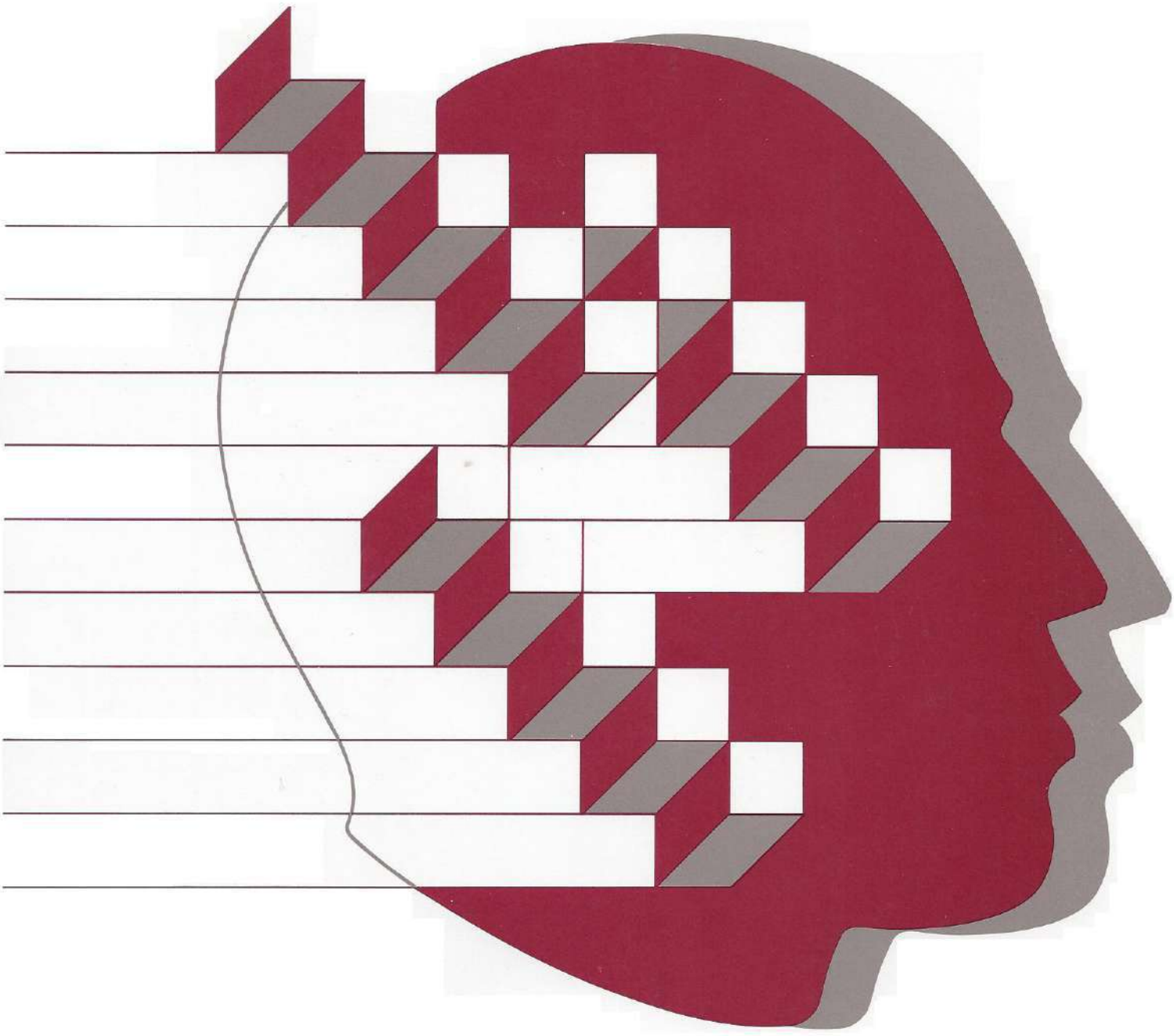
HP 3000 Computer Systems

HEWLETT PACKARD

HPToolset Reference Manual



HP 3000 Computer Systems

HPToolset

Reference Manual

19420 HOMESTEAD RD., CUPERTINO, CALIFORNIA 95014

Part No. 32350-90001

Printed in U.S.A. 07/82

NOTICE

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

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

LIST OF EFFECTIVE PAGES

The List of Effective Pages gives the date of the current edition and of any pages changed in updates to that edition. Within the manual, any page changed since the last edition is indicated by printing the date the changes were made on the bottom of the page. Changes are marked with a vertical bar in the margin. If an update is incorporated when an edition is reprinted, these bars are removed but the dates remain.

First Edition. . . . . July 1982

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. This edition does not change.

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

First Edition. . . . . July 1982 . . . . . 32350A.00

PREFACE

The HPTOOLSET REFERENCE MANUAL documents a set of programming tools designed to aid the COBOL programmer in developing programs on the HP 3000.

HPTOOLSET is a program development system containing an integrated set of tools that can aid you in all phases of program development. It consists of an Editor to aid in program creation and modification, function keys that compile and prepare source files for execution, a Symbolic Debug facility to execute, test and debug programs on-line, and a Workspace Manager to manage all the files that make up your program.

This manual is primarily for use by a programmer with a working knowledge of COBOL. It does not teach the COBOL language. New HP 3000 users as we as the most experienced HP 3000 users, can benefit from TOOLSET's easy to use tools and the resulting increase in programmer productivity. The following documentation may provide additional assistance for those users programming on the HP 3000.

USING COBOL	32213-90003
COBOL II REFERENCE MANUAL	32233-90001
LEARNING COBOL II COURSE	22832A
MPE COMMANDS REFERENCE MANUAL	30000-90009

NOTE:

HPTOOLSET is a new product and is not intended to be compatible with COBOL 68, EDITOR/3000, DEBUG or any other programming tools currently available on the HP 3000.

