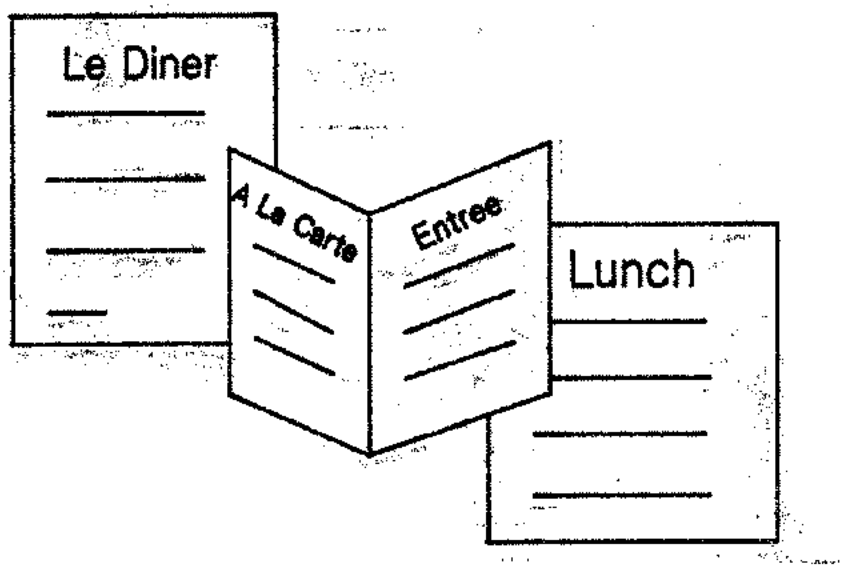
HPMenu



Training

E07/84
32112-90008

Table of Contents

* Indicates new or modified material for second release.

Introduction

Purpose of this training

This is the second version of the SE training for HPMenu. It is designed to give the SE a working knowledge of the product. After completing this training, an SE should be able to:

install the product
use customization
convert udc's
set up logon menus
take most PICS calls on the product
Discuss intrinsics, current object store and parameter passing

The entire training should take between 1 and 2 hours. The materials necessary to complete this training are:

HPMenu Administration Manual
HPMenu Quick Reference Guide
this booklet

What exactly is this product?

HPMenu provides the user with a friendly, menu-driven interface for transitions between system functions, including HP products, user-written applications, and some MPE commands. HPMenu is targeted at all HP3000 users. It is easy to learn and the customization facility allows each user to tailor his menus to match his view of the system. Each user may have different names for the same functions and organize them in different ways. A single user can set up several sets of menus corresponding to the various roles he assumes.

HPMenu shields the novice user by providing the means for him to invoke system functions without being aware of the MPE program and file structures which support them. Users select from a menu by point and push, by number, or by typing in the name of any defined function. They can make a menu choice by recognition rather than recall. In addition, HPMenu does not slow down the experienced users who are familiar with functions, because they can type function names for direct access.

The second release of the product provides a simple object management and current object passing facility. This can be done both interactively and programmatically. HPMenu provides users with a programmatically accessable extra data segment containing a 256 word "Scratchpad" area as well as a current object store. The current object store sets and yields current object names (e.g. chart file name) and configuration parameters (e.g. output device).

Disclaimer: It should be noted that HPMenu has no control over the actions of any programs that it runs. This means that HPMenu cannot change interface inconsistencies, error handling, hardware and software restrictions, or any feature of the program run under it, be it a user application or an HP subsystem.

Who is the product for?

The target customers are office workers who are not familiar with computers and operating systems, and need to have the operation of MPE transparent to them. The key market comprises office workers.

With the addition of the current object store capability, HPMenu is now more userful to EDP departments to connect applications and pass information from application to application. Current applications can be enhanced and new ones written to take advantage of the parameter passing capability, both interactively and programmatically.

Why Is HPMenu?

The introduction of HPMenu will show the marketplace that HP is moving towards integrating the existing office capabilities. HPMenu is the forerunner of a truly integrated word processing/data processing/graphics capability that will follow.

HPMenu provides the means for tying HP Office Products into a related set of capabilities which will allow HP to market the HP office packages and all the office products more effectively.

The operational value of HPMenu is that of making software applications more accessable to the typical office professional. HPMenu enhances the usability of the office system. It also reflects HP's commitment to developing products that make working easier for the typical office professional.

Where is HPMenu resident?

HPMenu is run as a program file from Pub.Sys, BUT it uses MENUCUST and LOCK files which must reside in the users group and account.

At the time of installation, the MENUCUST file is restored into MENUCUST.HPMenu.HPOFFICE. The first modifications to the menus are made at this time by the system manager to create the system specific default menus. When the users run HPMenu for the first time, the program checks for a MENUCUST file in the users group. When it is not found, the program copies in the system default from MENUCUST.HPMenu.HPOFFICE. After the default menu has been copied into the users group, the user can then use the menus and modify them to meet their group specific needs.

The LOCK file is created by the HPMenu program itself. It is named MLQ8X2Z. There must be a lock file in the users group in order for customization to occur.

