MPE V to MPE XL: Getting Started

900 Series HP 3000 Computer Systems

HEWLETT PACKARD

HP Part No. 30367-90002
Printed in U.S.A. 19890901

E00989
DRAFT 2/11/100 10:35

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 date on the title page and back cover of the manual changes only when a new edition is published. When an edition is reprinted, all the prior updates to the edition are incorporated. No information is incorporated into a reprinting unless it appears as a prior update.

Preliminary Edition	September 1987
First Edition	November 1987
Second Edition	July 1988
Third Edition	September 1989

Preface

What Is This Book?

MPE V to MPE XL: Getting Started is a self-paced training tool designed to familiarize experienced MPE users with the new features and commands of the MPE XL operating system. It also covers some of the MPE features that have been changed or deleted.

There is also a *Mentor's Guide* in the back of this binder for the benefit of users who wish the assistance of a mentor.

Who Should Use This Book?

MPE V to MPE XL: Getting Started was written for experienced users of the HP 3000 who are moving to a 900 Series HP 3000. It is interfaced for those users who regularly use MPE commands to interface with the HP 3000.

If you are a user who only logs on to the HP 3000 to run applications, then you do not need this training. (See Chapter 1 "Introduction" for an explanation of appropriate lessons for the end user, if you are unsure about going through this training.)

The basic groups of users for which this book has been written are programmers, system managers, system operators, and end users. Not all users will need to complete all of the chapters in order to use MPE XL successfully. Chapter 1 "Introduction" recommends the appropriate learning path for each type of user mentioned.

MPE V to MPE XL: Getting Started is prerequisite training for the following courses:

- *Moving From MPE V to MPE XL: System Operator* (HP 31117)
- *Moving From MPE V to MPE XL: System Manager* (HP 31110)
- *Moving From MPE V to MPE XL: Application Programmer* (HP 31114)

How Should This Book Be Used?

Mentored Versus Unmentored

This book can be used independently or with the assistance of a mentor. The Introduction on the *Mentor's Guide* explains who might need a mentor and what a mentor's responsibilities would be.

Organization

The information in this training has been organized into chapters. Each chapter is broken into individual lessons. Each lesson has realistic examples and activities that can be tried on the system. The activities in each lesson, called "Exercises", consist of questions to answer and things to try on the system in order to reinforce important concepts presented in the lesson. The answers to the exercises are at the end of each chapter.

The Appendix provides you with reference charts that list the unsupported, modified, and new commands, as well as the MPE utilities. These charts may be used for quick, at-a-glance reference for any of the commands and utilities mentioned in this course. The fold-out design is intended to allow you to refer to the charts, without removing them, while doing the lessons.

Procedure

- Proceed through the book in sequential fashion, beginning with Chapter 1, which describes the material that is most useful for each type of user. Decide which chapters are most appropriate for you.
- Log on to your 900 Series HP 3000.
- After reading each lesson introduction and trying out the examples, do the exercises in that lesson. (The correct answers to the exercises are found at the end of the chapter.)
- Not all lessons in a chapter may be appropriate for you-feel free to skip any lesson that you consider inappropriate according to the topic and/or the level of difficulty.

What Is Needed to Complete This Training?

You will need the following in order to complete the exercises in each lesson:

- Exclusive access to an account on a 900 Series HP 3000 (MPE XL Version A.01.01 or later).
- `A LABS` group, on addition to a home group.
- Basic user capabilities, plus `PH` capability if you need to go through Chapter 7.
- Working knowledge of an editor, such as MPE's EDIT/3000. If you wish to use EDIT/3000 and do not know how, *before starting this book* refer to Chapter2 of the *HP 3000 Guide for the New User* (32033-90009).

The subjects addressed in this training relate to subjects found in the following manuals, which your system manager should have:

- *MPE XL Commands Reference Manual* (32650-90003)
- *Introduction to MPE XL for MPE V System Administrators* (30367-90003)
- *Introduction to MPE XL for MPE V Programmers* (30367-90005)
- *MPE XL Volume Management Reference Manual* (32650-90045)
- *System Utilities Reference Manual* (32650-90081)

1 Introduction

What Has Changed?

MPE XL is very much like MPE V. You can continue to use most of the MPE V features and commands in MPE XL. However, you may wish to take advantage of the many new, exciting features of MPE XL.

Running programs and applications has become more flexible. New features have been added that give users more power than ever before. You can now do such things as change your prompt character to any character(s) you want, by using system variables. Programmers will like the new expression evaluation capabilities, too. If you create User Defined Commands (UDCs), you will appreciate the enhancements and a new kind of User Command called "Command Files".

Many of the changes to MPE are documented in the Appendix at the end of the book. Table 1 "Unsupported Commands" lists the MPE V commands that are not supported in MPE XL. Table 2 "Modified Commands" documents those MPE V commands that have been enhanced in MPE XL. The new commands in MPE XL can be found in Table 3 "New Commands". Table 4 "Utilities" documents new, modified, and unchanged MPE utilities. For more detailed information about anything in the reference charts, refer to the manuals listed in the Preface.

Which Lessons Do You Need?

Programmers

All chapters and Table 3 “New Commands” in the Appendix will be of interest to programmers. The changes that affect programming are taught in-depth in the classroom course *Moving From MPE V to MPE XL: Application Programmer* (HP 31114).

System Managers

Chapters 2-4 will be applicable to those of you who are system managers. Depending upon your level of expertise, you may also wish to take advantage of the information found in Chapter 5 “Variables”. All of the tables in the Appendix will be of particular use. Study Table 4 “Utilities”; find the account management commands in Table 2 “Modified Commands” and Table 3 “New Commands”. The changes that affect system management are taught in-depth in the classroom course *Moving From MPE V to MPE XL: System Manager* (HP 31110).

System Operators

If you are a system operator, concentrate on Chapters 2 and 3. Depending on your experience with creating UDCs, you may also want to study the new features of UDCs described in Chapter 4. The rest of the changes that affect system operators are taught in-depth in the classroom course *Moving From MPE V to MPE XL: System Operator* (HP 31117).

End Users

If you are an end user, concentrate on Chapters 2 and 3. Depending on your experience with creating UDCs, you may also want to learn about the new features of UDCs in Chapter 4. If you have had some experience using variables, you may wish to study Chapter 5 as well.

What Is This Training About?

Chapter 2 teaches you some of the enhancements that were made to the Command Interpreter. One such enhancement is a new feature called "implied `:RUN`" that lets you run programs without typing the command `:RUN`. Enhancements have also been made to allow you to reexecute commands previously entered. The `:REDO` command has been modified and the new commands `:LISTREDO` and `:DO` add flexibility to the reexecution of commands. Finally, the syntax of the new MPE XL commands has been changed to provide more flexibility.

In Chapter 3 you will learn about file manipulation. New commands are introduced that will make printing files (on printers and on your screen), copying files, and changing groups much easier.

Chapter 4 guides you through the enhancements made to user commands. UDCs have been modified to make them easier to create and maintain. A new type of User Command has been added—Command Files.

Chapter 5 introduces you to variables in MPE XL. You can now create and use variables. You can make use of many system variables that are now available. Some of the system variables are user modifiable, such as the variable that defines the prompt character. All variables can be used in User Commands, and the system variables can also be used in programs.

Chapter 6 teaches you about expression evaluation in MPE XL. The new command `:CALC` allows online calculation of numeric and alpha expressions.

In Chapter 7 you learn about the Command Interpreter (CI) program. It is now possible to run the CI from within other programs. This is handy for programmers who want to test programs or for users who wish to test UDCs while creating them.

2

Command Interpreter Enhancements

The MPE XL operating system offers a variety of improvements to some of the MPE V functions. Some of these enhancements provide you with convenient shortcuts, but are not required for the successful use of MPE XL.

- Lessons 1 and 2 in Chapter 2 introduce a new feature called "implied`:RUN`" that allows you to run programs without typing the`:RUN` command. These lessons will be useful to all users.
- Lessons 3 through 7 teach some new commands that will make working online much easier. The modified`:REDO` command and the new`:DO` command make reexecution of command lines more efficient; the new`:LISTREDO` command allows you to list the command lines you have entered during your session. These lessons will be useful for all users.
- Lesson 8 introduces the new, more flexible syntax for the new MPE XL commands. If you are unfamiliar with the syntax rules for MPE V, you may want to skip this lesson.

Lesson 1: Implied :RUN

```
             MPE V/E                          MPE XL

   +------------------------+    +------------------------+
   |                        |    |         Either         |
   |    :RUN SORT.PUB.SYS   |    |    :RUN SORT.PUB.SYS   |
   |                        |    |           or           |
   +------------------------+    |      :SORT.PUB.SYS     |
                                 +------------------------+
```

The MPE XL operating system allows you to run a program without specifying the `:RUN` command. This is called the implied `:RUN` feature.

Implied `:RUN` has not replaced the `:RUN` command—it is just a useful shortcut for invoking programs.

Exercises

1. How would you run the `QUERY` program in MPE V/E and MPE XL? (HINT: Like `SORT`, the `QUERY` program is in `PUB.SYS`)

   a. MPE V/E:

   b. MPE XL:

2. On your computer, run the `QUERY` program using the implied `:RUN` feature. What did you enter to do this? (Exit the program by entering >`exit`.)

Note The answers to the exercises are at the end of Chapter 2.

Lesson 2: Implied :RUN Parameters

Programmers who are accustomed to specifying several parameters when using the:`RUN` command will need to note the following:

The only parameters of the `:RUN` command that can be used with implied `:RUN` are `;INFO=` and `;PARM=`. Neither parameter is required, of course.

Examples of acceptable implied`:RUN` command lines are:

```
:TESTPROG;INFO="USER1"

:TESTPRG2;PARM=1

:TESTPRG3;INFO="USER1";PARM=2

:TESTPRG4
```

Additional Information

A more detailed discussion of these parameters is in the *MPE XL Commands Reference Manual* (32650-90003), Chapter 1.