CONTENTS

Section 1 Page

Section 2 Page

CONTENTS (continued)

Section 3 Page

Section 4 Page

Section 5 Page

CONTENTS (continued)

Section 6 Page

Section 7 Page

CONTENTS (continued)

ILLUSTRATIONS

Title **Page**

ILLUSTRATIONS (continued)

Title **Page**

SECTION 1

OVERVIEW

HPTOOLSET is a program development system that consists of an integrated set of software tools or utilities that aid in program development by minimizing the distinctions between the Creation, Translation, and Execution stages of program development. Reduction in development effort is possible because TOOLSET contains its own Command Interpreter. Unlike the MPE Command Interpreter where the composing, compiling, and debugging are handled by separate subsystems that must be operated independent of each other, the TOOLSET Interpreter makes all its software tools available for use at all times. This means, for instance, that one program can be compiling while you are editing a second program.

The features that make up HPTOOLSET are:

User Interface

Access to, and communication with, TOOLSET is accomplished through the use of commands and function keys, and information is displayed in either a command or message 'window'. TOOLSET allows you to alternately view information from different TOOLSET activities; for example, you can look at a program source listing and then a Symbolic Debug dialogue.

TOOLSET provides a HELP Facility that allows you to determine the purpose, syntax, and definition of TOOLSET commands and options, and the cause and corrective action for TOOLSET errors. The HELP facility makes it easy to learn HPTOOLSET simply by using it.

Workspace Management

TOOLSET manages the various files you use in the development of programs with a WORKSPACE. A Workspace is a collection of MPE files used for the development of one program, including the Sourcefiles, USLfile, and Program file.

The HPTOOLSET Workspace manager lets you assign multiple source version numbers so that when you make changes to a source file they are saved as a version of one file, and not as multiple files. These versions can be later accessed individually, shared, listed, or compiled.

Editor

A screen Editor is provided for easy composition and modification of your programs. If you are programming in a language other than COBOL, the HPTOOLSET Editor will allow you to set tabs and margins appropriate for the particular language you are using. If you are using COBOL, the appropriate format and function keys are set for you.

Program Translation

Compilation and preparation no longer need be explicitly performed by going through the HP 3000 Compiler and Segmenter. TOOLSET keeps track of which compiler and which files to use and performs program translation for you.

Compilation errors are displayed on your screen, and can easily be corrected by accessing the TOOLSET EDITOR where the file will be positioned at the offending point in the program. TOOLSET keeps up-to-date compilation listings on-line which you can display at any time, minimizing the need for hard copy listings.

Symbolic Debug

TOOLSET provides a number of powerful tools that enable on-line program testing and debugging:

- BREAKPOINTS can be set at section and paragraph heads, and at the beginning of lines.
- ASSERTIONS, such as whether a data value has changed, can be checked at paragraph and section heads.
- DATA ITEM VALUES can be displayed and modified, using source language identifiers and formats.
- PROGRAM FLOW and DATA ITEMS can be TRACED on a paragraph basis.

ACCESSING TOOLBOX

Hardware Requirements

HPTOOLSET operates on any HP 3000 Series III, 40, 44, and 64 which has a minimum of 1 megabyte of memory and runs MPE IV. To provide the needed visual and interactive user interface, one of the following terminal types must be used: HP2622A, HP2624, HP2626, HP2645, HP2647A.

Software Requirements

TOOLSET operates as an MPE IV subsystem in the PUB group of the SYS account. HPTOOLSET runs COBOL II and not COBOL/3000.

VPLUS Requirements

Version B.02.01, or subsequent versions, of VPLUS must be used to run TOOLSET. When debugging programs that reference VPLUS, you need to use two terminals because VPLUS uses block mode and TOOLSET uses character mode. To use a second terminal, specify a terminal file name in TERMFILE. Before running your program, do a file equation referencing the terminal file name you specified.

For example, if you specify TERM2 in VOPENTERM, you would enter the following file equation and reference it when running your program.

```
:FILE TERM2,NEW; DEV=LV; ACC=INOUT
:RUN
```

where LV is the logical device number of the terminal.

Symbolic Debug Requirements

The HPTOOLSET Symbolic Debug feature requires the KSAM (Keyed Sequential Access Method) utility.

Batch

HPTOOLSET can be run in batch mode on 2640, 2621, or HP125 terminals, or in a job. Menu driven commands such as HELP, SHOW and SET cannot be used, nor is the REDO command available, when running in batch mode. The Visual screen Editor is not available, but line editing can be done such as adding and listing.

Multipoint Terminals

All TOOLSET features are available with multipoint terminals except the Visual mode of editing and reading files.

Security

You manage your own source file versions, but MPE security at account, group, and file levels is still in effect. TOOLSET allows you to share and lock source files with other users, but you must have LOCK access at the account, group, and file level.

COMMANDS

Syntax

HPTOOLSET commands have the general format:

```
COMMAND   [operand [delimiter operand]...]
```

Items enclosed in brackets [] are optional; braces {} indicate that one of the enclosed options must be picked. An ellipsis (...) means that the items enclosed in the preceeding pair of parenthesis, brackets or braces may be repeated as many times as necessary.

Reserved words appear in upper case letters, and user-supplied words in lower case. Certain reserved words can be optionally included as delimiters for readability, and are enclosed in vertical bars ||. Besides these optional words, a blank, blanks or a comma can also be used to delimit operands in a command.

Defaults for TOOLSET commands are either underlined or specified in parameter definitions.

Abbreviating

TOOLSET reserved words may be abbreviated. The number of letters required is the minimum number necessary to ensure that the command will be uniquely identified within the current context. You may enter all reserved words in the lower case.

Certain command reserved words that are used frequently can be abbreviated even further. For example, the commands ADD and AT are abreviated as 'AD' and AT for uniqueness, but since ADD is used frequently, 'A' is accepted. Command abbreviations are indicated in their syntax as they are discussed. See Appendix A for a list of all HPTOOLSET commands and their abbreviations.