Minimum Requirements

1 block mode terminal
MPE IV (Q MIT or later)
VPLUS (B.03.03)
ND,SF,IA,BA,FH capability in users' group

Features & Functions

Default condition

After installation, when the user first types in
:run HPMenu.Pub.Sys
the system default first menu will come up. The installation default menu is an office oriented one and includes all of the HP office products and includes some of the common MPE commands. At the time of installation, the system manager (or installer) deletes those functions which do not reside on their system. The result of this first editing are the system specific default menus and functions.

At the time of this first running, one file is put into the users group. This is the MENUCUST file. The LOCK file, named MLQ8X2Z is copied into the group only when customization in invoked. The lock file locks the whole group during customization. This protects the MENUCUST file when changes are being made to it. This way, two individuals customizing at once do not cancel each others changes. More about lock files and customization later.

The * next to a choice means that this is another menu. Office functions, MPE commands, and Special Function are menus themselves and contain those types of functions.

HPMenu contains two classes of operators: menus and functions. FUNCTIONS are those specific commands which invoke a process (e.g. :run HPDRAW.pub.sys). MENUS are lists of functions or lists of other menus which contain functions. A function can appear on more than 1 menu. In this model, functions are like atoms, menus are like molecules and menus with menus are compounds.

The default functions contained in each of the installation menus are as follows:

Office Functions
- HPWORD
- HPMAIL
- Editor
- TDP/3000
- Inform
- HPDRAW
- HPEASYCHART
- HPSLATE
- DSG/3000
- HPWORD Utilities

MPE Commands
- Editor
- Fcopy
- Listdir2
- Listeq2
- Listf
- Listf2
- ShowDev
- ShowIn
- ShowJob
- ShowMe
- ShowOut
- ShowTime

Other Functions
- Learn HPMAIL
- Learn Inform

Customization and Command Mode (MPE mode) are not available to the user in the default state. Customization and Command Mode have to be specified at the time HPMenu is invoked with run string parameters. This is so that naive or low security users will not have the ability to change the default menus or use the more powerful system utilities.

Customization

This is HPMenu's most powerful and versatile feature. It allows the user to easily and non programmatically change all of the functions and menus. Customization must be turned on at run time by using the run string parameters:

```
:run HPMenu.Pub.Sys;info="okcustom"
```

This will invoke the customization ability and F5 will appear on the Main Menu with Customize written on it.

NOTE: In order to customize you MUST have a lock file. This file is called MLQ8X2Z. DO NOT purge this file from your group. If this file is purged by mistake, HPMenu will create a new one.

Press Customize to enter the customization facility. The Main Customization menu shows the first 30 of all the known choices. By pressing Next Page, you can see the rest of the known functions if there are more than 30 defined. These functions have already been defined within HPMenu. The * next to a choice means that the choices are menus themselves.

You can create new menus or functions, edit or delete existing ones, change which menu appears when you first enter HPMenu, read functions in from an ASCII file (such as a UDC), or print a list of all existing choices and their definitions. You can also add functions without attaching them to a menu.

To add an existing function to a menu, type in the number of the menu in the box and press Create/Edit. To add a new function, type in its name in the box provided and press Create/Edit. The choice number and choice name options are mutually exclusive. Use either one OR the other. For more about how to use this menu see "Customizing HPMenu" in the Administration Manual.

After you have added your functions, you will probably want to define menus. HPMenu knows about a function as soon as it has been defined. You don't need to press End Custom before changing or adding menus.

HPMenu will then bring up the Classify New Choice menu.

This is where you tell it whether the new additions are functions or menus. For more information, see 2-22 through 2-32 of the Administration Manual.

You will have to define new functions by entering the run strings, file equations and parameters for all functions you wish to perform. Once the functions and menus have been defined, they can be accessed through the Selection menus.

Reserved Commands

There are several reserved commands which can be useful in defining functions. These commands are defined only within HPMenu and serve to add control information flow within HPMenu. The reserved commands are:

$Liston	$MPE	$Pause
$Listoff	$Parm i = text	$Wait
$Message	$Password	

For more information on these commands, see Appendix A in the Administration Manual.

**NOTE: HPMenu has an upper limit of 100 defined choices. These choices are both menus and functions. (e.g. you can have 30 menus and 70 functions, 50 menus and 50 functions etc). If you approach the limit, a warning is issued. When the limit is reached, customization will stop.*

Incorporating UDC's

Many customers have already defined their functions via UDC's. In order to save retyping all that information into HPMenu, there is a feature which allows you to incorporate your UDC files. This is the MERGE ASCII feature. MERGE ASCII is found on the Main Customization menu by pressing the OTHER KEYS twice. This brings up the Merge ASCII Menu.

The UDC file needs to be an ASCII file with the following features:

1. Function name must be <= 16 characters (not counting leading blanks)
2. Command line must be <=79 characters
3. Functions must be separated by a line with * in the first character position
4. UDC files must be kept unnumbered. Otherwise the numbers are read in as information.

Even if these conditions are not met, HPMenu will attempt to copy anyway.

If the file contains a name which matches a Known Choice, a warning is issued and the choice is NOT copied. After the entire file has been processed, HPMenu returns to the Main Customization menu. The user will then decide whether or not to add these known functions to new or existing menus.

There are several restrictions, however, when copying in UDC's. HPMenu cannot execute if/then/else sequences or parameter passing from UDC's. Also, since UDC's do not work within HPMenu, UDC's that call one another do not work either.

