—Test	5-26
talk	5-26
task—Task Mode	5-28
v — Verify Display	5-28
visual — Visual	5-28
w — Word	5-28
x — Execute	5-28
Function Key Reference	5-29
Exit Menu	5-29
Get SSB Files	5-29
Repeat Previous Command Line	5-29
Display Previous Information	5-29
Move Cursor to Command Line	5-29
Move Cursor to Data Field in Form	5-29
Select a Menu Item	5-30
Clear Field	5-30
Find a Menu Item	5-30

Display Error Codes for a Procedure	5-30
Display General Help	5-30
Display All SSB Files	5-30
Display Different SSB Files	5-30
Display Field Data	5-31
Display Input Data Required for a Procedure	5-31
Display Commands	5-31
Special Key Reference	5-32

Enhanced Mode and Basic Mode Commands and Keys

This chapter provides a listing of all DEFINITY Proc Mode (Enhanced Mode and Basic Mode) commands for reference purposes. Included are the rules of syntax for Proc Mode commands and a discussion of the effect of each command. Most commands can be abbreviated down to three characters. For each of these commands, the shorthand notation is provided. Included also is a listing of function keys and a description of their use.

Command Syntax

A command line appears near the bottom of Proc Mode screens. When it is appropriate for you to enter a command, the cursor appears immediately after an **enter command:** prompt. Here are the general rules of syntax.

- Always type a space between items of data. (Proc Mode interprets *123* as data for one field, and *1 2 3* as data for three distinct fields.)
- Enter a series of commands, parameters, and data by pressing Return (or the equivalent of "carriage return" on your keyboard).
- Some commands require that Return be pressed before another command can be executed. For example, a Return must immediately follow the **task** command.

In general, commands and data are easier to read (for example, in log files) if they are separated by spaces (or tabs or newlines), so it is a good idea to use such space as a rule. However, you will find that the Proc Mode software correctly interprets most commands whether they are typed as *p150w2* or *p 150 w 2*.

Most Proc Mode commands do not accept parameters at all. A few, however, require parameters. A parameter is a way to tell a command to operate in a certain way or to give the command enough information to perform its function. Parameters immediately follow the command itself. For example:

- The procedure command **p** requires a procedure number: *p 100*
- The equipment location command **el** requires an equipment location: *el 1 0 c 2 23*
- The run file command **run** requires the name of the file: *run cmd-file*

Parameters are optional for a few commands. For example:

- The **get** command may take an option that modifies its effects or a file designator in double quotes that specifies the files on which it operates or both: *get -a file* or *get*
- The Switch Mode command **m** may take the number of the mode to be accessed (see the "m—Mode Procedure" later in this chapter for details): *m 123* or *m*

Command Reference

The commands listed below are either Enhanced Mode or Basic Mode commands. Some of them can be used in *run files*—that is, in command scripts executed by the Proc Mode software. Programming commands can be used in *programs*. All of them can be used on the Proc Mode command line. Of the commands listed below that apply to procedures, not all apply to every procedure. Some commands work only with administrative procedures (procedures numbered from 000 through 499). Some work only with maintenance procedures (procedures numbered from 500 on). Some are needed only with particular administrative or maintenance procedures. Press to see what commands are valid when in a procedure. If a procedure cannot execute a command that you have used, it responds with an error message.

``string``—Interpret a String

Name data, incoming-call-identification (ICI) data, and the dial codes * and # must be entered in switch procedures in the form of numeric codes that stand for letters, numbers, and punctuation. If you would rather not enter the codes, you can instead use the Enhanced Mode data-string capability. Note that this functionality is available only in Enhanced Mode; it cannot be used in Basic Mode. The Proc Mode software translates a string of characters between double quotes into its associated numeric codes and sends those numeric codes to the switch. Have Proc Mode thus interpret a string of characters by surrounding the string with double quotes as you enter it. (Do not enter more characters in a string than you have procedure fields to receive them.) For example, to enter *Doe, Jane* starting at field 2 of Procedure 012 Word 2, use the following command.

```
cf 2 ``JJ``
```

Name interpretation is available in Procedure 012 Word 2, Procedure 013 Word 1, Procedure 279 Word 1, and Procedure 497 Word 2. ICI interpretation is available in Procedure 031 Word 1 and Procedure 204 Word 1. Dial-digit interpretation is available in numerous procedures, among them Procedure 100 Word 1, Procedure 175 Word 1, and Procedure 350 Word 1.

You may be puzzled by the problem of quotes within a string. If you have to enter a double quote in a field, do so by escaping the quote with a backslash (\). Therefore, if Jane wants her nickname *JJ* appended to her name, so that it appears as *Doe, Jane "JJ"*, access segment 2 of Jane's record in Procedure 012 Word 2 and use the following command.

```
cf 2 ``\JJ``
```

@—One-Second Pause

Use @ to make the Proc Mode software pause for one second. For example, use @ on the command line or in a run file to insert a one-second pause between commands. The pause command may be of particular use when including a maintenance procedure in a run file. In such situations it is good practice to place several pauses after the execution of a test. This prevents Proc Mode from sending the next command while your PBX is executing that test. (The amount of time required for execution varies with the conditions under which a particular switch operates. You will learn with experience how many pause commands are necessary and under which circumstances.)

;-Advance One Field

Use **;** to leave a field's contents as they are. That is, use **;** to advance past the currently active field to the next-higher-numbered field without changing the contents of the field thus passed. If the current field is the last one (and therefore you cannot skip to a next field), use the change-field command **cf** to move to another field.

#-Comment

Use **#** to add non-executable comments to a Proc Mode-executable file. Everything that follows the comment command on that line is part of the comment. The example below consists of a request to access Procedure 100 followed by a comment.

```
p 100 # access procedure 100
```

Comments may also be entered from the Proc Mode command line. If a log file is active, these comments are added to the file. The comments can then be used as markers within the log file.

add-Add

Use **add** with administration procedures to add displayed data to the switch's translation memory. In most procedures, the **add** command must be followed by an *execute* command, **x**. The shorthand notation for **add x** is **ax**. Always follow an **ax** operation with a **dx** command to view the results of your work.

alarms-Display Switch Alarm Status

Use **alarms** to display the alarm status of the currently active connection. The alarm status may include information on the following switch indicators and alarms: CC0/CC1, ON-LINE, OFF-LINE, MAJOR, MINOR, RUN TAPE, BUSY OUT, IN USE, and WAIT. You can use the **alarms** command in programs. The shorthand notation for **alarms** is **ala**, **alar**, or **alarm**.

The **alarms** command uses the syntax given below.

```
alarms [-s] [-p /-l /-m /-n /-r /-b /-u /-w]
```

For further details, refer to the **alarms** command description in Chapter 7, "Proc Mode Programming."

ask-Display a Prompt and Wait for a Response

Use **ask** to display a prompt on the message line and wait for the user's response. You can use the **ask** command in programs. The shorthand notation for **ask** is **ask**. The **ask** command uses the syntax given below.

```
ask prompt
```

For further details, refer to the **ask** command description in Chapter 7, "Proc Mode Programming."

ax—Add Execute

Use **ax** with administration procedures to add displayed data to the switch's translation memory. (See the **add** command description.) Always follow an **ax** command with a **dx** command to view the results of your work.

basic—Basic Mode

Use **basic** to change to Basic Mode. (Find a description of Basic Mode in Chapter 4 of this document). The shorthand notation for **basic** is **bas** or **basi**.

bo—Busy Out

Use **bo** to place circuit locations in maintenance busy-out status. Use this command from a maintenance procedure that allows a circuit to be busied out. Not all procedures allow this. Those that do are:

- Procedure 635 - Cause of maintenance busy-out (on Generic 2 systems only)
- Procedure 632 - Carrier busy
- Procedure 631 - Trunk group busy-out
- Procedure 630 - Busy out/release busy-out

CAUTION: Equipment is removed from service when placed in busy-out status. Consider this effect before using the **bo** command.

cdx—Clear Data Execute

Use **cdx** to resolve alarms—that is, to clear the alarm-sent bits in the error log without clearing the record itself. Clearing the alarm sent bit of an error record turns off the alarm indicator on the alarm panel and on the Proc Mode status line if there is no other reason for the indicators to be on. Like certain other commands, **cdx** is effective only when used from within appropriate maintenance procedures (such as Procedure 600).

ce—Clear Entry

Use **ce** to clear the current input field. New data can now be entered in the cleared field. Or, to leave the field blank, use the **;** command (described above) to move to the next input field.

cf—Change Field

Use **cf** followed by a field number to select the specified field—that is, to make it the active one. This clears the specified field and makes it ready for input. For example, **cf4** changes the current input field to field 4 (provided, of course, that field 4 exists and accepts data).

chg—Change

Use **chg** with administration procedures to change displayed data in the switch's translation memory. In most procedures, the **chg** command must be followed by an execute, **x**. The shorthand notation for **chg x** is **cx**. Many procedures require that you issue a **dx** command to display the data you wish to change before you can use the **cx** command. Always follow a **cx** command with a **dx** command to view the results of your work.

continue

Use **continue** to terminate a "talk" session by returning to your program or run file execution once you are done interactively entering commands. The **continue** command is entered from within a "talk" session. As a result, Proc Mode resumes accepting input from the run file or program which initiated the "talk" session.

The shorthand notation for **continue** is **cont**, **conti**, **contin**, or **continu**. With VMAAP, this functionality is partially performed by the command **quit**.

You issue **continue** as follows:

```
continue 
```

To get a better understanding of how **continue** works, refer to the description of the **talk** command later in this chapter.

cp—Customer Procedures

Use the **cp** command followed by a System Management Terminal (SMT) procedure number to simulate the functions of an SMT. This enables access to the procedures that may be invoked from the SMT. This command has the same effect as the **p** command (described below) except that SMT procedure numbers are used. (This command cannot be used on Generic 2 communications systems.)

cx—Change Execute

Use **cx** with administration procedures to change displayed data in the switch's translation memory. (See the **change** command description.) Many procedures require that you issue a **dx** command to display the data you wish to change before you can use the **cx** command. Always follow a **cx** command with a **dx** command to view the results of your work.

disconnect—Switch Disconnect

Use **disconnect** to disconnect from the current switch (DOS only). This command releases the current connection and makes the currently used port and modem available for another **con** command. The shorthand notation for **disconnect** is **disc**, **disco**, **discon**, **disconn**, **disconne**, or **disconnec**.

dsp—Display

Use **dsp** to display data for the current procedure. In most procedures, the **dsp** command must be followed by an execute command, **x**. The shorthand notation for **dsp x** is **dx**.

dx—Display Execute

Use **dx** to display data for the current procedure and word. (See the **dsp** command description.) For most administration procedures, data must be displayed using **dx** before it can be changed using **cx** or removed using **rx**. Always follow an **ax** or a **cx** command with **dx** to view the results of your work.

el—Equipment Location

Depending on the mix of features required of it, a given DEFINITY Communications System Generic 2 may contain up to three types of network modules: *traditional*, *universal*, or *XE*. Because of differing hardware, the various types of modules call for different forms of equipment location codes. Equipment location codes for traditional modules are of the form used for DIMENSION and System 85 communications systems. Equipment location codes for universal or XE modules are of the form used for System 75 communications systems. Use **el** to translate an equipment location from universal or XE to traditional format or the reverse. The **el** command uses the syntax given below.

el module cabinet carrier [slot [circuit]]

Here, *module* is 0–30 and *cabinet* is 0–4. The *carrier* is *a - e* if the equipment location is universal or *XE* and 0–3 if the equipment location is traditional. The *slot* is 0–21, and the *circuit* is 0–23. The brackets indicate that the slot and circuit are optional. Of course, to get a complete equipment location, you must specify both slot and circuit. The results of the command are displayed on the message line just above the command line. For instance, the command *el 1 0 e 15 20* causes Proc Mode to display the following message line:

Universal or XE 1 0 E 15 20 corresponds to Traditional 1 3 1 0 4

The **el** command provides mapping only; there is no attempt to verify that a given equipment location exists on the current switch. Indeed, **el** works whether or not a switch connection exists. Although the **el** command is especially helpful when you are using Basic Mode to administer the DEFINITY Communications System Generic 2, it is also useful to verify mapping information when you are using Enhanced Mode.

enhanced—Enhanced Mode

Use **enhanced** to change to the Enhanced Mode. Enhanced Mode is available only if the applicable SSB has been installed on your PC. (Find a description of Enhanced Mode in Chapter 3 of this document). The shorthand notation for **enhanced** is **enh**, **enha**, **enhan**, **enhanc**, or **enhance**.

fields—Display a Procedure Image

Use **fields** to display a procedure image (sequence of data fields all on one line representing the state of a proc) for the currently active procedure. You can use the **fields** command in programs or run files. The shorthand notation for **fields** is **fie**, **fiel**, or **field**.

The **fields** command uses the syntax given below.

```
fields [-s] [field_number /-p /-e /-a /-w /-t]
```

For further details, refer to the **field** command description in Chapter 7, “Proc Mode Programming.”

get—Get Support Files

Proc Mode requires many files of switch data to support the switch-specific features of its Enhanced Mode. All Generic 2 systems contain such files on their magnetic storage facilities. These same files are also available on diskettes.

Use the **get** command to install the support files from the currently selected switch, provided that the connection is through a PPG port. Files cannot be copied from the switch by way of an RMATS port. To do so, use a command line of the following type:

```
get [-a] [-g] [-s] [filenameexpression] Return
```

Where:

- a Means get all of the SSB files. The default is to get only those from the switch that differ from those already installed.
- g Means get the specified information immediately. The default is to display a screen through which you can interactively redefine the list of files you wish to obtain.
- s Means suppress synchronization of the two copies of the SSB, one on the DTS disk and the other on the switch tape. If they are out of sync, the SSB on the switch tape is copied onto the DTS disk unless the *-s* option is specified. This option is for use by qualified technicians only.

filenameexpression are names of files that you want to specify for installation or viewing. Wildcard file name specifications may be used. See “help or h—General Help” later in this chapter for the conventions that apply to these specifications.

Unless the *-a* option was used, only files that differ between your communication system and those already installed will be listed on the screen produced by the *get* command, and they will all be marked for downloading (with a *yes* in the first column). To mark a file so that it will not be downloaded, use the Page Down and ↓ keys to highlight the line on which the file appears and press the function key labeled 3 Select to toggle the first field to *no*. If the menu is extensive, use the function key labeled 4 Find to search for a filename. To use this facility, press 4 Find and then enter an expression of the type described in “help—General Help” in this chapter (do not quote the expression in this context). Proc Mode attempts to match your expression with a menu item. If a match is found, the highlight bar is moved to the matched item. Otherwise, the highlight bar is moved to the first item in the menu.

To download the files marked *yes*, press **2 Get**. To return to the **enter command:** prompt, press **1 Exit**.

If the file you want to download is not listed, press **6 All**. This has the same effect as using the *-a* option with the **get** command. That is, all files are now available for your perusal, and all entries in the first column become *no*. Using the function keys just described, select the files you want downloaded by marking them with *yes* in the first column. When all files are listed, the label for the function key changes to **6 Delta**, and offers the option of selecting a list of files that differ between the switch and your current support file installation.

hc or ?—Command Help

View a screen that gives a brief description of the Proc Mode commands by typing *hc* or *?* at the command line, or by pressing a function key that is screen-labeled **8 Ccmds**. This screen associates each Basic Mode or Enhanced Mode command with a descriptive phrase that identifies its function. When you are ready to leave this screen, press **1 Exit** or **Return**.

help or h—General Help

Access help by typing *h* or *help* at the command line, or by pressing a function key that is screen-labeled **5 Help**. Use these commands to access a menu of subjects appropriate to the currently selected procedure (or a menu of general subjects, if no procedure is selected). However you choose to get help, the menu that appears will display a highlighted bar over one of its items. Use the **↑**, **↓**, **Page Up**, **Page Down**, **Home**, and **End** keys to position the bar over the subject on which you need information. Then press **3 Select** or press **Return** to select a subject. This may lead to a screen with information or to a subordinate menu. If a menu is on your screen, proceed as above. If information is on your screen, scroll through it as described above.

As you use the available help screens, several function key labels variously appear and disappear on the screen. These are **1 Exit**, **Return**, **3 Select**, and **4 Find**. Use **1 Exit** to exit the help facility. Use **Return** to return to the previous menu. Use **3 Select** to select the highlighted menu item. Use **4 Find** to search for a topic within a menu.

The *find* function is most useful when you are confronting a long list of possible topics. For example, there are hundreds of administrative procedures, each one with a set of help screens. Having accessed the administrative procedure help menu, suppose you want to find Procedure 100. To do so, simply press **4 Find** and type *100* on the command line followed by **Return**. Proc Mode searches the list of procedures and the highlighted bar is moved to the line containing the number 100.

You can type any string on the command line and have the *find* function look for a match. The function even looks for strings matched by the wildcard conventions described in Table 5-1. Therefore, for example, you could type *restrict.*search* to find the following string.

Procedure 283 Word 1-Facility Restriction Level Related Searches

TABLE 5-1
Wildcard Character Specifications

Character	Specification
c	A character <i>c</i> not listed in this table matches itself.
c*	A character <i>c</i> not listed in this table followed by an asterisk matches <i>zero</i> or more occurrences of the character.
c+	A character <i>c</i> not listed in this table followed by a plus sign matches <i>one</i> or more occurrences of the character.
*	An asterisk at the beginning of an expression matches any string—including *.* and the null string.
.	A period matches any single character.
[...]	Characters within brackets match any one of the enclosed characters. A pair of characters separated by a hyphen (-) matches any character that falls between the pair (in ASCII order), including the pair itself. If the first character that follows the opening bracket is a circumflex (^), any character <i>not</i> enclosed is matched. (Therefore, for example, [[^] abc3-7] matches any character except a, b, c, or numbers 3 through 7.)
^	A circumflex (outside of brackets) matches the beginning of a line.
\$	A dollar sign matches the end of a line. A specification that begins with a circumflex and ends with a dollar sign, therefore, matches an entire line or filename.

The *find* function is not case-sensitive. That is, it matches both uppercase and lowercase characters regardless of the case specified. This can be seen in the above example in which “Search” was found as a result of specifying “search”.

Having used the 4 Find function key to highlight the topic line, press the function key labeled 3 Select to view help on the highlighted topic.

hf—Field Help

Access descriptions of the individual field by typing *hf* at the command line, or by pressing a function key that is screen-labeled 6 Field.

While a procedure field on your screen is ready to accept input (that is, while the field is active), the 6 Field label appears on your screen. If you press this function key or if you type *hf* at the

command line, you will receive a description of the active field. This description may be in the form of a range of possible values, if appropriate. Or, if more information is called for, this description may take the form of a window with text that you can scroll by means of the **↑**, **↓**, **Page Up**, and **Page Down** keys.