Concatenating

Commands may be strung together in order to enter a sequence of commands at the same time. To use more than one TOOLSET command, follow each command except the last with a semi-colon (;). For example:

```
COPYFILE MYFILE TO NEWFILE ; EDIT NEWFILE
```

You cannot concatenate a TOOLSET command after an MPE command. If you are stringing together TOOLSET commands and MPE commands, the MPE command must be at the end of your list.

If an error occurs during the processing of a command contained in your string that causes the command to fail, the remainder of the commands in that sequence are not be executed. The same is true if a command in your string requires a menu display or puts HPTOOLSET in ADD mode. Any pending XEQ files or symbolic debug command lists are terminated.

TOOLSET commands are explained according to their function, that is, Editor commands are discussed in the Editor section of this manual and so on. You can type 'HELP' after a command syntax error to find out the correct syntax for that command.

Continuation Lines

You can continue commands and command sequences on the next line by either

(1) Using the cursor wraparound - pressing RETURN signifies the end of your command sequence and the start of execution.

(2) Using an ampersand (&) as the last non-blank character of each line except for the last line. If commands are being read from a file, you must use this option. Note that tokens, with the exception of strings, cannot be continued from one line to the next if this method (2) of continuation is used.

A total command may contain no more than 255 characters including ampersands, and no more than 24 continuation lines.

MPE Commands

You can execute MPE commands without exiting TOOLSET by typing a leading colon (:) before the command. Most commands which are preceded by a colon are passed to MPE for execution. The MPE commands which can be entered from TOOLSET are:

ALTSEC
BUILD
COMMENT
FILE
GETRIN
HELP
LISTF
LISTVS
PREP
PTAPE
PURGE
RELEASE
RENAME
REPORT
RESET
RESETDUMP
RUN
SAVE
SECURE
SETDUMP
SETJCW
SETMSG
SHOWDEV
SHOWIN
SHOWJCW
SHOWJOB
SHOWME
SHOWOUT
SHOWTIME
SPEED
STREAM
TELL
TELLOP

Consult the MPE Commands Reference Manual for the correct syntax of the above commands.

RUNNING HPTOOLSET

To enter the HPTOOLSET utility, use the MPE :RUN command:

```
:RUN TOOLSET.PUB.SYS
```

To exit TOOLSET, press the EXIT function key when available, or type the EXIT command following a TOOLSET prompt.

```
>> EXIT
```

SECTION 2

TOOLSET BASICS

MODES

You interact with HPTOOLSET either by pressing TOOLSET-defined function keys or by entering commands following a command prompt (>>) on the terminal screen. There are three forms or modes of interaction, VISUAL mode, COMMAND mode, and MENU mode.

Visual Mode

Visual mode is available to you when you are in the TOOLSET EDITOR or reading a file. You can perform edit functions with the Cursor Control and Field Edit keys on your terminal on those files that are open for editing. Visual mode contains a 'message window' near the top of the terminal screen. TOOLSET prompts and confirmation messages appear here when you use certain edit operations. Entering the EDIT, READ or LISTING command puts you in Visual mode unless one of these commands is entered from an XEQ file, followed by a concatenated command, or the EDITMODE parameter of the SET ENVIRONMENT command is set equal to Command. In these cases, you receive a message that your file is open and you remain in command mode.

Command Mode

Commands may be entered while the Command mode is in effect, designated by a >> prompt. You are automatically in Command mode upon entering HPTOOLSET, and can reenter it from Visual mode by pressing the CMDWIN permanent function key.

Menu Mode

Menu mode occurs when you are using the HELP facility, with certain commands such as SHOW, and when creating new files and workspaces. Menus either provide information (SHOW) or display default and previously defined options (SET command, new files and workspaces) and allow you to define or redefine these options.

TOOLSET menus highlight those options that you can change. To change a default or already defined option, tab to that option and type the new option over the existing one. When you have defined menu options to your liking, press the SETOK function key, or the ENTER key on your terminal. The location of the ENTER key depends on the type of terminal you are using. The terminal screen and keyboard diagrammed in Figure 1-1 is for a 2645. If you are using any of the 262x type terminals, the ENTER key is located at the lower right of your keyboard.

Menus can also display information, such as files, that you can Mark and perform various operations on. To mark items within a menu generated by the SHOW FILES command (ShowFils key), SHOW DEBUG command or the Program Key, enter a character in the Mark field at the left of each item you wish to mark. Then, press the function key corresponding to the operation you wish to perform on the Marked file, such as COMPILE.

Function keys can be pressed while in any of these modes; their selection is, of course, dependent on the subsystem and function you are performing at the time.

Terminal Control Keys

The terminal control keys which include Display functions, Block Mode, Memory Lock, Auto LF and Tape Test should not be hit while TOOLSET is in Visual mode. If any of these keys is mistakenly hit, TOOLSET checks to see if it is an appropriate function, and if not, displays a message in the message window and recovers.