HPMenu does not perform any kind of checking on the UDC's it copies. If UDC's are copied that do not meet the requirements, they are not executed, or are executed improperly when invoked. The user should review his UDC's before copying them into HPMenu.

For more information see "Security Specifications" and Appendix B of the Administration Manual.

**NOTE: When copying in large UDC files, HPMenu may reach the 100 choices limit before completing the copy. At this point, HPMenu stops, gives a message "Unable to Merge entire file. Maximum choices now in use." and then*

returns to Main Customization. In order to find out where the copying process stopped, press PREVIOUS PAGE on the Main Customization menu until it displays the last page of choices entered.

List Functions Offline

From time to time it may be useful to obtain a hardcopy listing of all the choices defined in HPMenu. You can do this through the PRINT CHOICES feature in Customization. This will send a listing to the system line printer. The listing can be directed to another device if a file equation is issued prior to invoking HPMenu. The output is automatically directed to MENULIST;Dev=lp. To direct the output to another device, issue the following file equation PRIOR to running HPMenu. (N.B. HPMenu runs nobreak)

```
:File MENULIST;dev=ldn
```

The printout shows all the known choices in the same order as they appear in Main Customization. The menus are shown with all their choices and the functions are shown with their full definitions.

*NOTE: *Offline listings are only available through Customization. The default user runs with no customization and cannot use this feature.*

Parameter passing

The purpose of parameter passing is to allow the customizer to prompt for data from the user and to set up function specific parameters which can be changed as needed. The commands which allow parameter passing can only be used within the function definition facility in customization and are not recognized anywhere else in the program.

The customizer can define 10 parameters within a given function $Parm 0 - $Parm 9. The parameters will prompt for and fetch information from the user at the time when the function is invoked through PERFORM CHOICE. Information which is passed as a parameter is kept in storage for the duration of the function's execution and can be called up repeatedly. The parameters can be set up with default information, or they can store the user's input as the parameter, or they can be set up with a default value but be overwritten with user input information.

The information to be substituted with parameter input must be enclosed in %'s when the function is defined.

EXAMPLES OF FUNCTION DEFINITION

1. Parameter set up with default information

```
$Parm 0 = @,1
Listf %0%
```

When executed, this function will perform a LISTF @,1.

2. Parameter is input by user and saved for future use

```
$Parm 1 = text
Listf %1Enter a filename%
```

When executed, this function will first prompt the user for a file name like this:

```
Enter filename>
```

After the user enters a file name, the function will perform a LISTF on that file name. Furthermore, $Parm 1 is now set to the file name entered. If $Parm 1 is used again later on in the

function without prompt text, the user entered file name will be passed. The parameter number will not appear on the terminal screen even though it appears between the %'s. When the user is prompted for this information, he has no way of knowing what the initial values are.

3. Parameter initial value set and user has input option

```
$Parm 2 = 2
File slp;dev=pp,%2 Enter # of copies%
```

When executed, this function will first prompt the user for a number. The user can either type in a number or hit the carriage return. If the user hits carriage return, the initial value is used and a file equation is issued. If the user types in a number, a file equation is issued for that number of copies and $Parm 2 is set to the number typed by the user.

It is also possible to get information from the user without saving it. You can simply define a function such that:

```
Listf %Enter filename%,1
```

The user enters a file name but it is not kept. In this case, however, the user MUST enter something. A carriage return is not sufficient input. In this case HPMenu will continue to prompt the user until something is typed in.

The value of parameters set in one function CANNOT be passed to another function. $Parm 0-9 must be set separately in each function where they are used.

Command Mode

The Command Mode feature lets the user exit from menu choices and enter a command environment. Command mode allows you to issue and run most MPE commands from within HPMenu. This feature is not offered in the default state. To run HPMenu with the command mode capability, you must start up HPMenu as follows:

```
:Run HPMenu.Pub.Sys;info="OKMPE"
```

If you wish to run with both Command Mode and Customization, you must run HPMenu like this:

```
:run HPMenu.Pub.Sys;info="OKCUSTOM;OKMPE"
```

When you press the COMMAND MODE softkey on the Selection menu, you will get a ! as a prompt. This lets you know that you are in command mode, but not out of HPMenu. From this prompt you can use most MPE commands. There are some restrictions however.

The following commands CANNOT be used within HPMenu.

BYE	MRJE	SYSDUMP
DEBUG	RJE	VINIT
DSCOPY	SETCATALOG	VSUSER
HELLO	SHOWCATALOG	UDC's

These commands are only accessible through the command interpreter.

The following functions are available through HPMenu, BUT operate with restrictions. (See Internals and Appendix B in the Administration Manual for more specifics.)

BASIC	COBOLPREP	FORTPREP	RPGPREP
BASICCOMP	EDITOR	PREP	SEGMENTER
BASICGO	FCOPY	PREPRUN	SPL
COBOL	FORTRAN	RPG	SPLGO
COBOLGO	FORTGO	RPGGO	SPLPREP

In Command Mode, MPE commands are performed as in MPE and all MPE error messages are returned.

When you wish to return to the menus, type RESUME. This will bring up the menu from which you pressed the COMMAND MODE key.