Exercises

3. Which of the following command lines are valid with the implied `:RUN` feature?

a. `:TEST;STDIN=*INFILE;STDLIST=RESULT`

b. `:TESTME;DEBUG`

c. `:MYPROG;INFO="Do this";PARM=3`

d. `:MYPROG;INFO="Do this";PRI=AM`

e. `:MYPROG;PARM=2;INFO="Do this"`

4. True (T) or False (F):

a. _ Parameters cannot be used with the implied `:RUN` feature.

b. _ The ;INFO=" and ;PARM=" parameters of the `:RUN` command are the only ones that can be used with implied `:RUN`.

c. _ The implied :RUN feature replaces the :RUN command.

Lesson 3: Using The Command History

Each time you enter a command, MPE XL saves that command in a list called the command line history stack. The command history, along with the new commands `:LISTREDO` and `:DO`, and the modified `:REDO` command, allow you to keep track of and reexecute earlier commands without having to completely retype them. By default, the command history keeps track of the last 20 commands entered. Each command is added to the bottom of the list as it is entered.

Exercise

5. The command line history stack is

a. a list of the system commands available to the user.

b. a list of the last 20 commands entered by a user in a session.

c. a list of all the commands entered by a user in a session.

Note The answers to the exercises are at the end of Chapter 2.

Lesson 4: :LISTREDO

You can use the new :LISTREDO command to see your command line history stack on your terminal screen.

To see your command line history stack, enter `listredo` at the MPE prompt (:). (Try it now.)

Notice that the :LISTREDO command appears at the bottom of the command history display as the most recent command entered.

```
:HELLO USER.MYACCT

Command Entered                History Stack   Command Line #
                             +---------------+
:LISTF           ---->       | LISTF         |       1
                             +---------------+---------------
:SORT.PUB.SYS    ---->       | SORT.PUB.SYS  |       2
                             +---------------+---------------
:SHOWME          ---->       | SHOWME        |       3
                             +---------------+---------------
:RUN MYPROG      ---->       | RUN MYPROG    |       4
                             +---------------+---------------
:EDITOR          ---->       | EDITOR        |       5
                             +---------------+---------------
:LISTREDO        ---->       | LISTREDO      |       6
                             +---------------+---------------
```

When the history stack has reached its maximum size, the "top" command on the list is deleted as each new command entered is added to the bottom of the list. Consecutive numbering of the commands continues until the session ends.

Exercises

6. Enter several simple commands, such as:LISTF and:SHOWME. After every couple of commands, display the contents of your command line history stack. Try entering a misspelled command. How is it recorded in the history stack?

7. Given a maximum history stack size of 20, what range of numbers will be displayed by:LISTREDO if :LISTREDO is command number 45?

Lesson 5: :LISTREDO Parameters

You can use the `:LISTREDO` parameters to display the command line history stack in three different ways.

The parameter `;ABS` is the default and lists your command line history stack by absolute number—the first command entered is 1, the second is 2, and so on.

If you specify the parameter `;REL`, the commands are listed in order relative to the current command line. The list will use negative numbers: the command most recently entered is shown as -1, the next most recent as -2, and so on.

Finally, you can display an unnumbered history stack with the parameter `;UNN`.

```
      ;abs
+-----------------+
| :LISTREDO;ABS   |
| 1) listf        |
| 2) showme       |
| 3) run myprog   |
| 4) editor       |         ;unn
| 5) sort.pub.sys |   +-----------------+
| 6) LISTREDO;ABS |   | :LISTREDO;UNN   |
+-----------------+   | listf           |
                      | showme          |
      ;rel            | run myprog      |
+-----------------+   | editor          |
| :LISTREDO;REL   |   | sort.pub.sys    |
|-6) listf        |   | LISTREDO;UNN    |
|-5) showme       |   +-----------------+
|-4) run myprog   |
|-3) editor       |
|-2) sort.pub.sys |
|-1) LISTREDO;REL |
+-----------------+
```

Exercise

8. Try listing your command line history stack with the `;REL`, `;ABS`, and `;UNN` parameters.

Lesson 6: :DO and :REDO

You can reexecute any of the commands in your history stack by using the familiar MPE V/E `:REDO` command or the new MPE XL `:DO` command.

```
+-------------------+     +-------------------+
|                   |     |                   |
| :REDO 3 [RETURN]  |     |  :DO 3 [RETURN]   |
| [RETURN]          |     |                   |
+-------------------+     +-------------------+
```

The advantage of using `:DO` instead of `:REDO` is that `:DO` immediately executes the command after you press Return. `:REDO` requires that you press Return twice before the command reexecutes.

Same-Line Editing

Both `:REDO` and `:DO` allow you to perform "same-line" editing before a command is reexecuted. That is, the syntax of these commands allows you to specify editorial changes on the same line as the command itself. `:REDO` also allows next-line editing in MPE XL, as it did in MPE V/E.

Refer to the *MPE XL Commands Reference Manual* (32650-90003), `:DO` or `:REDO`, for an explanation of using the "same-line" editing feature.

Both `:DO` and `:REDO` may be used with or without specifying a command line number from the history stack listing. When a line number is not specified, the command most recently issued is reexecuted.

```
+-------------------------------------------+
|          :LISTREDO                        |
|           1) listf                        |
|           2) showme                       |
|           3) showjob                      |
|           4) listredo                     |
|                                           |
| :DO or \     Command #4, LISTREDO         |
| :REDO  /     would be reexecuted          |
|                                           |
| :DO 2 or \   Command #2, SHOWME           |
| :REDO 2  /   would be reexecuted          |
+------------------------------------------+
```

Exercise

9. Display your command line history stack on your screen. Use `:DO` and `:REDO` with and without specifying line numbers to reexecute any command in your command history.

Lesson 7: Numbering with :DO and :REDO

`:DO` and `:REDO` allow you to specify command line numbers from the history stack using either absolute or relative numbers regardless of how the numbering on your history stack is displayed.

```
+-------------------------------------------+
|          :LISTREDO;REL                    |
|          -4) showme                       |
|          -3) showjob                      |
|          -2) setcatalog                   |
|          -1) LISTREDO; REL                |
|                                           |
| :DO -3 \                                  |
|    or   -->  :SHOWJOB would be reexecuted |
| :DO 2  /                                  |
+-------------------------------------------+
```

Exercise

10. Display your command line history stack on your screen and specify relative numbering. Use `:DO` with an absolute line number to reexecute any command in your history stack.

Lesson 8: MPE XL Syntax

In MPE XL, you can enter commands in your accustomed manner. Long-standing rules for syntax still apply.

If you use complex command syntax in MPE V/E, note that the parameters of the *new* commands in MPE XL have the option of being entered in a more flexible manner. Parameters for new commands are no longer limited to being *only* a keyword or *only* a positional parameter.

Note If you are not familiar with MPE syntax rules, you may wish to skip this lesson.

Keyword Parameters

Commands—old as well as new—can be entered in MPE XL by using only keyword specifications, in the same manner that you were accustomed to in MPE V/E.

Example:

```
:file lprint;dev=lp;env=envfile.pub
```

The syntax definition of new commands in MPE XL, however, makes the `;keyword=` specifications optional. Refer to Table 3 "New Commands" in the Appendix for more examples of the new syntax definition.

Example:

Syntax: PRINT [FILE=] *filename*
[[;OUT=] *outfile*]
[[;START=] *m*]
[[;END=] *n*]

Using only keyword specifications:

```
:print file=myfile;out=*printer;start=5
```

Using the optional positional specifications:

```
:print myfile,*printer,5
```

Positional Parameters

If the keywords are omitted, the familiar rules regarding positional parameters come into effect: commas separate parameters and hold the place of omitted parameters; sequence becomes important.

Example:

```
:print myfile,,5
          |  | |
 FILE=----+  |
             | |
;OUT=--------+ |
               |
;START=--------+
```

Combining Keyword and Positional Parameters

And, as always, you can combine keyword and positional parameters. Once a keyword has been specified, you must continue using keywords in the rest of the command line.

Example:

Acceptable: `:print myfile;out=*printer;end=10`

Unacceptable: `:print myfile;out=*printer,5,10`

Note The new syntactical rule applies to the commands that are *new* to the MPE XL system. By and large, "modified" and "unchanged" commands are restricted to MPE V/E syntax rules.

Additional Information

For more information about the syntax for new MPE XL commands, refer to Chapter 1 of the *MPE XL Commands Reference Manual* (32650-90003).

Exercises

For each command line listed below, indicate if it is executable in MPE XL or not, based on what you know about the new syntax. For each one that is not executable in MPE XL, explain why.

HINT: `:COPY`, `:PRINT`, `:LINK`, `:INPUT`, and `:LISTREDO` are new commands. `:HELLO` and `:RUN` are modified commands.

11. `:COPY FROM=fred;TO=wilma;NO`

12. `:PRINT FILE=MYFILE,*PRINTER,5,8`

13. `:LINK file1;TO=newfile,rlfile2;SHOW`

14. `:INPUT myname;WAIT=30;PROMPT=**`

15. `:HELLO barney.rubble,,;TIME=60`

16. `:LISTREDO OUT=*lp,-7,-3;UNN`

17. `:COPY FROM=fred,wilma`

18. `:RUN myprog,,,;DEBUG`

19. `:PRINT beatles,*lp;START=4;END=20`

Answers to Exercises

1. a. :`run query.pub.sys`
   b. `:run query.pub.sys` or:`query.pub.sys`
2. :`query.pub.sys`
3. c and e
4. a. F
   b. T
   c. F (Both are available in MPE XL.)
5. b
6. :`listf`
   ```
   ...
   : showme
   ...
   : listredo
   ...
   5) LISTF
   6) SHOWME
   7) LISTREDO
   :
   ```
   (Etc.)

   If you misspell a command, the misspelled command is added to the history stack*as is*, in its misspelled state.
7. Commands 26-45 will be displayed.
8. `:listredo;rel`
   :`listredo;abs` or`:listredo`
   :`listredo;unn`
9. `:listredo`
   :`redo`
   :`redo 2` (or any other line number)
   :`do`
   :`do 3` (or any other line number)

10. `:listredo;rel`

    ```
    -4) SORT.PUB.SYS
    -3) LISTF
    -2) SHOWJOB
    -1) SHOWCATALOG
    ```

    `:do 2` `LISTF` will execute.