In the latter case, a highlighted bar appears in the window. To select one of the encodes described in the window, place the highlighted bar on your choice and press the function key labeled **3 Select**. To exit the field help facility without making a selection, press the function key labeled **1 Exit**.

hi—Input Help

In many switch administration or maintenance procedures, certain data must be entered before the procedure can execute any given command. For example, to display switch data associated with a given phone, the phone must be identified by the extension number or equipment location. In this case, an identifying number must be entered before **dx** can be executed.

To display a list of commands and the data required for their execution, either type *hi* from the command line or press **7 Input**. Such a list may be much longer than the window it is displayed in. If so, use the **↑**, **↓**, **Page Up**, and **Page Down** keys to scroll through it to find the information you need.

history—Display History

Use the **history** command to display the last nine events that have taken place. Such events may include commands, error messages, or procedure data displays. To exit the *history* screen, press **Return** or **1 Exit**. The shorthand notation for **history** is **hist**, **histo**, or **histor**.

ignore

The **ignore** command instructs Proc Mode to ignore errors returned by switch procedures. Normally, when the switch returns an error code in response to a Proc Mode request, the following occurs:

- The current command line is aborted (i.e. any commands remaining on the command line are not executed).
- If a run file or program is executing, the error is counted in the total count of file errors.

When the **ignore** command specifies that a switch error code be ignored, the scenario changes as follows:

- Command line processing continues as though no error had occurred.
- The run file error total remains unchanged.

The settings specified by an **ignore** command remain in effect until another **ignore** command is issued.

The shorthand notation for **ignore** is **ign**, **igno**, or **ignor**.

You can issue the **ignore** command as follows:

Command: `ignore [ERROR [ERROR [ERROR [...]]]]` Return

Arguments: `[ERROR [ERROR [ERROR [...]]]]`

The *ERROR* argument is a one or two digit switch error code, a range of codes (e.g. 73–75), or the word *all*. Either all switch errors or only the specified switch errors may be ignored at one time.

To restore Proc Mode to its default condition so that all switch error codes are processed, enter *ignore* without parameters:

`ignore`

Then any switch error will cause Proc Mode to abort the current command line. If a run file or program is currently executing, the error contributes to the total number of file errors.

To find out the current ignore settings, enter *ignore* followed by a question mark:

`ignore?`

Returns: Not applicable.

Example 1: `ignore 75-77 99`

Proc Mode ignores switch error codes 75, 76, 77, and 99. Any previous ignore setting is lost.

Example 2: `ignore 99 all`

Proc Mode ignores all switch errors (i.e. 0–99).

Example 3: `ignore 253`

Proc Mode displays an error message stating that 253 is an invalid switch procedure error number. Proc Mode reverts to the default *ignore* setting (that is all switch procedure errors are processed).

Example 4: `ignore`

Proc Mode stops ignoring switch procedure errors.

Notes:

- It is only possible to ignore switch errors.
- Proc Mode generated errors may not be ignored.

See Also: `program, run.`

log—Display the Name of the Log File

Use **log ?** to display the name of the current log file, if one exists, on the message line. For further details, refer to the **log** command description in Chapter 6, "Run Files."

math—Evaluate Math Expressions

Use **math** to evaluate the math expressions (a combination of operators and operands). Up to 18 expressions can be interpreted at a time by a **math** command. You can also use the **math** command in programs and run files.

The **math** command uses the syntax given below.

```
math [-s] [exp1 [expr2 [...expr18]]] 
```

For further details, refer to the **math** command description in Chapter 7, "Proc Mode Programming."

m—Mode Procedure

Use **m** to specify the communications system mode for which you are in contention. Mode specification is only for Generic 2 and System 85 R2V3 and R2V4. The three modes that can be requested are the administration mode (1), maintenance mode (2), and tape mode (3). Type *m* and then *1*, *2*, or *3* to toggle the corresponding mode between off and on (0 and 1). Type *m* alone (without any arguments) to display the current modes.

A screen resembling Figure 5-1 appears. If the displayed mode is 1, it is active; if 0, it is inactive.

ENHANCED MODE – PROCEDURE: MODE	
SYSTEM MANAGEMENT ACCESS PORT STATUS	
CURRENT PORT 1. Administration: <input type="checkbox"/> Not Active 2. Maintenance: <input type="checkbox"/> Not Active 3. Disk/Tape System: <input type="checkbox"/> Not Active	MODE CONTROLLER 11. Administration: <input type="checkbox"/> Not Active 12. Maintenance: <input type="checkbox"/> Not Active 13. Disk Tape System: <input type="checkbox"/> Not Active 14. RAMP: <input type="checkbox"/> Not Active 15. SMAP: <input type="checkbox"/> Not Active
AGENTS 4. TN492 Port 0: <input type="checkbox"/> 5. TN492 Port 1: <input type="checkbox"/> 6. TN563 Port 0: <input type="checkbox"/> 7. TN563 Port 1: <input type="checkbox"/> 8. Pseudo Port 0: <input type="checkbox"/> 9. Pseudo Port 1: <input type="checkbox"/> 10. DCIU Port: <input type="checkbox"/>	
Connected to CC0 ON-LINE * MAJOR MINOR RUN TAPE <hr style="border: 1px solid black;"/> enter command: <input type="text"/> 2 Repeat <input type="text"/> <input type="text"/> <input type="text"/> <input type="text"/> 5 Help <input type="text"/> <input type="text"/> <input type="text"/> 8 Cmds	

* Asterisk (*) indicates active connection when flashing.

FIGURE 5-1
Switch Mode Screen

The use of system modes allows several users to access one system at the same time. For example, while one user is executing administrative procedures with the administration mode set, another user can be executing maintenance procedures on the same system with the maintenance mode set. With Generic 2 or System 85 R2V4, you do not have to set the administrative mode to use the administration procedures to display data. With System 85 R2V3, you must set the administrative mode to display data.

You must set the switch to *administration* mode to change switch translations, and you must set the switch to *maintenance* mode to access maintenance procedures. Also, set the switch to *tape* mode to copy to or from your switch's tape cartridge (or disk, if your switch is so equipped). Tape Mode is required in order to use the **ptx**, **rtx**, or **rld** commands.

nc—Next Circuit

Where appropriate, use **nc** to make maintenance procedures select and display the next circuit location. Two uses of **nc** are possible; to display stored circuit information and to select a circuit location for testing.

To display test results or failure history one circuit location at a time, type *nc* once for each new circuit location.

To select a single circuit location for testing, type *nc* to step through and display circuit locations one at a time until the circuit you wish to test appears.

nd—Next Data

Use **nd** to display the next data item. In administration procedures that have an **nd** function, use it to display data for an input field that contains multiple data entries. Use **nd** also to step manually through all entries associated with the input field.

For maintenance procedures, type *nd* to display (for example) demand test results, periodic failure history data, and call processing failure history data. (These records are arranged by circuit location.) Repeatedly type *nd* to display data records one at a time.

nf—Next Fault

Use **nf** to step through maintenance errors that have more than one fault code associated with them.

nt—Next Test

Use **nt** to advance the maintenance procedure to the next test that can be run. If the current test displayed is the last test that the procedure has, the first test is selected. The new test selected will not start until you type *x* (for *execute*). Also, see the **test** command described in this chapter.

nu—Next Unit

Use **nu** to select and display the next unit type for testing when a maintenance test is selected. You can select and display the first circuit of that unit type for testing by typing *nc* (for *next circuit*). When displaying failure history or test results, type *nu* (for *next unit*) to look at the first circuit location for the next unit type in the failure history list.

note—Print a Message

Use **note** to print a message on the message line. The **note** command is particularly convenient for displaying status messages during a program or run file execution. A new message can be displayed for each section of the program.

The **note** command uses the syntax given below.

```
note [message] Return
```

For further details, refer to the **note** command description in Chapter 7, “Proc Mode Programming.”

p—Procedure

Use **p** followed by a procedure number to access that procedure. For multiword administration procedures, word 1 is automatically accessed first. For a maintenance procedure, test 1 is automatically selected first. Leading zeroes are not significant when specifying procedure numbers. For instance, p12 is equivalent to p012, and p0 is equivalent to p000.

program—Initiate Execution of a Program

Use **program** to initiate execution of a program that interacts with Proc Mode. This program issues requests in the form of the typical Proc Mode commands terminated by a new line. Proc Mode executes each command in turn. The shorthand notation for **program** is **pgm**, **pro**, **prog**, **progr**, or **progra**.

The **program** command uses the syntax given below.

```
program [-e max_nbr_errors_to_permit] program_name [arg1 [arg2 [arg3 [...[argn]] ]]]
```

Return

For further details, refer to the **program** command description in Chapter 7, “Proc Mode Programming.”

ptx—Park Tape Execute

Use **ptx** to return the switch’s tape to the start or “parked” position. Obtain Tape Mode from the mode procedure (*m*) before using this command.

quit—Exit Proc Mode

Use **quit** to exit from Proc Mode in an orderly way. In response, Proc Mode disconnects any active connections and returns control to the MS-DOS operating system. To exit from a “talk” session, use the **continue** command.

rb—Release from Busy

Use **rb** to release a circuit from maintenance busy-out status. Whenever the BUSY OUT indicator is on, **rb** can be used within an appropriate procedure to release a circuit location from maintenance-busy status. (The indicator does not disappear until all busied-out circuits have been released.)

Use Procedure 635—the *cause of busy out* procedure—to search for circuits that are in busy-out status. Or you may use this command from any maintenance procedure that allows a circuit to be released from maintenance-busy. See the busy-out command **bo** for further information.

reload—Reload Common Control

Use the **reload** command to reload the switch's common control memory from its disk (DTS) or program tape. (For System 85 R2V3 and later versions, you must use the mode command **m** to set the switch *Tape Mode*.) This command reloads the common control (CC) memory of the currently selected connection from the DTS or program tape. The shorthand notation for **reload** is **rld**.

CAUTION: Executing *rld* takes the CC that is communicating with your Proc Mode out of service while the reload is in progress. If this is the only CC, or the active CC in a duplicated system, a service-affecting switch outage will occur. Since the reload causes the remote end to disconnect, the currently selected Proc Mode connection is disconnected as well. Reconnect after the reload completes.

CAUTION: As you administer your system with Proc Mode, translations are stored in the system's CC memory. These translations are lost when you reload the CC from the magnetic medium. Be sure, therefore, to save your work each session with the **rtx** (*run tape execute*) command. This way, your most recent work is recorded on the magnetic medium and is reloaded when you execute **rld**. Otherwise, a reload loses any translation that you had made.

rmv—Remove

Use **rmv** with administration procedures to remove data from the switch's translation memory. In most procedures, the **rmv** command must be followed by an **execute** command, **x**. The shorthand notation for *rmv x* is *rx*. Many procedures require that you issue a **dx** command to display the data you wish to remove before you can use the **rx** command. Always follow the **rx** command with a **dx** command to view the results of your work.

repeat

Proc Mode maintains a list of the last 500 previously issued command lines, except those containing **repeat** commands. The **repeat** command is used to repeat or display a previous command line or range of command lines. The shorthand notation for **repeat** is **r**, **rep**, **repe**, or **repea**.

With VMAAP, this functionality is provided through two commands: **exec** and **last**.

The previous command lines are repeated in the order they were initially issued. This order is maintained by sequentially assigning a line number to a command line being executed, starting with 1 and ending with 999. Once a 999 is reached, numbering resumes with 1. Only the last 500 command lines are saved; any of the saved 500 lines may be displayed or repeated using the **repeat** command. When command line 501 is issued, command line 1 is forgotten and so on.




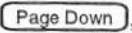


For a quick single command line repetition, Proc Mode provides a function key 2 Repeat. It is equivalent to issuing the **repeat** command with no arguments from the command line.

You can issue **repeat** as follows:

Command: `repeat[-d] [n] [-n]` 

Options: `-d`

The option `-d` specifies that the requested commands be displayed rather than executed. In screen format, a new `repeat` screen is overlaid on the current screen.

To move within the `repeat` screen, use the , , , , , and  keys.

By default, 18 command lines are displayed in scroll format. You may override the default by specifying the command lines to be displayed. If you specify more command lines than fit on the screen, you will be prompted whether or not to continue.

Arguments: `[n][/-n]`

The `repeat` command arguments are optional.

- If both the arguments and the `-d` option are omitted, the last command line is repeated.
- If the arguments are omitted and the `-d` option is specified, in screen format, the `repeat` screen is displayed; in scroll format, the last 18 commands are listed.
- If both the arguments and the `-d` option are used, the specified command line(s) are displayed.
- If the arguments are specified and the `-d` option is omitted, the associated command line(s) are executed.

Valid `repeat` command arguments are integers between 1 and 999 specifying command line numbers or an integer between 1 and 500 specifying a range of command lines.

- To execute or display the particular command, specify the command line number using a single integer.
- To execute or display a range of commands, specify the range of command line numbers using two integer numbers separated by a dash. Each integer must be a valid command line number in the set 1–999. At most, 500 command lines can be repeated at one time.
- You can also specify a range of commands using a single integer number. This approach allows you to quickly specify the last command lines to repeat without knowing the command line numbers. The maximum range may contain 500 command lines:
 - a To execute or display the previous 10 commands, use `-10` as an argument.
 - b To execute or display the last 500 command lines, use `-500` as an argument.

- Returns:** The result of issuing a **repeat** command depends on the contents of the command line or lines which are repeated and whether the **-d** option has been specified:
- If the **-d** option is used, the command lines are displayed on the screen, not executed. The displayed command lines are not returned to a program nor are they written in the log file.
 - If the **-d** option is not used, the command lines are executed. If a command line is repeated that returns data to a program, then this data is passed to a program provided a program issued the **repeat** command.
 - If the **repeat** command (or other commands) is issued by an interactive user or from within a run file while a program is active, data is not returned to a program. For example, if a program executes a run file which then issues a **repeat** command, no data is returned to the program by either the repeated commands or any other commands executed by the run file.

Example 1: `repeat -d -3`

This command displays the previous three command lines.

The resulting output in scroll format:

```

^    99 p100 34 dx
^    100 p0 32004 dx
^    101 32005 dx
> 1 32006 3_0 C 3_2_1_1 - 0 000 Proc:000 Err:___ Fld:___
$

```

Example 2: `repeat 100`

This command repeats command line numbered 100. Note the difference between **repeat 100** and **repeat -100**. **Repeat 100** re-executes a single command line, the one numbered 100. **Repeat -100** re-executes the last 100 command lines.

The resulting output in scroll format:

```

> 32006 3_0 C 3_2_1_1 - 0 000 Proc:000 Err:___ Fld:___
$ p0 32004 dx
> 1 32004 3_0 C 3_2_1_1 - 0 000 Proc:000 Err:___ Fld:___
$

```

- Notes:**
- The **repeat** command must be issued on a command line by itself.
 - If a log file is open, and the **repeat** command is issued with the **-d** option, the output is not written to the log file.

See Also: **program** (Refer to Chapter 7), **run** (Refer to Chapter 6).

rs—Reset

Use **rs** to reinitialize the current procedure. All displayed fields are empty. For administration procedures, **rs** reinitializes the currently displayed word. For a maintenance procedure, the nominal start configuration is displayed. Most often, this is test 1, displayed with blank fields. (Some maintenance procedures require a second reset before all fields are cleared.)

rtx—Run Tape Execute

Use **rtx** to copy translation memory onto your switch's tape cartridge (or to disk, if your switch is so equipped). On all Generic 2, and on System 85 R2V3 and R2V4, access all three system modes (administration, maintenance, and tape) in the mode procedure before issuing the **rtx** command.

run—Execute Run Files and Log Files

Use **run** to execute run files or log files. The **run** command uses the syntax given below.

```
run [-e max_nbr_errors_to_permit] filename 
```

For further details, refer to the **run** command description in Chapter 6, "Run Files."

rx—Remove-Execute

Use **rx** with an administration procedure to remove displayed data from the switch's translation memory. (See the **rmv** command description.) Many procedures require that you issue the **dx** command to display the data you want to remove before you can use the **rx** command. Always follow an **rx** command with a **dx** command to view the results of your work.

s—Stop

Use **s** to halt test execution when a maintenance procedure test is running. Not all maintenance tests recognize this command. To stop such tests, request another procedure using the **p** command.

scroll—Scroll Screen Format

Use *scroll* in Enhanced or Basic Mode to switch from visual (full-screen) format to scroll (line-by-line) format. Visual format is the default format. The shorthand notation for **scroll** is **scr**, **scro**, or **scrol**.

The scroll format displays information one line at a time. The user command line is displayed followed by error messages and switch responses as appropriate. The command prompt is a dollar sign (\$). The following is an example of **scroll** format.

```
$ p0
> 1 ----- Proc:000 Err:___ Fld: 01
$
```

Scroll format is particularly useful for examining and comparing repetitive data like that found in maintenance procedures using the **nc** command or administration procedures using the **nd** command.

Scroll format can be invoked in either of two ways. Invoke Proc Mode with the *-l* parameter (for example, *mgrii -l*) (if the *-l* option is not used, the default Proc Mode screen format is visual) or enter *scr* on the command line. To switch back to visual format, enter *vis* on the command line.

Either of the screen formats may be selected from Enhanced or Basic Mode. The screen format chosen has no effect upon Enhanced Mode features, such as equipment location and character mapping.

silent—Silent Screen Format

The **silent** command is used to turn off output to the screen until a **visual** or **scroll** command is issued. This capability is particularly useful from programs and run files. The screen output may be turned off during tedious tasks that you need not view. The **silent** command can only affect screen output. Output continues to be passed to a program if one is active and to be written to a log file if one is open.

The **silent** command can be issued from the command line interactively, from a run file, or program.

The equivalent VMAAP command is **set**.

The shorthand notation for **silent** is **sil**, **sile**, or **silen**.

See Also: **Visual and Scroll.**

ssb—Switch Support Base

Proc Mode allows you to specify which Switch Support Base (SSB) files are accessed to produce Enhanced Mode screens. A Switch Support Base consists of the Electronic Flip Chart (EFC) files associated with a switch version, release, issue, and dot issue. The EFC files include procedure screen layouts and help information.

The `ssb` command is used to override the default Switch Support Base selection made by Proc Mode. Generally, there is no reason to use a Switch Support Base other than the one chosen by Proc Mode. However, if the need arises, you may issue the `ssb` command to select another Switch Support Base. This command can also be used to re-select the default.

Any time you issue an `ssb` command with no arguments, whether or not a connection exists, Proc Mode chooses the default Switch Support Base.