UDC's cannot be accessed from within HPMenu. If a user wants to execute UDC functions from HPMenu, they must define this function in customization. This can be done either by defining the function as a new function, or copying in the UDC file using the MERGE ASCII feature.

*NOTE: *Caution should be used when invoking compilers from HPMenu. This is where the fatal errors occur. See Compiler Access in the Internals part of this document for more details.*

Current Object Store

HPMenu now provides the capability to share information between the applications running under it. The Current Object Store is an extra data segment, created by HPMenu, that contains:

--ScratchPad Area, a 256-word area used for applications to communicate with each other, or to store application dependent information. An application can perform any operation on a value in the ScratchPad area as long as the value can be expressed in a 16-bit word.

--User Password and Name, set and retrieved by HPMENU or applications.

--Current Object, contains values that can be passed from one application to another. Twenty of these are reserved for HP products (chart and file names, etc.), while five are user-definable. These can be accessed via the new $CURRENT n function, as well as intrinsics. This area also contains values for configuration specifications for plotters, accessed by the $CONFIG n functions. More on these later.

(NOTE: This means that the HP Office products are evolving to all share this common data store--i.e. INCREASING PRODUCT INTEGRATION!!)

--HPMENU Status, used to determine if an application was invoked by HPMENU. Only applications invoked by HPMENU can access the Current Object Store.

Thus, information can be passed from one program or application to another without the user intervening to supply the information themself.

Some HP Office Applications running under HPMenu have been designed to access the data in the Current Object Store. For example, if an HPMenu user runs DSG/3000 and plots a chart, the values supplied for the plotter (logical device number, plotter type, etc.) will be saved in the Current Object Store. If the user then runs HPDRAW, HPDRAW will retrieve this current data from the extra data segment and use it as the default plotter specifications for plotting in HPDRAW. This way, the user does not have to re-enter this informatio for every plot, but they still can modify the specifications as necessary.

New HPMenu reserved commands and extensions to the $PARM command allow easy access to the Current Object Store when they are used in building HPMENU functions. These commands allow direct access into the extra data segment. For example, when a user runs HPMENU, their USERID and USERPASS can be recorded in the Current

Object Store. Then a function designed for running HPDesk could retrieve this information from the Store. The $PARM functions could be set to the USERID and USERPASS and the function could call HPDesk using $PARMs in the INFO= string to pass the data. Thus, the user would not have to enter their name again.

A third use of the information saved in the Current Object Store is to access it programmatically through HPMENU intrinsics. An application which could be used to enhance system security would retrieve the user name and password from the Store. It could check this data against a security matrix to ensure the user has proper capability to execute the function before continuing in the program.

This Current Object Store feature greatly enhances HPMenu's feature set and provides for a very broad range of applications. It adds the flexibility needed to increase the user friendliness and reduce entry of redundant information.

HPMenu Intrinsics

Applications invoked by HPMenu may take advantage of a set of intrinsics which allow the passing of information between those applications. Only applications running under HPMenu can access the Current Data Store, the vehicle for the communication. This Store is a shared extra data segment which contains a scratchpad area of 256 words, as well as the user's name and password, 20 current object names reserved for use by HP and five current object names for use by users, and an area which is used to stor seven configuration values. An application could set information in this Current Object Store, and subsequent programs could retrieve the data, using it perhaps as input data or as control information.

All of the intrinsics but one (OLMenuStatus) require the passing of an array of words. The array is 23 words long and must be in the global area of the application. All parameters are passed by reference. See Appendix D of the new HPMenu Reference Manual for conventions used to define the parameters.

OLMenuStatus is used within an application to determine if the application itself was invoked by HPMenu.

This intrinsic should be called before any other HPMenu calls are made. Only applications invoked by HPMenu can use the current objects, so failure here should flag an error condition.

OLEnableCurrObj is used to initialize all other HPMenu intrinsics except OLMenuStatus.

This intrinsic should be called immediately after OLMenuStatus and before any other HPMenu intrinsics.

OLGetCurrObj returns the value of the specified current object.

OLSetCurrObj sets the value of the specified current object.

These intrinsics are used to set/return values in the current object area of the Current Object Store so data can be passed between user applications and also between a user application and several HP Office applications. These values can also be referenced by the new $CURRENT reserved word. Details on referencing these objects are provided in Appendix A of the HPMenu Reference Manual and in the "New Reserved Commands" section of this document.

For example, an application could access a data base to create a data file and set the file as Current Object 8 (DSG data file) for a particular Chart (Current Object 6) in a Chart File (Current Object 5). Then a DSG/3000 programmatic application could reference these objects in the Current Object Store and plot the chart without further operator intervention.

OLGetPassword returns the eight-character password stored by HPMenu.

OLSetPassword sets the password string in the Current Object Store.

OLGetUserId returns up to 46 characters containing the user id.

OLSetUserId sets the user id string in the Current Object Store.

Applications can retrieve or set the user's password and name using these intrinsics. The user name can include node and location codes for HPDesk. For some applications, it may be preferred that the user avoid re-entering their name and password for each application. In these cases, once the information has been entered into the Current Object Store, it can be accessed by the other applications. If name and password are supplied by the user when HPMenu is invoked, these values will automatically be entered into the Current Object Store. If no name or password is specified, HPMenu assigns blanks. When the user exits HPMenu, blanks are substituted for the password in the Current Object Store. Applications requiring high security should always require a password.

