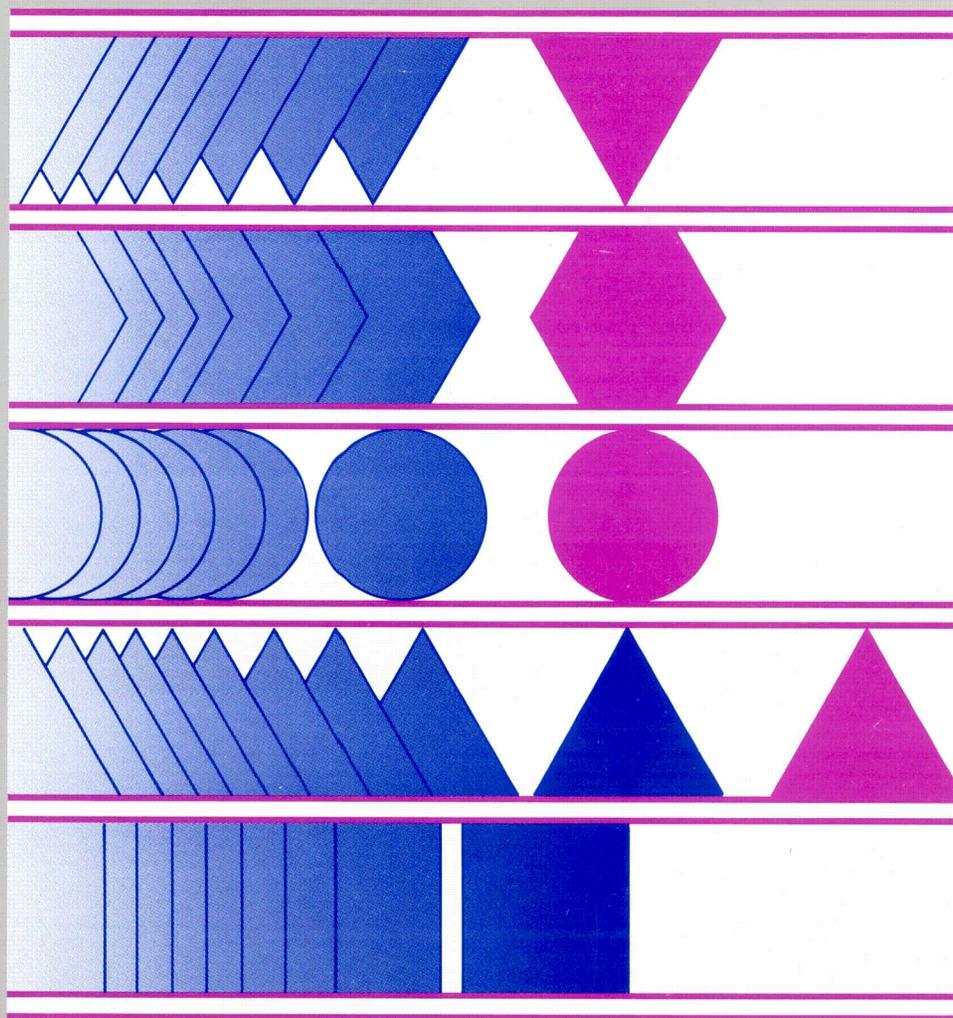


# PROCMON/3000 Reference Manual

hp HEWLETT  
PACKARD



**HP 3000 Computer Systems**

# **PROCMON/3000**

**Reference Manual**



10520 Ridgeview Court, Cupertino, Ca 95014

Part No. 99941-90001  
E1186

Printed in U.S.A. 11/86

#### **NOTICE**

The information contained in this document is subject to change without notice.

**HEWLETT-PACKARD MAKES NO WARRANTY OF ANY KIND WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. Hewlett-Packard shall not be liable for errors contained herein or for incidental or consequential damages in connection with the furnishing, performance or use of this material.**

Hewlett-Packard assumes no responsibility for the use or reliability of its software on equipment that is not furnished by Hewlett-Packard.

This document contains proprietary information which is protected by copyright. All rights are reserved. No part of this document may be photocopied, reproduced or translated to another language without the prior written consent of Hewlett-Packard Company.

# CONVENTIONS USED IN THIS MANUAL

NOTATION	DESCRIPTION
nonitalics	Words in syntax statements which are not in italics must be entered exactly as shown. Punctuation characters other than brackets, braces and ellipses must also be entered exactly as shown. For example:  EXIT;
<i>italics</i>	Words in syntax statements which are in italics denote a parameter which must be replaced by a user-supplied variable. For example:  CLOSE <i>filename</i>
[ ]	An element inside brackets in a syntax statement is optional. Several elements stacked inside brackets means the user may select any one or none of these elements. For example:  $\left[ \begin{array}{l} A \\ B \end{array} \right]$ User <i>may</i> select A or B or neither.
{ }	When several elements are stacked within braces in a syntax statement, the user must select one of those elements. For example:  $\left\{ \begin{array}{l} A \\ B \\ C \end{array} \right\}$ User <i>must</i> select A or B or C.
...	A horizontal ellipsis in a syntax statement indicates that a previous element may be repeated. For example:  [, <i>itemname</i> ]...;  In addition, vertical and horizontal ellipses may be used in examples to indicate that portions of the example have been omitted.
█	A shaded delimiter preceding a parameter in a syntax statement indicates that the delimiter <i>must</i> be supplied whenever (a) that parameter is included or (b) that parameter is omitted and any <i>other</i> parameter which follows is included. For example:  <i>itema</i> [█ <i>itemb</i> ][, <i>itemc</i> ]  means that the following are allowed:  <i>itema</i> <i>itema, itemb</i> <i>itema, itemb, itemc</i> <i>itema,, itemc</i>

# CONVENTIONS (continued)

$\Delta$  When necessary for clarity, the symbol  $\Delta$  may be used in a syntax statement to indicate a required blank or an exact number of blanks. For example:

```
SET[(modifier)] $\Delta$ (variable);
```

underlining When necessary for clarity in an example, user input may be underlined. For example:

```
NEW NAME? ALPHA
```

In addition, brackets, braces or ellipses appearing in syntax or format statements which must be entered as shown will be underlined. For example:

```
LET var[[subscript]] = value
```

shading

Shading represents the terminal's screen or key portions of an example.



The symbol  may be used to indicate a key on the terminal's keyboard. For example, RETURN indicates the carriage return key.

CONTROL *char*

Control characters are indicated by CONTROL followed by the character. For example, CONTROLY means the user presses the control key and the character Y simultaneously.

# PRINTING HISTORY

New editions are complete revisions of the manual. Update packages, which are issued between editions, contain additional and replacement pages to be merged into the manual by the customer. The dates on the title page change only when a new edition or a new update is published. No information is incorporated into a reprinting unless it appears as a prior update; the edition does not change when an update is incorporated.

The software code printed alongside the date indicates the version level of the software product at the time the manual or update was issued. Many product updates and fixes do not require manual changes and, conversely, manual corrections may be done without accompanying product changes. Therefore, do not expect a one to one correspondence between product updates and manual updates.

First Edition . . . . . AUG 85  
Second Edition . . . . . NOV 86



# LIST OF EFFECTIVE PAGES

The List of Effective Pages gives the date of the most recent version of each page in the manual. To verify that your manual contains the most current information, check the dates printed at the bottom of each page with those listed below. The date on the bottom of each page reflects the edition or subsequent update in which that page was printed.

Effective Pages	Date
all . . . . .	NOV 1986



# CONTENTS

<b>Section 1</b>	
<b>OVERVIEW</b> . . . . .	1-1

## **Section 2** **CREATING AND EXECUTING PROCEDURES**

Creating a Procedure . . . . .	2-1
Executing a Procedure . . . . .	2-2
Keying in the Procedure Name. . . . .	2-2
Selecting an Option from a Menu . . . . .	2-3
Calling one Procedure from Another . . . . .	2-3
Passing Parameters to Procedures . . . . .	2-4

## **Section 3** **COMMANDS**

MPE Commands . . . . .	3-1
PROCMON Commands. . . . .	3-1
ACTIVATE . . . . .	3-2
BLDDFU . . . . .	3-3
BLDKSAM . . . . .	3-5
BLDMENU . . . . .	3-8
BYE. . . . .	3-9
CANCEL . . . . .	3-10
CLEARACTIVE . . . . .	3-11
CONTINUE. . . . .	3-12
COPYFILE . . . . .	3-13
CREATE. . . . .	3-17
DATE . . . . .	3-18
DELETE. . . . .	3-19
DISPLAY . . . . .	3-21
ECHO . . . . .	3-22
ECHOOFF. . . . .	3-24
EDITOR . . . . .	3-25
ESCAPE . . . . .	3-26
EVALUATE . . . . .	3-27
FCOPY. . . . .	3-30
GOTO . . . . .	3-31
JOBQ . . . . .	3-32
KILL . . . . .	3-35
LISTFILE . . . . .	3-36
LOCAL . . . . .	3-39

# CONTENTS (continued)

MENU . . . . .	3-41
MENUOFF . . . . .	3-42
MSG . . . . .	3-43
OFF . . . . .	3-45
OPERATOR . . . . .	3-46
PATH . . . . .	3-47
PAUSE . . . . .	3-48
PFILE . . . . .	3-49
PREP . . . . .	3-50
PROCCONFIG . . . . .	3-51
PROMPT . . . . .	3-55
RENAME . . . . .	3-57
RESETPROC . . . . .	3-58
RETURN . . . . .	3-60
RFILE . . . . .	3-62
RPG . . . . .	3-63
RUN . . . . .	3-64
SECURITY . . . . .	3-65
SETPASS . . . . .	3-66
SHOWACTIVE . . . . .	3-68
SHOWMENU . . . . .	3-69
SWITCH . . . . .	3-70
TAG . . . . .	3-71
TASK . . . . .	3-72
WAIT . . . . .	3-75
WSID . . . . .	3-77

## Section 4 SUBSTITUTION WITHIN PROCEDURES

Substitution Expressions . . . . .	4-1
Substitution Expression Formats . . . . .	4-1
Nested Substitution Expressions . . . . .	4-5

## Section 5 CONDITION WITHIN PROCEDURES

Conditional Expressions . . . . .	5-1
IF Expressions . . . . .	5-1
ELSE Expressions . . . . .	5-4
BEGINIF, ENDIF, BEGINELSE, and ENDELSE Statements . . . . .	5-4
RESETPROC Statement . . . . .	5-5
CANCEL Statement . . . . .	5-5
RETURN Statement . . . . .	5-5
GOTO and TAG Statements . . . . .	5-6

# CONTENTS (continued)

## Section 6

### CREATING, UPDATING, AND EXECUTING MENUS

Creating Fixed Format Menus . . . . .	6-1
Creating Free Format Menus . . . . .	6-3
Updating Menus . . . . .	6-4

## Section 7

### ADDITIONAL FEATURES

FILE IPL= . . . . .	7-1
FILE MENU= . . . . .	7-1

## Appendix A

### CREATE, ACTIVATE, AND KILL



# PREFACE

This publication is the reference manual for the HP3000 Process Monitor software product, PROCMON/3000. It provides information on creating and using PROCMON procedures, commands and substitution expressions.

Section 1 gives a brief overview.

Section 2 describes the methods of making and using procedures and passing parameters.

Section 3 contains the actual manual pages of syntax diagrams, parameter descriptions, discussions of operation and examples. This section comprises the bulk of the manual.

Section 4 describes how substitutions can be used within procedures for use with passed parameters and other variables.

Section 5 explains the use of the IF/ELSE constructs with conditional parameters provided by PROCMON.

Section 6 deals with screen menus and gives descriptions and examples of their creation, modification and usage.

Section 7 explains the IPL and MENU file equation features of PROCMON.

Other publications which should be available for reference when using this manual are:

MPE Commands Reference Manual

MPE File System Reference Manual

FCOPY/3000 Reference Manual

KSAM/3000 Reference Manual



The process monitor program, PROCMON, provides a friendly, menu-driven, custom interface for transitions between system functions, user-written applications and MPE commands. PROCMON is designed for all HP3000 users.

PROCMON is easy to use and allows users to tailor menus to match their own needs. Users may have different procedure names for the same functions, organize them in different ways, and set up menus corresponding to tasks to be performed.

Inexperienced users can invoke system functions without being aware of the MPE programs and the underlying file structure. Users select from a menu by associated task number, procedure name or command. They can choose by recognition rather than recall.

## **PROCMON:**

- Provides an easy-to-understand interface with MPE.
- Reduces keystrokes.
- Allows users to create custom menus.
- Simplifies the system by handling parameter prompting, parameter passing, statement substitution, conditional statement control and procedure nesting.