11. Executable.
12. Not executable on MPE XL. Positional parameter cannot be specified after a keyword parameter.

    ```
    :  PRINT;FILE=MYFILE,*PRINTER,5,8
                        ^
    ```

13. Not executable on MPE XL. Positional parameter cannot be specified after a keyword parameter.

    ```
    :  LINK FILE1;TO=NEWFILE,RLFILE2;SHOW
                            ^
    ```

14. Executable.
15. Not executable. Syntactical changes only apply to new commands. `:HELLO` is a modified command.
16. Not executable. Positional parameters cannot be specified after a keyword parameter.

    ```
    :  LISTREDO OUT=*LP,-7,-3;UNN
                       ^
    ```

17. Not executable. Positional parameters cannot be specified after a keyword parameter.

    ```
    :  COPY FROM=FRED,WILMA
                     ^
    ```

18. Not executable. `:RUN` is not a new command.
19. Executable.

3

Working With Files

This chapter discusses three new commands. `:PRINT` and `:COPY` will make printing and copying files easier for you than they were on MPE V/E. The new command `:CHGROUP` provides an easier method for getting to other groups within your account.

- Lessons 1 through 3 teach you the new `:PRINT` command. You will learn how to print files on your terminal screen and on a printer. This new command allows you to print either a whole file or a portion of a file. These lessons are useful for all users.
- Lesson 4 introduces the new`:CHGROUP` command. It allows you to get to another group without logging off and on again. This lesson is useful if your job requires you to change groups.
- Lessons 5 through 7 teach you the new`:COPY` command. This command provides an easy method for copying files, even from other groups and accounts. These lessons are useful to all users.

Lesson 1: :PRINT

The new command `:PRINT` prints the contents of a file on your terminal screen.

```
:print file=myfile
This file is called
MYFILE.  It is only
three lines long.
:
```

Exercise

Create a ten-line file with your text editor and call it `LUCY`.

1. Display the contents of the file `LUCY` on your terminal screen.

Note For the rest of the exercises in this chapter, you will be using the file you create in this exercise. DO NOT PURGE the file LUCY" before you finish Chapter 3.

Lesson 2: Printing Portions of Files

If you want to display only a portion of a file on your terminal screen, you can specify the lines you want with the `;START=` and `;END=` parameters.

To print lines 5 through 20 of the file `MYFILE` to the terminal screen:

```
:print file=myfile;start=5;end=20
```

or

```
:print myfile,,5,20
```

Refer to Table 3, "New Commands", in the Appendix for the syntax of the new `:PRINT` command.

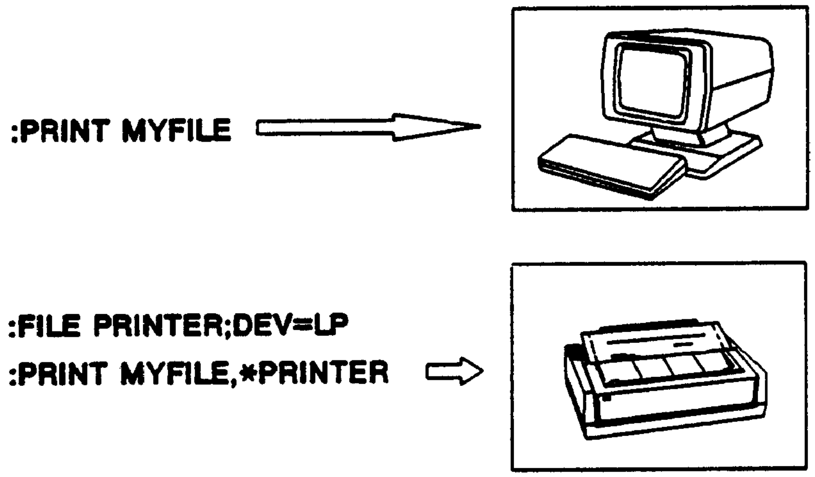
Exercises

2. Display lines 2 through 9 of the file `LUCY` on your terminal screen.

3. Display line 10 of the file `LUCY`.

4. Display line 5 of the file `LUCY`.

Lesson 3: Printing a File on a Line Printer



The contents of a file may also be sent to a printer with the `:PRINT` command.

For example,

```
:file printer;dev=lp
and:print file=myfile;out=*printer
or :print myfile,*printer
```

will print the file `MYFILE` to a printer as specified by the file equation.

Exercises

Given the file equation `:FILE PRINTER;DEV=LP`, fill in the blanks below with the command lines you would use in each case; verify by trying the command lines at your terminal (after entering an appropriate file equation).

5. Print the file `LUCY` to the printer.
   Command:

6. Print lines 6 through 9 of the file `LUCY` to the printer.
Command:

7. Enter the following command:
```
:print file=lucy;start=-3
```
What happened? Why?

Lesson 4: :CHGROUP

```
            MPE V/E                             MPE XL

   +----------------------------+     +---------------------+
   |                            |     |                     |
   |:HELLO USER.ACCOUNT,NEWGROUP|     |  :CHGROUP NEWGROUP  |
   |                            |     |                     |
   +----------------------------+     +---------------------+
```

The new command :CHGROUP gives you mobility to move from group to group within your account. However, it does not allow you to change to a group in another account.

Note The :CHGROUP command may only be useful to you if your work requires you to change groups.

To change from your current group to the group PAYROLL within your account:

```
:chgroup payroll
```

To change from your current group to a group called SALES that is password-protected:

```
:chgroup sales/password
```

To get into your home group, just enter :CHGROUP by itself. Entering the command :CHGROUP without specifying a group name will always put you into your home group. (Remember, you can use:SHOWME to find out what group you are in.)

```
   +------------------------------------------------------------+
   |                                                            |
   | Changing from your current group...                        |
   |                                                            |
   |    to another group           to another passworded group  |
   | +----------------------+    +-----------------------------+ |
   | | :CHGROUP NEWGROUP    |    | :CHGROUP NEWGROUP/PASSWORD  | |
   | +----------------------+    +-----------------------------+ |
|                                                              |
|                    to your home group                        |
|                    +-----------------+                       |
|                    |   :CHGROUP      |                       |
|                    +-----------------+                       |
+--------------------------------------------------------------+
```

Exercise

8. Change groups within your account; then get into your home group.

Lesson 5: :COPY

```
        MPE V/E                        MPE XL
   +-----------------+           +-----------------+
   |     :DSCOPY     |           |     :DSCOPY     |
   |     :FCOPY      |           |     :FCOPY      |
   |                 |           |     :COPY       |
   +-----------------+           +-----------------+
```

The new command `:COPY` copies one file to another.

Assume that you have a file named `MYFILE` that you want to copy, and that you want to name the copy `MYCOPY`.

```
   :copy from=myfile;to=mycopy
      or :copy myfile,mycopy
```

If a file named `MYCOPY` already exists, the prompt `"PURGE OLD?"` appears. A YES answer *overwrites* the old file. A NO answer terminates the copy process, leaving the original file `MYCOPY` intact.

Exercises

9. Use the file `LUCY` that you created earlier. Make a copy of the file `LUCY` and call it `RICKY`.

10. Make editing changes to the file `RICKY` so that it now looks different. Save it again as `RICKY`. Then`:COPY` the file `RICKY` to the file `LUCY`.

- **a.** Answer NO to the prompt and look at the file `LUCY`. What happened?
- **b.** Try copying again. Answer YES to the prompt and look at the file `LUCY`. What happened?
- **c.** What command did you use to look at the file `LUCY`?

Lesson 6: :COPY Parameters

When copying a source file to a target file you may specify one of three options:

- `;ASK` asks "`PURGE OLD targetfile.group. account?`" if the target file already exists. The `;ASK` option is the default for sessions.

  `:copy from=x;to=y;ask` or `:copy x,y;ask`

- `;YES` purges old targetfile.group.account *automatically*, if one exists. The `;YES` option is the default for jobs.

  `:copy from=x;to=y;yes` or `:copy x,y;yes`

- `;NO` terminates the copy process if a duplicate file name exists.

  `:copy from=x;to=y;no` or `:copy x,y;no`

Exercises

11. Again make editing changes to the file `RICKY` that you created earlier.

Copy `RICKY` to `LUCY` so that the system will terminate the copy process *without* giving you the prompt "`PURGE OLD?`".
Command:

12. Print the file `LUCY` to your screen to verify that the copy did not occur.
Command:

13. Copy `RICKY` to `LUCY` so that the system will perform the copy without giving you the prompt "`PURGE OLD?`".
Command:

14. Print the file `LUCY` to your screen to verify that the copy occurred.
Command:

Lesson 7: Copying From Other Groups & Accounts

Given proper file security, you can copy files from other groups within your account, and even from other accounts, with the `:COPY` command.

To copy a file named `INFORM` from the `PUB` group of the `MKTG` account and give it the name `MKTGINFO` in your own group and account:

```
:copy from=inform.pub.mktg;to=mktginfo
or:copy inform.pub.mktg, mktginfo
```

If you copy the file named `INFORM` from the `PUB` group of the `MKTG` account, *but do not give it a target file name*, the file is copied into your current group with the name `INFORM`.

Note You cannot use the`:COPY` command to copy a file *to* another group unless you have account manager (AM) capability. You cannot use`:COPY` to copy files *to* another account unless you have system manager (SM) capability.

Exercises

15. Check to be sure you have a group in your account called `LABS`. While still in your home group, copy `LUCY` into your `LABS` group. Get into your `LABS` group and verify that the copy was successful.

16. While in the `LABS` group, create a file called `RICARDO` and copy `RICARDO` back to your home group. Return to your home group and verify that the copy was successful.