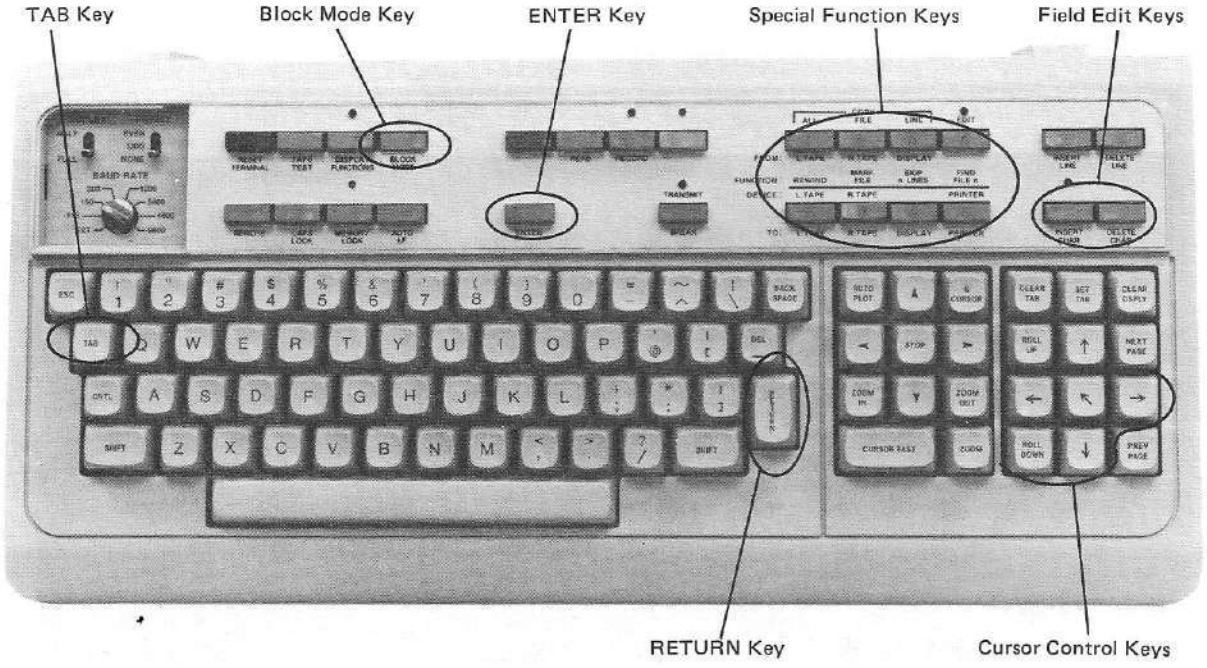


Figure 2-1. HP264x Terminal Keyboard.

Figure 2-2. HP262x Terminal Keyboard.

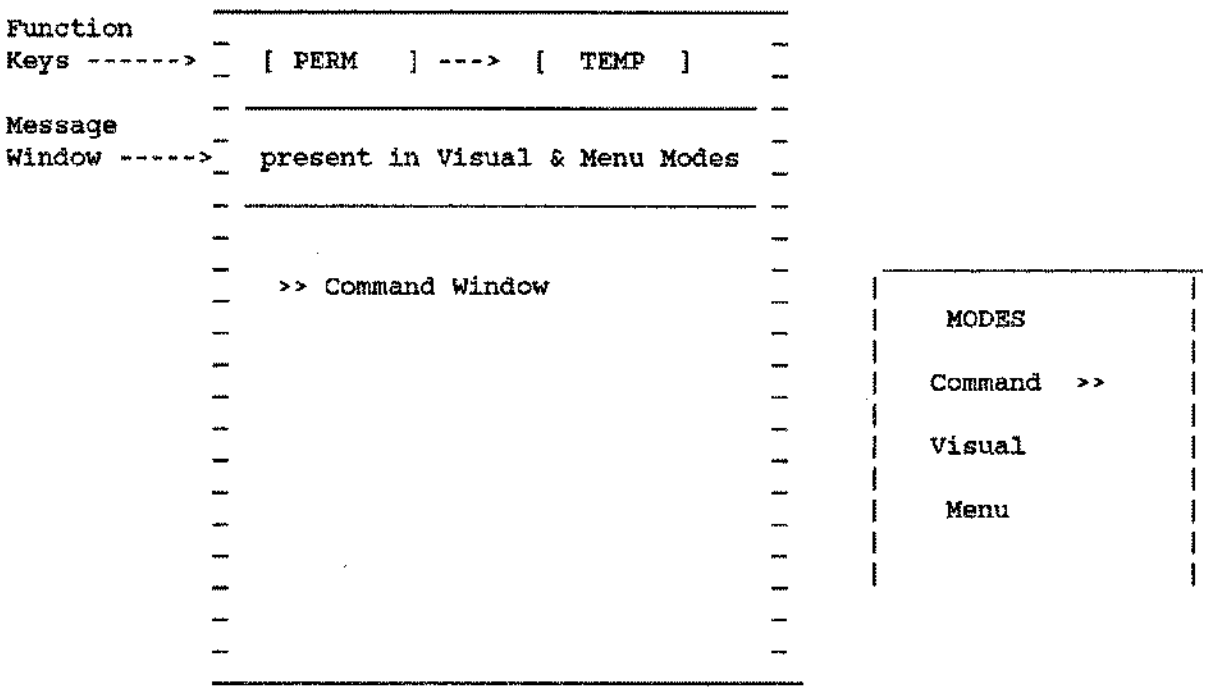


Figure 2-3. Terminal Screen for HP264x.

Upon entering TOOLSET, you are given a basic screen display that looks like the following:

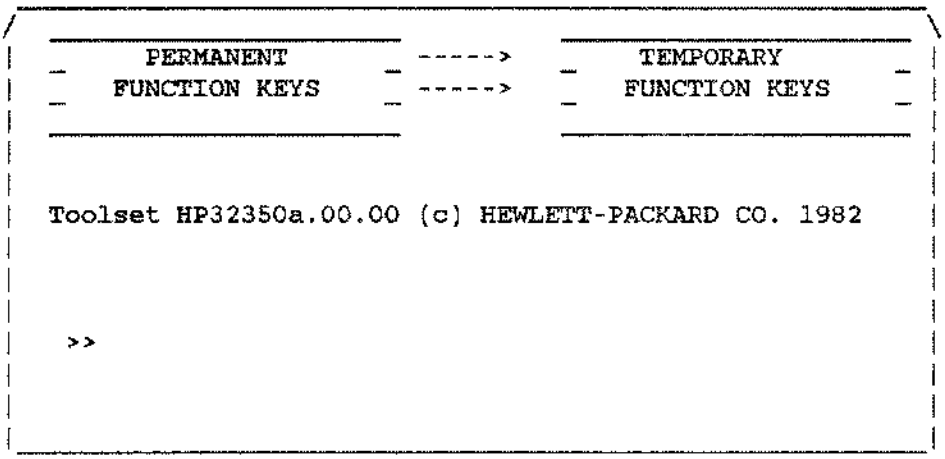


Figure 2-4. Basic Screen for HP264x.