# CREATING AND EXECUTING PROCEDURES

SECTION

2

A procedure is a set of MPE commands or PROCMON commands or both combined in a single, unnumbered editor file. The editor file is assigned a name and that name is used to reference the procedure.

Several application steps may be combined within one procedure to reduce keystrokes and provide a user-friendly application interface. A procedure file, for example, can replace the dozen or so job steps needed to create a job cost report.

## CREATING A PROCEDURE

A procedure file is created using EDIT/3000 or TDP/3000. Each command is entered on a separate line, with the exception of conditional statements (which may be combined on the same line for multiple conditioning).

### NOTE

Lines are limited to 72 characters. If a line is greater than 72 characters, a "&" continuation character may be used. For example:

```
FILE INPUT=INPUT.DATA
FILE OUTPUT=OUTPUT.DATA;DISC=2000;&
SAVE
RUN PROGA.PUB
LISTF @.DATA,1
```

PROCA-procedure file

When the procedure - PROCA is executed, the file statements for INPUT and OUTPUT are set, the PROGA program is executed and a listing of the files located in the DATA group is displayed at the user's terminal.

**NOTE**

File equations set by a procedure are cancelled when the procedure terminates with the exception of file equations set by the PFILE command. (These remain active until reset by an RFILE command or until the session ends.)

## EXECUTING A PROCEDURE

A procedure can be executed from PROCMON in three ways:

- keying in the procedure name
- selecting an option from a menu
- calling one procedure from another

### Keying in the Procedure Name

When PROCMON's inverse video cursor appears, enter the procedure command followed by the return key. The procedure command format is:

*procedurename* [.group [.account ]]

*procedurename*

Name of the editor file containing PROCMON and MPE commands. If the group and account are not specified, PROCMON will look for the procedure in the "P" group of the user's logon account.

## Selecting an Option from a Menu

When PROCMON's inverse video cursor appears, enter the menu task number followed by the return key. For example:

```

                                TIME CARD MENU

1) ENTER TIME - WK1   9) ENTER TIME - WK2
2) ENTER TIME - WK3  10) ENTER TIME - WK4

3) GOTO PRINT MENU
    
```

TCMENU - menu file

To enter week2 time cards, the operator enters the number 9 followed by the return key.

## Calling one Procedure from Another

One procedure can call another procedure. The procedure called is referenced as the nested procedure. After execution, the nested procedure returns to the calling procedure. For example:

```

FILE INPUT=INPUT.DATA
FILE OUTPUT=OUTPUT.DATA;DISC=2000;&
SAVE
RUN PROGA.PUB
LISTF @.DATA,1
PROCB
RUN PROGA1.PUB
    
```

PROCA-calling procedure file

```

FILE PAYIN=PAY01.DATA
FILE PAYOUT=PAUOUT.DATA
RUN PROGB.PUB
    
```

PROCB-nested procedure file

A maximum of 255 levels of procedure nesting is allowed.

## PASSING PARAMETERS TO PROCEDURES

Variable information can be passed to procedures by means of positional parameters. These parameters are placed after the procedure name in the procedure command. Commas are used to separate parameters and to indicate their position relative to the other parameters in the procedure call. If a parameter is omitted, a comma must still be used to hold the position. A maximum of 64 parameters (of 128 characters each) can be passed to a procedure. The total number of characters passed to a procedure may not exceed 1024. The "@" character may be used to pass all parameters from the current procedure to the called procedure.

Parameters are referenced within the procedure by means of substitution expressions. For more information on these expressions see Section 4.

Parameters may be undelimited or may be enclosed in single or double quotes. Quotes are taken as delimiters if the parameter string has both a beginning and ending quote of the same type, i.e. both single quotes or both double quotes. In all other cases they are taken as part of the string. If the character being used as a delimiter is also to be embedded in the string it must be specified twice. For example, if the literal "one o'clock" is being specified as a parameter enclosed in single quotes, it must be coded as follows: 'one o'clock'

```
IF DATA-?1?.DATA DELETE ?1?.DATA
FILE OUTPUT=?1?.DATA;DISC=1500;SAVE
RUN PROGC.PUB
```

PROCC NEWFILE -Passing parameter 1 value to procedure PROCC

**MPE COMMANDS**

The MPE RUN command and all MPE commands that can be executed via the COMMAND intrinsic are accessible from within PROCMON. They may be called from a procedure file or entered from the keyboard after the inverse video cursor appears.

**PROCMON COMMANDS**

In addition to the available MPE commands, PROCMON provides its own powerful command set. These commands can be called from a procedure file or entered from the keyboard after the inverse video cursor appears.

The PROCMON commands are:

ACTIVATE	FCOPY	PROMPT
BLDDFU	GOTO	RENAME
BLDKSAM	JOBQ	RESETPROC
BLDMENU	KILL	RETURN
BYE	LISTFILE	RFILE
CANCEL	LOCAL	RPG
CLEARACTIVE	MENU	RUN
CONTINUE	MENUOFF	SECURITY
COPYFILE	MSG	SETPASS
CREATE	OFF	SHOWACTIVE
DATE	OPERATOR	SHOWMENU
DELETE	PATH	SWITCH
DISPLAY	PAUSE	TAG
ECHO	PFILE	TASK
ECHOOFF	PREP	WAIT
EDITOR	PROCCONFIG	WSID
ESCAPE		
EVALUATE		

# ACTIVATE

Activates a process.

## Syntax

```
ACTIVATE progrname
```

## Parameters

*progrname* Name of program to be activated, in the following format:

*progrname[.group[.account]]*

## Operation

Activates a process previously created or suspended. Once activated, the process runs until it is suspended (using the RPG SUSP operation) or is deleted (using the PROCMON KILL command) or goes to end of job. See Appendix A for more information on CREATE, ACTIVATE, and KILL.

## Example

To activate the currently suspended program - PAY001 enter:

```
ACTIVATE PAY001
```

Creates and immediately allocates an empty direct file on disc.

## Syntax

```

BLDDFU filename

[;REC=[recsize],[blockfac],[F][,BINARY]]
[V][,ASCII]]
[U]]

[;CCTL] [;TEMP] [;DEV=[dsdevice]#][device]
[;NOCCTL]

[;CODE=[filecode]]

[;DISC=[numrec][,numext][,initalloc]]

[;RIO
;NORIO
;MSG
;CIR]

```

## Parameters

- filename* Name of file to be created, in the following format:  
*filename.group.account*]]
- recsize* Size of record. A positive number indicates words, while negative indicates bytes. Default is 128 words.
- blockfac* Number of logical records per physical block. Default is calculated.
- F, V or U File contains F-fixed, V-variable or U-undefined length records. Default is fixed.
- BINARY File contains binary-coded records. Default.
- ASCII File contains ASCII-coded records.
- CCTL Carriage control characters will be supplied with write requests.

## BLDDFU

<code>NOCCTL</code>	Carriage control characters will not be supplied with write requests. Default.
<code>TEMP</code>	File will be created in the job/session temporary file domain. Default is permanent.
<i>device</i>	Specifies the device on which the file will reside.
<i>filecode</i>	Code indicating a specially-formatted file.
<i>numrec</i>	Maximum number of logical records. Default is 1023.
<i>numext</i>	Maximum number of disc extents. Default is 8.
<i>initialloc</i>	Number of extents to be initially allocated to the file at the time it is opened. This is a value from 1 to 32. Default is 1.
<code>RIO</code>	Relative I/O file is created.
<code>NORIO</code>	Non-relative file is created.
<code>MSG</code>	Message file is created allowing communication between any set of processes.
<code>CIR</code>	Acts as normal sequential file until full, then the first physical block will be deleted when the next record is written, and remaining blocks will be logically shifted to the front of the file.

## Operation

Creates a direct file on disc and immediately allocates space for it. If ASCII is specified, the records are initialized to blanks. If BINARY is specified, the records are initialized to zeros.

## Example

To create the direct ASCII file - PAYDATA, with a record length of 130 and a blocking factor of 3, enter:

```
BLDDFU PAYDATA;REC=-130,3,F,ASCII
```



## BLDKSAM

<i>recsize</i>	Record size. A positive number indicates words, while negative indicates bytes. Default is 128 words.
<i>bloclac</i>	Number of logical records per physical block. Default is calculated.
A or P	Key type of the primary key. Valid values are A-alphanumeric or P-packed.
Y or N	Specifies whether duplicate primary keys are permitted. Valid values are Y-duplicate keys are permitted, N-duplicate keys are not permitted. Default is N.
KEY	The KEY specification describes an alternate key. There may be up to 15 alternate key descriptions. Refer to the BUILD command, in the KSAMUTIL section of the KSAM/3000 Reference Manual, for a detailed description of parameter values.
<i>akeytype</i>	Key type of the alternate key.
<i>akeylocation</i>	Location of the first character of the alternate key within the data record counting from the first character in the record.
<i>akeysize</i>	Length of the alternate key in characters.
<i>akeyblocking</i>	Number of alternate keys per block.
DUPLICATE or DUP	Duplicate alternate keys are permitted and are to be inserted chronologically in the duplicate key chain.
RDuplicate or RDUP	Duplicate alternate keys are permitted and are to be inserted randomly in the duplicate key chain.

**Operation**

Creates a KSAM file on disc and immediately allocates space for it.

**Example**

To create the permanent KSAM data file - GL01 in the DATA group, with a key file - GL01K, a key beginning in position 15, a key length of 5 and a record length of 250 enter:

```
BLDKSAM GL01.DATA,GL01K,15,5;REC=-250
```

# BLDMENU

Activates PROCMON's BLDMENU facility.

## Syntax

```
BLDMENU menuname [,FREEFORM]
```

## Parameters

*menuname* Name of menu to be created, in the following format:

*menuname*[.*group*].*account*]]

If group is not specified, PROCMON will build the menu in the "M" group of the user's log-on account.

**FREEFORM** Free format menu is created. Default is fixed.

## Operation

Activates PROCMON's BLDMENU facility. The facility creates and updates both fixed and free format menus. For details, please refer to section 6.

## Example

To create the free format menu - PAYMNU, enter:

```
BLDMENU PAYMNU,FREEFORM
```

To create the fixed format menu - PAYROL, enter:

```
BLDMENU PAYROL
```

Terminates the process monitor.

## Syntax

BYE

## Parameters

none

## Operation

Terminates the process monitor.

## Example

To terminate PROCMON and return the user to MPE's control enter:

BYE

### NOTE

Entering a 0 and **RETURN** at PROCMON's prompt has the same effect as the BYE command.

# CANCEL

Terminates a procedure.

## Syntax

```
CANCEL
```

## Parameters

none

## Operation

Terminates a procedure and returns to the PROCMON prompt. Typically used on a conditional statement. If the condition is met, then the procedure is terminated.

## Example

To terminate the procedure - PROCA, if the file - INPUT does not exist, enter:

```
IF NOT DATA-INPUT CANCEL
```

portion of a procedure file

```
This command can only be called from a procedure.
```

# CLEARACTIVE

Removes an "active procedure" entry from the PROCMON database.

## Syntax

```
CLEARACTIVE {procname}  
             {@}
```

## Parameters

*procname*        Name of "active procedure" entry to be removed from the PROCMON data base.

*@*                All "active procedures" entries are to be removed from the PROCMON data base. This parameter requires system manager capability.

## Operation

Removes an "active procedure" entry from the PROCMON data base. Invalid "active procedure" entries are often left in the data base when procedures abort abnormally. The CLEARACTIVE command is a housekeeping command.

## Example

To remove all "active procedure" entries from the data base, enter:

```
CLEARACTIVE @
```

To remove the "active procedure" entry - PROCA, from the data base, enter:

```
CLEARACTIVE PROCA
```

# CONTINUE

Overrides a procedure error.

## Syntax

```
CONTINUE
```

## Parameters

none

## Operation

Permits a procedure to continue even though the next command results in an error. Typically used when anticipating command errors and applies only to the command immediately following the CONTINUE command.

## Example

To continue with the next command after a DELETE command fails, enter:

```
CONTINUE  
DELETE OUTPUT.DATA
```

portion of a procedure file

```
This command can only be called from a procedure.
```

Copy, reorganize and create disc files.