17. How did you get from your group to the LABS group and back again?

Answers to Exercises

1. :`print file=lucy`
   or :`print lucy`
2. :`print file=lucy;start=2;end=9`
   or :`print lucy,,2,9`
3. :`print file=lucy;start=10;end=10`
   or :`print lucy,,10,10`
   or :`print file=lucy;start=-1` if the file is only 10 lines long.
4. :`print file=lucy;start=5;end=5`
   or :`print lucy,,5,5`
5. :`print file=lucy;out=*printer`
   or :`print lucy,*printer`
6. :`print file=lucy;end=9;start=6;out=*printer`
   or: `print lucy,*printer,6,9`
   (Note: in the first solution, the order of the parameters is unimportant.)
7. :`print file=lucy;start=-3`
   This command caused the last three lines of the file `LUCY` to print on the screen. The minus (-) specified before the number 3 caused this.
8. (example of a correct answer)
   :`chgroup newgroup`
   :`chgroup`
9. :`copy from=lucy;to=ricky`
   or: `copy lucy,ricky`
10. a. :`copy from=ricky;to=lucy`
    or :`copy ricky,lucy`
    PURGE OLD? `no`
    NO COPY WAS DONE (CIERR 9113)

    b. :`copy from=ricky;to=lucy`
    or: `copy ricky,lucy`
    PURGE OLD? `yes`
    :
    What happened? The file `RICKY` overwrote the file `LUCY`.

c. :`print file=lucy` or :`print lucy`

11. :`copy from=ricky;to=lucy;no`
    or :`copy ricky,lucy;no`
    NO COPY WAS DONE (CIERR 9113)

12. :`print file=lucy` or :`print lucy`

13. :`copy from=ricky;to=lucy;yes`
    or :`copy ricky,lucy;yes`

14. :`print file=lucy` or :`print lucy`

15.
```
:newgroup labs
:copy from=lucy.home;to=lucy.labs
:chgroup labs
:print file=lucy
```
    or
```
:newgroup labs
:copy lucy,lucy.labs
:chgroup labs
:print lucy
```

16.
```
:copy from=ricardo;to=ricardo.home
:chgroup
:print file=ricardo
```
    or
```
:copy ricardo,ricardo.home.account
:chgroup
:print ricardo
```

17.
```
:chgroup labs
:chgroup
```

4

User Commands

MPE XL provides you with a new type of User Command called Command Files. Along with the familiar User Defined Commands (UDCs), Command Files can be used to create, modify, and manipulate the MPE environment.

- Lesson 1 introduces you to Command Files, the new User Command in MPE XL.
- Lesson 2 makes a comparative study of both UDC files and Command Files and the benefits of having both available to you. If you are not familiar with the creation of UDC files and execution of UDCs, you may choose to skip Lesson 2 and continue with Lesson 3.
- Lesson 3 steps you through the process the MPE XL system uses in its attempt to execute whatever you type at the prompt. Lesson 4 introduces you to the new :XEQ command.
- Lesson 5 introduces you to enhanced features of the `:SETCATALOG` command that allow easy UDC file manipulation. Lesson 6 discusses the new `RECURSION` feature that allows one UDC to call upon any other UDC, making UDC file maintenance more flexible. Those of you who are not familiar with creating and maintaining UDC files may choose to skip these two lessons and the following two lessons, 7 and 8.
- Lesson 7 introduces new User Command `OPTION`s. Lesson 8 introduces the new MPE XL:`OPTION` command. These two lessons assume some prior knowledge of programming and of creating and maintaining UDCs.

Lesson 1: Creating Command Files

A Command File is a simple ASCII or binary data file that contains MPE commands, program file names, UDC names, and/or other Command File names. It is created by the user in a text editor. A Command File is executed like "implied:RUN", by entering the file name.

At its simplest, a Command File can contain a single MPE command. For example, instead of always having to enter the whole command:SHOWCATALOG, you could create a Command File called SC that simply contained the one word SHOWCATALOG. Then, every time you entered SC at the colon (:) prompt, SHOWCATALOG would execute. Many MPE V users use UDCs for this purpose; in MPE XL you can use either UDCs or Command Files.

Another example of a one-line Command File would be one that printed the last three lines of a particular file to the terminal screen. It would look like the following if you created it in :EDITOR:

```
        1   PRINT FILE=myfile;START=-3

   or

        1   PRINT myfile,,-3
```

If this example were to be saved under the file name PR, you would simply type PR at the MPE prompt to execute the file.

```
       :PR

        This is the file Myfile.
        This is a very short file.
        This is the last line of the file.
        :
```

The example below shows the Command File that you have named SH. It executes the Command File PR" and then executes the MPE command :SHOWTIME.

```
    1  PR
    2  showtime
```

To execute, you would enter:

```
:sh

 This is the file Myfile.
 It is a very short file.
 This is the last line of the file.
 WED, NOV18, 1987, 2:10 PM
 :
```

Exercises

1. Choose two MPE commands and create one Command File that executes both. Test the Command File.

2. Choose a third MPE command and create another Command File that executes it. Test this Command File.

3. Create a Command File that executes both of the Command Files from exercises 1 and 2. Test this Command File.

Lesson 2: Comparing User Commands

Note For this lesson, it is important that you be reminded of the distinction between the terms "UDC file" and "UDC". A *UDC file* is an MPE file that contains one or more UDCs separated by asterisks. In order to use the UDCs, the file must first be cataloged using the:`SETCATALOG` command.

A *UDC* is not an MPE file. It is contained within a file, and it consists of one or more MPE commands and/or User Commands. The UDC executes when its command header is typed at the prompt, if the file in which it resides has been cataloged. For more information on UDCs, refer to Chapter 3 of the *MPE XL Commands Reference Manual* (32650-90003).

For those of you who create your own UDC files, you will notice some similarities between UDC files and Command Files. The following comparisons will help you weigh the benefits of having both UDC files and Command Files available in MPE XL.

Contents:

AUDC File:	**A Command File:**
`Contains command header and (sometimes) option header. Consists of one or more command lists (list = one or more executable commands). Each list is called a UDC and is separated from the other UDCs in the file by one or more "*".`	`Contains NO command header; can contain option header (on first line). Consists of a single command list (list = one or more executable commands).`

Error in ENTITY parameter F0401 (File F0401)

Maintenance:

A UDC File:	A Command File:
The UDC file must be cataloged, via the :SETCATALOG command.	A Command File is not cataloged. It can be modified and PURGEd easily.

Example

A UDC File:

```
:SETCATALOG myudc

:SHOWCATALOG

MYUDC.PUB.MERTZ
GREETING     USER
PRINTIT      USER
ST           USER
```

Exercises

4. True or false: A Command File may be cataloged, but does not have to be.

5. True or false: When you enter a Command File name, everything in the file is executed.

Characteristics:

A UDC:	A Command File:
Has the option of being invoked when the user logs on to the system (OPTION logon).	LOGON option ignored. However, it may be called upon by a logon UDC. (See SHOWME below.)

May contain one or more executable commands.	May contain one or more executable commands.
A UDC must have a command header; the UDC is identified by its command header.	A Command File has no command header--it is identified by its file name.

Error in ENTITY parameter F0402 (File F0402)

Execution:

A UDC:	A Command File:
Is invoked at the prompt by its command header.	Is invoked at the prompt by its file name.
Should be used for frequently used, stable command lists.	Can be used to test potential UDCs. Should be used when contents change frequently.

Example

UDCs	Command File:
:GREETING	:SM
:PRINTIT	:PR
:ST	:TIME

Note Parameters can be defined in UDCs and Command Files, using a PARM statement defined in the User Command opening line. Refer to Chapter 3 of the *MPE XL Commands Reference Manual* (32650-90003).

Exercise

6. For each item below, put an X in the appropriate column(s) to show whether the item is a characteristic of UDCs, Command Files, or both.

Error in ENTITY parameter F0403 (File F0403)

Lesson 3: Understanding Search Priorities

What if your Command File or program has the same name as a UDC or system command? Which will execute? To answer this, it will be helpful for you to understand how MPE XL interprets what is typed at the prompt.

In brief, MPE interprets everything entered at the prompt to be a "command" and prioritizes its "search" for that "command" in the following order:

- UDCs first,
- MPE XL commands next,
- file names last (including Command Files and programs).

This hierarchy is called the Search Priority.

The detailed diagram on the next page illustrates the process of MPE XL's search for a "command" called `:TRYIT`.

Error in ENTITY parameter F0404 (File F0404)

This diagram is broken down in the next few pages to allow you to look at each step of this process.

UDC Directory

The system first searches for `TRYIT` in the UDC Directory. User created UDCs are searched first, Account UDCs next, and finally System UDCs. If `TRYIT` is a UDC, the search ends and `TRYIT` is executed.

Error in ENTITY parameter F0405 (File F0405)

Exercise

7. Which UDC in each of the following pairs would execute?

   a. The user UDC `ST` or the system UDC `ST`?

   b. The system UDC `LF` or the account UDC `LF`?

   c. The account UDC `RE` or the user UDC `RE`?

Command Directory

If `TRYIT` is not found in the UDC Directory, the search continues to see if `TRYIT` is an MPE XL system command. If `TRYIT` is an MPE XL system command, it executes.

Error in ENTITY parameter F0406 (File F0406)

Exercise

8. Which would execute, a UDC named `LISTF` or the MPE:`LISTF` command?

File Directory

If `TRYIT` is not found in the Command Directory, the search continues on to file names. If `TRYIT` were a qualified file name of a valid, executable file (`TRYIT.PUB.SYS` or `TRYIT.MYGROUP`, for example), it would execute.

Error in ENTITY parameter F0407 (File F0407)

Exercise

9. If you entered the qualified program file name `SORT.PUB.SYS`, would the system still begin its search in the UDC directory?

Search Path

Since `TRYIT` is not a qualified file name, the system follows a designated search path looking for a matching name. Following the default search path, the system first looks in your current group, next in your PUB group, and finally in PUB.SYS. (The search path is user modifiable.)

Error in ENTITY parameter F0408 (File F0408)

If `TRYIT` is not found in the search path or is found to be a non-executable file, the system issues an Error Message and returns you to the MPE prompt.

Error in ENTITY parameter F0409 (File F0409)

Exercise

10. Indicate in the space provided what would execute in each of the following situations.

- **a.** The `:LISTF` system command or a Command File named `LISTF`?
- **b.** A Command File or a UDC with the same name?
- **c.** A program file called `PRINT` or the MPE XL command `:PRINT`?
- **d.** A Command File in your `PUB` group or a program file with the same name in your current group?