With NO CONNECTION, the default Switch Support Base is selected as follows:

- The default switch release and version are extracted from the initialization file which should have an entry similar to:

```
DEFAULT_SWITCH=r2v5
```

- The most up-to-date issue and dot issue are extracted from the `$$SSB/DEFAULT_SWITCH/path.ssb` file. The Environment Variable `SSB` specifies the main SSB directory. Subdirectories for various switch releases/versions may appear under the SSB directory. Under each release/version subdirectory, there should be a `path.ssb` file. For more details on environment variables, refer to “Environment Variables” in Chapter 7.

With an ACTIVE CONNECTION, the default Switch Support Base is selected as follows:

- The release and version of the current switch are used as defaults.
- The issue and dot issue closest to the issue and dot issue of the current switch are extracted from the `$$SSB/current_switch/path.ssb` file. The `$$SSB/current_switch` subdirectory is determined using the release and version of the current switch. For example, if the connection is made to a release 2, version 5 switch, then the `path.ssb` file is located in the `$$SSB/r2v5/path.ssb` subdirectory.

If a `path.ssb` file is not found, Proc Mode cannot provide Enhanced Mode screens. The screen mode reverts to Basic Mode. Table 5-2 is an example `path.ssb` file for Generic 2.1 switches. It contains entries for issue/dot issues 3.0, 2.0, and 2.0.xxxx.

TABLE 5-2
Example path.ssb files located in the directory /mgr/ssb/r2v5

```
DOT_ISSUE: 03.00
ADM_PATH=/mgr/ssb/r2v5/e419/admin
MNT_PATH=/mgr/ssb/r2v5/e419/maint
HLP_PATH=/mgr/ssb/r2v5/e419/help

DOT_ISSUE: 02.00
ADM_PATH=/mgr/ssb/r2v5/e419/admin
MNT_PATH=/mgr/ssb/r2v5/e419/maint
HLP_PATH=/mgr/ssb/r2v5/e419/help

DOT_ISSUE: 02.00.xxxx
ADM_PATH=/mgr/ssb/r2v5/xxxx/e419/admin:/mgr/ssb/r2v5/e401/admin
MNT_PATH=/mgr/ssb/r2v5/xxxx/e419/maint:/mgr/ssb/r2v5/e401/maint
HLP_PATH=/mgr/ssb/r2v5/xxxx/e419/help:/mgr/ssb/r2v5/e401/help
```

You can issue the *ssb* command from the command line interactively, from a run file, or program. The resulting output will be displayed on the screen, and saved in a log file if one is open.

An SSB Directory

Proc Mode must have access to the Switch Support Base files which are stored in subdirectories of the directory pointed to by the Environment Variable **SSB**. Make sure that the Environment Variable is set to this directory. See “Environment Variables” in Chapter 7.

If the Environment Variable is not properly initialized, Proc Mode cannot access Switch Support Base files. Without the Switch Support Base files, Proc Mode displays Proc Mode procedures in *Basic Mode*.

Command Syntax

There are several forms of the **ssb** command. Depending on the syntax used, you may specify one of the following types of the Switch Support Base files:

- The Default Switch Support Base.
- A Particular Switch Support Base.
- A Particular Custom Switch Support Base.

The Default Switch Support Base contains information associated with the release, version, issue, dot issue, and custom product identifier (if applicable) of the current switch.

If the default Switch Support Base does not match the switch, you may need to change the default to a particular Switch Support Base or Custom Switch Support Base by issuing the **ssb** command with the appropriate arguments.

A particular Switch Support Base or Custom Switch Support Base specified by an **ssb** command is used only until another **ssb** command is issued, a new connection is made, or a switch disconnect occurs.

You may also request that Proc Mode display the name of the Switch Support Base currently in use.

Command: *ssb*
 ssb release_version issue.dot_issue [custom_identifier]
 ssb -c custom_identifier
 ssb ?

Options: *-c*

The *-c* option specifies the custom product identifier to use with the release, version, issue, and dot issue of the current switch connection.

The custom identifier option is only valid if there is a currently active connection.

Arguments: The **ssb** command with no arguments specifies that the default Switch Support Base be used.

release_version

The argument *release_version* is a switch release and version. The *release_version* is at most six characters.

issue

The argument *issue* is a switch issue. An issue number is always two digits. If only one digit is specified in the **ssb** command, Proc Mode automatically precedes that digit with a zero.

dot_issue

The argument *dot_issue* is a switch dot issue. A dot issue is made up of exactly two digits. If only one digit is specified in the **ssb** command, Proc Mode automatically precedes that digit with a zero.

custom_identifier

The argument *custom_identifier* specifies a Custom Product identifier.

This identifier consists of up to 4 characters. Custom identifiers only apply to custom switches. This argument is optional.

?

The argument ? means display on the message line the name of the Switch Support Base currently being used.

Returns: No data is returned to a program as a result of an *ssb* command. A status message with the name of the current Switch Support Base is displayed on the screen and entered in the log file if one is open.

Example 1: `ssb r2v5 1.3`

This command specifies that the Switch Support Base associated with switch release 2, version 5, issue 1, and dot issue 3 be used.

Example 2: Following is a sample log file that demonstrates how you may use the *ssb* command to manipulate the current Switch Support Base.

```
< ssb ?
~ 521=SSB in use: G2.1 Issue 03.00
< con switch_name
~ 235=Connecting to switch. Press <Del> key to abort.
~ 425=Dialing complete. Initiating switch login.
~          0003E 02-0300.06.00
~ Connected to: switch_name          0003E 02-0300.06.00
> 0 0 0 70 ----- Proc:mode Err:___ Fld:___
< ssb ?
~ 521=SSB in use: G2.2 Issue 00.00
> 0 0 0 70 ----- Proc:mode Err:___ Fld:___
< ssb g2.1 3.0
~ 521=SSB in use: G2.1 Issue 03.00
> 0 0 0 70 ----- Proc:mode Err:___ Fld:___
< ssb
~ 521=SSB in use: G2.2 Issue 00.00
> 0 0 0 70 ----- Proc:mode Err:___ Fld:___
< ssb -c cm2
~ 521=SSB in use: G2.2 Issue 00.00.CM2
> 0 0 0 70 ----- Proc:mode Err:___ Fld:___
< ssb
~ 521=SSB in use: G2.2 Issue 00.00
> 0 0 0 70 ----- Proc:mode Err:___ Fld:___
< disc
~ 100=SWITCH DISCONNECTED.
< ssb ?
~ 521=SSB in use: G2.1 Issue 03.00
```

- The initial **ssb ?** command displays the Switch Support Base *G2.1 Issue 03.00* (the default prior to making a connection).
- After making a connection, the **ssb ?** command displays the current Switch Support Base *G2.2 Issue 00.00* (default).
- The next **ssb** command selects the new Switch Support Base *G2.1 Issue 03.00*. This overrides the default selection corresponding to the current connection.
- The command **ssb** with no arguments restores the default *G2.2 Issue 0.0* Switch Support Base.
- The command **ssb -c cm2** changes the current Switch Support Base to the one with a custom product identifier *G2.2 Issue 00.00.CM2*.
- The **ssb** command with no options changes the current Switch Support Base to the default *G2.2 Issue 00.00* associated with the current connection.
- If a switch disconnect occurs, the original default Switch Support Base *G2.1, Issue 3.0* is selected. This Switch Support Base is not associated with a connection since there is none.

Note: Switch Support Base for a particular version, release, issue, and dot issue of a switch must be installed before Proc Mode may use it. If you request a Switch Support Base which cannot be found, Proc Mode displays an error message and does not change the current Switch Support Base selection.

See Also: **get**, **getsym**, **symbols** (Refer to the TSC document *Services Commands*).

status—Display Connection Status

Use **status** to display the status of the current connection. This includes switch name, customer id, product id (Procedure 497), type (for example, *G2.1 Issue 01.000*), and SSB (for example, *G2.1 Issue 01.00*). To exit the status screens, press the function key labeled **1 Exit**. You can use the **status** command in programs. The shorthand notation for **status** is **sta**, **stat**, or **statu**.

The **status** command uses the syntax given below.

```
status [-s] [-c /-d /-i /-p /-t /-b /-l]
```

For further details, refer to the description of the **status** command in Chapter 7, “Proc Mode Programming.”

sw—Switch (Change) Common Controllers

Use **sw** to switch the remote end of the currently selected connection to communicate with the other common controller (CC) in a duplicated CC machine. This command works only if your connection is through the RMATS port (TN492) of your system. For example, suppose communication is currently with common control 0 (CC0) through an RMATS port. Issuing the **sw** command switches communication to common control 1 (CC1). This change is reflected in the status line (fourth from the bottom) of the screen.

t—Test

Use **t** followed by the number of the desired test to change from one maintenance procedure test to another. For example, to request Test 2, enter *t2*

Most procedures have either words or tests. Procedures with words are generally used for administration while procedures with tests are used for maintenance. Use the word command **w** to change from one procedure word to another. Requesting a test from a procedure with words or requesting a word from a procedure with tests results in an error. Some procedures need only one screen, and therefore have neither words nor tests. It is not necessary to request test 1 or word 1 when calling up a procedure for the first time since test 1 and word 1 are the defaults. A command closely related to the test command is the next test command, **nt** (described elsewhere in this chapter).

talk

The **talk** command is used to briefly halt program or run file execution, so you can interactively execute Proc Mode commands. Only a program or run file may initiate a “talk” condition. To terminate a “talk”, enter the **continue** command.

The equivalent VMAAP command is **talk**.

Typically, before entering the **talk** command, a program or run file uses a **note** command to announce that a “talk” condition is about to begin. Also, any instructions may be displayed at that time via **note** commands.

You can run **talk** as follows:

Command: `talk`

Once you are finished interactively entering commands, use the **continue** command to return control to the program or run file execution.

Returns: Data is not returned to a program as the result of any command issued during a “talk” session, even if the session is initiated by a program.

Example: Suppose a program contains the following lines:

Issuer of Commands	Command
Program	note Adjust p000. Then enter the CONTINUE command. talk
Interactive User	32006 dx continue
Program	note Work on p000 is complete.

Resulting output in scroll format:

```

$ note Adjust p000. Then enter the CONTINUE command.
~ Adjust p000. Then enter the CONTINUE command.
> ----- Proc:000 Err:___ Fld:01
$ talk
> ----- Proc:000 Err:___ Fld:01
$ 32006 dx
> 1 32006 3_0 C 3_2_1_1 - 0 000 Proc:000 Err:___ Fld:___
$ continue
> 1 32006 3_0 C 3_2_1_1 - 0 000 Proc:000 Err:___ Fld:___
$
$ note Work on p000 is complete.
~ Work on p000 is complete.

```

Notes:

- A **talk** or **continue** command must appear as the last command on the command line (i.e. a **Return** must follow immediately after the command).
- If you enter the **continue** command when no “talk” session is active, the command is ignored, and an error message is displayed.
- A **talk** command may only be issued by a program or run file. Do not issue a **talk** command interactively.
- Once a **talk** command is issued by a program or run file, you are permitted to interactively enter commands until a **continue** command is issued.
- Once a **talk** session has begun, you are not allowed to interactively enter another **talk** command. Basically, multiple “talk” sessions are not permitted.
- Run files and programs may not be executed from within a “talk” condition.
- If you do not enter any command during the “talk” session, the activity timer will eventually time out. Proc Mode will abort the controlling run file or program.
- To terminate a “talk” session and the run file or the program which initiated it, press **Delete**.

See Also:

ask, continue, note, program

task—Task Mode

Use **task** to change Proc Mode to the Task Mode. See Chapter 2 in the *DEFINITY Manager III Operations* manual for a description of Task Mode.

v — Verify Display

Use **v**, if you suspect the validity of any of the procedure data on the screen, to completely redisplay the screen. For Generic 2, the command also gets the data from the switch.

visual — Visual

Use **visual** in Enhanced or Basic Mode to switch from scroll (line-by-line) format to visual (full-screen) format. Visual format is the default format. The shorthand notation for **visual** is **vis**, **visu**, **visua**.

To switch back to scroll format, enter `scr` on the command line.

Either of the screen formats may be selected from Enhanced or Basic Mode. The screen format chosen has no effect upon Enhanced Mode features like equipment location and character mapping.

w — Word

Use **w** followed by a word number to access a word other than the currently displayed word when executing multiword administration procedures. For example, to request word 2, enter `w2`.

Most administration procedures have words. The word command **w** is used to change from one procedure word to another. Requesting a word from a procedure with tests results in an error. Some administration procedures need only one screen, and therefore have no words. It is not necessary to request word 1 when calling up a procedure for the first time since word 1 is the default.

x — Execute

Use **x** to initiate some activity. The activity will depend on the current procedure. In many cases, **x** is used in conjunction with another command. For instance, administration procedures routinely accept the following command combinations:

ax: add execute

cx: change execute

dx: display execute

Some of these procedures interpret **x** or **dx** to mean the same thing: display execute. With respect to maintenance procedures, the execute command is usually used by itself. The **x** command typically tells the procedure to display fault information or to start a maintenance test.

Some maintenance procedures call for **x** to be entered twice. (Because of the importance of the test or activity, the procedure requires the execution command be confirmed.) When performing such a procedure, press **Return** after the first execution of **x**. Then enter the second **x** command on the next command line, followed by another **Return**.

Function Key Reference

Access the functions listed below by pressing function keys when their labels are present on the Basic or Enhanced Mode screen. Depending on the command you enter, function key labels change dynamically.

Exit Menu

Press **1 Exit** to exit the current menu or other auxiliary screen and return to the command line.

Get SSB Files

Press **2 Get** to initiate the process of getting SSB files from the system.

Repeat Previous Command Line

Press **2 Repeat** to re-execute the previous command line.

Display Previous Information

Press **2 Return** to return to the previous display.

Move Cursor to Command Line

Press **3 Line** to move the cursor from the active data field to the command line so that you can execute a command.

Move Cursor to Data Field in Form

Press **3 Form** to move the cursor from the command line to the active data field so that you can enter data directly into the field.

Select a Menu Item

Press **3 Select** to select the menu item that is currently highlighted.

Clear Field

Press **4 Clear** while you are editing fields in a menu or procedure to clear the active field.

Find a Menu Item

Press **4 Find** to search an extensive menu. To use this facility, press this key and then enter an expression. The Help facility attempts to match that expression with a menu item. If a match is found, the highlight bar is moved to the matched item. Otherwise, the highlight bar is moved to the first item in the menu. (See the description of the **help** command earlier in this chapter for further details).

Display Error Codes for a Procedure

Press **5 Errors** to display error message numbers and meanings that can arise while using a procedure.

Display General Help

Press **5 Help** to access a system of help menus that describe the details of Proc Mode operation. (Use these menus as described for the **help** command in "help or h—General Help"). When you are connected to a switch, general help on the active procedure is displayed. When you are not connected to a switch, general help on Proc Mode is available.

Display All SSB Files

Press **6 All** to make Proc Mode display all possible SSB files.

Display Different SSB Files

Press **6 Delta** to make Proc Mode display only those SSB files on the current switch that are different from your SSB.

Display Field Data

Press **6 Field** while you are viewing fields in a menu or procedure, to access a description of data that may be entered in the active field. This description may be in the form of a range of possible values, if appropriate. Or, if more information is called for, this description may take the form of a window with text that you can scroll by means of the **↑**, **↓**, **Page Up**, and **Page Down** keys.

In the latter case, a highlighted bar may appear in the window. To select one of the encodes described in the window, place the highlighted bar on your choice and press **Return** or **3 Select**. To exit the field-help facility without making a selection, press **1 Exit**.

Display Input Data Required for a Procedure

Press **7 Input** to find out the minimum amount of data required by the current procedure before a command can be executed. This information may appear in a window. If so, use the **↑**, **↓**, **Page Up**, and **Page Down** keys to scroll through it to find the information you need.

Display Commands

Press **8 Cmds** to access a display of all Proc Mode commands. If you require further information about these commands, press **1 Exit** to leave the display. Then press **5 Help** to view more detailed help.

Special Key Reference

Listed below are brief descriptions of all the special keys used with Proc Mode in procedure screens, help screens and on the command line.

When in a procedure screen (entered by pressing **3 Form**):

Key	Action
Backspace	Press to delete the character to the left of the cursor in a field.
Return	Press to enter data in the current field and move the cursor to the <i>next</i> field.
Tab	Press to enter data in the current field and move the cursor to the <i>next</i> field.
Shift Tab	Press to enter data in the current field and move the cursor to the <i>previous</i> field.
→	Press to enter data in the current field and move the cursor to the <i>next</i> field.
←	Press to enter data in the current field and move the cursor to the <i>previous</i> field.
↓	Press to enter data in the current field and move the cursor to the <i>next</i> field.
↑	Press to enter data in the current field and move the cursor to the <i>previous</i> field.

When in a help screen (entered by pressing, for instance, **5 Help**):

Key	Action
↓	Press to scroll <i>down</i> one line.
↑	Press to scroll <i>up</i> one line.
Page Down	Press to scroll <i>down</i> one page.
Page Up	Press to scroll <i>up</i> one page.

When on the command line:

Key	Action
Space Bar	Press to move the cursor one space to the right.
Tab	Press to move the cursor one space to the right.
Back Space	Press to delete the character to the left of the cursor.
Return	Press to execute the commands typed on the command line.

When in any screen or on the command line:

Key	Action
Print Scn	Press to print the screen on a printer.

6 Run Files

Run Files	6-1
New Terms	6-1
Commands Used with Run Files	6-2
Executing Run Files	6-3
run	6-3
Creating Proc Activity Logs	6-5
run proc	6-6
log	6-7
Executing Run Files from within a Manager III Script	6-8
Script Commands and Run Files	6-8
Scheduling Run Files Using Manager III	6-9
Checking Pre-Generic 2 Run Files	6-10
Canceling Pre-Generic 2 Run Files	6-10
Run File Results	6-10
Resubmitting a Run File	6-11

Run Files

Why?	To string together complex or frequently used procedure commands.
When?	Whenever you need to: <ul style="list-style-type: none">■ save a set of Basic or Enhanced Mode procedure commands to be executed or scheduled later.■ to schedule a "run tape" to run overnight.
Who?	Only the user that has UNIX "read" permission can run a run file.
Notes	<p>When using a run file to schedule a "run tape" to run overnight, be sure that</p> <ul style="list-style-type: none">■ The run file accesses the Mode Proc to obtain administration, maintenance, and tape modes (modes 1, 2, and 3).■ There is a writable tape in the switch tape drive.■ You don't schedule any other scripts for at least 30 minutes after the run tape is scheduled, so that it has enough time to complete. While the run tape is executing, any other scheduled scripts executed will fail. <p>Run files can be run from the command line, included in scripts for scheduling later, or be scheduled with the Work Organizer.</p> <p>A detailed description of run files is provided in <i>DEFINITY® Manager III Operations</i>.</p>

New Terms

Run File	a list of Proc Mode commands.
Script	a list of several Manager III Task Mode commands to be run in sequence.