TOOLSET COMMAND INTERPRETER

HPTOOLSET contains its own Command Interpreter, a feature that enables you to perform different program development tasks at the same time. For example, you can edit a program's source file while it is at an execution breakpoint.

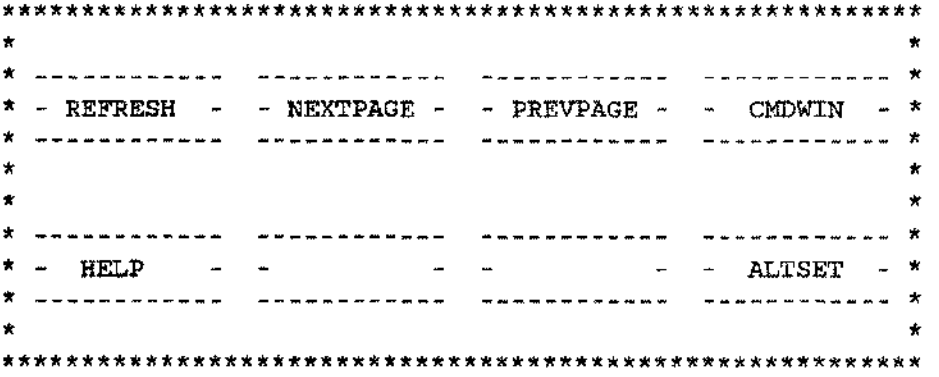
A second advantage of the TOOLSET CI is the ability to access the various TOOLSET features by pressing function keys or by entering commands. In some cases, entering a command produces a set of function keys appropriate to the entered command. For example, when EDIT is invoked, either by entering the command 'EDIT' while in command mode, or by pressing the EDIT function key, a secondary level of editing function keys is displayed.

Each screen within the TOOLSET facility contains a set of permanent function keys and a set of temporary function keys. The permanent set always remains the same while the temporary set is sensitive to the current TOOLSET activity.

When using 264x terminals both sets appear at the top of your screen and the active set is pointed to by the double set of arrows located between the two sets. On 262x type terminals, only the active set is displayed at the bottom of the screen. The Temporary function key set is the active set when you enter TOOLSET.

PERMANENT FUNCTION KEYS

The following are function keys that are available at any time from any TOOLSET subsystem and are located at the top left of your screen: on 264x terminals, and at the bottom of a 262x screen when active.



Their functions are as follows:

Key	Function
REFRESH (f1)	Redisplays the function key labels and current display in the event they are accidently erased.
** NEXTPAGE (f2)	Displays the next page within the current file. You can define the number of lines per page by entering the SET ENVIRONMENT command from which the PAGESIZE field of the menu can be set to the desired number of lines.
** PREVPAGE (f3)	Displays the previous page or 'n' number of lines while in Visual mode. You can define the number of lines per page by entering the SET ENVIRONMENT command and setting the PAGESIZE field of the menu to the desired number of lines.
CMDWIN (f4)	Accesses the Command mode with a TOOLSET prompt (>>) when you are in Visual mode. This is a toggle function key, so if you are currently in command mode, you are returned to Visual mode.

Key	Function
HELP (f5)	Invokes the TOOLSET Help facility. See the HELP command in this section in this section for more information.
ALTSET (f8)	Activates the key set that is currently inactive. This key is present for all Permanent and Temporary key sets.

** The functions for these keys can be performed in Visual or Menu (Show Files, Program, Cmdlst) mode.

TEMPORARY FUNCTION KEYS

The Temporary function key set is located at the top right of your terminal if you are using a 264x terminal, or at the bottom of your screen if you are using a 262x terminal and the temporary set is active. Function keys are grouped by like functions, and the meaning a function key has is dependent on what TOOLSET process is active. Pressing one of these function keys causes the TOOLSET Command Interpreter to execute the corresponding command.

TOOLSET function keys are constructed in a tree-like manner, so that pressing some of the keys causes TOOLSET to branch to another function key set. Those function keys which perform a TOOLSET function but do not change the function keys or function key displays are labeled in CAPITAL LETTERS. The function keys that are labeled with a beginning capital letter denote a branching function key that activates a new set of function keys.

All Temporary key sets have an ALTSET key that when pressed makes the Permanent key set active.

MARK Function Key

Several of the Temporary key sets contain a MARK function key that allows you to Mark characters or lines for further TOOLSET operations while in Visual mode. Characters are marked by positioning the cursor at that character and pressing the MARK key. Lines are Marked by positioning the cursor at the line you wish to Mark and pressing the MARK key twice. The character or line marked appears highlighted, and you can then press the temporary function key associated with the operation you want to perform on the marked character or line.

CLRMARK Function Key

A CLRMARK function is available in temporary key sets that contain the MARK key. Pressing this key clears all Marks you have made on characters or lines.

The following set of Temporary function keys are displayed when entering TOOLSET:

Key	Meaning
Edit	Accesses the HPTOOLSET Editor. See Section 5 for a detailed discussion. Selecting this key is the same as entering the EDIT command.
Wrkspc	Allows you to define a Workspace. A workspace must be defined before most TOOLSET functions can be performed. Pressing this key is the same as entering the WORKSPACE command. See the Files and Workspace section of this manual.
Read	Allows you to read a file. You can read a file in either Visual or Command mode, but the file cannot be edited while it is being read. Pressing this key is the same as entering the READ command.

Key	Meaning
ShowFils	Provides a display of all owned and used files in the current workspace. Pressing this key is the same as entering the SHOW FILES command. See the Files and Workspace section of this manual.
Program	Displays a menu of the owned files for a given workspace, their program-IDs, Languages, types, associated source files and list-files. Pressing this key is the same as entering the PROGRAM command.
EXIT	Allows you to Exit HPTOOLSET. Pressing this key is the same as entering the EXIT command.
ALTSET	Pressing this key activates the Permanent Function Key Set.