OLGetOneScratch returns the specified value from the scratchpad area.

OLSetOneScratch copies a value passed by an application into the scratchpad area.

The scratchpad is the application's communication area. The application can put information into the scratchpad by Index to be used in later communications or as storage for application-dependent information. These intrinsics provide much flexibility for tailoring application programs to share as much information as possible.

If a series of application programs are to be run in a single session, these intrinsics could be used to set values in the Current Object Store. Subsequent programs could access this information and use it as input data or to control program flow.

Note that only numeric data can be stored in the scratchpad area. There are five current objects in the current object area that can be used for passing textual information.

OLSetConfig sets one of the configuration values.

OLGetConfig returns one of the configuration values.

The values stored in the configuration area specify information for certain HP office products. They include logical device number, HPIB number, recovery count, expertise level, medium type, page size, and position. Again, these fields could be set by an application program so the operator does not have to enter the data each time.

Note that the Current Object Store is a shared extra data segment. Thus, it remains active until the end of the session. So even if the user exits from HPMenu, the data will be stored for use if they should return to HPMenu in the same session. For security, the password is changed to blanks upon exiting HPMenu.

The key to accessing the extra data segment is the required VAR parameter "WORDS." This is the first parameter in all the access intrinsics and provides the area which can be used for data segment communication. This space must be in the global area. See Appendix D of the HPMenu Reference Manual for examples of calling these intrinsics in Pascal, SPL, Basic, COBOL, and Fortran.

Although the user applications do not need to specify length or offset into the extra data segment when using these intrinsics, here is the structure of the Current Object Store.

* * * For your information * * *

The 'OL' prefix on all the intrinsic names stands for

OFFICE LIBRARY.

0 to 255	Scratchpad Area	Len=256 Words
256 to 278	UserId string	Len=46 Bytes
279 to 282	User Password String	Len=8 Bytes
283	Logical Device Number	Len=1 Word
284	HPIB channel number	Len=1 Word
285	Recovery Count(HPDRAW)	Len=1 Word
286	User Expertise Level	Len=1 Word
287	Medium	Len=2 Bytes
288	Page Size	Len=2 Bytes
289	Position	Len=2 Bytes
290 to 307	Ascii Current Object	Len=36 Bytes
308 to 325	Word Document Object	Len=36 Bytes
326 to 343	Slate Document Object	Len=36 Bytes
344 to 361	Chart File Name Object	Len=36 Bytes
362 to 379	Chart Name Object	Len=36 Bytes
380 to 397	EzChart File Name	Len=36 Bytes
398 to 415	Data File Object	Len=36 Bytes
416 to 433	SD Data File Object	Len=36 Bytes

434 to 451	HPList File Object	Len=36 Bytes
452 to 469	Drawing File Object	Len=36 Bytes
470 to 487	Drawing Name Object	Len=36 Bytes
488 to 505	Figure File Object	Len=36 Bytes
506 to 523	Figure Name Object	Len=36 Bytes
524 to 541	Roster File Object	Len=36 Bytes
542 to 569	Customization Object	Len=36 Bytes
570 to 587	Environment File Object	Len=36 Bytes
588 to 605	Raster File Object	Len=36 Bytes
606 to 623	Device Model Number	Len=36 Bytes
624 to 641	Spare 1	Len=36 Bytes
642 to 669	Spare 2	Len=36 Bytes
670 to 687	User 1 Object	Len=36 Bytes
688 to 705	User 2 Object	Len=36 Bytes
706 to 723	User 3 Object	Len=36 Bytes
724 to 741	User 4 Object	Len=36 Bytes
742 to 769	User 5 Object	Len=36 Bytes
770 to 1022	Reserved for Expansion	Len=252 Words

New Reserved Commands

HPMenu has reserved commands that are used in function definitions. The commands can be used to control the operation of a function ($PAUSE, $WAIT, $LISTON, etc.) or to pass information within the function ($MESSAGE, $PARM i, etc.).

Several new reserved commands have been added to access the Current Object Store, and the $PARM i command has been expanded. This broadens the data sharing capabilities not only between applications accessing the Current Object Store, but also between functions within HPMenu. Thus, a reserved command set in one function will retain its value when executed by another function.

$CURRENT n=text

This command sets the designated current object in the Current Object Store to the first 36 characters found in "text". The current object to be set is indicated by the value n where n is an integer from 1 through 25 as indicated in the following table:

Number	Object Name	
1	TDP File	
2	WORD Doc	
3	SLATE File	
4	Chart File	
5	Chart Name	
6	EZChart	
7	Data File	
8	SD Data File	
9	HPList	
10	HPDraw File	
11	Drawing Name	
12	Figure File	
13	Figure Name	
14	Roster File	
15	Custom File	
16	ENV File	
17	Raster File	
18	Device Type	
19	Spare 1	(For future
20	Spare 2	HPMenu expansion)
21	User Object 1	
22	User Object 2	(21 thru 25 for
23	User Object 3	user application)
24	User Object 4	
25	User Object 5	

The User Objects provide a store for textual information to be shared between application programs. Remember, the Scratchpad area can only be used for numeric values.