Lesson 4: :XEQ Command

You should always try to keep file names from duplicating those of UDCs or MPE XL commands. If you do encounter name duplications and do not wish to rename your Command File/program, you can employ the new MPE XL `:XEQ` command.

The `:XEQ` command ensures execution of valid Command Files/programs despite name duplications. The `:XEQ` command plus the name of a program file or Command File will cause that file to execute.

Suppose you had a UDC and a Command file with the same name, `MYFILE`.

```
        :MYFILE
```

would execute the UDC.

```
        :XEQ MYFILE
```

would ensure that your Command File `MYFILE` executed.

Exercise

11. Create a Command File that executes the `:SHOWTIME` command; save it as `"LISTREDO"`. Execute this Command File.

Lesson 5: Adding and Deleting UDC Files

Long-time users of UDCs will be pleased to know that they can continue to use UDCs in exactly the same manner as they always have. However, the `:SETCATALOG` command has added the new options `;APPEND` and `;DELETE` to make the task of maintaining UDC files much easier.

Let us look at how a task is done in MPE V/E and how that same task can be done in MPE XL.

Assume that you have a UDC directory with the cataloged files `UDC1`, `UDC2`, `UDC3`, and `UDC4`.

;DELETE

To delete `UDC2` from the directory, keeping `UDC1`, `UDC3`, and `UDC4` cataloged:

MPE V/E

```
     :SETCATALOG udc1,udc3,udc4
```

MPE XL

```
     :SETCATALOG udc2;DELETE

                    or

     :SETCATALOG udc1,udc3,udc4
```

;APPEND

To add `UDC2` back into the directory, where `UDC1`, `UDC3`, and `UDC4` are cataloged:

MPE V/E

```
     :SETCATALOG udc1,udc2,udc3,udc4
```

MPE XL

```
:SETCATALOG udc2;APPEND

                or

:SETCATALOG udc1,udc2,udc3,udc4
```

Note The appended file (`UDC2`) is added to the *end* of the UDC directory in the command `:SETCATALOG UDC2;APPEND`.

Replacing Existing UDC Files

To replace existing UDC files in the directory with specified UDC files:

Same on MPE V/E and MPE XL

```
:SETCATALOG newudc1,newudc2,newudc3
```

Exercises

If the files `TOKYO`, `LONDON`, `DALLAS`, and `LIMA` were cataloged UDC files in a UDC directory, how would you:

12. Delete `LONDON`, without affecting other cataloged files?

13. Add the UDC files `MOSCOW`, `KENYA`, and `BRASILIA` to the UDC directory without affecting other cataloged files?

14. Replace existing UDC files in your UDC directory with the following files: `NEWYORK`, `BOSTON`, `BOMBAY`, and `ZURICH`?

Lesson 6: RECURSION Option

Experienced users of UDC files will recall that in MPE V, a UDC can only call another UDC if the second UDC comes after the first in the UDC directory.

The new MPE XL `RECURSION` option allows the flexibility of having one UDC call upon another that may be cataloged before it. With the `RECURSION` option, the search for a UDC starts with the first UDC in the first UDC file cataloged on the system.

Consider how cataloging is done in MPE V and how flexible it is in MPE XL. In both of the following examples, the UDC `GETB`, in the UDC file `UDCFILE3`, calls upon the UDC `B` in the file `UDCFILE1`.

Error in ENTITY parameter F0410 (File F0410)

Note where `UDCFILE3` *must* be cataloged in MPE V. In MPE XL, `UDCFILE3` may be cataloged either before or after the UDC `B` as long as `OPTION RECURSION` has been invoked.

The following example shows a UDC file in which a UDC calls upon another UDC that precedes it.

Error in ENTITY parameter F0411 (File F0411)

Execution of the UDC "ED" would result in the following:

```
:ED

FILENAME

APPLE    BEATLES    DIANA    EMILY   NORM ...

HP32201A.07.17 EDIT/3000 WED,MAR 9,1988,10:16AM
(C) HEWLETT-PACKARD CO. 1985
/
```

Note The `RECURSION` option is only effective for the UDC in which it is specified. Other UDCs in the file are not affected.

Note also that recursion is an unchangeable characteristic of Command Files; `RECURSION` and `NORECURSION` are simply ignored in Command Files.

`OPTION NORECURSION` is the default and is consistent with MPE V/E. When invoked it allows a UDC to call other UDCs only if they follow the calling UDC in the catalog.

Exercise

15. Study the following MPE XL UDC file. Indicate how you would change this file to have the UDC SO'' call upon and execute the UDC ST'' without changing the order in which they now appear:

```
ST
showtime
**
SM
showme
**
SO
showout
**
```

Additional Information

Because recursion allows two UDCs to call each other and enables a UDC to call itself, limitations have been put in place to prevent "endless loops" (for example, two commands calling each other over and over again without end).

Example

A UDC called `GETA`

```
GETA
Option Recursion
SHOWTIME
GETA
**
```

The maximum number of times that `GETA` can call itself is 30 times. When that maximum is reached, the system interrupts the process with an error message. This safeguard may be helpful to you in "debugging" your User Commands.

Lesson 7: PROGRAM Option

The new MPE XL option `PROGRAM/NOPROGRAM` allows you to specify when a User Command may or may not be executed from within a program. `OPTION PROGRAM` is the default.

Note Programs and subsystems that use the new MPE XL intrinsic `HPCICOMMAND` execute UDCs that have `OPTION PROGRAM` specified. Programs and subsystems that use the old `COMMAND` intrinsic, however, will not allow programmatic execution of UDCs even when `OPTION PROGRAM` has been specified.

The new `PROGRAM` option (the default condition) will allow you to execute the following User Command:

```
TIME
SHOWTIME
**
```

from within an application program such as `VOLUTIL` (which is MPE XL's replacement for the MPE V/E command `:VINIT`):

```
volutil: :TIME
MON, MAR 9 1987, 11:07 AM
```

The use of `OPTION NOPROGRAM`:

```
TIME
OPTION NOPROGRAM
SHOWTIME
**
```

will suppress the User Command search when the User Command is executed from within a program:

```
volutil: :TIME
UNKNOWN COMMAND NAME.  (CIERROR 975)
```

Exercises

16. Study the following UDC and indicate how you would change this file so that the UDC `LF` would *not* be executable from within a program.

```
LF
LISTF
***
```

17. Assume that you have a Command File, called `LPRINT`. Could you execute `LPRINT` *from within a program*, if the program used the `HPCICOMMAND` intrinsic? If it used the `COMMAND` intrinsic?

Lesson 8: :OPTION Command

`:OPTION` is an MPE XL command and can be used in any User Command—both UDCs and Command Files.

The new `:OPTION` command allows `RECURSION/ NORECURSION` and `LIST/NOLIST` to be specified anywhere in the body of a User Command. All other options can only be set in the option header. Options remain in effect until the User Command has finished executing or, in the case of `RECURSION/NORECURSION` and `LIST/NOLIST`, until they are overridden by the `:OPTION` command in the body of the User Command.

In the following example, the header `OPTION LIST` is "activated" in the UDC `DOIT1` and "deactivated" later in the UDC body with the `:OPTION` command. Also note where `OPTION RECURSION` appears in UDC `DOIT3`.

Example:

```
DOIT1
OPTION LIST
showtime
OPTION NOLIST
listf
**
DOIT2
showme
**
DOIT3
listf
OPTION RECURSION
doit2
**
```

Exercises

18. Create a *single* UDC that does the following in the order given:

a. Executes the `:SHOWTIME` command without listing the `:SHOWTIME` command to the screen.

b. Executes the `:LISTF` command without listing the `LISTF` command to the screen.

c. Lists to the screen and executes the `:SHOWME` command.

19. What is wrong with the following Command File?

```
OPTION NOLIST
FILE OUT;DEV=LP
PRINT MYFILE, OUT=*OUT
OPTION LIST, NOHELP
LISTF MYFILE,2
```

Answers to Exercises

1. Example

   Command File: `MYFILE1`

   ```
   showme
   showtime

   :myfile1
      . . .
   ```

2. Example

   Command File: `MYFILE2`

   ```
   listf

   :myfile2
      . . .
   ```

3. Command File: `MYFILE3`

   ```
   MYFILE1
   MYFILE2

   :myfile3
      . . .
   ```

4. False. Before it could be cataloged, a command header would have to be added.

5. True

6.

Error in ENTITY parameter F0412 (File F0412)

7. a. User UDC `ST`

b. Account UDC `LF`

c. User UDC `RE`

8. A UDC named `LISTF`

9. Yes, it would.

10. a.`:LISTF` system command.

b. The UDC.

c. The MPE XL command`:PRINT`.

d. The program file in your current group.

11.
```
/a
1  showtime
2  //
/k listredo
...

:XEQ LISTREDO
```

```
MON, FEB 8, 1988, 4:27 PM
```

12. `:SETCATALOG LONDON;DELETE`

13. `:SETCATALOG MOSCOW, KENYA, BRASILIA;APPEND`

14 `:SETCATALOG NEWYORK, BOSTON, BOMBAY, ZURICH`

15.

```
SO
OPTION RECURSION
showout
ST
**
```

16.

```
LF
OPTION NOPROGRAM
LISTF
**
```

17. The HPCICOMMAND intrinsic allows Command Files and UDCs to be executed from within a program, as long as the User Command has not specified OPTION NOPROGRAM. If the program has used the COMMAND intrinsic, however, User Commands cannot be executed while the user is running the program, even if they have OPTION PROGRAM specified.

18.

```
SHOW
OPTION NOLIST
SHOWTIME
LISTF
OPTION LIST
SHOWME
**
```

19. The use of the:OPTION command within the body of a User Command is limited to RECURSION/ NORECURSION and LIST/NOLIST. All other options may only be specified in the OPTION header. Therefore, NOHELP should not have been specified in the body of the Command File.

5

Variables

A new feature of MPE XL is the availability of two kinds of variables. Now, user created variables are accessible, and can be given names and values. Predefined system variables also are available. Together they extend the "programming-like" capabilities of the Command Interpreter.