Commands Used with Run Files

Run files and log files are manipulated using standard UNIX commands, as summarized in the following table.

TABLE 6-1
Run File Commands

To	Use UNIX Command
Check results	cat or pg log file
Create or edit	vi or ed
Display or review	cat or pg run file
List	ls
Remove	rm
Reset	Run files always run from the beginning, regardless of success of transactions. Use vi to edit out successful transactions if rerunning a partially successful run file.

Executing Run Files

Run files allow you to display, change, or remove switch administration data by issuing Proc Mode commands. Most Proc Mode commands may be issued from a run file.

Run files may be executed by using the **run** command. The commands contained in the run file are executed in the order that they appear in the file. As a line is executed, it is displayed on the screen and entered in the log file, if one is open.

run

There are three types of files that can be executed from within Proc Mode: log files, run files and programs. The **run** command is used to execute run files and log files.

The equivalent VMAAP commands **:[** and **:<** perform similar functionality.

When run file execution completes, a count of the accumulated errors is displayed on the message line. If the run file was the only file to execute, the count includes only errors generated by that file. However, the cumulative errors continue to grow if one run file or program executes another.

For example, a program executes a run file. Then the error count displayed when the run file completes includes errors which occurred during program execution before the run file started and all run file errors. Another error count is displayed when the program completes. It tallies all errors generated by both run file and program including errors which occurred after the run file completed.

Following is the **run** command syntax:

Command: `run [-e max_nbr_errors_to_permit] filename` Return

Options: `[-e max_nbr_errors_to_permit]`

The `-e` option is used to specify the number of errors to permit before aborting run file execution. `Max_nbr_errors_to_permit` is an integer number specifying the maximum number of errors to permit. If the `-e` option is not specified, the default number of errors to permit is zero.

Arguments: `filename`

The argument `filename` specifies the name of a log file or a run file to be executed. If you execute a log file, the **run** command interprets the prefixes used in these files as follows:

- Lines beginning with a `<` prefix are treated as command lines. The `<` prefix is ignored. The rest of the command line is executed.
- Lines beginning with either `>` `^` `~` or `!` are treated as comment lines (that is, lines beginning with `#.`)

Returns: Data is not returned to a program as a result of any command issued by a run file, even if a run file is executed by a program.

Example: For example, run file *file1* generates one error and then executes *file2* which has five errors of its own. After *file2* completes, *file1* execution continues and three more errors are discovered. Two error count messages will be displayed, one after *file2* completes and the second after *file1* finishes:

- *File2 completed with 6 error(s) accumulated.*
- *File1 completed with 9 error(s) accumulated.*

Notes:

- Only one cumulative error count is maintained. Individual totals are not kept for each file.
- You may not execute programs using the **run** command.

See Also: **run proc, program** (*Refer to Chapter 7*).

Creating Proc Activity Logs

Proc Mode log files are used to log all user and switch activities. The following Proc Mode activities are written to the log file:

- Command lines prefixed by <
- Switch procedure images prefixed by >
- Error messages prefixed by !
- Data lines prefixed by ^
- Status messages prefixed by ~

This is a sample log file:

```
! Note: 15 minutes of inactivity will terminate this session.
! 235=Connecting to switch.
! Dialing complete, initiating switch login.
! BLX10 - SPAM Systest 0003E 02-0302.05.00
! Connected to: pluto BLX10 - SPAM Systest 0003E 02-0302.05.00
! Alarms/Status: CC0 ON-LINE MAJOR MINOR RUN TAPE BUSY OUT
! Date: Mon Mar 12 11:11:45
! Alarms/Status: CC0 ON-LINE MAJOR MINOR RUN TAPE BUSY OUT
> 0 0 0 51 ----- Proc:mode Err:___ Fld:___
< m1
! Alarms/Status: CC0 ON-LINE MAJOR MINOR RUN TAPE BUSY OUT
> 1 0 0 51 ----- 0 ----- Proc:mode Err:___ Fld:___
< p010 10 dx cf4 1 cx
> 1 10 1 1 1 0 0 0 1 0 0 0 0 0 0 0 Proc:010 Err:___ Fld:01
< 11 dx cf4 1 cx
> 1 11 0 1 0 0 1 0 1 0 0 0 1 0 0 0 0 0 Proc:010 Err:___ Fld:01
< 12 dx cf4 1 cx
> 1 12 1 1 1 0 1 1 1 1 1 1 0 0 0 0 0 0 Proc:010 Err:___ Fld:01
< 13 dx cf4 1 cx
> 1 13 1 1 1 0 1 1 1 1 1 0 1 0 0 0 0 0 0 Proc:010 Err:___ Fld:01
< 14 dx cf4 1 cx
> 1 14 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 Proc:010 Err:___ Fld:01
< 15 dx cf4 1 cx
> 1 15 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 Proc:010 Err:___ Fld:01
! Time: 11:12:20
```

From Manager III, Proc Mode activities are logged in the server log `$$SWITCH/logs/srvrmmddlog`. In addition, you may have the data copied to your own personal log by using the `log` option of the `run proc` command.

run proc

The Manager III **run proc** command is used to execute Proc Mode and thereby create proc activity logs. The command syntax is as follows:

Command: **run proc** *runfile log logfile error number*

The equivalent VMAAP command is :>

Options: *log*

The option *log* causes a copy of the log from the run proc session to be written to the log file specified. The output for the session is copied to the requested log when you terminate the session by entering the **task** or **quit** command. The log copy is also placed in the server log **\$SWITCH/logs/srvrmmdd log**.

Arguments:

- The argument *runfile* specifies the name of an existing run file.
- The argument *logfile* specifies the name of the log to be created. For more details on log files refer to Chapter 5 in *DEFINITY® Manager III Operations*.
- The argument *number* is the number of errors that can occur before the run file stops. The default is 1.

Returns: Not applicable.

Notes:

- Make sure you provide the correct name for your log file.
- If you do not provide a full path name, the log is placed in **\$HOME/mgr3/logs**.
- Since the log will record your entire Proc Mode session, to record only your program output, follow the steps:
 - a Enter the **run proc log logfile** command.
 - b Execute your program by entering the command:
program program_name
 - c When the execution completes, enter the **task** command to terminate the Proc Mode session. All your program activities will be logged in your log file.

See Also: **run, program** (*Refer to Chapters 5 and 7*).

log

The **log ?** command is used to display the name of the current log file, if one exists, on the message line.

The command syntax is as follows:

Command: **log ?**

You may also use the *status -l* command to obtain the name of the current log file.

See Also: **status** (*Refer to Chapters 5 and 7*).

Executing Run Files from within a Manager III Script

Enhanced Mode commands can be included in a script by calling a run file, or script of proc commands, from the task mode script.

Script Commands and Run Files

When a run file is called from a Manager III task mode script, the "call" is considered a transaction in the script. That transaction can be viewed or edited using any of the script commands. However, script commands do not manipulate the actual run file or allow you to view run file results. You are not placed into Enhanced or Basic Mode while editing the Manager III script. The command to call the run file is simply placed in the script and executed when the script runs.

Run files and log files are manipulated using standard UNIX commands, as summarized in the table "Run File Commands" earlier in this chapter.

To include Enhanced Mode commands in a Manager III script, enter

```
run proc runfile log logfile error number
```

where *runfile* is the name of an existing run file and *logfile* is the name of the file where the results are written.

If the file *logfile* exists, the new log file will append to it.

Procedure: Running a Script Interactively

This procedure explains running a Manager III script called "newuser" interactively.

- 1 Enter the command `run script newuser`

You will be led through each transaction in the script, and given the opportunity to change any data on the command line and the form before submitting the transaction.*

- 2 Each command line is displayed. Press to execute the command or use arrow keys to edit it.
- 3 As the script runs, each transaction screen is displayed. The activity line of the screen shows the script name and step number, and the command line without parentheses:
scriptname-001: add station

- 4 Press to run each transaction as it displays.

The results of each transaction are displayed on the screen.

- 5 If an error is displayed, you can press to cancel that transaction, or fix the field in error and press to submit the transaction again.

* Add the optional qualifier "nui" (no user input) to the "run script" command to run a script immediately without an opportunity to display and correct screen information.

Note: If the command line calls a Proc Mode Run File, the run file executes without any opportunity for change. When the run file is complete, the next command in the script is displayed on the command line.

Scheduling Run Files Using Manager III

Run files can be included in scripts for scheduling later. Scheduling increases the amount of time available for switch administration to 24 hours per day.

Procedure: Scheduling Generic 2 Run Files

To schedule a Generic 2 run file, include the run proc command in a script, and schedule the script. For example, to schedule the run file "callhold5", enter the following command in a script:

```
run proc callhold5 log callhold5.log error 1
```

You can specify a full path name for the run file; otherwise, the run file must be in the current working directory. If you do not specify a log name (which can include a full path name), the log file will be written to the default file name **\$HOME/mgr3/logs/callhold5.log**.

Procedure: Scheduling Pre-Generic 2 Run Files

Run files for pre-Generic 2 switches cannot be scheduled through scripts. They can be scheduled using the UNIX **at** command.

Before a user can schedule with the **at** command, the System Administrator must add the user to the file **/usr/lib/cron/at.allow**. See the *UNIX System V Release 3 User's Reference Manual* for information about the **at** command.

Users with permission to use this command can follow these steps to schedule a run file. This is done from the UNIX shell.

- 1 Create the run file.
- 2 At the UNIX **\$** prompt, enter the **at** command using military time as in this example:

```
at 23:55
$SPAMDIR/bin/runfile switch runfile [logfile] [errors] >resultsfile
```

where:

switch is the switch name

runfile is the full path name of the run file, for example, /user/jcm/run1

logfile is the full path name of the log file, for example, /user/jcm/run1.log

[errors] is the number of errors encountered before the run file stops

resultsfile is the name of the file where results will be written.

You can substitute /dev/null if you do not want to see results.

- 3 Press `Ctrl-d` to end the `at` entry.

A message in this format is displayed:

```
job 123456754.a at Tues Feb 20 23:55 1990
```

- 4 The system forwards a mail message to the user with the results of the `at` command.

Checking Pre-Generic 2 Run Files

To list jobs that are scheduled with the `at` command, at the `$` prompt, enter: `at -l`.

This command will display the job number required to cancel the job.

If you do not get a mail message with results, check the `$$SWITCH/logs/errmdd` log after the time that the job was scheduled to run.

Canceling Pre-Generic 2 Run Files

To cancel a job that is scheduled with the `at` command, enter: `at -d <job number>`. The job number is obtained by using the `-l` option as discussed above. See the *UNIX System V Release 3 User's Reference Manual* for more information.

Procedure: Reusing a Script

Use the command `reset script` to clear the old results so the script can be rerun from the first transaction. To reset the script called "newuser", enter the command `reset script newuser`.

Use this command to:

- rerun a script that ran successfully.
- completely rerun a partially successful script. If you run a script that was partially successful without editing or resetting it, the script will start at the first failed step.

Note: The `reset script` command does not reset Enhanced Mode run files. See "Resubmitting a Run File" later in this chapter.

Run File Results

Run file results are written to either:

- the log file that was specified when the `run proc` command was run, or
- the default file `runfile.log` (where `runfile` is the name of the run file) in the `$$HOME/mgr3/logs` directory. For example, results of run file "callhold5" would be written to `callhold5.log`.

If you do not find any results after a scheduled run file has run, first check the log in `SWITCH/logs/err` and `SWITCH/logs/srvrmdd` and then look in `$$SPAMDIR/logs/errmdd` after the time the run file was scheduled to execute.

The results file contains status messages from the switch, Proc Mode commands from the run file, and the output or error codes from the switch for each proc command. Use the UNIX commands **pg** or **vi** to view the results file. If desired, edit the run file to correct any errors.

Resubmitting a Run File

If you are executing a run file that has already been submitted once, use the UNIX command **rm** to remove the log file before resubmitting the job. Otherwise, the results file of the run file will be appended to the existing run file. When you resubmit a run file, all procs will re-execute, even if some were run successfully.

To correct errors in run files, or to remove procs that have already executed, edit the run file using the UNIX commands **vi** or **ed**.

The **remove script** command does not remove run files or their associated log files. They must be removed with the UNIX **rm** command.

7 Proc Mode Programming

Proc Mode Programming	7-1
Writing Programs	7-2
General Guidelines	7-2
Environment Variables	7-3
Conventions and Considerations	7-4
Executing Programs	7-6
program	7-6
Exiting the Program	7-8
How To Use Some Typical Programming Commands	7-9
Converting VMAAP Child Processes to Proc Mode Programs	7-11
Programming Commands	7-13
alarms	7-13
ask	7-17
continue	7-19
fields	7-20
math	7-23
note	7-29
status	7-31
talk	7-34

Proc Mode Programming

The Proc Mode programming feature allows you to automate your switch administration and maintenance by writing programs that interact with Enhanced and Basic Modes (Proc Mode) to provide a sophisticated and flexible way of issuing Proc Mode commands. Enhanced and Basic Modes are described in detail in Chapters 3 and 4 of this manual.

You can access Proc Mode to use the programming feature from the Manager III, IV, or Trouble Tracker application. You may write programs in the C Programming Language, UNIX Shell Command Language, or other similar languages (within certain limitations) that use the commands discussed in this chapter. Once a program is written, it is executed by issuing the **program** command. See the **program** command description.

While executing, a program writes Proc Mode commands to its standard output. Proc Mode reads the program output and then executes the commands in the order that they are received. Some of these commands may cause Proc Mode to return information to the program. The program retrieves the data by reading its standard input.

Proc Mode commands can be issued from a run file. Run files are explained in detail in Chapter 6, "Run Files." Logic structures available from programming languages make programs a more flexible way to issue Proc Mode commands than simple run files. Programs may vary the commands issued depending upon the current situation. When a run file executes, the same commands are always done in the same order.

While most Proc Mode commands may be used from a run file or a program, there are a few, in particular, that are the most useful. These commands are referred to collectively as the programming commands. The Proc Mode programming commands provide the following functionalities:

- Execute run files or programs from within Proc Mode.
- Provide an interface between your executable programs and Enhanced and Basic Mode procedures.
- Evaluate math expressions.
- Extract switch data.
- Read and write switch memory, Common Control and Input/Output ports (*Services* commands, TSC only).
- Execute previously entered command lines.
- Provide interface to switch public symbols (*Services* commands, TSC only).

Note: *Services* commands: **fm**, **getsym**, **nio**, **rio**, **symbols**, **rm**, **wm** can be used by a Super User only. These commands are described in detail in the TSC document *Services Commands*.

Writing Programs

This chapter explains in detail how to write programs using the Proc Mode commands that return data to the program: **alarms**, **ask**, **fields**, **math**, and **status**. A description of **note** is included, as well. All of these commands are available for use by any class of user.

A list of the commands and a brief description of their use are provided in Table A-3 "Programming Commands Which Do Return Data"; see "Appendix A. Proc Mode Quick Reference." Examples of C and Shell programs using these commands are explained in "Appendix B. Programming Examples." The programming examples demonstrate how to correctly and most efficiently utilize the Proc Mode programming functions in your programs. Follow the guidelines and programming examples to obtain the best possible results.

General Guidelines

Proc Mode programming commands provide you with a powerful tool to administer and maintain your switch. Writing programs that may access the switch database through these commands, change the contents of the memory locations, and reconfigure the switch puts upon you responsibility for the consequences to your switch that may result from a programming error. A detailed description of programming commands is provided later in this chapter. For information on the equivalent VMAAP commands, refer to *VMAAP Switch Administration and Maintenance with a UNIX® PC (585-206-701)*.

WARNING:

AT&T takes no responsibility for any program you write. You are responsible for the use of the Proc Mode programming feature and the effect that the execution of your program may have on your equipment.

To write programs which perform the desired operations with less chance of error, follow these guidelines:

- You are strongly urged to take a course in C or Shell programming if you do not have any programming experience.
- Make sure you have documentation for C and/or Shell.
- Do not write a large program at once. Get the overall logic in and then expand your program by adding additional functions.
- Since your program gives commands to Proc Mode by writing to the standard output device and reads the response from Proc Mode by reading from the standard input device, test the logic and decision making of a program by running your program from UNIX. The commands you issue to Proc Mode will then be printed on your terminal, and you can enter possible responses from your terminal.
- Always comment your code. The comments always help you or anyone who will have to work with your program later to follow the logic of the code.

- Include **note** commands throughout your code. This will notify you about the actions performed by your program. For example, in a C program, include a statement like:

```
puts ("note Removing trunk group 107 from switch b1x6.")
```

For a Shell script, the command is:

```
echo "note Removing trunk group 107 from switch b1x6."
```

Environment Variables

Proc Mode uses certain Shell Environment Variables to locate its programs and other files:

MGRII	Set the MGRII Environment Variable to the directory path where all Proc Mode executable files are installed.
MII_PGM	Set the MII_PGM Environment Variable to the directory path where an executable program or UNIX Shell script is installed. If the MII_PGM is not set, Proc Mode will search for this program in the current directory.
MII_RUN	Set the MII_RUN Environment Variable to point to the directory containing run files. The default is the current working directory.
MII_LOG	Set the MII_LOG Environment Variable to specify the directory where <i>log</i> files are created. The default is the current working directory.
PUB_SYM	Set the PUB_SYM Environment Variable to the directory path where all Public Symbol files are installed. There is no default directory. If this variable is not initialized, Proc Mode will not access Public Symbol files.
SSB	Set the SSB Environment Variable to the directory path where all Switch Support Base versions are installed. If this variable is not initialized, Proc Mode will not access Switch Support Base files.

Conventions and Considerations

When using Proc Mode programming commands, consider the following:

- Most commands can be executed from the command line, a run file, or a program. *However, a few commands are restricted to the command line and cannot be used in run files or programs. Also, some commands may only be used within a program.* Restrictions placed on the commands are explained for each command later in this chapter.
- The output of a command is consistent whether it is directed to the screen, a log file or a program. For example, when an **alarms** command is issued, the data displayed on the screen is identical to the data entered in the log file or the data returned to a program executing from Proc Mode.
- The **-s** option is used consistently throughout the programming commands to mean *silent* (i.e. no screen output).
- Proc Mode displays and logs three types of messages which are indicated by different prefixes appearing at the beginning of these messages:

User Input	A < prefix appears in front of all user command lines in log files. A \$ prefix is the prompt used in scroll format instead of the prompt Enter command found in screen format.
Error, Status, and Data Messages	The ! prefix is used with an error message both when displayed on the screen and when entered in a log file. A ~ (tilde) prefix appears in association with a status message. A ^ (caret) prefix denotes a data message.
Switch Procedure	A switch procedure image has the prefix > .