$CONFIG n=value

This command sets the configuration item specified by n to "value". These items specify characteristics of the output device and operational features of certain HP office products.

Item	Number
Logical Device number	26
HPIB number	27
Recovery Count (HPDRAW)	28
Expertise Level	29
Medium	30
Paper Size	31
Position	32

This example shows how to set up an HPMenu function that will call EZChart and set the initial specifications to plot chart PIECHART with the SD data file PIEDATA to transparency on a 7221C plotter on logical device 66:

```
$CURRENT 6=PIECHART.PUB.DEMO
$CURRENT 7=PIEDATA.PUB.DEMO
$CURRENT 18=7221C
$CONFIG 30=T
RUN EZCHART.PUB.SYS;info="ldev=66"
```

Note that EZChart ONLY accepts LDEV information through the INFO= string Thus, the $CONFIG 26 reserved word can not be used to pass a logical device number to EZChart, although it can be used with DSG/3000 and HPDraw.

$SCRATCH n=value

This command sets the scratchpad location specified by n (1 thru 256) to a number contained in "value", providing the capability to pass information to an application program via the Current Object Store.

Again, the scratchpad area can be used for communication between a series of programs run in a single session. The values can be retrieved from the Current Object Store and passed to the RUN command in the function via the PARM= or INFO= string.

$USERID=text

This command sets the user identification string to the first 46 characters found in text.

$USERPASS=text

This command sets the user password string to the first 8 characters in text.

$CLEAR

This command sets all the textual values in the Current Object Store to blanks, and all the numeric values to zero (0). The one exception is the object name corresponding to the HPMenu customization file name (number 15). This object name remains as before to prevent reliability problems with HPMenu.

Perhaps a user wants to be able to return to EZChart sometimes with their chart as the default and other times with the standard defaults displayed. Two functions could be specified--one containing the standard RUN command (so EZChart will access the Current Object Store) and the other containing the $CLEAR command before the RUN so that the Current Object Store will not have the past information. Thus, the user could specify which default they wanted by selecting the appropriate function.

Current EZChart	Fresh EZChart
RUN EZCHART.PUB.SYS	$CLEAR RUN EZCHART.PUB.SYS

$PARM i=Keyword

This extension to the PARM command is used to retrieve the values of the Current Object Store. You may place the Current Object indicated by Keyword into a $PARM storage location and pass this value to an application, probably with the PARM= or INFO= parameters on the RUN command. The available Keywords are:

```
CONFIG n
CURRENT n
SCRATCH n
USERID
USERPASS
```

For example, when a user runs HPMENU, their USERID and USERPASS could be recorded in the Current Object Store. The function designed for running HPDesk could set $PARM functions to the USERID and USERPASS, then call HPDESK using the INFO= string to pass the data. Thus, the user would not have to enter their name again. The function definition for HPDesk would be:

```
$PARM 1 = USERID
$PARM 2 = USERPASS
RUN HPMAIL.HPMAIL.SYS;PARM=1;INFO="%1%:%2%"
```

*** BEWARE of upper and lower case in password. ***

If no password is set for the user in HPDesk, it does not matter whether or not the second parameter is passed in the INFO= string. However, if the password is set, this parameter MUST be included in the INFO= string. HPDesk will not accept the name from the INFO= string then prompt the user for the password.

Security for individual functions

In addition to the normal MPE file security, HPMenu can protect specific functions. This can be useful in situations where members of the same group do not have the same security levels for application usage.

HPMenu allows passwords to be specified for individual functions defined in HPMenu. When the function is invoked, HPMenu prompts for the passwords. Terminal echo is turned off at this point, and the user gets three trys to correctly type in the password. If the password is not correctly entered after the third try, HPMenu returns to the most recent menu.

You can specify a lockword for a Customization file. HOWEVER, you must issue a file equation specifying the lockword PRIOR to running HPMenu. (e.g. :file menucust=menucust/lockword) If this is not done, the lockword is requested when the terminal is in block mode. It is very difficult to respond to this. The usual outcome is that the terminal hangs and the session has to be aborted.

Passworded functions are much more suitable for security than passworded MENUCUST files.

*NOTE: *Any user who can customize can list all of the functions and passwords. Also, if the $Liston option is in effect when the password check is performed, HPMenu will list the correct password before prompting for it.*

Recovery after a crash

HPMenu goes to great lengths to protect the user from data loss in the Custom file during a system crash. Menu data loss can occur only if the user is customizing at the time of the crash. The user can choose the level of backup desired. There are three options:

Transact- this option backs up the file after every transaction made during customization. Performance during customization is slower because of this.

On_Save - backup occurs only when the user leaves customization.

Never - no backup is done during customization. File errors can occur if the file is being written to when the crash occurs. This is the run-time default.

If a crash should occur when either Transact or On_Save is specified, one or more recovery files may be left in the user's group. These files are named M0Q8X2Z_ (where _ is a alphanumeric) and MRQ8X2Z. These files should not be purged. When HPMenu is run again after the crash, it will look for these files and recover on its own. The user may receive a message that recovery is going on.

During this process, the recovery files are located, the new information is incorporated into MENUCUST, and the recovery files are then purged. The user should not see the recovery files after HPMenu has been run again.