## Syntax

```

COPYFILE fromfile,tofile[/keyfile][;TEMP] [;ADD
                                           ;APPEND]

[ {;INCLUDE} {= } [column,"characters"]
  {;EXCLUDE} {<>}
              {<}
              {<=}
              {>}
              {>=} ]

[;REC=recsize] [;DISC=numrec] [;ORG={SAME}
                                     {S
                                     I
                                     D}]

[;KEY=keylocation,keysize] [;NOREORG]
                             [;REORG]

[;DUPL={Y}] [;MAX=maximum records]
  {N}

```

## Parameters

*fromfile* Name of the file to be copied in the following format:

*fromfile* [. *group* [. *account* ]]

*tofile* Name of the file to receive the data in the following format:

*tofile* [. *group* [. *account* ]]

The file will be created unless the ;ADD or ;APPEND options are used.

*keyfile* Name of the KSAM keyfile to be created. This parameter is optional. If it is not specified when a KSAM file is being created, PROCMON will name the key file. PROCMON's naming convention is as follows: If the data file name is less than 8 characters in length, a "K" is appended to it to produce the key file name. If the data file name is 8 characters long, the last character of the name is overlaid with a "K" to produce the key file name.

# COPYFILE

- TEMP** The *tofile* will be created in the job/session temporary file domain. Default is the permanent domain.
- ADD or APPEND** Appends records to an existing file.
- NOREORG** If the file being copied is a KSAM file, the records will be placed in chronological sequence in the new file. Default.
- REORG** If the file being copied is a KSAM file, the records will be placed in key sequence in the new file.
- INCLUDE or EXCLUDE** Specifies whether selected records in the *fromfile* are to be included in or excluded from the *tofile*. Selected records are those that satisfy the column and characters comparison test.
- =** If the characters in the specified columns of the *fromfile* record are the same as the comparison characters, the record will be included in or excluded from the *tofile*.
- <>** If the characters in the specified columns of the *fromfile* record are not the same as the comparison characters the record will be included in or excluded from the *tofile*.
- <** If the characters in the specified columns of the *fromfile* record are less than the comparison characters the record will be included in or excluded from the *tofile*.
- <=** If the characters in the specified columns of the *fromfile* record are less than or equal to the comparison characters the record will be included in or excluded from the *tofile*.
- >** If the characters in the specified columns of the *fromfile* record are greater than the comparison characters the record will be included in or excluded from the *tofile*.
- >=** If the characters in the specified columns of the *fromfile* record are greater than or equal to the comparison characters the record will be included in or excluded from the *tofile*.
- column** Specifies the location within the record at which character comparison is to begin. If this parameter is not specified, then each location in the record is

compared to the comparison characters until the condition is met.

- characters* Specifies the comparison characters. Up to 30 characters may be compared. The character string may be undelimited or may be enclosed in single or double quotes.
- recsize* Record size of the *tofile* record. A positive number indicates words, a negative number indicates bytes. Default is the size of the *fromfile* record.
- numrec* Maximum number of logical records to be allocated to the new file. Default is the number of records in the *fromfile*.
- SAME** Specifies that the *tofile* is to have the same organization as the *fromfile*. If this parameter is blank SAME is assumed.
- S** Specifies that the *tofile* is to be a sequential file.
- I** Specifies that the *tofile* is to be a KSAM file.
- D** Specifies that the *tofile* is to be a direct file.
- keylocation* Location of the first character of the key within the data record for the KSAM file being created. If a value is not specified, the key location of the *fromfile* is assumed. If the *fromfile* is not an indexed file, but the *tofile* is, this parameter must be specified.
- keysize* Length of the key in characters. If a value is not specified the key length of the *fromfile* is assumed. If the *fromfile* is not an indexed file, but the *tofile* is, this parameter must be specified.
- Y or N** Specifies whether to allow duplicate keys in the *tofile* when the *tofile* is a KSAM file. Y indicates that duplicate keys are allowed. N indicates that duplicate keys are not allowed. Default is N.
- maximum records* Specifies the total number of records to be copied to the *tofile*.

# COPYFILE

## Operation

Allows the user to:

- Copy an entire disc file or selected portion of a disc file to a new or existing disc file.
- Create the new file with different file organization.
- Include or exclude selected records.
- Reorganize KSAM files.
- Create the new file with different record size and/or number of logical records.

If the optional parameters are allowed to default, the new file will be built with the same characteristics as the *fromfile* (i.e. KSAM or sequential, record length, number of records,...).

## Example

To make a duplicate copy of the KSAM file - OLDDATA and name the duplicate file - NEWDATA, enter:

```
COPYFILE OLDDATA,NEWDATA
```

(The KSAM key file will be named "NEWDATAK")

To copy the KSAM file - OLDDATA, to the existing file - PAY01 and ignore all records containing a "D" in column 1, enter:

```
COPYFILE OLDDATA,PAY01;ADD;EXCLUDE=1,"D"
```

To copy the MPE file - INPUT to a new KSAM file=OUTPUT including those records containing the characters "ABC" in columns 72-74.

```
COPYFILE INPUT,OUTPUT;INCLUDE=72,"ABC";ORG=I;KEY=2,9
```

Creates a process.

## Syntax

```
CREATE progrname
```

## Parameters

*progrname* Name of program to be created in the following format:

*progrname[.group[.account]]*

<*empty*> Displays all existing son processes for the session.

## Operation

Loads the program to be run by the new process into virtual memory, creates the process, initializes its data stack and returns the Process Identification Number (PIN). The program is then ready to be "ACTIVATED." See Appendix A for more information on CREATE, ACTIVATE and KILL.

## Example

To create the program - PAY001 enter:

```
CREATE PAY001
```

# DATE

Changes session date (RPGUPDATE file).

## Syntax

```
DATE [ <empty> ]  
      [ mm/dd/yy ]  
      [ mmddyy ]  
      [ mm-dd-yy ]
```

## Parameters

*mm*                Session month.

*dd*                Session day.

*yy*                Session year.

*<empty>*           Displays current session date.

## Operation

Loads the specified date into the RPGUPDATE file. If the DATE command is not issued, the RPGUPDATE file contains the system date. The date will be accessed by the RPG run time library, when an "F" is placed in column 17 of the program's header specification.

## Example

To load the RPGUPDATE file with the date - JULY 28, 1956, enter:

```
DATE 072856    or  
DATE 07/28/56   or  
DATE 07-28-56
```

Purges a file or fileset from disc.

## Syntax

```
DELETE {filename}[,TEMP]
       {fileset }
```

## Parameters

*filename* Name of the file to be purged, in the following format:

*filename*[.*group*].*account*]]

The TEMP parameter may be specified to purge a file from the temporary domain.

*fileset* *fileset*[.*group*].*account*]]

The characters @, #, and ? may be used as wildcard characters in the fileset parameter.

@ Specifies zero or more alphanumeric characters

# Specifies one numeric character

? Specifies one alphanumeric character

## Operation

Purges a file or fileset from disc. When purging a KSAM file, only the data file name need be specified since PROCMON will automatically purge the associated key file. The TEMP parameter may not be specified with the *fileset* parameter.

# DELETE

## Example

To purge the KSAM data file - GL01 and it's associated key file, enter:

```
DELETE GL01
```

To purge the temporary file - PAYTEMP, enter:

```
DELETE PAYTEMP,TEMP
```

# DISPLAY

Displays a message at the user's terminal.

## Syntax

```
DISPLAY [message]
        [MSGmic]
```

## Parameters

*message*            Message to be displayed at the user's terminal.

*mic*                4-position numeric specifying the catalog entry to be displayed at the user's terminal.

## Operation

Displays a message at the user's terminal. Typically used to inform the user of the next procedure step or request a response from the user.

## Example

To display the message "The payroll sort is executing", enter:

```
DISPLAY The payroll sort is executing
```

To display the catalog message entry number 0004, enter:

```
DISPLAY MSG0004
```

# ECHO

Displays contents of a procedure, as the procedure is executing.

## Syntax

```
ECHO [CRT][;STEP][;TO=filename][;LIST]
      [;OUTCLASS=[device[,outpriority][,numcopies]]
```

## Parameters

<b>CRT</b>	Each procedure step is echoed to the user's terminal.
<b>STEP</b>	Before a procedure step is executed, it is displayed at the terminal. To execute the step in its current state, the return key is pressed. If modifications are desired, the modifications are made on the screen, then the return key is pressed.
<b><i>filename</i></b>	Name of the file to contain the echoed output.
<b>LIST</b>	Echoed output is sent to the line printer.
<b><i>device</i></b>	Class name or logical device number to receive listing output. Used in configuration with the ;LIST parameter. Default is the line printer.
<b><i>outpriority</i></b>	The output priority for the echo list file. Must be a value from 1 (lowest priority) to 13 (highest priority). Default is 8.

## Operation

Displays each procedure step as it is executed. This is often helpful when debugging a sophisticated procedure with parameter passing, substitution and procedure nesting. The echo facility will remain on until the ECHOOFF command is executed.

**Example**

To display each procedure step to and be given the option of modification before execution, enter:

```
ECHO STEP
```

prior to running the procedure.

To generate a history of each procedure step in a file named - PROCHIST, enter:

```
FILE PROCHIST=PROCHIST;REC=80, ,F,ASCII;NEW;SAVE;CIR  
ECHO TO=*PROCHIST
```

prior to running the procedure.

To log procedure and commands executed to the line printer and defer the output:

```
ECHO LIST;OUTCLASS=LP,1
```

# ECHOOFF

Turns the previous ECHO command display off.

## Syntax

```
ECHOOFF
```

## Parameters

none

## Operation

Turns the previous ECHO command display off.

## Example

To turn off the display from a previous ECHO command, enter:

```
ECHOOFF
```

Activates the EDIT/3000 subsystem.

## Syntax

```
EDITOR
```

## Parameters

None

## Operation

Activates the EDIT/3000 subsystem. For further information, please refer to the EDIT/3000 reference manual.

## Example

To activate the EDIT/3000 subsystem, enter:

```
EDITOR
```

# ESCAPE

Sends an escape sequence to the terminal.

## Syntax

```
ESCAPE character code[character code. . . . .]
```

## Parameters

*character*        The escape character. This may be any keyboard character. Each time this character occurs in the sequence it is taken as the escape character.

*code*             The escape code. Refer to the Display Terminal Reference Manual, for the specific terminal, for a list of escape codes and their functions.

## Operation

Sends an escape sequence to the terminal.

## Example

To home the cursor and clear the screen on an HP2392A terminal:

```
ESCAPE *H*J
```

To set the RETURN key equal to the ENTER key on an HP2624B terminal:

```
ESCAPE *&q8te1{1R
```

### NOTE

Be sure to select a unique escape character. That is, do not select a code already in use as an escape code.

Assign values, return codes, arithmetic and substitution expression results.

## Syntax

```
EVALUATE [Pn[,length][:startpos]=expression]
          [CD=nnnn]
```

## Parameters

<i>P</i>	Signifies that a parameter is being set.
<i>n</i>	Is the parameter number. This is a value from 1 to 64.
<i>length</i>	Is the number of characters in the result. This is a value from 1 to 15.
<i>startpos</i>	Is the position in the expression used as the starting position of the result.
<i>CD</i>	Signifies that the return code is being set.
<i>nnnn</i>	Is the 4-digit return code value.
<i>expression</i>	Expression can be of several formats:

- 1) an alphanumeric string
- 2) a numeric expression, signed or unsigned
- 3) an arithmetic expression using
  - \*,/,+ and - as operators
  - exponentiation in the formats En, Enn, E+n, E-n,E-nn
  - parentheses to indicate order of operations
  - integer or real operands

- 1) Expression as an alphanumeric string

A string may contain letters, digits, and special characters. If the string is to contain only digits and special characters or embedded blanks, it must be enclosed in double quotes. If the value specified for LENGTH is greater than the actual length of the string then the result is padded with blanks on the right. If the value specified for LENGTH is less than the actual length of the string then the result is truncated on the right. If LENGTH exceeds 15 or is not specified the actual length of the string is used. When STARTPOS is

# EVALUATE

specified the result begins at the given position. One is the default starting position. STARTPOS can only be specified for string values.

## 2) Expression as a numeric value

A numeric value may be signed or unsigned and must be in integer form. Leading zeros can be used but may be truncated in the result. If the value specified for LENGTH is greater than the actual length of the number the result is padded with zeros on the left. If the value specified for length is less than the length of the number the result is truncated on the left. If LENGTH is not specified the actual length of the number is used. STARTPOS may not be specified for numeric values.

## 3) Expression as an arithmetic expression

An arithmetic expression may contain any of the four operations: \*,/,+, or -. (\*, /, +, and - denote multiplication, division addition, and subtraction respectively). Multiplication and division operations are performed before addition and subtraction. Parentheses may be used to group operations, operations within parentheses are performed before those outside of parentheses. Multiplication is assumed when a digit appears directly before a left parentheses. Exponents are specified using any of the following formats: En, Enn, E+n, E+nn, E-n, E-nn. Either integer or real operands are allowed. The LENGTH parameter is used in the same way as with numeric expressions - see 2) above.