- *BEFORE WRITING ANYTHING TO YOUR PROGRAM'S STANDARD OUTPUT, MAKE SURE THAT YOU DEFINED THE STANDARD OUTPUT AS UNBUFFERED.*

In your C program, define the standard output as unbuffered using the function `setbuf(stdout, (char *)NULL)`. For a UNIX Shell script, standard output is automatically unbuffered.

- *TO AVOID COMMUNICATION DEADLOCK, MAKE SURE THAT YOUR PROGRAM CAREFULLY READS BACK ANY OUTPUT IT HAS REQUESTED.*

Your executing program communicates with Proc Mode via pipes. It is important to realize that pipes have a standard finite size, and it is possible to totally fill a pipe. To avoid blocking a pipe, your program must consistently read all data it has requested from Proc Mode.

- *THE PROGRAM SHOULD CONTINUE READING FROM STANDARD INPUT UNTIL THE <EOB> MESSAGE IS RECEIVED.*

Some commands cause Proc Mode to return information to an executing program. The information is passed to a program in blocks of data which may consist of data only, or a mixture of error messages and data. Proc Mode marks the end of such blocks with a special character string appearing on a line by itself:

<EOB> *End-Of-Block* *Marks the end of a data or mixed data and an error message block*

Graphically, data returned to a program may take either of the two general forms shown below. Error messages (! prefix) are optional. Each line is terminated by a new line character.

```

^ xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
[! nnn=error message]
^ xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
[! nnn=error message]
...
^ xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
[! nnn=error message]
<EOB>

~ xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
[! nnn=error message]
~ xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
[! nnn=error message]
...
~ xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
[! nnn=error message]
<EOB>

```

Each line includes a prefix at the beginning of the string which allows you to differentiate between three types of messages: data, status and error. A ~ (tilde) marks a status message; an ! (exclamation mark) denotes an error message, and a ^ (caret) indicates a data message. A single blank space separates the prefix character from the rest of the line.

- Occasional references are made to the AT&T VMAPP product. Such references are intended to simplify learning similar Proc Mode commands for those users already familiar with VMAAP.

Executing Programs

Programs may be executed by issuing the **program** command. Programs allow you to display, change, or remove the switch administration data by issuing Proc Mode commands.

The command **program** is equivalent to the VMAAP command **&**.

program

The command **program** is used to initiate execution of a program that interacts with Proc Mode. To convey requests to Proc Mode, your executable must write to its standard output (*stdout*). Requests must be in the form of typical Proc Mode commands terminated by a new line. Proc Mode receives these commands across the pipe it established for use in communication with your program.

As commands are received, Proc Mode behaves much as it does if the commands are entered at the command line. The commands are displayed provided the **silent** command has not been used to suppress the output to the screen. When a complete command line is received, Proc Mode executes each command in turn. Depending on the command and the current screen format, data may be displayed or changed.

Certain commands cause Proc Mode to return data to your program. These commands are: **fields**, **alarms**, **status**, **ask** and **math**. No other commands send information to your program. To obtain the requested information, your program must read its standard input (*stdin*). The format of the data returned by each command is explained in the section for that command under the heading **Returns**.

When program execution completes, a count of the accumulated errors is displayed on the message line. If the program was the only one to execute, the count includes only errors generated by that file. However, the cumulative errors continue to grow if a program or run file executes another.

If a run file executes a program, then the error count displayed when the program completes includes errors which occurred during run file execution before the program started and all program errors. Another error count is displayed when the run file completes. It tallies all errors generated by both the program and the run file including errors which occurred after the program completed.

These commands cannot be executed from within a program or a run file: **get**, **hc**, **?**, **hf**, **hi**, **help**, **hist**, **put**, **task**, and **udb**.

IMPORTANT: *Programs must define the standard output as unbuffered before anything is written to the standard output. For example, in a C program, use*

```
setbuf(stdout, (char *)NULL);
```

For a UNIX Shell, standard output is automatically unbuffered.

Use the **program** command as follows:

Command: `program [-e max_nbr_errors_to_permit] program_name [arg1 [arg2 [arg3 [...[argn]]]]]`

Return

Options: *-e max_nbr_errors_to_permit*

This option is used to specify the number of errors to permit before aborting program execution. The *max_nbr_errors_to_permit* argument to this option follows these conventions:

- *Max_nbr_errors_to_permit* is an integer number specifying the maximum number of errors to permit during the program execution.
- The number of errors argument must follow the option letter *-e*.
- The number of errors argument may be separated from the option letter by zero or more spaces.

If the *-e* option is not specified, Proc Mode stops the program after the first error.

Arguments: **program_name** [*arg1* [*arg2* [*arg3* [...*argn*]]]]]

The argument *program_name* is the name of an executable program or UNIX Shell script. It may be a simple or a full path name. If a simple name is used, Proc Mode searches for the program in the directory pointed to by the Environment Variable **MII_PGM**. If the Environment Variable is not set, Proc Mode looks for the program in the current directory.

arg1, arg2, arg3, ... argn are arguments to the program *program_name*.

Example: **program -e2 shell_pgm 52025**

where **shell_pgm** is the name of the Shell script that displays data for a specified extension.

This command executes the Shell script *shell_pgm* for extension *52025*; two errors are allowed before aborting the program.

- Notes:**
- You may not execute run files using the **program** command.
 - Only one program may be active at a time.
 - Programs may execute run files by issuing a **run** command.

See Also: **alarms, ask, continue, fields, note, status, talk**

Exiting the Program

Normally, the program itself exits by reaching the end of the file or by executing an "exit" call. Proc Mode resumes operation at the level whence the program was originally executed:

- If the program was initiated from the command line, Proc Mode resumes processing command line input.
- If a run file started the program, Proc Mode resumes reading input from that run file.

To terminate an out-of-control program, press **Delete**. Proc Mode will abort the program. In the event of "nested execution" (when a run file executes another run file or program), pressing **Delete** will abort the entire "nested execution." This means that all run files and programs are aborted.

How To Use Some Typical Programming Commands

Your programs interact with Proc Mode via commands which may accept one or more options or arguments. This table provides examples of C and Shell program excerpts which send commands to Proc Mode and read the data back from Proc Mode. Use this table as your reference guide. These examples make use of several C language functions. The code for these functions [*getstreob()*, *getinteob()*, and *mathinteob()*] appears in Appendix B.

TABLE 7-1
C and Shell Programs Using Commands with One Option Specified

Command	Type	Program
note	shell	echo "note ..."
	C	puts("note ...");
ask	shell	echo "ask ..." read PREFIX RESPONSE read EOB
	C	char response[BUFSIZ]; puts("ask ..."); getstreob(response);
alarms	shell	echo "alarms -s -p" read PREFIX PROCESSOR read EOB
	C	char processor[BUFSIZ]; puts("alarms -s -p"); getstreob(processor);
fields	shell	echo "fields -s 1" read PREFIX FIELD read EOB
	C	int field; puts("fields -s 1"); getinteob(&field);
math	shell	echo "math -s ..." read PREFIX EXP OCT DEC HEX PAREN read EOB
	C	int value; puts("math -s ..."); mathinteob(&value);

TABLE 7-1 (Continued)
C and Shell Programs Using Commands with One Option Specified

Command	Type	Program
status	shell	echo "status -s -i" read PREFIX CUSTOMER ID CUSTOMERID read EOB
	C	char customerid[BUFSIZ]; puts("status -s -i"); getstreob(customerid);

Note: To send a command to Proc Mode which depends on the value of a variable, use *printf("... %d ...\\n", variable)* instead of *puts("...")*.

Converting VMAAP Child Processes to Proc Mode Programs

Proc Mode programs and VMAAP child processes are similar. Most VMAAP commands have an equivalent Proc Mode command. Consider the following when converting a VMAAP child process to a Proc Mode program:

- 1 Continue to use uppercase VMAAP commands, for example, the VMAAP commands **RS** is accepted by Proc Mode.
- 2 The VMAAP **enter** command is a dot (.). The Proc Mode **enter** command is a semicolon (;). Normally, in Proc Mode, the **enter** command is implied, i.e. you do not have to specify this command. If you have more than one dot (.) in VMAAP, use a space for the first dot, and semicolons for the remaining dots.
- 3 To display intermediate messages, the VMAAP command is **:c** . The equivalent Proc Mode command is **note**. It is also possible to write to standard error (*stderr*) by using the C **fprintf** function call.
- 4 Use **ask** instead of **note** to pause and display the last message. For example,
ask "The answer is ?... Press to continue"
- 5 Use **fields n** (not **math #n**) instead of **:q mn** to get a procedure field without a math error. A math error occurs when the field contains a dash or a blank.

Use this cross-reference table as your guide when performing the VMAAP-to-Proc Mode conversion:

TABLE 7-2
VMAAP-to-Proc Mode Cross-Reference List

VMAAP Child Process	Proc Mode Program
char response[BUFSIZ];	char response[BUFSIZ]
puts(":i ..."); gets(response);	puts("ask ..."); getstreob(response);
puts(":C"); gets(response);	puts("alarms -s"); getstreob(response);
puts(":d"); gets(response);	puts("fields -s"); getstreob(response);
puts(":N"); gets(response);	puts("status -s -i"); getstreob(response);
puts(":o"); gets(response);	puts("status -s -c"); getstreob(response);
puts("smath ...");	puts("math -s ..."); getstreob(NULL);
int value;	int value;
puts(":q ..."); scanf("%x\n", &value);	puts("math -s ..."); mathinteob(&value);
puts(":q mn"); scanf("%x\n", &value);	puts("fields -s n"); getinteob(&value);
puts(":i ..."); scanf("%x\n", &value);	puts("ask ..."); getinteob(&value);

Programming Commands

alarms

The **alarms** command is used to produce and display the alarm status of the currently active connection. The alarm status may include information on the following switch indicators and alarms: *CC0/CC1*, *ON-LINE/OFF-LINE*, *MAJOR*, *MINOR*, *RUN TAPE*, *BUSY OUT*, *IN USE*, and *WAIT*. When a connection exists, some or all of these indicators appear near the bottom of the Proc Mode screen. The **alarms** command provides the same information without the screen graphics. The shorthand notation for **alarms** is **ala**, **alar**, and **alarm**.

The equivalent VMAAP alarm command is **:C**.

You can issue **alarms** as follows:

Command: **alarms** [*-s*] [*-p* | *-l* | *-m* | *-n* | *-r* | *-b* | *-u* | *-w*]

Options: *-s*

This option (*silent*) is used to suppress displaying the output related to this command. This is an optional argument and when specified, output is still passed to a program if active, and to a log file if open.

The **alarms** command with no options or only with the *-s* option produces the status of all switch alarms. The information is composed of alarm indicators separated from each other by three blanks.

By specifying an appropriate command option, you may request the status of a single alarm.

-p (processor)

This option requests information about which switch processor is involved in the currently active connection (i.e. *CC0/CC1*).

-l (line)

This option requests the status information of the current processor (i.e. *ON-LINE/OFF-LINE*).

Note: Each of the first two alarms has two possible values when they are on. The alarms can either be off or display one of the two **on** values. The remaining alarms have only one **on** value. These alarms are either **on** or **off**.

-m (major)

This option requests the *MAJOR* alarm status (i.e. *MAJOR* or --).

-n (minor)

This option requests the *MINOR* alarm status (i.e. *MINOR* or --).

-r (run tape)

This option requests the *RUN TAPE* alarm status (i.e. *RUN TAPE* or *--*).

-b (busyout)

This option requests the *BUSY OUT* alarm status (i.e. *BUSY OUT* or *--*).

-u (in use)

This option requests the *IN USE* alarm status of the currently active connection (i.e. *IN USE* or *--*).

-w (wait)

This option requests the *WAIT* alarm status of the currently active connection (i.e. *WAIT* or *--*).

Returns:

- The results of an **alarms** command are always returned to an executing program if one is active.
- If no options are specified, the data is returned in the following format (provided that all switch indicators are on and the current connection is with the on-line processor CC0):

```
^ CC0 ON-LINE MAJOR MINOR RUN TAPE BUSY OUT IN USE WAIT
<EOB>
```

If some of the indicators are off, the data might look like:

```
^ CC0 ON-LINE --- MINOR --- BUSY OUT --- ---
<EOB>
```

- If one of the **alarms** command options, except *-s* is used, the command returns only the status of the requested alarm. Otherwise, the status of all alarms is returned.

For example, if the *-n* option is specified, the following data might be returned:

```
^ MINOR
<EOB>
```

- If for any reason the command fails, an error message is returned.

```
! error message
<EOB>
```

Example 1: **alarms -n**

where *-n* requests the *MINOR* alarm status.

This example assumes that the switch *MINOR* alarm indicator is active. The *MINOR* alarm status of the currently active connection is displayed on the message line (in screen format).

Resulting output in screen format:

```
MINOR
```

Resulting output in scroll format:

```
$ alarms -n
^ MINOR
> 1 32006 3_0 C 3_2_1_1 - 0 000 Proc:000 Err:___ Fld:___
$
```

Example 2: **alarms**

Resulting output in screen format (provided all switch indicators are on):

```
CCI  ON-LINE  MAJOR  MINOR  RUN TAPE  BUSY OUT  IN USE  WAIT
```

Resulting output in scroll format:

```
^ CCI ON-LINE MAJOR MINOR RUN TAPE BUSY OUT IN USE WAIT
> 1 32006 3_0 C 3_2_1_1 - 0 000 Proc:000 Err:___ Fld:___
$
```

- The first entry is either *CC0* or *CCI*.
- The second entry is either *ON-LINE* or *OFF-LINE*.
- If the switch major alarm is on, the next entry is *MAJOR*. Otherwise, 2 dashes are added to the alarm string.
- If the switch minor alarm is on, the next entry is *MINOR*. Otherwise, 2 dashes are added to the alarm string.
- If the switch run tape is on, the next entry is *RUN TAPE*. Otherwise, 2 dashes are added to the alarm string.
- If the switch busy out alarm is on, the next entry is *BUSY OUT*. Otherwise, 2 dashes are added to the alarm string.

- If the switch in use alarm is on, the next entry is *IN USE*. Otherwise, 2 dashes are added to the alarm string.
- If the switch wait alarm is on, the next entry is *WAIT*. Otherwise, 2 dashes are added to the alarm string.

Notes: Only one of the above-mentioned options, except *-s*, is permitted at a time. The *-s* option may be used in conjunction with any of the other options or by itself.

See Also: **program**

ask

The **ask** command allows a program to display a prompt on the Proc Mode message line and wait for the user's response.

The equivalent VMAAP command is **:i**.

The command syntax is as follows:

Command: **ask** *prompt*

Arguments: *prompt*

This argument is composed of the string of characters that you wish to display in order to prompt a user for some response. Construct the prompt as follows:

- Begin the prompt message with the first non-white space character (consists of blanks and tabs) following the **ask** command.
- End the prompt with the last character before .
- Prompts are limited to 80 characters. Longer prompts are truncated.
- White space between the first character of the prompt and the last is displayed as entered, except that tabs are changed to blank spaces.
- The **ask** command returns the user's response as a string of characters.
- If **ask** is used with no prompt, an error message followed by an end-of-block marker **<EOB>** is returned.

Returns: The user's response is returned to a program. The response consists of whatever characters the user enters up to and including the first new line. An end-of-block marker **<EOB>** is sent to the program to indicate the end of the response.

The format of the data returned to a program is:

^ maximum 125 character response.
<EOB>

If the **ask** command is issued without a prompt or some other problem occurs, the format of the data returned to a program is:

! error message
<EOB>

Example 1: Before the prompt appears on the message line, the **ask** command, as issued by the program, appears briefly on the Proc Mode command line. Then the command line is blanked for the user response. Suppose the program contains this command:

ask Specify an output file name. Then press <Return>.

Resulting output in screen format:

```
Specify an output file name. Then press <Return>.
```

Resulting output in scroll format:

```
< ask Specify an output file name. Then press <Return>.  
~ Specify an output file name. Then press <Return>.  
$ maximum 125 character response  
> xxx xx x x...x xx xxx Proc:xxx Err:xx Fld:xx  
$
```

Notes:

- Any characters following the **ask** command up to the end of the line are considered a part of the prompt argument. Therefore, it is not possible to enter another command on the same command line after an **ask**.
- You may enter a response of up to 125 characters including a **Return**.
- If you do not respond to an **ask** command, the activity timer will time out and the program aborts.

See Also:

continue, note, program, talk (Refer to Chapter 5).

continue

Use **continue** to terminate a "talk" session by returning to your program or run file execution once you are done interactively entering commands. The **continue** command is entered from within a "talk" session. As a result, Proc Mode resumes accepting input from the run file or program which initiated the "talk" session.

The shorthand notation for **continue** is **cont**, **conti**, **contin**, or **continu**. With VMAAP, this functionality is partially performed by the command **quit**.

You issue **continue** as follows:

continue

To get a better understanding of how **continue** works, refer to the "talk" command description later in this chapter.

fields

The **fields** command is used to produce and display a procedure image (sequence of data fields all on one line representing the state of a proc) for the currently active procedure. Proc Mode generates procedure images for several different purposes. These images may or may not have a prefix character at the beginning of the image.

Normally, procedure images displayed on the message line in screen format have no prefix. When produced in response to a **fields** command, procedure images have a caret (^) prefix if displayed in scroll format, entered in log files, or returned to programs.

When produced automatically in scroll format and in log entries after every user command line, procedure images have a "greater than" sign (>) prefix.

The equivalent VMAAP command is **:d**.

You can issue **fields** as follows:

Command: **fields** [*-s*] [*field_number* | *-p* | *-e* | *-a* | *-w* | *-t*]

Options: *-s*

This option (*silent*) is used to suppress displaying the output related to this command. This is an optional argument and when specified, output is still passed to a program if active, and to a log file if open.

The **fields** command with no options other than *-s* produces an entire procedure image.

The **fields** command options, except the *-s*, allow you to request data from a specific procedure field.

field_number

This option is an integer specifying one particular field. The maximum allowed value is **25**, though the *true* maximum is the last data field of the current procedure.

-p

This option specifies the procedure number/name field.

-e

This option specifies the error number field.

-a

This option specifies the field containing the number of the current active data field.

-w

This option specifies the word number field.

-t

This option specifies the test number field.

Returns:

- The results of a **fields** command are always returned to an executing program if one is active.
- If no options are used, a procedure image and end-of-block *<EOB>* marker are returned to a program (provided a switch connection exists with an active procedure).

```
^ x x x xx xx xxx xxxx xx Proc:xxx Err:xx Fld:xx
<EOB>
```

- If one of the **fields** command options (except *-s*) is used, the command returns only the data from the requested field. Otherwise, the entire procedure image is returned.