User created variables are new tools for both interactive and programmatic use. They are similar to MPE V Job Control Words (JCWs), but more versatile. The lessons on user created variables introduce the new commands `:SETVAR, :SHOWVAR`, and `:DELETEVAR`.

- Lesson 1: Names for user created variables
- Lesson 2: Creating and displaying variables
- Lesson 3: Deleting a variable

System variables give the user greater access to global items. Some are user modifiable. The lessons on system variables describe the new commands `:SETVAR` and `:SHOWVAR`.

- Lesson 4: System variables
- Lesson 5: User modifiable system variables
- Lesson 6: Modifying and restoring system values

Two new commands `:ECHO` and `:INPUT`, offer the user some helpful options for both kinds of variables.

- Lesson 7: The`:ECHO` command
- Lesson 8: Using`:INPUT` to change your prompt

Aspects of Dereferencing and some simple variable applications furnish the user with more choices in MPE XL.

- Lesson 9: Dereferencing variables

- Lesson 10: Dereferencing with `:ECHO`
- Lesson 11: Recursive dereferencing
- Lesson 12: Changing your search path

Lesson 1: User Created Variable Names

The first step in creating a user variable is to select a name for it. This name can be as simple as the letter X, or many lines long. It must follow these simple rules:

- It must start with either an alphabetic or an underbar character.
- The rest must be alphanumeric or underbar characters; nothing else.
- It can be from 1 to 255 characters long.
- It can have no spaces.

ACCEPTABLE	**UNACCEPTABLE**
PACIFIC_TIME	#@%_$&
A	MAX#
_007	123
NUM_array	error flag
employee4	4employee
A6_B12_X	A6_B12+X

Exercise

1. Which of the following are acceptable names for user created variables?

   a. $DOLLARS

   b. FRED_FLINTSTONE

   c. 4FSNE1

   d. _BED ROCK

   e. Z

   f. INSPECTOR!

g. _JAMES_BOND

h. RINGO*

i. The_quick_red_fox_jumped_over_the_lazy_b

j. _!@#$%&*()

k. LOG&LOG

Lesson 2: Creating and Displaying Variables

User created variables must be given values. Use:`SETVAR` to create and modify them. Use:`SHOWVAR` to display them.

They are similar to JCWs, but are more flexible and extensive. MPE XL variables can be used interactively, in sessions, in jobs, or in programs. JCWs and variables can be used in both UDCs and Command Files.

User created variables can contain 32-bit integers, string values, Boolean values, expressions, or the names of other variables. Any process in a job or session can read, change, or delete user created variables. They end with the session.

:SETVAR

:`SETVAR` is used to create variables and to assign values. Using it, create a variable `f` to equal the value `Fred`,

a variable _e" to equal the value eTHEL", a variable k" to equal the integer 1024", and a variable j" to equal the decimal 10.24", as follows:

```
:setvar f,'Fred'
:setvar _e,'eTHEL'
:setvar k,1024
:setvar j, '10.24'
```

:SHOWVAR

To show that these variables were set, use:`SHOWVAR`:

```
:showvar f,k,_e,j
 F  = Fred
 K  = 1024
 _E = eTHEL
 J  = 10.24
```

Entering:`showvar` alone displays a list of all user created variables defined during a session, and their values. The variables are shown in the order of their creation.

Exercises

2. Set the variable `PI` to equal the value `3.14159`, and the variable `BOND` to equal the value `007`.

3. Display the values of `PI` and `BOND`.

Lesson 3: Deleting a Variable

The new:`DELETEVAR` command is used to purge user created variables from the system.

Remove the variables `f`, `_e`, and `k` from the system with :`DELETEVAR`, as shown below. Use:`SHOWVAR` to verify that they were deleted.

```
:deletevar f,_e,k
:showvar f
        ^
VARIABLE NOT FOUND IN TABLE. (CIERR 8106)
```

To delete all user created variables, enter:`DELETEVAR` followed by the wildcard character `@`.

Exercises

4. Delete the variable `PI` created in Exercise 2. Verify that it is gone.
5. Display all variables created in this session that are still in place. Delete them. Show that they are gone.

Lesson 4: System Variables

There are a great many predefined system variables. Most of them are read-only. To see the variable table (a list of all of the variables on your system) with their current values, use :`SHOWVAR` plus the wildcard character `@`:

```
:showvar @
CIERROR = 0
HPACCOUNT = SYS
HPAUTOCONT = FALSE
   .
   .
HPWAITJOBS = 0
HPYEAR = 88
JCW = 0
```

The values on your system may not be the ones shown in the example. Any user defined variables added during the current session appear at the bottom of the list.

You will find additional information on system variables and their uses in Appendix A of the *MPE XL Commands Reference Manual* (32650-90003).

Exercise

6. Where will you find definitions of system variables?

Lesson 5: User Modifiable System Variables

These are the system variables you can modify. The modifications you make will be unique to your session. They will disappear at the end of the session and be replaced by default values in your next session.

Variable	Definition	Default
CIERROR	last CI error #	0 (zero)
HPAUTOCONT	en-/disable :CONTINUE	FALSE
HPCMDTRACE	en-/disable cmd tracing	FALSE
HPCMEVENTLOG	shows # of CM events	0(zero)
HPERRDUMP	# of levels to dump	0 (zero)
HPMSGFENCE	CI message level fence	0 (zero)
HPPATH	system search path	!hpgroup ,pub ,pub.sys
HPPROMPT	CI prompt string	: (colon)
HPREDOSIZE	CI redo stack max #	20
HPRESULT	last value from:CALC	none
HPSYSNAME	computer system name	none
HPTIMEOUT	# of min. for CI reads	0 (zero)
JCW	Job Control Word	0 (zero)

Exercises

7. True (T) or False (F):

a. `HPWAITJOBS'' is a user modifiable system variable. *b.``` HPCMEVENTLOG is a user modifiable system variable.

c. `HPCPUNAME'' is a user modifiable system variable.`

8. What system variable would you look at to find out what your current user capabilities are? How?

Lesson 6: Modifying and Restoring System Values

The new:SETVAR command is used to change the value of a user modifiable system variable. An example is to change the size of the command line history stack from its default of 20 to 15 using the variable HPREDOSIZE.

```
:setvar hpredosize,15
```

To give a name to your system, or to change the one it already has, use the variable HPSYSNAME.

```
:setvar hpsysname,'harmony'
```

To verify the changes, enter:

```
:showvar hpredosize,hpsysname
 HPREDOSIZE = 15
 HPSYSNAME = harmony
```

To change the value of HPREDOSIZE back to the default, 20, either log off and back on again, or use:

```
:setvar hpredosize,20
```

To verify the change, enter:

```
:showvar hpredosize
 HPREDOSIZE = 20
```

Note You cannot use :DELETEVAR on a system variable. If you try, you will get an error message, because system variables cannot be removed.

Exercises

9. How do you change the prompt from a colon (:) to `Yes?` ?

10. How do you verify that the variable value has changed?

Lesson 7: The :ECHO Command

The new`:ECHO` command repeats, or echoes, whatever string of characters follows it. The character string is displayed on the screen with no further result, even if it is a command name. For example:

```
:echo ShowTime (25)
 ShowTime (25)
```

The only result is the appearance of the character string `ShowTime (25)`. Uppercase and lowercase letters, digits, and symbols appear as they were entered. (MPE XL has not responded to ShowTime as a command.)

Exercise

11. If you insert a colon in front of `ShowTime (25)` in the example, what response appears on your screen?

Lesson 8: Using :INPUT to Change Your Prompt

The new:INPUT command can create a user variable. It also allows you to interactively assign a value to any user created variable or user modifiable system variable. It has two optional parameters: a prompt string, and a timed delay.

Example:

If you want to be able to interactively change the system prompt from its default colon (:) to some other character(s), you could use the following command line in a UDC, or in a Command File called CHPROMPT:

```
input hpprompt,"Enter New Prompt: ";wait=15
```

Executing such a UDC or Command File (CHPROMPT) would look like this:

```
:chprompt
 Enter New Prompt:
```

The Command File causes the prompt string Enter New Prompt: to appear for up to 15 seconds, giving the user time to respond. If the user supplies this response to the waiting system:

```
Enter New Prompt: Hi There ...
Hi There...
```

Hi There ... is then assigned to the variable HPPROMPT. The prompt on the screen changes immediately. :SHOWVAR HPPROMPT will also confirm the new value.

The system returns to a colon prompt (:) every time you start a new session.

Exercises

12. What happens in the case of the example given for the `:INPUT` command if you don't respond within 15 seconds?

13. How would you use `:INPUT` simply to change `HPPROMPT` from the default colon (:) to three dots (...) without either a prompt string or timed delay? (Hint: The three dots are *not* entered on the `:INPUT` command line. They are entered *after* the `:INPUT` command line has been executed.)

Lesson 9: Dereferencing Variables

Dereferencing replaces the variable name with its value. An exclamation point placed in front of the variable name tells the system to dereference that variable.

For example:

```
:setvar x,'BLUE'
:showvar x
 X = BLUE
:setvar y,'!X is best.'
:showvar y
 Y = BLUE is best.
```

The:`SETVAR` command assigns the string value `!X is best.` for the variable `y` in the previous example. The exclamation point in `!X is best.` tells the system to substitute `BLUE` (the variable value) for `X` (the variable name). This process is dereferencing.

Changing the value of variable `x` will have no effect on the value of `y`, as shown below:

```
:showvar x,y
 X = BLUE
 Y = BLUE is best.
:setvar x,'RED'
:showvar x,y
 X = RED
 Y = BLUE is best.
```

Exercises

14. What do you call the resolving of `!X is best.` to `Y = BLUE is best.`?

15. At the end of the second example on the previous page, how would you use dereferencing to make `Y = Your eyes are RED`?

Lesson 10: Dereferencing With :ECHO

The new:`ECHO` command, and an exclamation point immediately followed by a variable name, will also extract the value of a variable.

For example, a variable named `e_m` is created that has a value of `Ethel Mertz`. It can be dereferenced with :`ECHO` by starting the command line with an exclamation point, like this:

```
:setvar e_m,'Ethel Mertz'
:echo !e_m
 Ethel Mertz
```

The value is displayed.