## Operation

Allows the user to:

- Assign values to a parameter.
- Perform addition, subtraction, multiplication and division of values.
- Evaluate substitution expressions.
- Set the return code.

The *n*, *length*, *startpos*, *nnnn* and *expression* parameters may all be supplied as substitution expressions. Multiple expressions may appear on a line but must be separated by one or more blanks. Expressions are evaluated from left to right.

## Example 1

The following statement assigns a value of 128 to parameter 10:

```
EVALUATE P10,3=54128
```

Because the length of the numeric expression is greater than the specified length the result is truncated on the left.

## Example 2

The following statement assigns a value of "PLEASE WAIT" to parameter 5:

```
EVALUATE P5="PLEASE WAIT"
```

## Example 3

The following statement assigns a value of "YOUR JOB IS EXECUTING" to parameter 1:

```
EVALUATE P1,21:15="PLEASE WAIT - YOUR JOB IS EXECUTING"
```

## Example 4

The following statement assigns a value of 8 to parameter 2:

```
EVALUATE P2=10+-2
```

## Example 5

The following statement assigns a value of 128 to P64

```
EVALUATE P64=4*(12+20)
```

# FCOPY

Activates the FCOPY subsystem.

## Syntax

```
FCOPY FROM [= (filename)]
           [*]
           ;TO [= ((dfile,kfile)) [;optionlist]]
              { filename
                *
                <empty> } ]
```

## Parameters

For parameter description, please refer to the FCOPY/3000 reference manual.

## Operation

Activates the FCOPY/3000 subsystem. For further information, please refer to the FCOPY/3000 reference manual.

## Example

To create a duplicate copy of the file - GL01 and name it GL02, enter:

```
FCOPY FROM=GL01;TO=GL02;NEW
```

Performs branching within a procedure file.

## Syntax

```
GOTO label
```

## Parameters

*label*            Name of tag the procedure will branch to. May be up to 8 characters in length.

## Operation

Performs branching within a procedure. Typically used on a conditional statement. When the condition is met, processing branches to the corresponding TAG command.

## Example

To branch to another portion of PROCA if the file - INPUT does not exist, enter:

```
IF NOT DATA-INPUT GOTO BUILDA     1  
FILE INPUT=INPUT.DATA  
RUN PROGA.PUB  
TAG BUILDA                         2  
BUILD INPUT;REC=15,,F,ASCII       3
```

PROCA - procedure file

This command can only be called from a procedure.

# JOBQ

Executes a procedure file in the DS job queue.

## Syntax

```
JOBQ procname [parm1 [,parm2 [, ... [,parmN]]]]  
@  
  
[;USER  
;WSID  
;ID="jobname"]  
  
[;username [/i>userpass] .acctname [/i>acctpass]  
[;groupname [/i>grouppass]]]  
  
[;TIME=cpusecs]  
  
[;PRI= {BS  
CS  
DS  
ES}]  
  
[;HIPRI  
;INPRI=inputpri]  
  
[;RESTART]  
  
[;OUTCLASS=device] [,outputpri]  
  
[,numcopies]]
```

## Parameters

- procname* Name of the procedure to be placed on the job queue.
- parmN* Parameters to be passed to the procedure. A maximum of 64 parameters may be specified.
- @ All 64 parameters are to be passed to the specified procedure.
- USER Name of the user who launched the procedure will be printed with the procedure output.

WSID	Identification number of the terminal the procedure was launched from will be printed with the procedure output.
<i>jobname</i>	User assigned name of the procedure. Name will be printed with the procedure output.
<i>username</i>	User name which allows you to log on to this account. Must contain 1 to 8 alphanumeric characters beginning with an alphabetic character. Default is user who launched the job.
<i>userpass</i>	User password. Must contain from 1 to 8 alphanumeric characters beginning with an alphabetic character.
<i>acctname</i>	Account name. Must be from 1 to 8 alphanumeric characters beginning with an alphabetic character. Default is account from which job was launched.
<i>acctpass</i>	Account password. Must contain from 1 to 8 alphanumeric characters beginning with an alphabetic character.
<i>groupname</i>	Group name. Must contain from 1 to 8 alphanumeric characters beginning with an alphabetic character. Default is user's home group.
<i>grouppass</i>	Group password. Must contain from 1 to 8 alphanumeric characters beginning with an alphabetic character.
<i>cpusecs</i>	Maximum CPU time allowed in seconds.
DS,ES,BS,CS	Execution priority class.
<i>inputpri</i>	Relative input priority used in checking against access restrictions imposed by jobfence (if one exists).
HIPRI	Request for maximum input priority, causing job to be scheduled regardless of current jobfence or execution limit for jobs. This parameter requires System Manager capability.
RESTART	Request or restart a spooled job interrupted by system termination/restart.
<i>device</i>	Class name or logical device number of device to receive listing.

## **JOBQ**

*outputpri*            Output priority for job list file, if destined for spooled line printer or card punch.

*numcopies*            Number of copies of job listing to be produced.

### **Operation**

A stream file is automatically built to permit execution of the procedure file in the DS job queue. A copy of PROCMON, including the current local data area contents and user switch settings, is launched with the file to control the execution of each procedure step.

If the logon username, acctname and groupname are not specified, the job will log on in the same way as the user who launched it. If the account is secured, the SETPASS command (page 3-66) must be executed prior to execution of the JOBQ command.

### **Example**

To execute the procedure - PROCA in the job que and pass the parameter - 0123 to it, enter:

```
JOBQ PROCA 0123
```

```
JOBQ CUSTPRT ABC,DEF;USER.SALESLIB;WSID;ID=CUSPRT
```

Deletes a process.

## Syntax

```
KILL {programe}  
      {@}
```

## Parameters

*programe* Name of program to be deleted in the following format:

*programe[.group[.account]]*

*@* Deletes all previously "created" programs.

## Operation

Deletes a process previously "created". See Appendix A for more information on CREATE, ACTIVATE, and KILL.

## Example

To delete all previously "created" programs associated with the user's session, enter:

```
KILL @
```



- REORG** If the file is a KSAM file, its contents are listed sequentially by key.
- INCLUDE or EXCLUDE** Specifies whether selected records in the *fileset* are to be included in or excluded from the listing. Selected records are those that satisfy the column and characters comparison test.
- =** If the characters in the specified columns of the *fileset* record are the same as the comparison characters, the record will be included in or excluded from the listing.
- <>** If the characters in the specified columns of the *fileset* record are not the same as the comparison characters, the record will be included in or excluded from the listing.
- <** If the characters in the specified columns of the *fileset* record are less than the comparison characters the record will be included in or excluded from the listing.
- <=** If the characters in the specified columns of the *fileset* record are less than or equal to the comparison characters, the record will be included in or excluded from the listing.
- >** If the characters in the specified columns of the *fileset* record are greater than the comparison characters the record will be included in or excluded from the listing.
- >=** If the characters in the specified columns of the *fileset* record are greater than or equal to the comparison characters, the record will be included in or excluded from the listing.
- column** Specifies the location within the record where character comparison is to begin. If this parameter is not specified, each location in the record is compared to the comparison characters until the condition is met.
- characters** Specifies the comparison characters. Up to 30 characters may be compared. The character string may be undelimited or may be enclosed in single or double quotes.
- recsize** Record size of the *fileset* record. A positive number indicates words, a negative number indicates bytes. Allows the user to list portions of records. Default is the complete record is listed.
- maximum records** Specifies the total number of records to be listed. Allows the user to list part of a file. Default is the complete file is listed.

# LISTFILE

## Operation

Allows the user to:

- List the contents of a fileset or portion of a fileset to the line printer or to the user's terminal.
- Include or exclude selected records.
- List portions of records.
- List the hexadecimal representations of characters.

## Example

To list the contents of all files beginning with "PROC" at the user's terminal, enter:

```
LISTFILE PROC@;CRT
```

Places a string of characters in the local data area, LDA.

## Syntax

```
LOCAL [offset], "string" [; BLANK={length}]
      [; AREA={filename}]
           {LDASYS }
```

## Parameters

- offset* Specifies the starting position for a string of characters to be placed in the local data area. Can be any value from 1 through maximum size of LDA being referenced. Default is 1.
- string* Specifies the string of characters to be placed in the local data area. The string may be undelimited or may be enclosed in single or double quotes.
- length* Specifies the number of positions to be set to blanks, starting with the position specified by the offset parameter.
- @ Specifies that all positions following, and including, the offset position are to be set to blanks.
- filename* Name of the user local data area to be referenced. This must be a temporary or permanent file residing in the user's logon group and account.
- LDASYS Specifies that the system local data area is to be used. That is the permanent file LDASYS.PUB.SYS.

# LOCAL

## Operation

Places a string of characters in the local data area (LDA). The LDA is accessible from procedure and program files.

If both a character string and the BLANK keyword are specified on the same LOCAL statement, the LDA is first set to blanks and then the character string is placed in the LDA.

The area specified on the AREA keyword is referenced by the current LOCAL statement and by all subsequent LOCAL statements and LDA substitution expressions, until another AREA keyword on a LOCAL statement is processed. If the AREA keyword is not specified the user local data area is assumed.

The maximum LDA size that can be referenced is dependent upon the the PARM value used on the RUN statement when PROCMON is invoked. This value is a number from 1 through 32 indicating the number of multiples of 256 byte records in the LDA file. For example, to use an LDA of 1024 bytes, enter:

```
RUN PROCMON.PUB.SYS;PARM=4
```

The default PARM value is 2 indicating a 512 byte LDA.

<b>NOTE</b>
-------------

Local Data Area files must be built as direct files containing 1 through 32 multiples of 256 byte records. This may be done using the RPGINIT program or via the BLDDFU command. For more information on RPGINIT see the RPG Utilities Reference Manual.

## Example

To place the value - PAY001, in the local data area beginning at position 12, enter:

```
LOCAL 12,"PAY001"
```

Displays a specified menu at the user's terminal.

## Syntax

```
MENU [menuname]
```

## Parameters

*menuname* Name of menu to be displayed, in the following format:

```
menuname[.group].account]]
```

If the group is not specified, PROCMON will look in the user's log-on group for the "menuname." If the menu is not in the log-on group, PROCMON will look in the "M" group of the log-on account.

Note: A file equation can be used to display a menu immediately following PROCMON initialization. The formal file designator is:

```
FILE MENU=menuname[.group].account]]
```

If this parameter is not specified, the menu currently stored in memory will be displayed.

## Operation

Displays a specified menu at the user's terminal.

## Example

To display the menu - PAYMENU, located in the user's log-on group, enter:

```
MENU PAYMENU
```

# MENUOFF

Clears a menu display from the user's terminal.

## Syntax

```
MENUOFF
```

## Parameters

none

## Operation

Clears a menu display from the user's terminal. This can be helpful when the user desires to use the display space for something other than the current menu. Once the MENUOFF command is issued and the current menu display is cleared, a MENU command without parameters will redisplay the previous menu.

## Example

To clear the current menu from the user's terminal, enter:

```
MENUOFF
```

```
This command can not be called from a procedure.
```

Sends a message to a selected terminal. Recalls the user's pending messages.

## Syntax

```
MSG [wsid,msgstring]  
    [<empty>]
```

## Parameters

*wsid* Two character terminal identifier in the following format:

Logical device	20	Terminal ID	B0
	25		B5
	40		D0
	45		D5

*msgstring* The string of characters to be sent to the terminal identifier (*wsid*).

*<empty>* Recalls all messages previously sent to the user's terminal.

## Operation

Sends a message to a terminal. If the terminal is not currently in session mode, the message is stored in PROCMON's data base, until a user at the specified terminal issues a MSG command to recall all pending messages.

### NOTE

If a user has pending messages the word "MESSAGE" flashes in the top right hand side of the currently displayed menu.

# MSG

## Example

To send the message "Please mount payroll tape!!" to logical device 25, enter:

MSG B5,Please mount payroll tape !!

To recall the user's pending messages, enter:

MSG

**OFF**

Terminates the process monitor.

## Syntax

OFF

## Parameters

none

## Operation

Terminates the process monitor.

## Example

To terminate PROCMON and return the user to MPE's control enter:

OFF

# OPERATOR

Displays a message at the system console.

## Syntax

```
OPERATOR [message]
          [MSGmic]
```

## Parameters

*msgstring*      The message to be displayed at the system console.

*mic*              4-position numeric specifying the catalog entry to be displayed at the system console.