HELP FACILITY

HPTOOLSET contains a HELP facility that provides you with online information about how to use TOOLSET commands and how to interpret error messages. The HELP facility displays information about the current activity you are performing when the HELP command is entered. HELP is accessed by either typing HELP after a TOOLSET prompt, by pressing the HELP key located in the Permanent function key set, or by entering a question mark (?) after a command. If you incorrectly enter a command and request Help, TOOLSET gives you the correct syntax for that command.

Overview

A Help Overview is provided automatically if you press the HELP key when you first access HPTOOLSET and have not yet performed any TOOLSET functions. The Overview gives you a general description of TOOLSET and its functions and a menu of operations for which you can obtain additional help. To choose any of these menu items, type the letter next to that item in the selection box and push the SELCT OK key.

Functions

When you request help for any of the major TOOLSET functions, (Edit, Workspace, Program translation, Symbolic Debug), the commands for that function are displayed. To obtain help about a TOOLSET function, do one of the following:

(1) Enter the character of that function in the selection box while at the HELP Overview screen and press the SELCT OK key.

(2) Type HELP <function>

(3) Type HELP while the function is active

(4) Push the OVERVIEW key while at a HELP screen for any command in that function.

Further help on any of the function commands appearing on the menu can be obtained by entering the letter of the desired command in the selection box and pressing the SELCT OK key, or by typing HELP <command name>.

Commands

Requesting help for a specific command will provide you with the syntax of that command. To get further information about parameters or operation press the FORWARD function key or enter the letter next to the subject desired in the selection box located at the bottom of your screen and press the SELCT OK key.

All Commands

COMMANDS is a selection option from the HELP Overview menu. Requesting this option displays the first page of a list of TOOLSET commands. To display the second page press the FORWARD function key. More information on any of the commands can be obtained by entering the letter in the selection box next to the desired command, and pressing the SELCT OK key.

Errors

Pressing the HELP key or typing the HELP command following a syntax error accesses the HELP facility and gives you the correct syntax for the command in error. For errors other than syntax errors typing the HELP command or pressing the HELP function key displays the cause of the error and the action you should take to correct that error, but does not access the HELP facility. To access the HELP facility in these cases, press the HELP key a second time. An overview the the current TOOLSET utility is then displayed.

Exiting Help

You can exit the HELP facility in one of three ways:

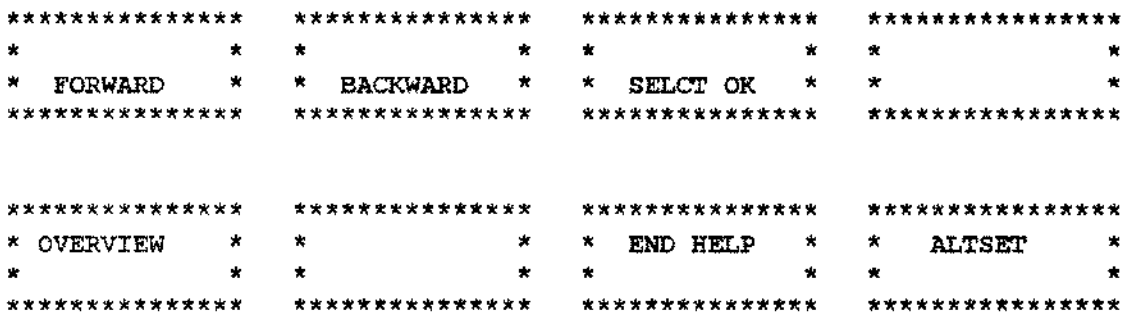
(1) Pressing the END HELP function key

(2) Typing a TOOLSET command in response to the prompt at the bottom of your screen and pushing the SELCT OK key.

(3) Pressing the ALTSET function key. This activates the Permanent Function Key set and allows you to enter the command window (>>) with the CMDWIN key.

HELP Function Keys

When you enter the TOOLSET Help facility, the following set of Temporary Function keys are available:



Their functions are as follows:

Key	Function
FORWARD	Allows you to advance to the next screen in the Help catalog.
BACKWARD	Allows you to return to the previous screen in the Help catalog.
SELCT OK	This key enters the selection you have made on the Help screen into TOOLSET.

Key	Function
OVERVIEW	If you are in a command help screen, pressing this key displays the overview for the TOOLSET function in which the command is used. If you are in a function screen, this produces the TOOLSET overview screen.
END HELP	Allows you to exit from the Help facility and return to the screen previously displayed.
ALTSET	Activates the Permanent Function Key set.

WORKSPACE

TOOLSET keeps track of the files used to develop a program file through a 'workspace'. A workspace is the collection of source (text), USL, Program, and list files that are used in the development of one program. You can be associated with only one workspace at a given time.

Each time you enter TOOLSET you must enter a Workspace name by either entering the command 'WORKSPACE' or by pressing the WRKSPC temporary function key. TOOLSET then prompts you for the name of the workspace:

```
--> Workspace? _ ONE
```

If the workspace you name does not already exist, TOOLSET attempts to create it and prompts you with

```
--> Create Workspace ONE ?
```

At this point you can cancel the creation of ONE by entering any character but "Y" or "YES" or by entering a Control Y. If you cancel a new workspace, TOOLSET displays the following message

```
*** Operation Cancelled. (211)
```

In order to create the workspace, a menu of the default SET options for the USLfile, Program file, PREP options and Priority queue is displayed. Any of the defaults may be changed, but whether you choose to keep the defaults or supply your own options, the filenames must be unique. Once you are satisfied with the options, press the SET OK temporary function key.

TOOLSET sets up default USL and program filenames by appending a U or P to the first seven characters of the workspace name you have entered to designate the files for program translation and execution. The default Priority Queue is DS. Compiles are executed in the queue defined here.