For example, if the data is limited to the procedure number/name field by specifying the *-p* option, the following data is returned:

```
^ xxx
<EOB>
```

- If for any reason the command fails, an error message is returned.

```
! error message
<EOB>
```

Example:

```
fields 1
```

where *1* specifies field 1.

This command displays the data from field 1 on the message line (in screen format).

Resulting output in screen format:

32006

Resulting output in scroll format:

```
$ fields 1
^ 32006
> 1 32006 3_0 C 3_2_1_1 - 0 000 Proc:000 Err:___ Fld:___
```

- Notes:**
- Only one **field** option other than *-s* is permitted at a time. The *-s* option may be used in conjunction with one of the other options or by itself.
 - Use of one of the last six options limits the scope of the **fields** command to one field of the procedure image.
 - Depending on the Enhanced or Basic Mode selected, procedure image fields appear differently. In Enhanced Mode, the width of the fields is exactly as it is defined in the Switch Support Base EFC (Electronic Flip Chart) files.

In *Basic Mode*, Switch Support Base is not available; therefore, there is no way to determine the exact width of fields. It is assumed that each new field begins where the prior field ends. If there are a few blanks between two adjacent fields, these blanks are included in the second field thus increasing its width.

For example, a field which is only five digits wide in Enhanced Mode may consist of nine digits in Basic Mode (field 5 in Generic 2.1 Issue 3.0 procedure 100 word 4).

See Also: **alarms, program, run** (*Refer to Chapter 6*).

math

The **math** command provides access to an on-line calculator or math interpreter for evaluating the math expressions (a combination of operators and operands). Up to 18 expressions can be interpreted by a **math** command at one time.

With VMAAP, the math functionality is provided through three commands: **math**, **smath**, and **:q**.

You can issue the **math** command from the command line interactively, from a run file, or from a program. The resulting output will be displayed on the screen, saved in a log file if one is open, and returned to a program if one is active.

Math Operands

There are three types of math operands:

- A Number:** A number operand can be presented in one of three bases: octal (*o*), decimal (*i*), or hexadecimal (*x*). The default is hexadecimal. For example, to specify the decimal number 16, enter *i16*; the octal number 20 is presented as *o20*; and the hexadecimal 10 is entered as *10*. The range of this operand is X(0) to X(ffffffff).
- Register Value:** A value stored in one of 62 Proc Mode registers. Valid register names are 0–9, a–z, and A–Z. A register is specified by the operand *r* followed by a valid register name. For example, to access register *a*, enter *ra*; register 3 is addressed as *r3*. Proc Mode registers are the only operands that may be assigned values. That is, only registers may appear on the left-hand side of an assignment operator. The range of this operand is X(0) to X(ffffffff).
- Proc Field Value:** A switch procedure field value is specified by the operand *#* followed by the field number. For example, to address the value in field 5, enter *#5*. Normally, almost all switch procedure fields are decimal. Only a few contain data in a format other than decimal. For example, procedure 490 contains some octal values. Proc Mode assumes that all procedure fields are decimal. Therefore, if a non-decimal field value is included in a math expression, Proc Mode erroneously interprets the octal value as a decimal number.
- Note that in a DEFINITY Generic 2 switch, many switch procedures contain mapped information. When alpha characters or universal equipment locations appear in the procedure data fields, it is mapped information. The **math** command returns the unmapped value interpreted as an integer, rather than the mapped value. In the event a field holds part of a universal equipment location, the corresponding traditional equipment location value is returned.

Math Operators

The valid math operators are listed in the following table:

TABLE 7-3
Math Operators

Operator	Definition
+	addition
-	subtraction
*	multiplication
/	division
%	modulo
	Boolean OR
&	Boolean AND
^	Boolean XOR
=	assignment
(<i>expr</i>)	logical grouping
' <i>symbol</i> '	symbolic address
< <i>expr</i> >	contents of memory address

Notes:

In Table 7-3, the following conventions apply:

- (*Expr*) operator defines the order of evaluation of an expression. *Expr* can be any valid math expression.
- The '*symbol*' operator specifies a switch memory address using a public symbol (symbolic name for a switch memory address). The public symbol must be enclosed in single quotes.
- The <*expr*> operator is used to access the contents of a switch memory address. The *expr* in the <*expr*> operator must be a valid math expression which evaluates to switch memory address. The '*symbol*' operator may be part of the expression. The <*expr*> operator first evaluates *expr* as a memory address, and then determines the contents of that memory address.
- The operators '*symbol*' and <*expr*> are only available to the Services Organization. Only a Super User has access to these operators.

You can issue **math** as follows:

Command: **math** [-s] [expr1 [expr2 [...[expr18]]]] Return

Options: -s

This option (*silent*) is used to suppress displaying results to the screen. Note that output to a log file or to a program is not affected by the -s option.

Arguments: [expr1 [expr2 [...[expr18]]]]

These arguments (math expressions) are optional. An expression is a combination of operands and operators. No white space (space, tab, etc.) is permitted within an expression. Multiple expressions are separated by a white space. The **math** command can interpret up to 18 expressions at a time.

- Returns:**
- If no expression is given, then all current register values are returned.
 - The results of evaluating expression(s) are returned to the screen (unless the -s option is given), to an executing program if one is active, and entered in the log file if one is open.
 - Error messages which occur are also returned to a program.
 - For each expression in the command, one line of data is returned. For example, if one expression is specified, one line containing the results plus another line with the end-of-block marker are returned. The various values in the line are separated by blanks. The caret (^) prefix is used to indicate a data line.

```
^ expr oct-value dec-value hex-value (oct, dec, hex)
<EOB>
```

The following is the format of data returned when multiple expressions are specified:

```
^ expr1 oct-value dec-value hex-value (oct, dec, hex)
^ expr2 oct-value dec-value hex-value (oct, dec, hex)
...
^ exprn oct-value dec-value hex-value (oct, dec, hex)
<EOB>
```

- If Proc Mode detects an error in one of the expressions, it returns an error data line for that expression, an error message, and an end-of-block marker. Any remaining expressions are not processed.

```

^ expr1 oct-value dec-value hex-value (oct, dec, hex)
...
^ exprerr --- --- ---
! error message
<EOB>

```

Screen Presentation of the Math Results:

For each math expression in a **math** command, one line of output is generated containing the expression itself followed by the calculated result in three bases: octal, decimal, and hexadecimal. For example, if the command **math r1=i25*o5** is issued, the results appear on the message line (in screen format) as:

r1=i25*o5	175	125	7d	(oct, dec, hex)
-----------	-----	-----	----	-----------------

If more than one math expression is specified, the current screen is overlaid with a math screen. The results are displayed one line per expression in this new screen. The format of each line is identical to the format used for a single expression. Press **Return** to exit the math screen and return to the previous screen.

If no expression is specified, like in this example:

math

then the contents of all 62 math registers are displayed in a math screen. If all registers are 0, the message **The 62 math registers all contain zero.** is displayed on the message line and the current screen remains unchanged.

Any expression of up to 25 characters is displayed in this format. Larger expressions are truncated to leave room in the line for the results.

If an error occurs, the **math** command aborts. Any remaining expressions are not evaluated. The output line for the expression with the error contains dashes.

```
r3/*5      ---  ---  ---
```

An error condition occurs:

- If there is an error in a specified expression.
- If an expression contains the *<expr>* operator while there is no switch connection.
- If an expression contains the *'symbol'* operator while there is no access to public symbols.

Example 1: **math (i21+b)*o10**

This math command specifies a single expression. The result appears on the message line in screen format:

```
(i21+b)*o10    400    256    100    (oct, dec, hex)
```

Resulting output in scroll format:

```
$ math (i21+b)*o10
^ (i21+b)*o10    400    256    100    (oct, dec, hex)
> 1 32006 3_0 C 3_2_1_1 - 0 000 Proc:000 Err:___ Fld:___
$
```

Example 2: **math #1+1**

This example uses the # operand to obtain the value in procedure field 1. Resulting output in screen format (message line):

```
#1+1            76407    32007    7d07    (oct, dec, hex)
```

Resulting output in scroll format:

```
$ math #1+1
^ #1+1    76407    32007    7d07    (oct, dec, hex)
> 1 32006 3_0 C 3_2_1_1 - 0 000 Proc:000 Err:___ Fld:___
$
```

Example 3: **math 'dirnumb'+10**

The symbols operator, '*symbol*', is used to obtain the switch memory address of public symbol *dirnumb*. However, in this case, public symbols are not available.

Resulting output in screen format (message line):

```
xxx=Public symbol file not found: /public/symbol/path/to/er2v50300
```

Resulting output in scroll format:

```
$ math 'dirnumb'+10
! xxx=Public symbol file not found: /public/symbol/path/to/er2v50300
> 1 32006 3_0 C 3_2_1_1 - 0 000 Proc:000 Err:___ Fld:___
$
```

Example 4:

```
math r1=<'pubsymA'> r2=r1&1111 r3='pubsymB' r4=<r3+r2>
```

This example shows a **math** command with multiple expressions. Several of the expressions reference switch memory. This is only possible if a connection exists.

Resulting output in screen format (message line):

```
100=SWITCH DISCONNECTED.
```

Resulting output in scroll format:

```
$ math r1=<'pubsymA'> r2=r1&1111 r3='pubsymB' r4=<r3+r2>
! 100=SWITCH DISCONNECTED.
^ r1=<'pubsymA'>   --   ---   ---
$
```

Notes:

- An expression with the *symbol* operator can be evaluated as long as public symbols are available, even if there is no switch connection.
- Multiple expressions must be separated by white space.
- No white space (space, tab, etc.) may exist within an expression itself.
- Multiple expressions are evaluated in the order given.

See Also:

symbols (The TSC document *Services Commands*).

note

The **note** command is used to print a message on the message line. This message remains on the message line until one of the following occurs:

- Another message is issued by the **note** command.
- A **note** command with no message is issued.
- A switch error message (0–99) occurs.
- A Proc Mode error or status message (>99) occurs.

The **note** command is particularly convenient for displaying status messages during program or run file execution. A new message can be displayed for each section of the program. To clear a note message, enter the **note** command with no message.

The **note** command is similar to the VMAAP **echo** and **:c** commands.

You can use the **note** command as follows:

Command: **note** *[message]*

Arguments: The message is composed of the string of characters that you wish to appear on the message line. Construct the message as follows:

- Begin the message with the first non-white space character (consists of blanks and tabs) following the **note** command.
- End the message with the last character before .
- Messages are limited to 80 characters. Longer messages are truncated.
- White space between the first character of the message and the last is displayed as entered, except that each tab is changed to a blank.

Returns: Not applicable.

Example: **note** *There are many blanks before this sentence which do not appear on this screen*

This command displays the message **There are many blanks before this sentence which do not appear on this screen** on the message line (in screen format). Resulting output in screen format:

```
There are many blanks before this sentence which do not appear on this screen.
```

Resulting output in scroll format:

```
~There are many blanks before this sentence which do not appear on this screen.  
> 1 32006 3_0 C 3_2_1_1 - 0 000 Proc:000 Err:___ Fld:___  
$
```

Notes:

- The **note** command with no message clears any note message currently displayed. There is no message if only white space appears between the **note** command and .
- You cannot clear the Proc Mode or switch error messages by using the **note** command. Proc Mode automatically clears its own messages. Switch messages are cleared when the procedure itself removes the error code.

See Also: ask

status

The **status** command is used to produce and display the status information available from the Proc Mode status screen.

With VMAAP, this functionality is provided through two commands: **:N** and **:o**.

You can issue **status** as follows:

Command: **status** [-s] [-c | -d | -i | -p | -t | -b | -l]

Options: -s

This option (*silent*) is used to suppress displaying the output related to this command. This is an optional argument and when specified, output is still passed to a program, if active, and to a log file, if open.

By specifying an appropriate command option, except the -s, you may limit your request to the specified status information.

-c (connection)

This option requests information on the current connection.

-d (device)

This option requests information on the device/port involved in the current connection.

-i (id)

This option requests the customer identification information for the current connection.

-p (product id)

This option requests the product id associated with the current connection.

-t (type)

This option requests the type of switch.

-b (base)

This option requests the Switch Support Base.

-l (log file)

This option requests the log file name.

Returns: The results of a **status** command are always returned to an executing program if one is active.

- If one of the **status** command options (except -s) is used, the command returns only the limited status as specified by that option. Otherwise, all of the status information is returned.

For example, if the `-l` option is specified, the following data might be returned:

```
Log File: log file
        <EOB>
```

- If no options are specified, the format of the data returned to a program is:

```
~ Connected To: switch-name
~ Using Port: tty23
~ Customer Id: ...
~ Product Id: ...
~ Switch Type: ...
~ SSB in Use: ...
~ Log File ...
        <EOB>
```

- If for any reason the command fails, an error message is returned.

```
! error message
<EOB>
```

Example: `status -c`

where `-c` limits the data to switch connection only.

This command displays the status information of the currently active connection.

Resulting output in screen format:

```
Connected To: switch_name
```

Resulting output in scroll format:

```
$ status -c
~ Connected To: switch_name
> ... Proc:xxx Fld:xxx Err:xxx
<EOB>
```

The **status** command with no options produces the entire status screen.

The resulting screen in scroll format:

```
~Connected To: switch_name
~ Using Port: tty23
~ Customer Id:          003E 02-0303.05.00
~ Product Id: 0000000000
~ Switch Type: G2.1 Issue 03.00
~ SSB in Use: G2.1 Issue 02.00
~ Log File: log_file_name
<EOB>
```

Notes:

- Only one **status** option other than **-s** is permitted at a time. The **-s** option may be used in conjunction with any of the other options or by itself.
- The last seven options are used to limit the status request to specific information.

See Also:

alarms, fields, math

talk

The **talk** command is used to briefly halt program or run file execution, so you can interactively execute Proc Mode commands. Only a program or run file may initiate a "talk" condition. To terminate a "talk," enter the **continue** command.

The equivalent VMAAP command is **talk**.

Typically, before entering the **talk** command, a program or run file uses a **note** command to announce that a "talk" condition is about to begin. Also, any instructions may be displayed at that time via **note** commands.

You can run **talk** as follows:

Command: **talk**

Once you are finished interactively entering commands, use the **continue** command to return control to the program or run file execution.

Returns: Data is not returned to a program as the result of any command issued during a "talk" session, even if the session is initiated by a program.

Example: Suppose a program contains the following lines:

Issuer of Commands	Command
Program	note Adjust p000. Then enter the CONTINUE command. talk
Interactive User	32006 dx continue
Program	note Work on p000 is complete.

Resulting output in scroll format:

```

$ note Adjust p000. Then enter the CONTINUE command.
~ Adjust p000. Then enter the CONTINUE command.
> ----- Proc:000 Err:___ Fld:01
$ talk
> ----- Proc:000 Err:___ Fld:01
$ 32006 dx
> 1 32006 3_0 C 3_2_1_1 - 0 000 Proc:000 Err:___ Fld:___
$ continue
> 1 32006 3_0 C 3_2_1_1 - 0 000 Proc:000 Err:___ Fld:___
$
$ note Work on p000 is complete.
~ Work on p000 is complete.

```

Notes:

- A **talk** or **continue** command must appear as the last command on the command line (i.e. a **Return** must follow immediately after the command).
- If you enter the **continue** command when no "talk" session is active, the command is ignored, and an error message is displayed.
- A **talk** command may only be issued by a program or run file. Do not issue a **talk** command interactively.
- Once a **talk** command is issued by a program or run file, you are permitted to interactively enter commands until a **continue** command is issued.
- Once a "talk" session has begun, you are not allowed to interactively enter another **talk** command. Basically, multiple "talk" sessions are not permitted.
- Run files and program may not be executed from within a "talk" condition.
- If you do not enter any command during the "talk" session, the activity timer will eventually time out. Proc Mode will abort the controlling run file or program.
- To terminate a "talk" session and the run file or the program which initiated it, press **Delete**.

See Also:**ask, continue, note, program**

A Proc Mode Quick Reference

Proc Mode Quick Reference	A-1
Enhanced and Basic Mode Commands	A-2
Enhanced and Basic Mode Function/Control Keys	A-4

Proc Mode Quick Reference

Enhanced and Basic Mode Commands

TABLE A-1
Proc Mode Commands

Cmd	Description	VMAAP	Cmd	Description	VMAAP
#	Enter a Comment	#	nt	Next Text	NT
;	Advance One Field	.	nu	Next Unit	NU
@	One Second Pause		p	Select Procedure	p
"..."	Interpret a Character String	"..."	program	Execute a Program	&
add	Add Data	a	ptx	Park Tape Execute	PTx
ax	Add Data Execute	ax	quit	Quit	quit
bas	Enter Basic Mode		rb	Release Busy Out	RB
bo	Busy Out Facility	BO	rld	Reload Switch Memory	
cdx	Clear Data Execute	CDx	rmv	Remove Data	r
ce	Clear Active Field Entry	CE	r	Repeat or Display Previous Command Line(s)	;, exec, last
cf	Change Active Field	CF	rs	Reset Procedure	RS
chg	Change Data	c	rtx	Run Tape Execute	RTx
cont	Terminate a "Talk" Session	quit	run	Run a Script File	:<, :[
cp	Select Customer Procedure	CP	run proc	Log All Activities	:>
cx	Change Data Execute	cx	rx	Remove Data Execute	rx
dsp	Display Data	d	s	Stop Procedure	s
dx	Display Execute	dx	scroll	Enter Scroll Mode	
el	Convert Equipment Location	x	silent	Turn off Output	set

TABLE A-1 (Continued)
Proc Mode Commands

Cmd	Description	VMAAP	Cmd	Description	VMAAP
enh	Enter Enhanced Mode		note	Print a Message	echo,:c
get	Get Switch Support Base File(s)	x	np	Next Procedure	NP
help, h	General Help and Information	help, h	ssb	Select Switch Support Base	
hc, ?	Command Help	x	stat	Display Status	:N,:o
hf	Procedure Field Help	x	sw	Switch Control Processors	sw
hi	Procedure Input Help		t	Select Procedure Test	
hist	Display Activity Log		talk	Suspend a Program or Run File Execution	talk
ignore	Ignore Errors	:e	task	Enter Task Mode	
log -l	Display Current Log File Name		v	Verify Procedure	v
m	Select Mode Procedure	m	visual	Exit Scroll Mode	x
nc	Next Circuit	NC	w	Select Procedure Word	w
nd	Next Data	ND	x	Execute Procedure	x
nf	Next Fault	NF			

Enhanced and Basic Mode Function/Control Keys

Use control sequences if your terminal does not have function keys. To use three-character control sequences, press **f** while holding down the **Ctrl** key, release, then press the third character.