## Operation

Displays a message at the system console. Typically used to request a response from the operator.

## Example

To prompt the operator to mount the tape - GLTAPE, enter:

```
OPERATOR Please mount GLTAPE tape!!
```

Sets and displays the current file search path.

## Syntax

```
PATH [account1[\account2[\...[\account7]]]]  
      <empty>
```

## Parameters

*account*            The account to be searched. Up to seven accounts can be designated.

<empty>            Displays the current search path.

## Operation

Sets and displays the current file search path for programs, procedures, and menus. If a file is not found in the specified account, PROCMON will search for it along the defined account path. Only the account name changes, not the group name. The path is initially set to: *Log-on account*,SYS. Data files are excluded from the search. The search path remains active for the duration of the session or until another PATH command is executed.

## Example

To set the search path to the P group of the Library account (when a procedure is not found in the P group of the PAYROLL account), enter:

```
PATH PAYROLL,LIBRARY
```

To display the current search path, enter:

```
PATH
```

# PAUSE

Displays a message at the user's terminal and waits for the user to reply.

## Syntax

```
PAUSE [message]
      [MSGmic]
```

## Parameters

*message*            The message to be displayed at the user's terminal.

*mic*                4-position numeric specifying the catalog entry to be displayed at the system console.

## Operation

Displays a message at the user's terminal and waits for the user to reply.

## Example

To display the message "END OF STEP 2 - press any key to continue", enter:

```
PAUSE End of step2
```

Declares the file attributes to be used at file open time.

## Syntax

PFILE

## Parameters

For parameter description, please refer to the :FILE command parameter description located in the MPE Commands Reference Manual.

## Operation

File equations set by a procedure are canceled when the procedure terminates. The PFILE command allows a programmer to retain a file equation for the duration of a session. PFILE attributes can be temporarily altered by executing an MPE FILE command. When in cursor mode, the previous PFILE attributes are reset when the next procedure terminates. When in a procedure file, the attributes are reset in accordance with the previous PFILE command. PFILE attributes can be permanently altered by executing another PFILE command. PFILE attributes can be temporarily canceled by executing an MPE RESET command. When in cursor mode, the previous PFILE attributes are reset when the next procedure terminates. When in a procedure file, the attributes are reset in accordance with the previous PFILE command. PFILE attributes can be permanently canceled by executing an RFILE command (page 3-62).

Four PFILE equations are permitted per session.

## Example

To set the PFILE equation - FILE DATA=PAY1.DATA, enter:

```
PFILE DATA=PAY1.DATA
```

# PREP

Prepares a compiled source program for execution.

## Syntax

```
PREP uslfile,progfile  
    [;ZERODB]  
    [;PMAP]  
    [;MAXDATA=segsiz]  
    [;STACK=stacksiz]  
    [;DL=dlsiz]  
    [;CAP=caplist]  
    [;RL=filename]  
    [;PATCH=patchsiz]  
    [;NOSYM]  
    [;FPMAP]  
    [;NOFPMAP]  
    [;CHECKSUM]
```

## Parameters

For parameter description, please refer to the MPE commands reference manual.

## Operation

Prepares a compiled source program for execution. For further information, please refer to the MPE commands reference manual.

## Example

To prepare the USL file - PAYUSL for execution as PAYROLL, enter:

```
PREP PAYUSL,PAYROLL
```

# PROCCONFIG

Provides customization of the PROCMON facility.

## Syntax

PROCCONFIG	[WARN NOWARN]	[;BANNER ;NOBANNER]
	[;CRRPARAM ;NOCRRPARAM]	[;CREATEMSG ;NOCREATEMSG]
	[;CRMPARAM ;NOCRMPARAM]	[;SUSPMSG ;NOSUSPMSG]
	[;JOBMSG ;NOJOBMSG]	[;KILLMSG ;NOKILLMSG]
	[;PAUSEANY ;PAUSEZERO]	[;WAITPROMPT ;NOWAITPROMPT]
	[;AIDSMODESON ;AIDSMODESOFF]	[;UPRPARMON ;UPRPARMOFF]
	[;EXCWAIT ;NOXCWAIT]	[;UPMPARMON ;UPMPARMOFF]
	[;DATE={ <i>mly</i> <i>dmy</i> <i>ymd</i> }]	

## Parameters

WARN	Warning messages are displayed at the user's terminal. Default.
WARN	Warning messages are not displayed at the user's terminal.
CRRPARAM	Carriage return is allowed for a blank passed as a required parameter. Default.
NOCRRPARAM	Space bar and carriage return are allowed for a blank passed as a required parameter.
CRMPARAM	Carriage return is allowed for a blank passed as a missing parameter.

# PROCCONFIG

Default.

NOCRMPARM	Space bar and carriage return are allowed for a blank passed as a missing parameter.
JOBMSG	When a procedure placed on the job queue completes, a message is displayed at the user's terminal. Default.
NOJOBMSG	When a procedure placed on the job queue completes, a message is not displayed at the user's terminal.
PAUSEANY	When a procedure executes the PAUSE command, the user can press any key to continue procedure execution. Default.
PAUSEZERO	When a procedure executes the PAUSE command, the user can press only the "0" key to continue procedure execution.
AIDSMODESON	Aids and Modes keys are enabled. Default.
AIDSMODESOFF	Aids and Modes keys are disabled.
EXCWAIT	If an exclusive access violation occurs during an attempt by an RPG program to open a file, processing will be suspended for 60 seconds, then the program will be re-RUN. The EXCQUIT option must be specified on the \$CONTROL record in the RPG source program. See the RPG Reference Manual for more information on this feature.
NOEXCWAIT	If an exclusive access violation occurs during an attempt by an RPG program to open a file, processing will continue with the next command after the RUN statement. Default.
BANNER	PROCMON's banner appears on lines 1 and 2 of user's terminal when a menu is displayed. Default.
NOBANNER	PROCMON's banner is not displayed along with menus. This is only supported with menus which were compiled with SIGEDIT (i.e. display text source consisted of S & D specs).
CREATEMSG	When a process is created, via the CREATE command. A message is displayed at the user's terminal. Default.

# PROCCONFIG

NOCREATEMSG	When a process is created, via the CREATE command. A message is not displayed at the user's terminal.
SUSPMSG	When a process is suspended, via the RPG operation SUSP, a message is displayed at the user's terminal. Default.
NOSUSPMSG	When a process is suspended, via the RPG operation SUSP, a message is not displayed at the user's terminal.
KILLMSG	When a process is deleted, via the KILL command or by program termination a message is displayed at the user's terminal. Default.
NOKILLMSG	When a process is deleted, via the KILL command or a program termination a message is not displayed at the user's terminal.
WAITPROMPT	When a procedure executes an IF FILEOPEN-filename WAIT conditional expression, if the condition is true a prompt is displayed at the user's terminal requesting the user to respond "Y" or "N". Default.
NOWAITPROMPT	When a procedure executes an IF FILEOPEN-filename WAIT conditional expression, if the condition is true a prompt is not displayed at the user's terminal and the response defaults to "Y".
UPRPARMON	If the user enters the required parameter in lowercase PROCMON will upshift it.
UPRPARMOFF	If the user enters the required parameter in lowercase PROCMON will not upshift it. Default.
UPMPARMON	If the user enters the missing parameter in lowercase PROCMON will upshift it.
UPMPARMOFF	If the user enters the missing parameter in lowercase PROCMON will not upshift it. Default.
DATE	The DATE parameter is used to set the format for the RPGUDATE.
<i>mdy</i>	Specifies that the RPGUDATE is in month, day, year format. Default.
<i>dmy</i>	Specifies that the RPGUDATE is in day, month, year format.

# PROCCONFIG

*ymd* Specifies that the RPGUPDATE is in year, month, day format.

none Procmon default values are set.

## Operation

Provides customization of the PROCMON facility.

Note: A file equation can be used to execute a procedure file during PROCMON initialization. The formal file designator is:

```
FILE IPL=procfile[.group[.account]]
```

It is suggested that the PROCCONFIG command be placed within the PROCMON IPL procedure, since the command need only be executed once during a PROCMON session.

## Example

To suppress warning messages and allow a carriage return for a blank passed as a required parameter, enter:

```
PROCCONFIG NOWARN;CRRPARAM
```

Displays a screen format which prompts for procedure substitution parameters.

## Syntax

```
PROMPT formsfile[.group[.account]]formname  
      [;START=parameter number][;LENGTH=n,n.....]
```

## Parameters

*formsfile* Name of file containing screen format to be displayed, in the following format:

*formsfile*[.group[.account]]

*formname* Name of the screen format to be displayed.

START Specifies the number of the first positional parameter to be prompted for.

*parameter number* Any value from 1 through 64. Default is 1.

LENGTH Specifies the length in bytes of the parameters being prompted for.

*n* Specifies the length of each parameter being prompted for, starting with the parameter specified by START. *n* can have a value from 0 through 128. The maximum number of bytes which can be prompted for is 1024. If *n* is not specified for a particular parameter its length defaults to 8.

### NOTE

If the parameter length defined in the screen format differs from that defined by the LENGTH keyword, data truncation may occur.

## PROMPT

### Operation

Displays a substitution screen, allowing the user to enter parameters by filling in fields (as opposed to answering prompts). The user's screen entries are passed to the current procedure as substitution values. If both the START and LENGTH parameters are omitted the user will be prompted for 11 parameters, each of 8 bytes in length, starting at parameter 1.

### Example

To display the screen - SCRNO1, from the file - PAYROLFM, located in the "PUB" group of the log-on account, enter:

```
PROMPT PAYROLFM.PUB,SCRNO1
```

This command can only be called from a procedure.

# RENAME

Changes identity (file name or group name) of a disc file.

## Syntax

```
RENAME oldfile,newfile[;newkey]
```

## Parameters

*oldfile* Current name of file, in the following format:

*filename[group]*

*newfile* New name of file in the following format:

*newfile[group]*

*newkey* New name of the KSAM key file. This parameter is optional. If it is not provided when a KSAM file is being renamed, PROCMON will name the key file. If the data file name is less than 8 characters in length, the key file name will be the data file name with a "K" appended to it. If the data file name is 8 characters long, the key file name will be the data file name with a "K" overlaying the last character.

## Operation

Changes the system file identification of a permanent disc file. The file can be a sequential or KSAM file. The user must be the creator.

## Example

To rename the file - OLDFILE, to the file - NEWFILE, enter:

```
RENAME OLDDATA,NEWDATA
```

# RESETPROC

Executes a procedure and resets it's nesting level.

## Syntax

```
RESETPROC procname [parm1 [, parm2 [, ... [, parmN ]]]  
[@]
```

## Parameters

*procname* Name of procedure to be executed. The format is:

*procname*[.*group*].*account*]

*parmN* Parameters to be passed to the procedure.

*@* All parameters being used by the current procedure are to be passed to the specified procedure.

## Operation

255 levels of procedure nesting are permitted. Some application tasks, however, can require more. For example, a procedure may need to call itself multiple times to complete a task. The RESETPROC command executes the specified procedure and resets the procedure's nesting level to 1. When the specified procedure completes, control is not returned to the procedure containing the RESETPROC command. Instead, it is returned to the user.

## Example

To permit the procedure - PROC A to be executed multiple times, enter:

```
DISPLAY SELECT WEEK1, WEEK2, WEEK3 OR WEEK4
IF ?1R?/FINISH CANCEL
IF ?1?/WEEK1 WEEK ONE
IF ?1?/WEEK1 RESETPROC PROC A
IF ?1?/WEEK2 WEEK TWO
IF ?1?/WEEK2 RESETPROC PROC A
IF ?1?/WEEK3 WEEK THREE
IF ?1?/WEEK3 RESETPROC PROC A
IF ?1?/WEEK4 WEEK FOUR
IF ?1?/WEEK4 RESETPROC PROC A
DISPLAY INVALID CHOICE, PLEASE RE-ENTER
RESETPROC PROC A
```

PROC A - procedure file

This command can only be called from a procedure.

# RETURN

Causes termination of a procedure or a return to the calling procedure.

## Syntax

```
RETURN [@]
```

## Parameters

@ All parameters being used by the current procedure are to be passed to the calling procedure.

## Operation

If the RETURN command exists in a first-level procedure, its execution causes termination of the procedure and the user is returned to the PROCMON cursor. If the command exists in a nested procedure, its execution causes a return to the calling procedure.