If Never, the default, is active during a crash, it is possible that the MENUCUST file will be damaged and unusable. This will occur if the file is being written to during the crash. The only recovery for this situation is to RESTORE an old copy of MENUCUST and start over.

Internals

File structure

The file structure for MENUCUST is dynamic. It is roughly divided into 4 main parts: header, list of choices, definitions of choices and ender.

You can examine a MENUCUST file very easily by fcopying it to the screen or a file.

The first line of the header contains a date and time as returned by the CALENDAR and CLOCK intrinsics. The second line contains the count of the number of choices found in the file.

The list of choices is the second major division of the file. It contains information on the number of the choice in Main Customization and whether it is a MENU or CMD (i.e. function).

In the third section are the details of each of the above listed choices. It is indexed into the second section by number of choice as appears in Main Customization. Each of the menus are listed with the index of all their choices. Each of the functions are listed with their full definitions.

The last part lists the number of the user's selected first menu and an end of file marker. The end of file marker is *****. If this is missing then the file is declared invalid by the program.

The following is a dump of the default MENUCUST file.

```
      -23030   185536773
          27
Main Menu                      MENU
HPSLATE                        COMD
HPWORD                         COMD
HPMAIL                         COMD
HPEasyChart                    COMD
HPDRAW                         COMD
DSG/3000                       COMD
Inform                         COMD
MPE Commands                   MENU
Fcopy                          COMD
Listf2                         COMD
ShowDev                        COMD
ShowIn                         COMD
ShowMe                         COMD
ShowOut                        COMD
ShowTime                       COMD
Editor                         COMD
ListDir2                       COMD
```

```
ListEq2                         COMD
Office Functions                MENU
TDP/3000                        COMD
ShowJob                         COMD
HPWORD Utilities                COMD
Other Functions                 MENU
Learn HPMAIL                    COMD
Learn Inform                    COMD
Listf                           COMD
 MENU     3
          20
           9
          24
 COMD     1
run hpslate.pub.sys
 COMD     1
run HPWORD.pub.sys
 COMD     1
run hpmail.hpmail.sys;lib=g
 COMD     1
run ezchart.pub.sys
 COMD     1
run hpdraw.pub.sys
 COMD     1
run graph.pub.sys
 COMD     1
run inform.pub.sys
 MENU    12
          17
          10
          18
          19
          27
          11
          12
          13
          22
          14
          15
          16
 COMD     1
run fcopy.pub.sys
 COMD     3
$parm 0=@
listf %0Files to be listed (<return> for all) %,2
$wait
 COMD     3
$parm 0=
showdev %0Enter device class or number (<return> for all) %
$wait
 COMD     2
showin
$wait
 COMD     2
```

```
showme
$wait
 COMD     2
showout
$wait
 COMD     2
showtime
$wait
 COMD     1
run editor.pub.sys
 COMD     1
run listdir2.pub.sys
 COMD     2
run listeq2.pub.sys
$wait
 MENU    10
           3
           4
          17
          21
           8
           5
           2
           6
           7
          23
 COMD     1
run tdp.pub.sys
 COMD     2
showjob
$wait
 COMD     1
run hpword.pub.sys,wordutil
 MENU     2
          25
          26
 COMD     1
run training.hpmail.itf3000
 COMD     1
run preview.inform.itf3000
 COMD     2
listf
$wait
           1
*****
```

Debug/Tracing

HPMenu contains a trace facility very similar to the one used by DSG. It allows a user with two terminals to the same system to monitor the calls made by the program while in progress. This is a very useful tool when trying to troubleshoot a problem. Most often a user can determine which procedure is the one causing the problem.

In order to obtain the trace, log off the terminal on which you want to have the HPMenu menus appear. The terminal on which you run the trace should be logged on to the group and account where the problem occurs.

You can obtain the trace listing in one of two ways, on the terminal or as a disc file.

For a disc file listing, issue the following file equation.

```
:file DEBUGDEV = disc;save
```

For a listing on the terminal, issue this file equation.

```
:file DEBUGDEV;dev=logical device # of terminal
```

You also need to issue a file equation naming the logical device number of your menu output terminal. This is done with the following file equation.

```
:file MENUDEV;dev=ldn of logged off terminal
```

If the terminal to be used for MENUDEV is busy, the process will abort.

The whole sequence to run the trace is as follows:

```
:file menudev;dev=ldn
:file debugdev=$stdlist    (or disc;save)
:run hpmenu.pub.sys;info="DEBUG" (or DEBUG;OKCUSTOM if necessary)
```

When this first starts up nothing happens for a few seconds. This is because the message catalog is being opened and read. There is no debug information in the message catalog so none of this will show up. Once this has been completed, the terminal opening procedures will appear and it will proceed as normal.

Fatal errors and recovery from them

There are two errors in HPMenu and one is more fatal than the other.

If :EOD or :EOF is issued in Command Mode, HPMenu will try to recover but will abort.

If :EOD or :EOF is issued in a function definition, then they are treated as Unknown Commands and the program returns and lets the user make another choice.

The recovery for the first case is to run HPMenu again. The remedy for the second case is to log on again and then run HPMenu.