TABLE A-2
Proc Mode Function/Control Keys

Key	Screen Label	Control Key	Function
F1	1 Exit	Ctrl + f , 1	Exit the screen.
F2	2 Repeat	Ctrl + f , 2	Repeat the last command line.
F3	3 Form	Ctrl + f , 3	When cursor is on command line, moves cursor to screen form to accept entries.
F3	3 Line	Ctrl + f , 3	When cursor is on screen, moves cursor from active field to the command line.
F4	4 Clear	Ctrl + f , 4	When cursor is on screen, clears field.
F5	5 Help	Ctrl + f , 5	Display a menu to access information about using the system (how to access on-line help, command syntax, a list of command options, a list of procedures, and information about procedures.)
F6*	6 Field	Ctrl + f , 6	Display a range of valid entries (such as 1–120) or a list of available field values displayed in a window.
F7*	7 Input	Ctrl + f , 7	Display required procedure input data.
F8	8 Cmds	Ctrl + f , 8	Display a list of valid commands.
↓		Ctrl + d	Scroll down one line.
↑		Ctrl + u	Scroll up one line.
Page Up		Ctrl + n	Display next screen.
Page Down		Ctrl + p	Display previous screen.
Home		Ctrl + b	Display beginning screen (i.e. show first screen).
End		Ctrl + e	Display end screen (i.e. show last screen).

* Enhanced Mode only

TABLE A-3
Programming Commands Which Do Return Data

Command Syntax and Typical Data Format Returned to a Program by this Command	Command Description	V M A A P
alarms [-s] [-p -l -m -n -r -b -u -w] ^ CC0 ON-LINE MAJOR MINOR RUN TAPE BUSY OUT IN USE WAIT <EOB>	Determine current alarm status. Display, log, and return to program, as appropriate.	:C
ask prompt <Return> ^ maximum 125 character user response <EOB>	Prompt interactive user for response. Return response to program. Log, as appropriate.	:i
fields [-s] [[field_number -p -e -a -w -t] ^ x x x -- xx -- xx -- -- -- Proc:xxx Err:___ Fld:___ <EOB>	Determine current contents of procedure data fields. Display, log, and return to program, as appropriate.	:d
math [-s] [expr1 [expr2 [... [expr18]]]]<Return> ^ expression oct_value dec_value hex_value (oct, dec, hex) <EOB>	Evaluate math expression(s). Display, log, and return to program, as appropriate.	math smath :q
status [-s] [-c -d -i -p -t -b l] ~ Connected To: switch_name ~ Using Port: port ~ Customer Id: customer_identification_code ~ Product Id: product_identification_code ~ Switch Type: release_version_issue_dot_issue ~ SSB in Use: release_version_issue_dot_issue ~ Log File: log_file_name <EOB>	Determine current status information. Display, log, and return to program, as appropriate.	:N :o

TABLE A-4
Useful Programming Commands Which Do Not Return Data

Command Syntax	Command Description	V M A A P
continue <Return>	Exit interactive talk session. Continue with program or run file execution.	quit
ignore [ERROR [ERROR [ERROR [. . .]]]]<Return>	Ignore a switch error or range of errors. Insures that command line processing continue even when a switch error would normally abort the line.	:e
note [message] <Return> <i>This command does not return data to a program.</i>	Display message on message line.	echo :c
program [-e max_nbr_errors_to_permit] program_name [arg1 [arg2 [arg3 [...[argn]]]]]<Return>	Execute a program.	&
repeat [-d] [n] [-n] <Return>	Display or re-execute previously entered command line(s). The -d option causes repeat to only display command lines. Repeat with no arguments repeats the last command line. Repeat with one number n repeats the command line numbered n. Repeat with -n repeats the last n command lines. Repeat with a range n-m repeats the command lines numbered from n to m.	; exec last
run [-e max_nbr_errors_to_permit] filename <Return>	Execute a run (or log) file.	:< :[
silent	Suppress all screen output until a visual or scroll command is issued.	set
ssb <Return> ssb release_version issue.dot issue [custom_identifier] <Return> ssb -c custom_identifier <Return> ssb ? <Return>	Select a Switch Support Base (EFC files). Selection remains in effect until another ssb command is issued, a connection is made, or the current connection disconnects.	-
talk <Return>	Initiate an interactive talk session from within a program or run file. Session terminates when the user enters a continue command.	talk

B Programming Examples

Programming Examples	B-1
Sample Programs	B-1
Example 1: program <i>wait</i>	B-1
Example 2: program <i>getmodes</i>	B-3
Example 3: program <i>scrn_cntl</i>	B-5
Example 4: program <i>getfunctest</i>	B-9

Programming Examples

Sample Programs

This chapter contains examples of programs which might be executed from Proc Mode. These programs are written using UNIX Shell commands. The programs are based on Proc Mode programming commands.

Example 1: program *wait*

The program *wait* is a Shell script that waits for the alarm lamp *WAIT* to go off. You can run the program as follows:

```
program wait 5
```

The program accepts one argument, the number of seconds to wait between the switch inquiries (*alarms -s -w*). This argument is optional. If the parameter is provided, then it (*\$1*) is used as a waiting interval between the inquiries. Otherwise, the program sets this interval (*\$INTERVAL*) to the default value (5 seconds).

The inquiry command *alarms -s -w* is sent to the switch in an interval of time specified by the *\$1* variable or every 5 seconds (default). In response, the switch returns *WAIT Alarm Status* to the program. If the *WAIT* alarm is on (*\$(WAIT) = WAIT*), then the program waits another *INTERVAL* before sending the command *alarms -s -w* to the switch. Otherwise, the program terminates.

```

# wait UNIX Shell program.
#
# If there is no active switch connection, this program simply exits.
# Otherwise, the program waits for the switch WAIT indicator to turn off.
# The program polls for the status of the indicator.
# There is one optional argument which specifies the polling interval
# in seconds. The default interval is 5 seconds.
#
# This program is particularly useful when executed from within
# other programs or run files.
# Try executing it immediately after executing a maintenance procedure,
# a run tape (rtx), or park tape (ptx).
# It will wait for the operation to complete before allowing
# the remainder of the parent program or run file to continue.
# Check for a connection.
# There is no WAIT indicator if there is no connection.
echo "status -s -c"
read PREFIX CONNECTED TO SWITCH
read EOB
case ${PREFIX} in
  ~)
    # Status -c returns a switch name of "None" when no connection
    # exists. There is no reason to wait if there is no connection.
    if [ ${SWITCH} = "None" ]
    then
      exit 0
    fi
    # Determine the polling interval. It is specified as the
    # first (and only) argument to this program.
    # If no argument is given, use 5 seconds as the default.
    if [ $# -eq 0 ]
    then
      INTERVAL=5
    else
      INTERVAL=$1
    fi
    # Check the WAIT indicator status of the current connection.
    # If WAIT is on, sleep for a polling interval and then try again.
    while
      echo "alarms -s -w"
      read PREFIX WAIT
      read EOB
      test ${WAIT} = WAIT
    do
      echo "note WAITING"
      sleep ${INTERVAL}
      echo "note"
    done
    ;;
    \! | \^ | \> | \< | *)
      echo "note Problem with switch connection status. Wait program aborted."
      exit 1
      ;;
  esac
exit 0

```

SCREEN B-1
program wait

Example 2: program *getmodes*

The program *getmodes* is a Shell script that allows you to set a switch mode (1, 2, or 3) or any combination of these modes from the command line. You can run the program as follows:

program *getmodes* 123

You are allowed to specify one argument on the command line (1 to 3 digits long) representing the requested modes. The program verifies that the argument is valid, otherwise the program exits.

If the argument is valid, the program determines whether the requested switch modes are already set. If not, the program attempts to obtain the modes. Note that the switch may not grant access to the requested modes.

```
# getmodes UNIX Shell program.
#
# This program takes one argument, the requested switch modes.
# The modes are specified using any combination of the digits 1, 2, and 3.
# Only the first three digits of the argument are examined.
# Specifying digit 1 requests the switch Administration mode.
# Digit 2 corresponds to the Maintenance mode.
# And digit 3 represents the Tape mode.
# For example:  pgm getmodes 123
#
# If no argument is specified, this program exits.
# If there is no active switch connection, this program ABORTS.
# Otherwise, the program accesses the switch mode procedure, if necessary.
# If the requested switch mode(s) are already turned on, the program
# does nothing.  If not, the modes are requested from the switch.
# THE SWITCH MAY REFUSE TO GRANT ACCESS TO THE REQUESTED MODE(S).

function getmode
{
    # Check if the switch mode is already set.
    echo "fields -s $1"
    read PREFIX VALUE
    read EOB
    if [ $VALUE -eq 0 ]
    then
        # The mode is not set.  Request the mode.
        echo "$1"

        # Did the switch grant the mode request?
        echo "fields -s $1"
        read PREFIX VALUE
        read EOB
        if [ $VALUE -eq 0 ]
        then
            # The mode request was not granted.
            echo "note Mode $1 is not available."

            return 1
        else
            return 0
        fi
    fi
}
```

SCREEN B-2
program *getmodes*

```
        fi
    }
    # Exit if the modes argument is not supplied.
    if [ $# -eq 0 ]
    then
        exit 0
    fi

    # Verify that the modes argument is at most 3 digits.
    ARG_LEN='expr $1 : '[0-9]*''
    if [ $ARG_LEN -gt 3 ]
    then
        echo "note Getmodes argument too long (max 3 digits). Program aborted."
        exit $ARG_LEN
    fi

    # Verify that the modes argument consists only of the digits 1, 2, & 3.
    NBR_DIG='expr $1 : '[123]*''
    if [ $NBR_DIG -ne $ARG_LEN ]
    then
        echo "note Invalid getmodes argument (use digits 1, 2, & 3). Program aborted."
        exit $ARG_LEN
    fi

    # Suppress screen output.
    echo "silent"

    # Access the mode procedure if it is not already displayed. Note that
    # the program aborts at this point if there is no switch connection.
    echo "fields -s -p"
    read PREFIX PROC
    read EOB
    if [ "$PROC" != "mode" ]
    then
        echo "m"
    fi

    FAILED=0
    while test $ARG_LEN -ne 0
    do
        # Check the next digit of the argument.
        MODE='echo $1 | cut -c${ARG_LEN}'
        getmode $MODE
        FAILED='expr $FAILED + $?'
        ARG_LEN='expr $ARG_LEN - 1'
    done

    # Clear any switch mode procedure errors by accessing the procedure again.
    echo "m"

    # reestablish screen output.
    echo "scroll"

    # Exit indicating how many requests failed.
    if [ $FAILED -ne 0 ]
    then
        echo "note Failed to obtain $FAILED of the requested mode(s)."
    fi
    exit $FAILED
```

SCREEN B-3
program *getmodes* (continued)

Example 3: program *scrn_cntl*

The program *scrn_cntl* demonstrates two concepts:

- How to write a C program which interacts with Proc Mode.
- How a program may control screen output.

You can run the program as follows:

program *scrn_cntl*

The program *scrn_cntl* does not accept arguments. However, the program prompts a user for a switch memory address. Then, *scrn_cntl* issues the **silent** command to suppress Proc Mode screen output. The program reads 32 switch memory data words starting from the specified address. This data is displayed by writing directly to the terminal device.

Next, the *scrn_cntl* program issues the **scroll** command to reinitiate Proc Mode screen output. The user is prompted for another address. The cycle stops when the user responds with a blank line.

```

/* scrn_cntl C Program.
 *
 * The main purpose of the C program scrn_cntl is to demonstrate how a program
 * may take control of the terminal screen from Proc Mode. This allows a
 * program to display data of its own. The first step is to change Proc Mode
 * to silent screen format. This prevents Proc Mode from displaying anything
 * on the screen and thereby interfering with the data displayed by the program.
 * The second step is for the program to open a stream to the terminal device
 * currently in use. The program writes data to the stream. That data appears
 * on the terminal screen.
 *
 * Scrn_cntl changes Proc Mode from silent to scroll screen format repeatedly
 * to permit the Proc Mode ask command to appear on the screen. When silent
 * format is in effect, this program writes its own data to the terminal screen.
 *
 * In order to display data, scrn_cntl reads switch memory. Scrn_cntl
 * prompts the user for the memory address via the "ask" command. The address
 * should be specified just as it would be for a Proc Mode read memory (rm)
 * command. Scrn_cntl then (arbitrarily) displays NBR_WDS (32) number of memory
 * words which are read consecutively from the specified address.
 */
#include <stdio.h>
#include <string.h>
extern void exit();
extern char *getenv();
FILE *terminal_stream();

/* maximum response line length */
#define MAX_LINE 130

/* number of switch memory words in decimal */
#define NBR_WDS 32
#define NBR_WDS_STR "132"

```

SCREEN B-4
program *scrn_cntl*


```

#define TRUE (1)
#define FALSE (0)

main()
{
    FILE *termfile;          /* stream file connected to terminal */
    char response[MAX_LINE]; /* user and command responses */

    /* Define Standard Output as unbuffered. This is necessary to
     * prevent blocked communications with Proc Mode.
     */
    setbuf(stdout, (char *)NULL);

    /* Check for an active switch connection. Read memory
     * operations require a connection.
     */
    if (connection() == FALSE) {
        exit(1);
    }

    /* Open a terminal device stream for writing. */
    if ((termfile = terminal_stream()) == (FILE *)0) {
        exit(1);
    }

    /* Repeatedly ask the user for the next memory address.
     * Read and display NBR_WDS words starting from that address.
     * The user may terminate the cycle by responding with a
     * blank line instead of a memory address.
     */
    while (prompt_addr(response) == TRUE) {
        /* The user's address response is in the array response. Take control
         * of the screen from Proc Mode. Change to silent format to suppress
         * Proc Mode screen output. Read and display switch memory data.
         */
        puts("silent");
        get_memory_data(response, termfile);

        /* Restore scroll format before prompting for the next address. */
        puts("scroll");
    }

    exit(0);
}

int
connection()
{
    char response[MAX_LINE]; /* command responses */
    char eob[MAX_LINE];      /* End-of-Block Marker */

    /* Check for an active switch connection. Return TRUE if one exists.
     * Otherwise, return FALSE.
     */
    puts("status -s -c");
    gets(response);
    gets(eob);

    if (strcmp(response, "~ Connected To: None") == 0) {
        /* There is NO connection. Exit. */
        puts("note scrn_cntl: An active connection is required.");
        return(FALSE);
    }
}

```

SCREEN B-5
program scrn_cntl (continued)

```

    } else {
        /* There IS a connection. */
        return(TRUE);
    }
}

FILE *
terminal_stream()
{
    FILE *termfile;          /* stream file connected to terminal */
    /* Open a stream to the controlling terminal device. Return the FILE pointer
     * if successful. Otherwise, return a NULL pointer.
     *
     * Determine if the Environment Variable TTY has been correctly initialized.
     * The information in TTY is needed in order to take control of the terminal
     * screen for displays. If TTY is set, open the terminal device for writing.
     */
    if ((termfile = fopen("/dev/tty", "w")) == (FILE *)0) {
        /* Attempting to open the terminal device for output did not work. */
        printf("note scrn_cntl: Unable to open the terminal for output.\n");
    }

    return(termfile);
}

int
prompt_addr(response)
char *response;
{
    char user[MAX_LINE];    /* user response */
    char eob[MAX_LINE];    /* End-of-Block Marker */

    /* Prompt the user for a memory address. Read the
     * response and the end-of-block marker. Return the address.
     */
    printf("ask Enter a memory address or press <RETURN> to quit: \n");
    gets(user);
    gets(eob);

    if (user[2] != '\0')
    {
        /* The user specified an address. Ignore the prefix (^) on the
         * response and return the address.
         */
        strcpy(response, &user[2]);
        return(TRUE);
    } else {
        /* The user simply pressed return (i.e. no memory address). */
        response[0] = '\0';
        return(FALSE);
    }
}

int
get_memory_data(address, termfile)
char *address;            /* switch memory address */
FILE *termfile;          /* terminal device Stream pointer */

```

SCREEN B-6
program scrn_cntl (continued)

```
{
    char line[MAX_LINE];          /* read memory command or data line */
    /* Create the read memory command and send it to Proc Mode. */
    sprintf(line, "rm %s %s", address, NBR_WDS_STR);
    puts(line);
    /* Read the Proc Mode response to the read memory command a line at a time.
     * Keep reading until the End-of-Block marker is encountered. Display each
     * line of actual data on the terminal. Use the stream pointer pointing to
     * the device to control the write. Separate the data block from other
     * screen output by displaying new lines.
     */
    gets(line);
    fprintf(termfile, "\n\n\n\n");
    while (strcmp(line, "<EOB>") != 0) {
        /* Display the data directly on the terminal. */
        fprintf(termfile, "%s\n", line);
        gets(line);
    }
    fprintf(termfile, "\n");
}
```

SCREEN B-7
program *scrn_cntl* (continued)

Example 4: program *getfunctest*

The program *getfunctest* tests the three get functions, *getstreob()*, *getinteob()*, and *mathinteob()*. These functions are used to verify whether the commands issued from the *getfunctest()* program interacted successfully with Proc Mode.