## Example

To return control from procedure - PROCE, to procedure - PROCD, if the file - INPUT does not exist, enter:

```
RUN PAY01.PUB  
PROCE  
DISPLAY INPUT FILE DOES NOT  
DISPLAY EXIST - TERMINATE
```

PROCD - procedure file

```
IF NOT DATA-INPUT.DATA RETURN  
FILE INPUT=INPUT.DATA  
RUN PROGE.PUB
```

PROCE - procedure file

This command can only be called from a procedure.

# RFILE

Removes the file attributes set by the PFILE command.

## Syntax

```
RFILE {filename}
      @
```

## Parameters

*filename*            Name of the file equation to be canceled.

@                    All PFILE equations are canceled.

## Operation

File equations set by a procedure are canceled when the procedure terminates. The PFILE command (page 3-49) allows a programmer to retain a file equation for the duration of a session. The RFILE command cancels the PFILE equation.

## Example

To cancel the PFILE equation - FILE DATA=PAY1.DATA, enter:

```
RFILE DATA
```

To cancel all PFILE equations, enter:

```
RFILE @
```

Compiles an rpg program.

## Syntax

```
RPG [textfile][,uslfile][,[listfile]
    [,masterfile][,newfile]]]
```

## Parameters

For parameter description, please refer to the MPE commands reference manual.

## Operation

Compiles an RPG program into a User Subprogram Library (USL) file. For further information, please refer to the MPE commands reference manual.

## Example

To compile the source file - TESTS, into the USL file TESTUSL, enter:

```
RPG TESTS,TESTUSL
```

# RUN

Executes a prepared program.

## Syntax

```
RUN progfile[,entrypoint]  
  [;NOPRIV]  
  [;LMAP]  
  [;DEBUG]  
  [;MAXDATA=segsiz]  
  [;PARAM=parameternum]  
  [;STACK=stacksize]  
  [;DL=dlsiz]  
  
  [;LIB={P  
         G  
         S}]  
  
  [;NOCB]  
  [;INFO=string]  
  
  [;STDIN=[*formaldesig]  
          [fileref  
          $NULL]]  
  
  [;STDLIST=[*formaldesig]  
            [fileref[,NEW]  
            $NULL]]
```

## Parameters

For parameter description, please refer to the MPE commands reference manual.

## Operation

Executes a prepared program. For further information, please refer to the MPE commands reference manual.

## Example

To execute the program file - PAY, enter:

```
RUN PAY
```

Provides an extra level of application security.

## Syntax

```
SECURITY
```

## Parameters

None

## Operation

Provides an extra level of application security by allowing the user to select menu item numbers, but only enter the BYE, OFF and MSG commands directly from the keyboard. Typically used to control the user's capabilities, by locking them into an application menu and preventing access to the MPE and PROCMON command sets. To terminate PROCMON, the user can enter BYE, OFF or item number 0.

Note: A file equation can be used to execute a procedure file during PROCMON initialization. The formal file designator is:

```
FILE IPL=procfile[.group[.account]]
```

It is suggested that the SECURITY command be placed within the PROCMON IPL procedure, since the command need only be executed once during a PROCMON session.

## Example

To turn menu security on, enter:

```
SECURITY
```

### NOTE

The security feature may also be activated by running PROCMON with the "SECURITY" entry point.

```
RUN PROCMON.PUB.SYS,SECURITY
```

# SETPASS

Provides PROCMON with account, user and group passwords.

## Syntax

```
SETPASS [acctpass[/userpass[/grouppass]]]  
        <empty>
```

## Parameters

<i>acctpass</i>	Log-on account password.
<i>userpass</i>	Log-on user password.
<i>grouppass</i>	Log-on group password.
<empty>	User is prompted for passwords.

## Operation

When PROCMON executes the JOBQ and TASK commands, a stream file is automatically built to permit execution of a procedure file in the job queue. If the account is secured, the SETPASS command must be executed prior to execution of the JOBQ and TASK commands. The SETPASS command provides PROCMON with the passwords required for building a stream file job card.

Note: A file equation can be used to execute a procedure file during PROCMON initialization. The formal file designator is:

```
FILE IPL=procfile[.group[.account]]
```

It is suggested that the SETPASS command be placed within the PROCMON IPL procedure, since the command need only be executed once during a PROCMON session.

**Example**

To provide PROCMON with the account password - MARY, the user password - PAY and the group password - ROLL, enter:

```
SETPASS MARY,PAY,ROLL
```

# SHOWACTIVE

Displays the "active" status of a procedure file.

## Syntax

```
SHOWACTIVE [procname1 [, procname2 [, ... [procnameN]] ]  
           @
```

## Parameters

*procnameN*      Name of the procedure file.

@                All active procedures are displayed.

## Operation

Displays the "active" status of a selected procedure file and displays all currently active procedures.

## Example

To display the "active" status of procedure files - PROCA and PROCB, enter:

```
SHOWACTIVE PROCA,PROCB
```

To display all active procedure files, enter:

```
SHOWACTIVE @
```

# SHOWMENU

Displays the commands associated with the current menu.

## Syntax

```
SHOWMENU
```

## Parameters

None

## Operation

A menu consists of 2 files, a display text file and a command file. The display text file is displayed at the user's terminal when the MENU command is executed. For every item number in the display text file there is a corresponding command in the command file. The SHOWMENU command displays, at the user's terminal, the commands associated with the current menu display.

## Example

To display the commands associated with the user's current menu, enter:

```
SHOWMENU
```

# SWITCH

Sets one or more of the external indicators.

## Syntax

```
SWITCH indset
```

## Parameters

*indset*            Eight-position indicator setting. The first, or leftmost character sets indicator U1, the second character sets U2, and so on.

Character 0 sets the specified indicator off, character 1 sets the indicator on and character X leaves the indicator as is.

## Operation

Sets one or more of the external indicators. The switch setting remains in effect until one of the following occurs:

- Another switch command is used
- User's session is ended
- User's program changes the external indicators

## Example

To set indicators U1 and U2 off, U3 through U7 on and leave U8 as is, enter:

```
SWITCH 0011111X
```

Used with the GOTO command to perform branching within a procedure file.

## Syntax

```
TAG label
```

## Parameters

*label* Name of tag, referenced by GOTO command. May be up to eight characters in length.

## Operation

Used with the GOTO command to perform branching within a procedure file. The GOTO command is typically used on a conditional statement. When the condition is met, processing branches to the corresponding TAG command.

## Example

To execute another portion of PROCA if the file - INPUT does not exist, enter:

```
IF NOT DATA-INPUT GOTO BUILDA 1  
FILE INPUT=INPUT.DATA  
RUN PROGA.PUB  
TAG BUILDA 2  
BUILD INPUT;REC=15,,F,ASCII 3
```

PROCA - procedure file

This command can only be called from a procedure.

# TASK

Executes a procedure file in the CS job queue.

## Syntax

```
TASK procname [parm1 [, parm2 [, ... [, parmN]]]]

[;USER
;WSID
;ID="jobname"]

[;username [/i>userpass].acctname [/i>acctpass]
[, groupname [/i>grouppass]]

[;TIME=cpusecs]

[;PRI = { BS
          CS
          DS
          ES } ]

[;HIPRI
;INPRI=inputpri]

[;RESTART]
[;OUTCLASS=device] [, outputpri]
[, numcopies]]
```

## Parameters

<i>procname</i>	Name of the procedure to be placed on the job queue.
<i>parmN</i>	Parameters to be passed to the procedure. A maximum of 64 parameters may be specified.
@	All 64 parameters are to be passed to the specified procedure.
USER	Name of the user who launched the procedure will be printed with the procedure output.
WSID	Identification number of the terminal the procedure was launched from will be printed with the procedure output.

## TASK

<i>jobname</i>	User assigned name of the procedure. Name will be printed with the procedure output.
<i>username</i>	User name which allows you to log on to this account. Must contain from 1 to 8 alphanumeric characters beginning with an alphabetic character. Default is user who launched the job.
<i>userpass</i>	User password. Must contain from 1 to 8 alphanumeric characters beginning with an alphabetic character.
<i>acctname</i>	Account name. Must be from 1 to 8 alphanumeric characters beginning with an alphabetic character. Default is account from which job was launched.
<i>acctpass</i>	Account password. Must contain from 1 to 8 alphanumeric characters beginning with an alphabetic character.
<i>groupname</i>	Group name. Must contain from 1 to 8 alphanumeric characters beginning with an alphabetic character. Default is user's home group.
<i>grouppass</i>	Group password. Must contain from 1 to 8 alphanumeric characters beginning with an alphabetic character.
<i>cpusecs</i>	Maximum CPU time allowed in seconds.
CS,DS,ES,BS	Execution priority class.
<i>inputpri</i>	Relative input priority used in checking against access restrictions imposed by jobfence (if one exists).
HIPRI	Request for maximum input priority, causing job to be scheduled regardless of current jobfence or execution limit for jobs.
RESTART	Request or restart a spooled job interrupted by system termination/restart.
<i>device</i>	Class name or logical device number of device to receive listing.
<i>outputpri</i>	Output priority for job list file, if destined for spooled line printer or card punch.
<i>numcopies</i>	Number of copies of job listing to be produced.

# TASK

## Operation

A stream file is automatically built to permit execution of the procedure file in the CS job queue. A copy of PROCMON, including the current local data area contents and user switch settings, is launched with the file, to control the execution of each procedure step.

If the logon username, acctname and groupname are not specified, the job will log on in the same way as the user who launched it. If the account is secured, the SETPASS command (page 3-66) must be executed prior to execution of the TASK command.

## Example

To execute PROCA in the CS job queue and pass a single parameter - 0123, enter:

```
TASK PROCA 0123
```

# WAIT

Causes a procedure to suspend execution for a given amount of time or until a certain time.

## Syntax

```
WAIT [INTERVAL=hhmmss]  
      [TIME=hhmmss]
```

## Parameters

- INTERVAL** Specifies the amount of time that a procedure will wait before it resumes processing. This amount is given in *hhmmss*.
- TIME** Specifies that the procedure will wait until a specified time of day occurs. This time is given in *hhmmss*.
- For both parameters a 6-digit number must be entered for *hhmmss*.
- hh** *hh* specifies hours. Valid values are in the range 00 to 24. 00 through 12 represents the hours from 12 midnight to 12 noon, and 13 through 24 represents 1p.m. to 12 midnight. Both 000000 and 240000 represent midnight.
- mm** *mm* specifies minutes. Valid values are in the range 00 to 59.
- ss** *ss* specifies seconds. Valid values are in the range 00 to 59.
- 240000 is the maximum possible value. if *hh* is 24, *mm* and *ss* must both be 00.
- none** Procedure will wait 60 seconds before it resumes processing.

# WAIT

## Operation

Causes a procedure to suspend execution. Typically used on a conditional statement. When used with the IF FILEOPEN-*filename* expression, if the condition is met, the user is given the choice of waiting or terminating. If the user chooses to wait, the condition will again be tested and if it is still true, the user will be asked once again to wait or terminate.

## Example 1

To execute the procedure file - PROCF, only after the procedure file - PROCA has completed execution, enter:

```
IF ACTIVE-PROCA WAIT  
FILE INPUT=INPUT.DATA  
RUN PROGF.PUB
```

PROCF - procedure file

## Example 2

To run a program PROGA at 6:30p.m. execute the following procedure:

```
WAIT TIME-183000  
FILE INPUT=INPUT.DATA  
RUN PROGA
```

## Example 3

To wait 5 minutes between program retries

```
IF ACTIVE-PROCA WAIT INTERVAL-000500  
RESETPROC PROCB  
RUN PROGB
```

This command can only be called from a procedure.

Changes the user's terminal identification number.

## Syntax

```
WSID wsid
```

## Parameters

*wsid*

Two character terminal identifier in the following format:

Logical device	20	Terminal ID	B0
	25		B5
	40		D0
	45		D5

## Operation

Changes the user's terminal identification number.

## Example

To change the user's terminal identification number from D4 to B6, enter:

```
WSID B6
```



## SUBSTITUTION EXPRESSIONS

Substitution expressions allow the programmer to insert variable information into a procedure statement, prior to execution of the statement. For example, a programmer can substitute:

- Positional parameters on the procedure command.
- Information supplied in response to prompts.
- Specified locations in the user's local data area.
- Specified message members.

## Substitution Expression Formats

Substitution expressions always begin and end with a question mark. A substitution expression begins any time a question mark is followed by a number or one of the following characters: C, D, F, L, M, R, S, U, or W. A question mark followed by any other character is not treated as a substitution expression. Substitution or prompting occurs whenever a valid expression is encountered.

Following are descriptions of the available substitution expressions.