System variables can also be dereferenced with `:ECHO`. For instance, you can find out what time you started your current session by entering:

```
:echo !hpintrotime
 10:07 AM      (or your actual start time)
```

To find out what kind of computer you are working with, enter:

```
:echo !hpcpuname
 SERIES 950    (or whatever your system is)
```

Exercises

16. Re-create variable `K` equal to `1024`. How do you dereference it in an:`ECHO` statement?

17. How do you dereference in order to see `1024` ... `Ethel Mertz` on your screen?

Lesson 11: Recursive Dereferencing

If you want to assign the value of one variable (M1) to another variable (M2), but want the value of M2 to change dynamically as the value of M1 changes, you must use two exclamation points in the :SETVAR command:

```
:setvar M1,'Message'
:setvar M2, '!!M1'
```

See what happens when you check the value of M2 with :SHOWVAR and :ECHO:

```
:showvar M2
 M2 = !M1
:echo !M2
 Message
```

In the example above, every time the value of M1 changes, the value of M2, which is !M1, will also change. This phenomenon is called recursive dereferencing.

A practical example of recursive dereferencing would be to add some status information to your MPE prompt. Assume that you want, as a constant reminder, your current group name followed by the current history stack number. You can accomplish this by using the two appropriate system variables HPGROUP and HPCMDNUM. If you then checked with :LISTREDO, your screen might look like the following:

```
:setvar hpprompt,'!!hpgroup/!!hpcmdnum:'
PUB/2:listredo
   1) setvar hpprompt,'!!hpgroup/!!hpcmdnum:'
   2) listredo
PUB/3:
```

Notice that the command number changed from 2 to 3. To check further, use :CHGROUP to change from PUB to LABS, or any other group in your account. You should see:

```
PUB/3:chgroup labs
LABS/4:
```

This indicates that you succeeded in changing groups and that the command history continues to grow, which means that it is independent of your location.

Exercises

18. Recursive dereferencing must be done using which of these commands?

    _: ECHO _: SETVAR

19. Recursive dereferencing characteristically uses how many exclamation points?

20. How would you change this command line in order to include the account name after the group name? `setvar hpprompt,'!!hpgroup/!!hpcmdnum:'`

21. How would the resulting prompt look in the `PUB` group of a `SALES` account?

Lesson 12: Changing Your Search Path

Another application for recursive dereferencing (when two exclamation points: `!!` are used) would be to simplify file execution by changing your search path.

You can execute any command file, or have read access to any working file on the system, without qualifying it and without having to change groups. Just get `:RELEASE` permission, observe security, and include the proper location in your `HPPATH`.

The default value is:

```
        HPPATH = !hpgroup,pub,pub.sys
```

Consider an example in your own account. Suppose you are in the `RECORDS` group. There is a Command File that you created named `DOIT` located in the `MIS` group of your account. To execute this file frequently and most conveniently, add the `MIS` group to your search path.

```
       :setvar hppath,'!!hpgroup,mis,pub,pub.sys'

   or

       :setvar hppath, '!hppath,mis'
```

Notice the two exclamation points in front of `HPGROUP`.

Exercises

22. How do you verify your search path with `:ECHO`?

23. What “permission” is needed to access the files in other groups in your account?

Answers to Exercises

1. Selections b, e, g, and i are all acceptable names.
2. `:setvar pi,'3.14159'`, and `:setvar bond,'007'`
3. `:showvar pi,bond` or just `:showvar`
4. `:deletevar pi`, then `:showvar pi`
5. `:showvar`, `:deletevar @`, and `:showvar`
6. In Appendix A of the *MPE XL Commands Reference Manual* (32650-90003)
7. a. F b. T c. F
8. `HPUSERCAPF` Enter `:showvar hpusercapf`
9. `:setvar hpprompt,'Yes?'`
10. The new prompt appears on your screen as soon as you enter the command string.
11. `:ShowTime (25)` (not the time and date).
12. It displays the first line of the Command File, plus the error message: `THE INPUT TIMED READ HAS EXPIRED. (CIWARN 9003)`, and reverts to the previous prompt.
13. Enter `:input hpprompt`. Press return, then enter ... (three periods entered into the empty space).
14. It is called dereferencing.
15. Enter `:setvar y,'Your eyes are !x'`
16. Enter `:echo !k`
17. Enter `:echo !k ... !e_m`
18. The `SETVAR` command.
19. 2 (or a pair)
20. Put `.!!hpaccount` between `hpgroup` and `/`, as follows:

```
          ... !!hpgroup.!!hpaccount/!!hpcmdnum:'
```

21. `PUB.SALES/2:`
22. Enter:`echo !hppath`
23. The file must be released.

6

Expression Evaluation

MPE XL provides you with an online calculator. This powerful expression evaluator evaluates whole numbers, strings, TRUE/FALSE (Boolean) expressions, and supports a rich set of functions. A full list of the expression evaluator functions can be found in the *MPE XL Commands Reference Manual* (32650-90003).

- Lessons 1, 2, 3, and 4 discuss the four MPE XL commands that permit implicit expression evaluation: `:CALC`, `:SETVAR`, `:IF`, and `:WHILE`. Lessons 3 and 4 require knowledge of programming.
- Lesson 5 introduces you to the MPE XL syntax that allows for expression evaluation and value substitution to occur at the user's discretion.

Lesson 1: :CALC

Error in ENTITY parameter F0601 (File F0601)

The new MPE XL command `:CALC` lets you use the operating system as an online calculator. `:CALC` evaluates an expression and stores the result in the system variable `HPRESULT`.

The answer to any mathematical problem is displayed as a decimal numeral first, as a hexadecimal numeral second, and as an octal numeral last.

Note Release 1.1 of MPE XL only recognizes whole numbers.

```
:CALC 1.5+2
          ^
ILLEGAL CHARACTER FOUND, EXPECTED A NUMBER.
(CIERR 9810)
```

Examples:

- Whole Number Evaluations

```
:CALC 1+4
 5,  $5,  %5

:CALC 10-4
 6,  $6,  %6

:CALC 5*12
 60,  $3C,  %74

:CALC 100/2
 50,  $32,  %62

:SHOWVAR hpresult
 HPRESULT = 50
```

- String Evaluation

```
:CALC 'abc'+'def'
 abcdef
:SHOWVAR hpresult
HPRESULT = abcdef
```

- Boolean Evaluation

```
:CALC 4=5
 FALSE
:SHOWVAR hpresult
HPRESULT = FALSE
```

Note Mixed expressions (for example, whole numbers and strings) are not accepted; an error will result.

```
:CALC 'a' + 2
            ^

ILLEGAL CHARACTER FOUND, EXPECTED A STRING
```

```
                (CIERR 9815)
```

Exercises

1. Use the :CALC command to do the following:
   a. Add 3 and 4.
   b. Evaluate the "truth" of 3=3.
   c. Subtract "abc" from "abcdefg".

Lesson 2: :SETVAR

:SETVAR also allows for evaluation of expressions:

Examples

- Whole Number Evaluation

```
:SETVAR a, 5
:SETVAR b, 4
:SETVAR c, a+b
:SHOWVAR c
C = 9
```

- String Evaluation

```
:SETVAR E, "Hi there,"
:SETVAR F, E + " friend."
:SHOWVAR F
F = Hi there, friend.
```

- Boolean Evaluation

```
:SETVAR a, 5
:SETVAR b, 4
:SETVAR c, a>b
:SHOWVAR c
```

```
C = TRUE
```

Exercises

2. Assign the value "John" to the variable "name". Assign the value "Doe" to the variable "lastname".

Link together (concatenate) "name" and "lastname", leaving a space " " between the two so that "John Doe" appears on the screen.

3. Calculate name=lastname.

Lesson 3:
:IF

The `:IF ( ... ENDIF)` command has been modified in MPE XL to allow for expression evaluation.

Consider the following Command File:

```
IF LFT (HPDATEF,3) = "MON" THEN
 ECHO Reminder--Staff Meeting at 11:00
ELSE
 IF LFT (HPDATEF,3) = "THU" THEN
  ECHO Reminder--Weekly Dept. Meeting at 9:30
 ELSE
  ECHO Have a nice day!
 ENDIF
ENDIF
```

In the example above, if it is Monday (the first three letters of `HPDATEF` = MON), "Reminder ... Staff Meeting at 11:00" will echo to the screen; if it is Thursday (the first three letters of `HPDATEF` = THU), the reminder of the 9:30 meeting will appear; and if it is any other day of the week, "Have a nice day!" is printed on the screen.

Exercise

4. Using the `:IF` command, write a logon UDC that will remind you to do something every other day. HINT: use (!hpday MOD 2).

Lesson 4: :WHILE

The new `:WHILE ( ... ENDWHILE)` command allows for expression evaluation, which adds a looping capability that can be used to control the sequence of command execution.

Examples

```
SETVAR filenum, 3
WHILE (filenum >= 0)
  PURGE myfile!filenum
  SETVAR filenum, filenum - 1
ENDWHILE
```

The above example will purge all occurrences of the file `MYFILE#` starting with `MYFILE3` until the `FILENUM` value of -1 is met (for example, `myfile3`, `myfile2`, `myfile1`, `myfile0`).

Exercises

5. Using the `:WHILE` command, write a Command File that will count to 10 and then end when the set limit has been reached.

6. Write a Command File that counts to 50 by twos.

Lesson 5: Expression Substitution

Expressions are evaluated—and variables are dereferenced—*implicitly* in the commands `:CALC`, `:IF`, `:WHILE`, and `:SETVAR` (if `:SETVAR` is defining a *numeric* variable). Expression evaluation and variable dereferencing are not done implicitly in other MPE commands, but they can be done *explicitly*.

You have already seen examples of implicit and explicit dereferencing:

Implicit Dereferencing (no !)

```
:SETVAR X,4
:SETVAR Y,X          (<-----numeric variable)
```

in which the value of `X` is assigned to the variable `Y` without the use of ! to dereference the `X`. (Note that if you did use a ! to dereference the `X`, the system would accept that syntax, too.)