The default SET options for Workspace ONE are:

```
USL -->  ONEU.group.account

PROG -->  ONEP.group.account

PREP -->        (this left blank by default; ONEU is prepped
                 into ONEP with MPE default options)

PRIORITY -->  DS
```

Valid Commands without a Workspace

The following TOOLSET commands can be entered without having first declared a workspace:

WORKSPACE	RESTORE
RECOVER	REDO
REFRESH	HELP
PURGE	EXIT
STORE	

You can also enter MPE commands except :PREP and :RUN without declaring a workspace.

All-Purpose Commands

Certain TOOLSET commands are global and are not unique to any one of the four program development activities included in this utility. They can be entered whether you are editing, reading, compiling or running your program. They are:

END	SET
EXIT	SHOW
HELP	XEQ
REDO	

END

Terminates a TOOLSET function or activity.

Syntax

```
-----------------------------
-   EN[D]  [ALL      ]      -
-          [EDIT     ]      -
-          [LISTING  ]      -
-          [READ     ]      -
-          [RUN      ]      -
-          [COMPILE  ]      -
-----------------------------
```

Parameters

ALL - Terminates all ongoing TOOLSET activities: editing, reading, running, and compiling. TOOLSET prompts you before terminating the compilation.

EDIT - Terminates editing of the current file.

LISTING - Terminates reading of the current listfile.

READ - Terminates reading of the current file.

RUN - Terminates running of the program file.

COMPILE - Terminates program compilation.

Operation

The END command is used to terminate a TOOLSET function such as editing or compiling, or to end an activity such as a menu display. When the command is entered, TOOLSET closes any associated files. If you use the ALL parameter when a compile is active, you are asked if you wish to terminate the compile. Any other on-going activities or functions are automatically terminated when END ALL is entered. When END is entered without any parameters, TOOLSET prompts to determine which operation you wish to end in the following order: RUN, EDIT, READ or LISTING, and COMPILE.

Example

```
>>END ALL
>>
```

Function Key

Pressing the END function key in visual mode ends the current edit, listing or read operation, closes the file, and puts you into command mode.

EXIT

Exits TOOLSET and returns you to MPE.

Syntax

```
-------------
-   EX[IT]  -
-------------
```

Operation

The EXIT command exits the HPTOOLSET utility and returns you to MPE, closing any open files associated with TOOLSET. If a program compilation is in progress, TOOLSET prompts you to see if you wish to abort the compile. If you respond 'NO' the compile continues and you do not exit.

Function Key

An EXIT function key exists as part of the Temporary function key set that is available after first entering TOOLSET. Pressing this key is the same as entering the EXIT command.

Example

```
>>EXIT
:END OF PROGRAM
:
```

HELP

Accesses the TOOLSET Help facility.

Syntax

```
-------------------------------------------
-                                         -
-  H[ELP] [OVERVIEW  ]                    -
-         [function name]                 -
-         [command name]    [SYNTAX]      -
-                           [PARAMETERS]  -
-                           [OPERATION]   -
-                           [EXAMPLE]     -
-------------------------------------------
```

Operation

Entering the HELP command accesses the TOOLSET on-line help facility that provides information on command syntax and operation. Typing Help after you receive a command syntax error provides the correct syntax for that command. Typing just 'Help' provides information about the current context If there is no context, such as when you first enter TOOLSET, an Overview of TOOLSET is provided. The HELP command is not available when you are in batch mode.

Function Key

A HELP function key is available as part of the Permanent function key set. Pressing this key is the same as entering the HELP command without any parameters.

Example

```
>>HELP
```

REDO

Enables correction and re-execution of the last command or command list entered at the command prompt.

Syntax

Operation

The REDO command allows you to correct and re-execute the previous command or command list you entered. When REDO is entered, the last command is displayed and characters can be inserted, deleted or replaced by entering I, D, or R respectively on the line below the command. Inserting and replacing (I and R) must be followed by the characters you wish to insert or replace with.

Editing functions can be undone by typing one U after the function, or, to return to the original command, type two consecutive U's. If neither an I, D, R or U is typed following the REDO command, a Replace is performed by default.

Note that a REDO command cannot be used on a function key, as a command in an XEQ file, or in a symbolic debug command list. If REDO is included in an XEQ file or command list, the commands following it are not executed.

You do not need to define a workspace to use this command.

Example

```
>>EDT
***Undefined HPToolset command keyword. (101)
>>REDO
EDT
  iI
EDIT
```

SET

Allows you to define Edit, Program, and Environment options.

Syntax

```
-------------------------
-                       -
-   SET  [ED[IT]       ] -
-        [PR[OGRAM]    ] -
-        [EN[VIRONMENT]] -
-------------------------
```

SET EDIT Syntax

```
[SOURCE LANGUAGE [COBOL] [OTHER]]
[STRUCTURE  [MAIN][SUBPROGRAM][INCLUDE][OTHER]]
[PROGRAM-ID  [              ]]
[LINELENGTH  [74]]
[FILESIZE    [6000]]
[INCREMENT    [1]           ]
[GUIDE    [ON] [OFF]]
[TABS    [OFF] [LANGUAGE] [OTHER]]
```

Parameters

LANGUAGE - Tells TOOLSET which language is being used in the source program. This selection effects the edit function keys and other EDITOR and PROGRAM KEY options. The LANGUAGE option cannot be reset after creation of the file.

STRUCTURE - Informs TOOLSET of the source language structure contained in the edit file. This option is set when you create a file and may be reset at any time.

PROGRAM-ID - Matches the one you use in the Identification Division of your COBOL source program. It is needed if the language is specified as COBOL and the structure is MAIN or SUBPROGRAM.

LINELENGTH - Sets the length of the file's logical records in bytes. The default is 74 for all files. The linelength cannot be reset after creation of the file.

FILESIZE - The number of logical records in the EDIT file. This number should allow for adequate file growth and for multiple versions of a file. The default is 6000 logical records. Note that this option cannot be reset after the creation of the file.

INCREMENT - Defines a default increment to be used when you add lines to your sourcefile with the ADD, MOVE or COPY commands. TOOLSET uses this increment if you do not explicitly give one in these commands. It is overridden if the amount of space remaining to add, move or copy lines is very small, or the implied increment given in the command is smaller than the default. The default increment is 1.