FORMAT	DESCRIPTION
<code>?n?</code>	Substitutes the value of the nth positional parameter.  FILE INPUT=?2?
<code>?nF'value'?</code>	Forces a new value to replace the nth positional parameter.  FILE INPUT=?2F'GL001'? ... FILE OUTPUT=?2?
<code>?n'default'?</code>	Substitues the value of the nth positional parameter; if the parameter is not defined, the default is permanently assigned as the value of the nth positional parameter.  FILE INPUT=?2'GL001'?

Substitution within Procedures

?nT'default'?

Substitutes the value of the nth positional parameter; if the parameter is not defined, the default is temporarily assigned as the value of the nth positional parameter.

```
FILE INPUT=?2T'GLOO1'?  
...  
FILE OUTPUT=?2T'PAY001'?
```

?nR'cat'?

Substitutes the value of the nth positional parameter; if the parameter is not defined, the catalog message is displayed and the user enters the value of the nth positional parameter.

```
FILE INPUT=?3R'1234'?.DATA
```

?R'cat'?

Catalog message is displayed and the user enters the value to be substituted.

```
FILE INPUT=?R'1234'?.DATA
```

?nR?

Substitutes the value of the nth positional parameter; if the nth parameter is not defined, the message "ENTER THE MISSING PARAMETER" is displayed and the user must enter the value of the nth positional parameter.

```
FILE INPUT=?nR?
```

?R?

Displays the message "ENTER THE MISSING PARAMETER" and the user enters the value to be substituted.

```
FILE INPUT=?R?FILE
```

?WS?

Substitutes the 2-character station ID of the user's terminal. The format is as follows:

Ldev	20	Term	ID	B0
	25			B5
	40			D0
	45			D5

```
FILE INPUT=?WS?FILE
```

?USER?

Substitutes the user's ID. The format is as follows:

```
HELLO userid.group.account  
-----  
FILE INPUT=?USER?
```

?L'offset ,lng'?	Substitutes a value from the local data area. The offset specifies the starting location of the value to be substituted. The length (lng) specifies the number of characters to be substituted from the local data area.
	FILE INPUT=?L'12,8'?
?M'xxxx,offset ,lng'?	Substitutes a value from a record in the catalog file. The offset specifies the starting location of the value to be substituted. The length (lng) specifies the number of characters to be substituted from the catalog record.
	FILE INPUT=?M'1255,12,8'?
?Mxxxx?	Substitutes a value from a record in the catalog file.
	FILE INPUT=?M1255?
?Cn?	Substitutes the length of the nth positional parameter.
?C'value'?	Substitutes the length of the specified value.
?MENU?	Substitutes the menu name of the current menu.
?PROC?	Substitutes the name of the first-level procedure that is running.
?TIME?	Substitutes the current system time in the format HHMMSS.
?GROUP?	Substitutes the name of the logon group.
?CD?	Substitutes a 4-character code returned by the process monitor. The possible return codes are:
	0000 The previous job step terminated normally. 2001- 2024 Command keys 1 through 24 were returned from the PROMPT command.
?CLIB?	Substitutes the name of the account used in the last RUN command.
	FILE INPUT=INPUT.DATA.?CLIB?
?SLIB?	Substitutes the name of the user's logon account.

Substitution within Procedures

FILE INPUT=INPUT.DATA.?SLIB?

?DATE?

Substitutes the current session date.

FILE INPUT=DF?DATE?

?F'A, *filename*'?

Substitutes the number of records found in the specified file (filename).

BUILD INPUT;DISC=?F'A,PAYROLL'?

?F'S, *filename*'?

Substitutes the number of records allocated in the specified file (filename).

BUILD INPUT;DISC=?F'S,PAYROLL'?

?JCW?

Substitutes the value of the JCW.

DISPLAY THE JCW IS CURRENTLY SET TO ?JCW?

## Nested Substitution Expressions

Substitution expressions may be nested. For example:

```
?M'1234,1,23?
```

substitutes the first 23 characters of the catalog message number - 1234. The characters are:

```
FILE DATA=?WS?TEMP.DATA
```

The statement resulting from the first substitution expression contains another substitution expression. Procmon processes the second substitution expression. If the the terminal is logical device 24, then the resulting statement is:

```
FILE DATA=B4TEMP.DATA
```

One substitution expression can be the default for another substitution expression. For example:

```
?1'?2'??
```

If the first parameter is not specified, PROCMON substitutes the value of the second parameter.



## CONDITIONAL EXPRESSIONS

Conditional expressions within a procedure allow the programmer to conditionally generate command statements when the procedure is called. There are two types of conditional statements: IF and ELSE expressions.

### IF Expressions

The IF expression can only be used within a procedure. An IF expression tests for a specified condition; if the condition is true, the corresponding statement is executed.

An IF expression can have one of the following formats:

*IF conditional parameter* (If the condition is true, the corresponding statement is executed)

*IF NOT conditional parameter* (If the condition is false, the corresponding statement is executed)

A conditional parameter must be a string of nonblank characters. A blank indicates the end of the parameter. Following are descriptions of the available conditional parameters:

FORMAT	DESCRIPTION
<i>ACTIVE-procname</i>	True if the specified procedure is executing on the system.  IF ACTIVE-PROCA LISTF @.DATA,1
<i>DATA-filename</i>	True if the specified file exists. Filename can be qualified. ,TEMP or ,PERM can be used to specify domain. If the domain is not specified, both are checked.  IF DATA-PAYROLL.DATA PURGE PAYROLL.DATA  IF DATA-PAY01,TEMP DELETE PAY01,TEMP
<i>FILEOPEN-filename</i>	True if the specified file is open. Filename can be qualified.  IF FILEOPEN-PAYROLL WAIT
<i>JOBQ-YES</i>	True if the called procedure is executed from the

Condition within Procedures

JOBQ.

IF JOBQ=YES TELLOP Mount formAA

JOBQ-NO

True if the called procedure is not run from the JOBQ.

IF JOBQ-NO TELL ?USER?,Mount formAA

LOAD-*progname*

True if the specified file exists. Filename can be qualified. ,TEMP or ,PERM can be used to specify domain. If the domain is not specified, both are checked.

IF LOAD-PROGA.PUB RUN PROGA.PUB

PROC-*procname*

True if the specified file exists. Filename can be qualified. ,TEMP or ,PERM can be used to specify domain. If the domain is not specified, both are checked.

IF PROC-PROCA.P PROCA.P

SOURCE-*progname*

True if the specified file exists. Filename can be qualified. ,TEMP or ,PERM can be used to specify domain. If the domain is not specified, both are checked.

IF SOURCE-PROGA.S PURGE PROGA.S

SWITCH-*indsetting*

True if all external indicators for the user's session are in the state indicated by the indicator setting value (*indsetting*).

0 - indicator is off
1 - indicator is on
X - indicator is not checked

IF SWITCH-11110000 RUN PROGB.PUB

SWITCHn-*indsetting*

True if the specified indicator is in the state indicated by the indicator setting value (*indsetting*).

	IF SWITCH4-0 DELETE OUTPUT.DATA
IF CONSOLE-YES	True if the specified procedure is running on the system console. Console must be logical device 20.  IF CONSOLE-YES RUN PROGA.PUB
IF CONSOLE-NO	True if th specified procedure is NOT running on the system console. Console must be logical device 20.  IF CONSOLE-NO CANCEL
<i>string1/string2</i>	True if string1 is equal to string2. Each character string can be up to 128 characters long.  IF ?2?/YES RUN PROGB.PUB
<i>string1=string2</i>	True if string1 is equal to string2. Each character string can be up to 128 characters long.  IF ?4?=123456 DELETE OUTPUT.DATA
<i>string1&lt;&gt;string2</i>	True if string1 is not equal to string2. Each character string can be up to 128 characters long.  IF ?4?<>123456 DELETE OUTPUT.DATA
<i>string1&gt;string2</i>	True if string1 is greater than string2. Each character string can be up to 128 characters long.  IF ?4?>123456 DELETE OUTPUT.DATA
<i>string1&lt;string2</i>	True if string1 is less than string2. Each character string can be up to 128 characters long.  IF ?4?>123456 DELETE OUTPUT.DATA
<i>string1&gt;=string2</i>	True if string1 is greater than or equal to string2. Each character string can be up to 128 characters long.  IF ?4?>=123456 DELETE OUTPUT.DATA

## Condition within Procedures

*string1* <= *string2*

True if *string1* is less than or equal to *string2*. Each character string can be up to 128 characters long.

```
IF ?4?>=123456 DELETE OUTPUT.DATA
```

String values may be enclosed in single or double quotes and may contain character data or substitution expressions. Numeric data may be either positive or negative, for example +10 or -5. If a sign is not specified the value is taken as positive. If a sign is specified the string must be enclosed in quotes. Alphanumeric strings are left-justified and blank-filled to a length of 128. Numeric strings are right-justified and zero-filled to a length of 128. Comparison is done on a character-by-character basis beginning with the leftmost character.

## ELSE Expressions

The ELSE expression can be used only in conjunction with the IF expression. The ELSE expression is executed when the IF expression is false. The ELSE expression must begin on the line following the IF expression. The format is as follows:

```
IF conditional-parameter  
ELSE procedure statement
```

## BEGINIF, ENDIF, BEGINELSE, and ENDELSE Statements

It is often desirable to condition several procedure statements on an IF condition. The BEGINIF, ENDIF, BEGINELSE and ENDELSE statements provide a straightforward way of accomplishing this task. The BEGINIF statement marks the beginning of a group of procedure statements to be executed when an IF condition is true.

```
IF NOT DATA-PAY01.DATA  
  BEGINIF  
  BLDKSAM PAY01.DATA,PAY01K,1,5;REC=-80  
  RUN PAYFILE.PUB  
  ENDIF
```

portion of a procedure file

The BEGINELSE statement marks the beginning of a group of procedure statements to be executed when an IF condition is false. The ENDELSE statement marks the ending of the group. For example:

```
IF DATA-PAY01.DATA RUN PAYFILE.PUB
BEGINELSE
BLDKSAM PAY01.DATA,PAY01K,1,5;REC=-80
RUN PAYFILE.PUB
ENDELSE
```

portion of a procedure file

## RESETPROC Statement

The process monitor allows 255 levels of nesting. The nesting level may be reset by using the RESETPROC statement. When the procedure called by the RESETPROC statement completes, control is not returned to the procedure calling the RESETPROC statement. The procedure called by RESETPROC is considered a first level procedure. The RESETPROC format is:

```
RESETPROC [procname parm1,parm2...parmn
@]
```

## CANCEL Statement

The cancel statement cancels the user's procedure. The format is:

```
CANCEL
```

## RETURN Statement

If a RETURN statement is executed in a first level procedure, control is returned to the user. If a RETURN statement is executed in a nested procedure, control is returned to the calling procedure. The format is:

```
RETURN [@]
```

## **GOTO and TAG Statements**

The GOTO and TAG statements allow a programmer to use branching within a procedure. A GOTO statement causes the procedure to branch to the statement following the GOTO's corresponding TAG statement.

```
GOTO label
```

```
...
```

```
TAG label
```

The BLDMENU facility creates and updates both FIXED FORMAT and FREE FORMAT menus.

A FIXED FORMAT menu consists of twenty four menu items, numbered 1 through 24, with twelve items in each column.

A FREE FORMAT menu consists of up to twenty four menu items, numbered 1 through 24, with the layout (display lines 3 through 20) defined by the programmer.

To display a menu, the user executes the PROCMON MENU command (page 3-41) or provides the menu name at PROCMON initialization time. To terminate a menu display, the user executes the PROCMON MENUOFF command (page 3-42) or enters a 0 (zero) instead of an item number.

A menu consists of two files: a *command* file and a *display text* file.

The *command* file defines the procedure file or command to be used as input when the user selects an item number. The command file contains a header (always the first line of the file) and a body (the remainder of the file). The header consists of the menu name (maximum of six positions), followed by a "##,2". The body contains up to twenty-four procedure statements. Each procedure statement begins on a new line. The statement consists of the item number in the first four positions, followed by a blank and the procedure name or command.

The *display text* file defines the item description to be displayed along with each item number on the menu. It contains a header (always the first line of the file) and a body (the remainder of the file). The header consists of the menu name (maximum of six positions), followed by a "DT,1". The body contains up to twenty-four item descriptions. Each item description must begin on a new line. The description consists of the task number in the first four positions, followed by a blank and the item description.

## CREATING FIXED FORMAT MENUS

To create a FIXED FORMAT menu, execute the PROCMON BLDMENU command (page 3-8). The menu name should not be more than six positions in length.

EXAMPLE: BLDMENU PAYROL

## Creating, Updating, and Executing Menus

PROCMON will display a screen of twenty four item numbers and eight function key labels.