If :EOD or :EOF are defined as functions within Customization there may be a dramatic reaction with flashing screens. This condition persists until the session is aborted. HPMenu runs nobreak so the session has to be aborted from another terminal.

*NOTE: *If the user does not have 103 free sectors of disc space the first time they try to run HPMenu, it will abort. This space is necessary for copying in the MENUCUST file. The remedy for this is to increase the available disc space.*

Compiler access

In order to successfully run compilers from HPMenu, you must RUN them (e.g. :run spl.pub.sys). If you run the compilers interactively (e.g. SPL) then there is a problem. This is a problem because most compilers use :EOD or :EOF to quit. As was mentioned earlier, this is a fatal error. HPMenu can recover from the : but $Stdin becomes closed. HPMenu cannot then call any other program which requires $Stdin. HPMenu opens $Stdinx which allows the user to exit gracefully, but that is about all.

The suggested way to run compilers from HPMenu is as follows:

```
$Parm 0=$Stdlist
file PASTEXT=%Enter source file name%
file PASLIST=%0 Enter list file name%
run Pascal.pub.sys;parm=3
```

Any ABORT's of the compilers will abort HPMenu as well because they call QUITPROG.

Segmenter

The segmenter must be run the same way as the compilers, except that no file names are needed to run it. The segmenter function should be defined:

```
run SEGDVR.pub.sys
```

An abort here will abort HPMenu.

Prep

To prepare a program, you need to use the PREPARE command from within SEGMENTER.

Installation

Why does it take so long to load?

In order to make the product localizable, HPMenu uses message catalogs for warning and error messages as well as function key labels. When HPMenu is first invoked it reads in all of the message catalog and key labels to insure optimal performance for the duration of the session. There are 4 message catalogs involved. The information in one is read and the other 3 are opened. In addition VPLUS, MPE, and HPMenu are initialized. The forms files are opened and the appropriate localized text read into the forms. Finally, the current customization file is opened and read onto the stack in the global area. This insures optimal performance in invoking known functions and menus. (see previous pages)

The net result of this process is a lot of IO at startup. The processes discribed above cause a 90-120 second delay at startup. HOWEVER, once the program is loaded, screen to screen performance, message and error reporting, and function execution occur at maximum possible speed.

Since HPMenu is a menu processor, it is assumed that it will be invoked ONCE during the day, usually in the morning, and then left running for the rest of the day. Under these conditions, the product performs extremely well, with little if any delay. HPMenu was not designed to be run as an application or subsystem with multiple entrances and exits during the day.

Installation procedures

HPMenu version A.01.00 will be available to customers only on MIT tapes. There will be no SE or customer installation of the product.

After the MIT tape is installed, the system manager shouldd alter the system default menu. Functions for non-resident office products should be deleted. Other system-wide functions can be added.

All of the initial customization files will be copied from MENUCUST.HPMENU.HPOFFICE so it is important that this file be correct.

In order to customize this system default file, use the following procedure.

1. :Hello mgr.hpoffice,hpmenu
2. :run HPMENU.pub.sys;info="okcustom"
3. Make your changes, (see Administration Manual)
4. Bye

HPMenu is now ready to run. If you wish,make up some udc's for your novice users. You may also desire to personalize some users' customization file. To do this do the following:

1. :Hello manager.useracct,usergroup
2. :Altuser username;cap=ia,ba,ph,nd,sf;home=usergroup

3. :Hello username.useracct
4. :run hpmenu.pub.sys;info="okcustom"
5. Make your changes (hint: keep info in customization small)
6. Set up a logon udc

Your user now has a personalized customization file. (It resides in his group). Multiple customization files per group requires a file equation for MENUCUST, see Administration M anual)

Limitations

Limitations

1. :EOD and :EOF cause aborts

2. 40 sessions running HPMenu take 120+ PCB's. The user's system should be configured to have 160+ PCB entries

3. 100 choices of both functions and menus

4. No nesting of HPMenu within HPMenu

5. 30 commands per function

6. 30 entries per menu

7. Must have 103 sectors of free disc space the first time HPMenu is run to accommodate the MENUCUST and LOCK files

8. While copying in large UDC files, HPMenu may run out of room and stop

9. There are no checks on UDC's copied in

10. No UDC's can be used from within HPMenu

11. Does not run in batch mode

Appendix

Setting up menus for security

There are some applications that may require users to be "locked out" from all but predefined functions. HPMenu can be used in this type of situation very successfully.

HPMenu can be called from a logon UDC with a BYE option. This would insure that when a user logged on they would only be able to perform tasks predefined within HPMenu and when they exited HPMenu they would be logged off from the system.

In this situation it is important that the user be set up with no customization and no mpe.

**NOTE: There is a change to the heirarchy of execution of UDC's in the Q MIT. UDC's are executed with strict precedence such that account udc preceeds user's and system udc preceeds account. If the system manager sets up a system UDC to logon/run HPMenu/bye then no other UDC's will be accessed.*

Using MENUCUST files from other groups

It is very easy to run HPMenu and access customization files from other groups. This can be done with file equations e.g.:

```
:file menucust=menucust.alpha.mom
:run hpmenu.pub.sys
```

In this situation you CANNOT customize, even if the CUSTOM option is turned on.