Explicit Dereferencing (using !)

```
:ECHO !X
:PRINT MYFILE!X
```

in which ! is used to dereference the variable `X` in echoing "4" to the screen and in printing the contents of `MYFILE4` to the screen.

If you wanted to evaluate an expression in a command other than `:CALC`, `:SETVAR`, `:IF`, and `:WHILE`, you would use this syntax: ! [*expression*].

Examples

```
:SETVAR a, 3 + 1
:ECHO a+2=![a+2]
A+2=6
```

In this example, note that the first expression in the "ECHO" line is not evaluated. The `:ECHO` command itself does not allow for evaluation of expressions. However, when ! [a+2] is encountered in the same command line, it is evaluated and its value is substituted therein.

Exercises

7. Use each of the following commands to evaluate 3+2:
`:SETVAR`, `:CALC`, and `:ECHO`.

8. Use `:ECHO` to evaluate the following, where X=24, Y=362, and Z=1000:
(X + Y + Z - 252) * 4

9. Write a Command File that prompts the user for his/her favorite year, and that uses the input in displaying the following (completed) sentence on the screen:

Had you been born in the year _____, you would be _____ years old in the year 2010!

(HINT: `:INPUT` cannot accept numeric data, but the input can be converted to numeric data using `:SETVAR`. However, you have to use *explicit dereferencing*!)

Answers to Exercises

1a.
```
:CALC 3+4
7,  $7,  %7
```

b.

```
:CALC 3=3
TRUE
```

c.

```
:CALC 'abcdefg' - 'abc'
DEFG
```

2.

```
:SETVAR name, "John"
:SETVAR lastname, "Doe"
:SETVAR C,name+" "+lastname
:SHOWVAR C
John Doe
```

3.

```
:CALC name=lastname
FALSE
```

4.

```
IF (hpday mod 2)=0 THEN
 ECHO Do this.
ELSE
 ECHO Do that.
ENDIF
```

5.

```
SETVAR counter, 1
WHILE counter<11
 ECHO !counter
 SETVAR counter,counter +1
ENDWHILE
```

6.

```
SETVAR counter, 2
WHILE counter<51
 ECHO !counter
 SETVAR counter,counter+2
ENDWHILE
```

7.

```
:SETVAR a, 3+2
:SHOWVAR a
A = 5
:CALC 3+2
5, $5, %5
:ECHO ![3+2]
5
```

8.

```
:SETVAR X, 24
:SETVAR Y, 362
:SETVAR Z, 1000
:ECHO ![(X + Y + Z - 252) * 4
4536
```

9.

```
INPUT YEAR,"What is your favorite year? "
SETVAR NYEAR,!YEAR
ECHO Had you been born in the year !nyear,
ECHO you would be ![2010-nyear] years old
ECHO in the year 2010!
```

7 The Command Interpreter Program

Have you ever been in the middle of an application and wished there were a quick way to access the MPE prompt and MPE commands without first exiting your application?

The Command Interpreter is a program that interprets MPE commands and executes them. MPE XL users can now access the Command Interpreter as a program. The CI program resides in PUB.SYS and can be run from within itself (at the MPE prompt) or from within *some* application programs.

- Lesson 1 introduces the `CI.PUB.SYS` program and how it may be accessed (at the MPE prompt) and exited.
- Lesson 2 describes how more than one level of the Command Interpreter can be accessed and what special capability is needed to run programs from those levels.

Lesson 1: Running CI.PUB.SYS

Running `CI.PUB.SYS` gets you out of the "root" level of the Command Interpreter and enters you into a "nested" level. The system variable `HPCIDEPTH` keeps track of your current CI level.

```
     ROOT Level CI               NESTED Level CI
 +--------------------+     +---------------------+
 | :SHOWVAR HPCIDEPTH |     | :SHOWVAR HPCIDEPTH  |
 |                    |     |                     |
 | HPCIDEPTH = 1      |---->| HPCIDEPTH = 2       |
 |                    |     |                     |
 | :CI.PUB.SYS        |     |                     |
 +--------------------+     +---------------------+
```

:EXIT and :BYE

The new `:EXIT` command and the familiar `:BYE` command allow you to exit any level of the CI. `:EXIT` backs you out of nested levels, one level at a time. If `:EXIT` is used from the root level, the session ends.

The `:BYE` command, used from any level, directly ends the session.

The [Break] key, as always, exits you temporarily from the program. From a nested level, it returns you temporarily to the root level.

`:RESUME` returns control to the appropriate CI level.

Error in ENTITY parameter F0701 (File F0701)

Running `CI.PUB.SYS` and using the`:EXIT` command:

Examples

```
:CI.PUB.SYS      or    :CI
MPE XL CI X.02.07 Copyright Hewlett-Packard
```

```
***********************************
*                                 *
*  WELCOME TO YOUR MPE XL SYSTEM  *
*                                 *
***********************************

:SHOWVAR hpcidepth
HPCIDEPTH = 2

:EXIT

:SHOWVAR hpcidepth
HPCIDEPTH = 1

:EXIT     or     :BYE
CPU=3. CONNECT=1. WED,MAR 9, 1988, 11:05 AM
```

Running CI.PUB.SYS and using the :BYE command:

Examples

```
:CI.PUB.SYS      or    :CI
MPE XL CI X.02.07 Copyright Hewlett-Packard

***********************************
*                                 *
*  WELCOME TO YOUR MPE XL SYSTEM  *
*                                 *
***********************************

:SHOWVAR hpcidepth
HPCIDEPTH = 2

:BYE
CPU=3. CONNECT-1. WED,MAR 9, 1988, 11:30 AM
```

Running CI.PUB.SYS and using the Break key and : RESUME command:

Examples

```
:CI.PUB.SYS      or    :CI
MPE XL CI X.02.07 Copyright Hewlett-Packard

***********************************
*                                 *
*  WELCOME TO YOUR MPE XL SYSTEM  *
*                                 *
***********************************

:Break

:SHOWVAR hpcidepth
HPCIDEPTH = 1
```

```
:resume
READ pending

:SHOWVAR hpcidepth
HPCIDEPTH = 2
```

Exercises

1. At the MPE prompt, run the CI program.

2. Verify that you were successful.

3. Explain the difference between ending your session using `:EXIT` and using `:BYE`.

4. Choose either method to exit your session.

Lesson 2: Accessing Multi-Nested Levels

The program `CI.PUB.SYS` does not have Process Handling (PH) capability, the ability to access nested levels. Therefore, the ability to run `CI.PUB.SYS` from levels other than the root level of the Command Interpreter is dependent upon either the user or the application program having PH capability.

Users with PH capability can run `CI.PUB.SYS` or any other program from all levels of the CI, root or nested. PH capability enables users to access multiple nested levels of the CI.

Error in ENTITY parameter F0702 (File F0702)

Users with PH capability can run any program, including CI.PUB.SYS,from the ROOT or NESTED levels.

A good example of the convenience of this feature is the ability to access another level of the CI while in the middle of developing a program. Programmers can save the code they have just written. Then, without having to exit their editor environment, they can run CI.PUB.SYS"; from this next-level CI they can compile and test their program. Finally, they can `:EXIT` that CI level, return to the earlier level, and continue working on the program code without having to redefine their environment.

Examples

```
:editor
    . . .
/a
   1     (User writes program code)
    . . .
/k prog
    . . .
/:run ci.pub.sys
    . . .
:(Nested CI level--user compiles/runs  PROG)
    . . .
:exit       (User returns to root CI)
/
```

With the exception of `:SETCATALOG` and `:CHGROUP`, which can only be executed from the root level, MPE commands are functional in nested levels.

A new command line history stack is created for each new nested level. As you`:EXIT` a level, the history stack for that level is permanently erased, and the history stack for the previous level is restored.

All user created variables and most user defined system variables remain constant across all nested levels. Exceptions include `HPREDOSIZE, HPCONTINUE, HPUSERCMDEPTH, HPAUTOCONT, HPCMDTRACE, and HPMSGFENCE`.

Note User PH capability is needed to complete the following exercises.

Exercises

You might want to change the prompt to reflect the `HPCIDEPTH` level before doing the following exercises.

5. Get to nested level 4.

6. Try to reset your UDC catalog.
Explain what happened.

7. Assign a value to the variable `A`; display it.

8. Back out of level 4 to nested level 3.

9. Display the value given to variable `A` on your screen.
Explain what happened.

10. Display your history stack to the screen. What commands are listed?

11. Back out of level 3 to nested level 2.

12. Display your history stack to your screen. How many commands are listed? Explain the discrepancy between this history stack and the one from Step 10.

13. Back out of level 2 to the root level.

14. Reset your UDC catalog. Explain what happened.

15. Use an editor to write a short program; save it. From within the editor, run `CI.PUB.SYS`. Compile/run your program; then`:EXIT` back to the editor at the root level.

Answers to Exercises

1. :CI (Return)
2. :SHOWVAR hpcidepth
3. Each time you enter :EXIT, you return to the previous CI level, until you reach the root level, at which point the :EXIT command ends your session. :BYE ends your session regardless of what level of the CI you are in.
4. :EXIT or :BYE
5. :CI (Return): CI (Return): CI (Return)
6. :SETCATALOG can only be executed from the root level of the Command Interpreter. You would get an Error Message: THIS COMMAND CAN ONLY BE EXECUTED FROM THE CI. (CIERR 9063).
7. :SETVAR a, 3
   : SHOWVAR a
8. :EXIT
9. :SHOWVAR a

   The value of A remains constant in all levels of the CI.
10. The history stack should reflect the commands you entered when you were at that level. The history stack is reset when you enter the next level of the CI. The history stack is restored as you exit back to previous levels of the CI.
11. :EXIT
12. The history stack is restored as you return to previous levels of the CI.
13. :EXIT
14. UDC catalog can now be reset. You are in the ROOT level of the CI.
15.

```
   :editor
       . . .
   /a
      1      (User writes program code)
       . . .
   /k prog
   . . .
/:run ci.pub.sys
    . . .
:(Nested CI level--user compiles/runs  PROG)
   . . .
:exit        (User returns to root CI)
  /
```