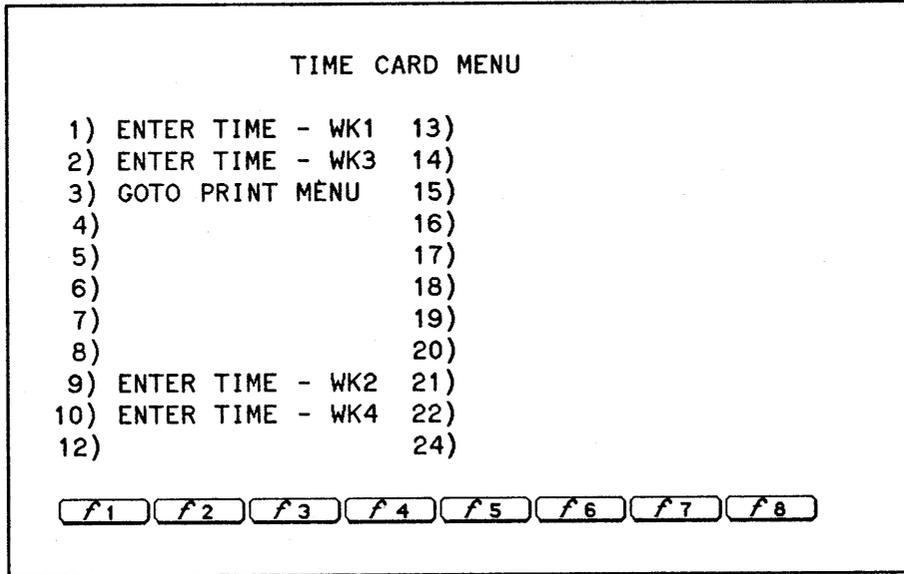
1)	13)
2)	14)
3)	15)
4)	16)
5)	17)
6)	18)
7)	19)
8)	20)
9)	21)
10)	22)
11)	23)
12)	24)

f1 f2 f3 f4 f5 f6 f7 f8

Enter the item description (text seen by user) to the right of the corresponding item number. **TAB**, **RETURN** and cursor control keys can be used to position the cursor for input. After entering the item descriptions, move the cursor to the function key labels and enter a brief item description for each. The eight function keys correspond to the first eight item numbers (1-8). Both item descriptions and function labels are optional.

Once the display is complete, press **ENTER**. The cursor will automatically move to the lower left portion of the display. PROCMON will prompt for the procedure statement (procedure name or command) associated with each specified item description. When the item number appears, press **ENTER**, then enter a procedure name or command, press **ENTER**. Continue until all item numbers have been defined. If an item number does not contain a description, PROCMON will not prompt the programmer for a procedure statement.

After the the last item number has been completed, PROCMON will clear the display and terminate the BLDMENU facility. The user can then call the newly created menu.



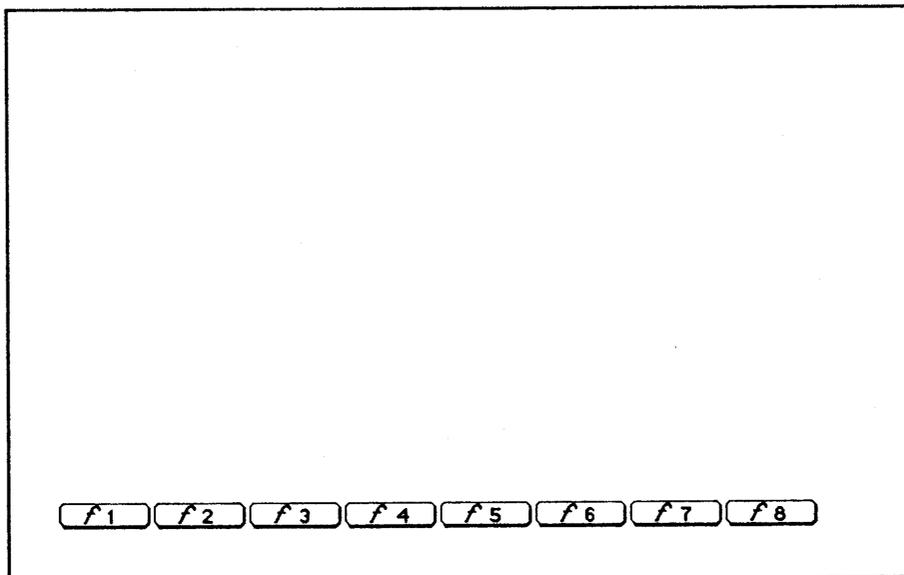
fixed format menu

## CREATING FREE FORMAT MENUS

To create a FREE FORMAT menu, execute the PROCMON BLDMENU command (page 3-8). The menu name should not be more than six positions in length.

EXAMPLE: BLDMENU PAYROL ,FREEFORM

PROCMON will display a blank screen and eight function key labels.



Enter the item number and description anywhere on the screen. **TAB**, **RETURN** and cursor control keys can be used to position the cursor for input. After entering the item numbers (1-24) and descriptions, move the cursor to the function key labels and enter a brief item description for each. The

## Creating, Updating, and Executing Menus

eight function keys correspond to the first eight item numbers (1-8). Both item descriptions and function labels are optional.

Once the display is complete, press **ENTER**. The cursor will automatically move to the lower left portion of the display. PROCMON will prompt for the procedure statement (procedure name or command) associated with each specified item description. When the item number appears, press **ENTER**, then enter a procedure name or command, press **ENTER**. Continue until all item numbers have been defined.

After the the last item number has been completed, PROCMON will clear the display and terminate the BLDMENU facility. The user can then call the newly created menu.

```

                                TIME CARD MENU

1) ENTER TIME - WK1   9) ENTER TIME - WK2
2) ENTER TIME - WK3  10) ENTER TIME - WK4

                                3) GOTO PRINT MENU

f1 f2 f3 f4 f5 f6 f7 f8

```

free format menu

## UPDATING MENUS

To update an existing menu, execute the PROCMON BLDMENU command. Specify the name of the menu to be modified. PROCMON will display the existing menu.

Modify the item and function key label descriptions. **TAB**, **RETURN** and cursor control keys can be used to position the cursor for input.

Once the display is complete, press **ENTER**. The cursor will automatically move to the lower left portion of the display. PROCMON will prompt for the procedure statement (procedure name or command) associated with each specified item description. When the item number appears, press **ENTER**, then enter a procedure name or command, press **ENTER**. Continue until all item numbers have been defined.

After the the last item number has been completed, PROCMON will clear the display and terminate the BLDMENU facility. The user can then call the modified menu.

## **FILE IPL=**

A file equation can be used to execute a procedure file during PROCMON initialization. The formal file designator is:

**FILE IPL=***procfile*[.group[.account]]

The file equation can be set in the user's log-on UDC. When the user logs-on, his/her personal procedure file is automatically executed.

## **FILE MENU=**

A file equation can be used to display a menu immediately following PROCMON initialization. The formal file designator is:

**FILE MENU=***menuname*[.group[.account]]

The file equation can be set in the user's log-on UDC. When the user logs-on, his/her personal menu is automatically displayed.



# CREATE, ACTIVATE, AND KILL

APPENDIX

A

**CREATE** - creates a process. (PROCMON command page 3-17)

Loads the program into virtual memory, creates the process, initializes its data stack and returns the Process Identification Number (PIN). The program is then ready to be activated.

**ACTIVATE** - activates a process. (PROCMON command page 3-2)

Activates a process previously created or suspended. Once activated, the program runs until it is suspended (using the RPG SUSP operation) or is deleted (using the PROCMON KILL command) or goes to end of job.

**KILL** - deletes a process. (PROCMON command page 3-26)

Deletes a process previously created.

**SUSP** - suspends a program. (RPG operation)

The RPG SUSP operation suspends a program at a specific point in execution, posts the local data area (LDA) and the user switches (U1 to U8), clears the screen and releases the terminal.

When the suspended program is re-activated, via the ACTIVATE command, execution begins at the instruction following the SUSP operation.

To use the operation, enter SUSP in columns 28 through 31 of the RPG calculation specification. Conditioning indicators can be placed in columns 7 through 17. All other columns must be blank.

## POINTS TO REMEMBER WHEN IMPLEMENTING CREATE, ACTIVATE & SUSP

- 1) The SUSP operation should only be called by a program using random file processing. Record pointers are not reset when a program is suspended. If a file is being processed sequentially when execution is suspended, the program will begin with the next sequential record (rather than the first record in the file) when execution continues.
- 2) If file locking is active, care should be taken to ensure that files are not left in a locked state when program execution is suspended. Files should be unlocked prior to the SUSP operation, either by forcing a write to the file or by using the UNLCK operation.
- 3) 16 processes are permitted per session.



**A**

ACTIVATE, 4-1  
ACTIVE, 6-1  
Additional features, 8-1

**B**

BLDDFU, 4-3  
BLDKSAM, 4-4  
BLDMENU facility, 7-4  
BLDMENU, 4-7  
BYE, 4-8

**C**

Calling a Procedure from another Procedure, 2-3  
CANCEL, 4-10  
CLEARACTIVE, 4-11  
Condition within procedures, 6-1  
Continuation character, 2-1  
CONTINUE, 4-12  
Creating a Procedure, 2-1  
CREATE, 4-17  
Cursor, 2-3

**D**

DATE, 4-18  
Displaying a menu, 7-1

**E**

EDIT/3000, 2-1  
Example, creating fixed format menus, 7-1  
Example, creating free format menus, 7-3  
Executing a Procedure, 2-2

**F**

File equations, 2-2  
Fixed format menus, 7-1  
Free format menus, 7-1

## Index

### I

IF expressions, 6-1  
IPL file, 8-1

### K

Keying in the Procedure Name, 2-2  
KILL, 4-34

### L

Line length, 2-1  
Literals, 2-4  
LISTFILE, 4-36  
LOCAL, 4-38

### M

menus, 7-1  
MENU file, 8-1  
MENU, 4-40  
MENUOFF, 4-41  
MSG, 4-42

### O

OFF, 4-44  
OPERATOR, 4-45  
OVERVIEW, -2

### P

Passing Parameters to Procedures, 2-4  
PATH, 4-46  
PAUSE, 4-47  
PFILE command, 2-2  
PFILE, 4-48  
Procedure execution, 2-2  
Procedure variables, 2-4  
Procedures, 1-1  
PREP, 4-49  
PROCCONFIG, 4-50  
PROCMON, 1-1  
PROMPT, 4-54

**R**

RENAME, 4-56  
RESETPROC, 4-57  
RETURN, 4-59  
RFILE command, 2-2  
RFILE, 4-61  
RPG, 4-62  
RUN, 4-63

**S**

Selecting an option from a Menu, 2-3  
SECURITY, 4-64  
SETPASS, 4-65  
SHOWACTIVE, 4-67  
SHOWMENU, 4-68  
Substitution expression formats, 5-1  
Substitution expressions, 5-1  
Substitutions allowed, 5-1  
SWITCH, 4-69

**T**

TAG, 4-70  
TASK, 4-71  
TDP/3000, 2-1

**U**

Updating menus, 7-4

**W**

WAIT, 4-74  
WSID, 4-77

**SPECIAL CHARACTERS**

?GROUP?, 5-3  
?PROC?, 5-3  
?TIME?, 5-3



# READER COMMENT SHEET

HP 3000 Computer Systems

PROCMON/3000  
Reference Manual

99941-90001 November 1986

We welcome your evaluation of this manual. Your comments and suggestions help us to improve our publications. Please explain your answers under Comments, below, and use additional pages if necessary.

Is this manual technically accurate?

Yes  No

Are the concepts and wording easy to understand?

Yes  No

Is the format of this manual convenient in size, arrangement, and readability?

Yes  No

Comments:

This form requires no postage stamp if mailed in the U.S. For locations outside the U.S., your local HP representative will ensure that your comments are forwarded.

FROM:

Date \_\_\_\_\_

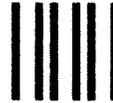
Name \_\_\_\_\_

Company \_\_\_\_\_

Address \_\_\_\_\_  
\_\_\_\_\_  
\_\_\_\_\_

FOLD

FOLD



NO POSTAGE  
NECESSARY  
IF MAILED  
IN THE  
UNITED STATES

**BUSINESS REPLY MAIL**

FIRST CLASS PERMIT NO. 1070 Cupertino, California

POSTAGE WILL BE PAID BY ADDRESSEE

Publications Manager  
Hewlett-Packard Company  
Computer Systems Division  
19447 Pruneridge Avenue  
Cupertino, CA 95014

FOLD

FOLD