Getfunctest() performs testing in the following sequence:

- 1 Issues a command to Proc Mode which does not request specific data through an option specified, such as *status -s*, *fields -s*, *math -s r1=1* etc.
- 2 Calls the appropriate get function by passing a *NULL* parameter to it. *NULL* means that no response is wanted by *getfunctest()*. The get function is used to test whether the command interacted with Proc Mode successfully.
- 3 Reads and prints on *stderr* the return code (*error*) from the get function. If the programming command executed successfully, *error* is equal to *OK (1)*, otherwise, *error* contains *FAIL (-1)*.
- 4 Issues a command to Proc Mode which requests for the specific data through an option specified, such as *status -s -i*, *fields -s 4*, etc.
- 5 Calls the appropriate get function. The request to return data to *getfunctest()* is specified through the passed parameter (*value*), for example, *getinteob(value)*. *Getfunctest()* reads and prints on *stderr* the response from the get function. This response contains a return code and data, status, or error message.

```

* getfunctest tests the three get functions, getstreob(), getinteob(),
* and mathinteob().
*
* First con to RMATS port 0, stay in Mode Proc, then pgm getfunctest.
*/
#include <stdio.h>
main()
{
    char string[BUFSIZ];
    int value;
    int error;

    setbuf(stdout, NULL);

    /* open new getfunc.log */
    (void) puts("log -d getfunc.log");

    (void) puts("# testing getstreob()");

    (void) puts("# get one response and discard it");
    /* without interfering with subsequent tests */
    (void) puts("math -s r1=1");
    error = getstreob(NULL);
    (void) printf("# e%d\n", error);
}

```

SCREEN B-8
program *getfunctest*

```
(void) puts("# get more than one response and discard all of them");
/* without interfering with subsequent tests */
(void) puts("status -s");
error = getstreob(NULL);
(void) printf("# e%d\n", error);

(void) puts("# get one response without title");
(void) puts("fields -s");
error = getstreob(string);
(void) printf("# e%d %s\n", error, string);

(void) puts("# get one response with title (title gets discarded)");
(void) puts("status -s -i");
error = getstreob(string);
(void) printf("# e%d %s\n", error, string);

(void) puts("# testing mathinteob()");

(void) puts("# get one response and discard it");
/* without interfering with subsequent tests */
(void) puts("math -s r1=1");
error = mathinteob(NULL);
(void) printf("# e%d\n", error);

(void) puts("# get more than one response and discard all but first one");
/* without interfering with subsequent tests */
(void) puts("math -s r1=1 r2=2 r3=3");
error = mathinteob(&value);
(void) printf("# e%d o%o i%d x%x\n", error, value, value, value);

(void) puts("# get agent id out of Mode Proc field 4");
(void) puts("math -s #4");
error = mathinteob(&value);
(void) printf("# e%d o%o i%d x%x\n", error, value, value, value);

(void) puts("# testing getinteob()");

(void) puts("# get one response and discard it");
/* without interfering with subsequent tests */
(void) puts("math -s r1=1");
error = getinteob(NULL);
(void) printf("# e%d\n", error);

(void) puts("# get more than one response and discard all of them");
/* without interfering with subsequent tests */
(void) puts("status -s");
error = getinteob(NULL);
(void) printf("# e%d\n", error);

(void) puts("# get agent id out of Mode Proc field 4");
(void) puts("fields -s 4");
error = getinteob(&value);
(void) printf("# e%d o%o i%d x%x\n", error, value, value, value);

(void) puts("# get dashes out of Mode Proc field 5");
(void) puts("fields -s 5");
error = getinteob(&value);
(void) printf("# e%d o%o i%d x%x\n", error, value, value, value);

(void) puts("# done testing get functions");
```

SCREEN B-9
program *getfunctest* (continued)

```

    /* close getfunc.log */
    (void) puts("log");
}
/*
 * getinteob() gets responses from Proc Mode (pm) commands
 * that return only one response and an end-of-block to
 * a Proc Mode program (pmp). A response consists of a
 * prefix followed by a space and a decimal number or
 * an error message. An end-of-block is the 5-character
 * string <EOB>.
 *
 * getinteob() gets the first response from pm, skips over
 * the prefix in the response, converts the rest of the
 * response to an integer and returns the integer to pmp.
 * If the response is not an integer, it returns -1 to pmp.
 * Then it gets and discards additional responses from pm
 * until it gets the end-of-block. Normally, the second
 * response from pm is the end-of-block. It checks the
 * prefix in every response from pm to determine whether
 * or not the response is an error message, and returns
 * an error message indication to pmp.
 *
 * getinteob() can be invoked in one of two ways:
 *     getinteob(&value);
 *     getinteob(NULL);
 * In the first case, the response is needed by pmp (e.g. :i)
 * and in the second, it isn't (e.g. smath).
 *
 * getinteob() facilitates replacing VMAAP child process :i, :q
 * and smath commands with Proc Mode program ask, field and
 * math commands:
 *
 * VMAAP Child Process:           Manager II Proc Mode Program:
 *
 * int value;                     int value;
 *
 * puts(":i ...");                 puts("ask ...");
 * scanf("%x\n", &value);         getinteob(&value);
 *
 * puts(":q mn");                 puts("field -s n");
 * scanf("%x\n", &value);         getinteob(&value);
 *
 * puts("smath ...");             puts("math -s ...");
 *                                 getinteob(NULL);
 *
 * Use mathinteob() when replacing :q with math -s.
 */

#include <stdio.h>
#include <string.h>

#define OK 1
#define FAIL -1

getinteob(value)
int *value;
{

```

SCREEN B-10
program getfunctest (continued)

```

char buffer[BUFSIZ];
int error = OK;

/* get response from pm */
(void) gets(buffer);

/* if response is wanted by pmp, return it to pmp */
if (value != NULL)
{
    /* check prefix in response */
    switch (*buffer)
    {
        case '^':
            /* response is data */

            /* if data contains decimal number, skip over prefix
             * and return number to pmp, otherwise return -1 to pmp */
            if (sscanf(buffer, "%d\n", value) == 1)
                break;

        case '~':
            /* response is status (treated like error message) */

        case '!':
            /* response is error message */

            /* return -1 to pmp */
            *value = -1;

            break;
    }
}

/* get responses from pm until response is end-of-block */
do
{
    /* if response is error message,
     * indicate error message occurred */
    if (*buffer == '!')
        error = FAIL;

    /* get response from pm */
    (void) gets(buffer);
}
while (strcmp(buffer, "<EOB>") != 0);

/* return error message indication to pmp */
return(error);
}

/*
 * getstreob() gets responses from Proc Mode (pm) commands
 * that return only one response and an end-of-block to
 * a Proc Mode program (pmp). A response consists of a
 * prefix followed by a space, an optional title followed
 * by a colon and a space, and data, status or error message.
 * An end-of-block is the 5-character string <EOB>.
 *
 * getstreob() gets the first response from pm, skips over
 * the prefix and title in the response, and returns the rest
 * of the response as a string to pmp. Then it gets and discards
 * additional responses from pm until it gets the end-of-block.
 */

```

SCREEN B-11
program *getfunctest* (continued)

```

* Normally, the second response from pm is the end-of-block.
* It checks the prefix in every response from pm to determine
* whether or not the response is an error message, and returns
* an error message indication to pmp.
* getstreob() can be invoked in one of two ways:
*     getstreob(response);
*     getstreob(NULL);
* In the first case, the response is needed by pmp (e.g. :i)
* and in the second, it isn't (e.g. smath).
*
* getstreob() facilitates replacing VMAAP child process
* commands with Proc Mode program commands:
*
* VMAAP Child Process:           Manager II Proc Mode Program:
*
* char response[BUFSIZ];         char response[BUFSIZ];
*
* puts(":i ...");                puts("ask ...");
* gets(response);               getstreob(response);
*
* puts(":C");                    puts("alarm -s");
* gets(response);               getstreob(response);
*
* puts(":d");                    puts("field -s");
* gets(response);               getstreob(response);
*
* puts(":N");                    puts("status -s -i");
* gets(response);               getstreob(response);
*
* puts(":o");                    puts("status -s -c");
* gets(response);               getstreob(response);
*
* puts("smath ...");             puts("math -s ...");
*                                getstreob(NULL);
*
* Use mathinteob() when replacing :q with math -s.
*
* The following pm commands will work when only one word
* is read or written. The responses for subsequent words
* are checked for error messages and then discarded.
*
* puts(":b ...");                puts("rm -s ...");
* gets(response);               getstreob(response);
* gets(eob);
*
*                                puts("wm ...");
*                                getstreob(response);
*
*                                puts("fm ...");
*                                getstreob(response);
*
* puts(":B ...");                puts("rio -s ...");
* gets(response);               getstreob(response);
* gets(eob);
*

```

SCREEN B-12
program *getfunctest* (continued)


```

*      puts(":w ...");          puts("nio -s ...");
*      gets(response);        getstreob(response);
*      gets(eom);
*
*      puts(":W ...");        puts("wio -s ...");
*      gets(response);        getstreob(response);
*      gets(eom);
*/
#include <stdio.h>
#include <string.h>
#define OK 1
#define FAIL -1
getstreob(response)
char *response;
{
    char buffer[BUFSIZ];
    char *pointer;
    int error = OK;

    /* get response from pm */
    (void) gets(buffer);

    /* if response is wanted by pmp, return it to pmp */
    if (response != NULL)
    {
        /* check prefix in response */
        switch (*buffer)
        {
            case '^':
                /* response is data (treated like status) */
            case '~':
                /* response is status */
                /* if response begins with title followed
                 * by colon and space, skip over it (only
                 * status command response begins with title;
                 * field command response contains colon
                 * but there's no space after colon) */
                pointer = strchr(buffer, ':');
                if (pointer == NULL || *(pointer + 1) != ' ')
                    pointer = buffer;

                /* return rest of response to pmp */
                (void) strcpy(response, pointer + 2);
                break;
            case '!':
                /* response is error message */
                /* return null response to pmp */
                *response = ' ';
                break;
        }
    }
}

```

SCREEN B-13
program *getfunctest* (continued)

```
/* get responses from pm until response is end-of-block */
do
{
    /* if response is error message,
     * indicate error message occurred */
    if (*buffer == '!')
        error = FAIL;

    /* get response from pm */
    (void) gets(buffer);
}
while (strcmp(buffer, "<EOB>") != 0);
/* return error message indication to pmp */
return(error);
}
```

SCREEN B-14
program *getfunctest* (continued)

Index

A

About Programming Commands, 1-1

Access

Basic Mode procedures, 4-3

Enhanced Mode procedures, 3-3

Accessing Proc Mode

from Manager III, 2-1

from Manager IV, 2-2

from Trouble Tracker, 2-3

Add

Call appearance, Example, 3-7, 3-8

Administration mode, 3-3, 4-3

Administrator

Switch, 1-3

System, 1-3

Alarms, 7-13

ask command, 7-17

at command

Run Files, Pre-Generic 2, 6-9

Audience

Document, 1-3

B

Basic Mode

Chooser, 4-5

Commands and Keys, 5-1

ELLS, 4-2

Extension Class of Service, 4-6, 4-7

from Enhanced Mode, 4-5

Function keys, A-4

Leave, 4-5

Procedures, 4-1

Procedures, Access, 4-3

Sample screen, 4-6

Screen, enter data, 4-4

C

Call appearance

Add, Example, 3-7, 3-8

Cancel

Run files, Pre-Generic 2, 6-10

Check

Run files, 6-10

Chooser

From Basic Mode, 4-5

Class of Service

Enhanced Mode, 3-5, 3-7

Extension, Basic Mode, 4-6, 4-7

Command Reference, 5-2

Command Syntax, 5-1

Commands

alarms, 7-13

ask, 7-17

Basic Mode, 5-1

continue, 7-19

Enhanced and Basic Mode, A-2, A-3

Enhanced Mode, 5-1

Enhanced Mode, Schedule, 6-8

Enhanced Mode, Script, 6-8

fields, 7-20

math, 7-23

note, 7-29

Proc Mode programming, 7-1, 7-13, 7-17, 7-19,
7-20, 7-23, 7-29, 7-31, 7-34

program, 7-6

Run Files, Scripts, 6-8

status, 7-31

talk, 7-34

Task, 3-4

Communications analyst, 1-3

continue command, 5-5, 7-19

Conventions and Considerations, 7-4

Convert VMAAP Child Process to

Proc Mode Program, 7-11

Creating Proc Activity Logs, 6-5

D

DEFINITY

Supporting documents, 1-5

DEFINITY Generic 2

Enhanced Mode procedures, 3-1

Dial User, 1-2

DIMENSION

Procedures, Basic Mode, 4-1

Document organization, 1-4

Documentation

Conventions, 1-7

Document(s)

Ordering, 1-5

Supporting, 1-5

E

Enhanced and Basic Mode

commands, A-2, A-3

Enhanced Mode

Class of Service, 3-5, 3-7

Extension Class of Service, 3-5, 3-7

from Basic Mode, 4-5

Function keys, A-4

Help key, 3-2

Leave, 3-4

Logs, 3-1

- Procedures, 3-1
- Procedures, access, 3-3
- Run files, 6-1
- Scheduling commands, 6-8
- Screens, Enter data, 3-4
- Enter data
 - Basic Mode screen, 4-4
- Enter date
 - Enhanced Mode screens, 3-4
- Environment Variables, 7-3
- Error
 - Status, and Data Messages, 7-4
- Example 1: program wait, B-1
- Example 2: program getmodes, B-3
- Example 3: program scrn_cntl, B-5
- Example 4: program getfuncstest, B-9
- Example
 - Add call appearance, 3-7, 3-8
- Executing Run Files, 6-3
 - from within a Manager III Script, 6-8
- Exiting the Program, 7-8
- Extension
 - Class of Service, Basic Mode, 4-6, 4-7
 - Class of Service, Enhanced Mode, 3-5, 3-7

F

- Fields, 7-20
- Files
 - Run, Cancel, Pre-Generic 2, 6-10
 - Run, Check, 6-10
 - Run, Resubmit, 6-11
 - Run, Results, 6-10
 - Run, Schedule, 6-9
 - Run, Schedule, Pre-Generic 2, 6-9
- Function Key Reference, 5-29
- Function keys
 - Basic Mode, A-4
 - Enhanced Mode, A-4

G

- Generic 2
 - basic mode, 4-1
 - Enhanced Mode procedures, 3-1
 - Scheduling run files, 6-9
- Getting Started With Proc Mode, 2-1

H

- Help key
 - Enhanced Mode, 3-2

I

- Ignore, 5-10
- Intended Audience, 1-3
- Interactive scripts, 6-8
- Interval
 - Time-out, 3-2

K

- Keys
 - Basic Mode, 5-1
 - Enhanced Mode, 5-1
 - Help, Enhanced Mode, 3-2
 - Function, Basic Mode, A-4
 - Function, Enhanced Mode, A-4
 - In this manual, 1-7

L

- Leave
 - Basic Mode, 4-5
 - Enhanced Mode, 3-4
- Limited Super User, 1-2
- Log command, 6-7
- Logfiles, 6-1
 - Script, Modeling, 6-1
- Login permissions, 3-1, 4-1
- Logs
 - Enhanced Mode, 3-1

M

- Maintenance mode, 3-3, 4-3
- Manager III
 - class of user, 1-2
 - Programming Feature, 1-1
- Manager IV
 - class of user, 1-2
 - Programming Commands, 7-1
- math command, 7-23
- Mode
 - Administration, 3-3, 4-3
 - Maintenance, 3-3, 4-3
 - Procedure, 3-3, 4-3
 - Tape, 3-3, 4-3
- Modeling
 - Scripts, Logfiles, 6-1

N

- note command, 7-29

O

- Ordering documents, 1-5

P

- Permissions
 - Login, 3-1, 4-1
- Pre-Generic 2
 - Schedule run files, 6-9
- Proc activity logs, 3-1
- Proc Mode, 1-1
 - Operation, 1-1
 - Programming Commands, 7-1, A-5, A-6
 - Programming Considerations, 7-4
 - Quick reference, A-1
 - Users, 1-1

Proc Mode Command Availability, 1-1
 Proc Mode commands, A-2, A-3
 alarms, 7-13
 ask, 7-17
 continue, 7-19
 fields, 7-20
 math, 7-23
 note, 7-29
 program, 7-6
 status, 7-31
 talk, 7-34
 Proc Mode commands which do not return data, A-6
 Proc Restriction, 1-2
 Procedure Mode Screens, 2-4
 Procedures
 Basic Mode, Access, 4-3
 Basic Mode, Sample screen, 4-6
 Enhanced Mode, Access, 3-3
 Mode, 3-3, 4-3
 Procs
 Basic Mode, 4-1
 Basic Mode, DIMENSION, 4-1
 Basic Mode, System 85, 4-1
 Enhanced Mode, 3-1
 Help, 3-2
 scheduling, 6-8
 program command, 7-6
 Programming Commands
 Manager IV, 7-1
 Proc Mode, 7-1, A-5, A-6
 Trouble Tracker, 7-1
 usage, 7-9
 Programming Commands which do not return data, A-6
 Programming Commands which do return data, A-5
 Programming Considerations
 Proc Mode, 7-4
 Programming Feature
 Manager III, 1-1

Q

Quick reference
 Proc Mode, A-1

R

Recommended training, 1-5
 Reference
 Quick, Proc Mode, A-1
 Reference documents, 1-5
 Regular User, 1-1
 Remove
 Script, Run Files, 6-11
 repeat command, 5-16
 Reset
 Script, 6-10
 Resubmit
 Run files, 6-11
 Results
 Run files, 6-10
 Reusing a script, 6-10

run command, 6-3
 Run Files, 6-1
 Cancel, Pre-Generic 2, 6-10
 Check, 6-10
 Enhanced Mode, 6-1
 Remove Script, 6-11
 Resubmit, 6-11
 Results, 6-10
 Schedule, 6-9
 Schedule, Pre-Generic 2, 6-9
 Script, 6-8
 UNIX commands, 6-8
 run proc command, 3-1, 6-6
 Run Script, 6-8
 Run Tape, 6-1
 Running Programs, 7-6

S

Schedule
 Enhanced Mode commands, 6-8
 Scheduling
 a run tape, 6-1
 Run files, Generic 2, 6-9
 Run files, Pre-Generic 2, 6-9
 Run Files Using Manager III, 6-9
 Screens
 Basic Mode, 4-6
 Basic Mode, Enter data, 4-4
 Enhanced Mode, Enter data, 3-4
 Script, 6-1
 Enhanced Mode, 6-1
 Enhanced Mode commands, 6-8
 Modeling, Logfiles, 6-1
 Reset, 6-10
 Reuse, 6-10
 Run, 6-8
 Run File, 6-8
 Security, 3-1, 4-1
 Security Code, 1-2
 Select codes
 Documentation, 1-5
 Services commands, 1-2
 silent command, 5-20
 Software
 Switch Support Base (SSB), 3-2
 Special Key Reference, 5-32
 SSB command, 3-2, 5-21
 status command, 7-31
 Super User, 1-2
 Switch
 Administrator, 1-3
 Technician, 1-3
 Switch selection menu, 4-5
 Switch Support Base (SSB), 5-21
 software, 3-2
 System 85
 Procedures, Basic Mode, 4-1
 System Administrator, 1-2, 1-3

T

- talk command, 5-26, 7-34
- Tape
 - Run, 6-1
- Tape mode, 3-3, 4-3
- task command, 3-4
- Technician Switch, 1-3
- Tests, 3-3, 4-3
- Time-out
 - Interval, 3-2
- Training
 - Recommended, 1-5
- Trouble Tracker
 - class of user, 1-2
 - Programming Commands, 7-1
- Typeface
 - In this manual, 1-7
- Typical Programming Commands, 7-9

U

- Universal ELL
 - And basic mode, 4-2
- usage
 - programming commands, 7-9

V

- VMAAP child process, 7-11
 - convert to Proc Mode program, 7-11
- VMAAP-to-Proc Mode
 - Cross-Reference List, 7-12

W

- Words, 3-3, 4-3

585-222-201
Issue 1

Graphics © AT&T 1988