GUIDE - The GUIDE=ON option places a column guide across the first line of the visual screen, which remains there until Visual mode is exited. When GUIDE is set equal to OFF, the column guide does not appear on your screen. The default is OFF.

TABS - LANGUAGE - You get the default tabs for the language specified.

OTHER - You may enter up to 10 tabs in the column guide shown on the SET MENU by placing a character in each of the column positions where you want a tab.

OFF - Erases all tabs.

SET PROGRAM Syntax

```
[USLFILE uslfilename                ]
[PROGFILE progfilename              ]
[PRIORITY queue priority            ]
[PREP prep command default options]
```

Parameters

uslfilename - Allows you to modify the workspace uslfile name. By default TOOLSET takes the first seven letters of your workspace name and appends a "U". This must be a unique name, that is, you cannot have any other files with this name.

progfilename - Allows you to change the workspace program file name. By default TOOLSET takes the first seven letters of your workspace name and appends a "P". This must be a unique name, that is you cannot have any other files with this name.

PRIORITY - Execution priority class for the compile (CS, DS,ES). The default is DS.

PREP - Allows you to make changes to the PREP command parameters. These parameters must be separated with semicolons with no semicolon preceding the first parameter. The default parameter values are set by MPE. See the PREP command.

SET ENVIRONMENT Syntax

```
[ECHO=[ON][OFF]                ]
[QUIET = [OFF][ON]             ]
[N=integer                     ]
[EDITMODE=[VISUAL][COMMAND]    ]
[FIND STRING=string            ]
[PAGESIZE=integer              ]
[CHANGE STRING=string          ]
```

Parameters

ECHO - If ECHO is set ON (default), commands executed from XEQ files and breakpoint command-lists are echoed on your terminal screen. If ECHO is set OFF, commands from these input sources are not echoed back. This option can be entered as a command.

PAGESIZE - The number of lines of an EDIT or READ file that appears on your terminal screen at one time while in the visual mode. The default is 20 lines. The integer specified must be between 2 and 40 inclusive. This option can be entered as a command.

N - The value used for N in the EDITOR function keys FIND+N and FIND-N. These keys enable you to go forward or backward in an EDIT, READ or LISTING file by the number of lines defined by N. N is set to 10 lines by default. This option can be entered as a command.

EDITMODE - If set to COMMAND, you stay in command mode when you enter the Listing, Read or Edit commands. If EDITMODE is set to VISUAL, you are put into Visual mode when the Listing, Read, and Edit commands are entered unless they are entered from a command list or XEQ file. In these instances, you enter command mode. Visual is the default. This option can be entered as a command.

QUIET - If QUIET is set on, you receive a listing of the lines affected when an operation is performed on the source file by the command editor. If QUIET is set to OFF, the lines affected by editor operations such as Delete will not be listed. The default is OFF. This option can be entered as a command.

FIND STRING - The current value of the EDITOR find-string. This value is set by entering a find-string in a FIND or CHANGE command, or by changing the string in the menu brought up by this (SET) command. The string can be no longer than 78 characters, including delimiters. This option can be entered as the command SET FIND = string.

CHANGE STRING - The current value of the EDITOR change-string. This value is set by entering a change-string in a CHANGE command or by entering a new string in the menu brought up by this (SET) command. The string can be no longer than 78 characters, including delimiters. This option can be entered as the command SET CHANGE string.

Operation

The SET command allows you to define EDITOR, PROGRAM, and ENVIRONMENT options. Entering the SET command with no parameters produces a general SET selection menu from which you can choose one of these three set options.

EDIT options are set when you create the Edit file, and all can be reset except the LANGUAGE, LINELENGTH and FILESIZE options. The values you define in the EDIT options are stored with the file and are not limited to the edit session. A SET EDIT function key is available while in the EDIT subsystem. When pressed a menu of the current edit options is displayed. It is the same as entering the SET EDIT command.

The PROGRAM options which you can change are the workspace uslfile name, the workspace program file name, the PREP command parameters, and the compile priority. A SET PROG function key is available within the PROGRAM KEY subsystem and is the same as entering the SET command with the PROGRAM option. When initially pressed, it produces a display of the current program translation default options. Note that the default parameters for the PREP command are set to the MPE values. The command syntax for PREP can be found in Section 6 of this manual.

The ENVIRONMENT is a general user, screen-oriented parameter that includes such options as page size and the number of lines you can scan forward or backward for reading and editing purposes. There is no function key for the SET ENVIRONMENT, but each of the ENVIRONMENT options can be entered as a command. For example, you can enter the command SET ECHO = OFF without bringing up the entire SET ENVIRONMENT menu.

The SET menus are not available when you are in batch mode.

SHOW

Displays information about the current user environment.

Syntax

```
-------------------------------------
-                                   -
-   SHO[W]  [F]ILES [workspacename] -
-           [D]EBUG                 -
-           [E]QUATES               -
-           [A]CTIVITIES            -
-           [L]ABELS [filename]     -
-                                   -
-------------------------------------
```

FILES [workspacename] -

Provides a display of all the owned and used files in the workspace you designate. If you do not designate a workspace, the display is for the current workspace.

The information provided for each file includes:

(1) Name of the owner workspace for all used TSAM files. An asterisk next to a used file indicates that a file equation is generated when the workspace is entered.

(2) The number of active versions, that is, the file versions that have not been purged in each file, for owned files.

(3) The REFERENCE version number of each owned file.

(4) The LATEST version number of each owned file.

(5) The type of each file - whether it is Used, Owned, a USL, or a Program file.

A ShowFils function key is available from the Workspace Management subsystem and provides the same file information as listed above. When the SHOW FILES menu is displayed, a second set of temporary function keys, EDIT, READ, SETV is made available. To use any of these functions on any of the listed files, type any letter in the box next to the file name and press the function key of the desired operation.

DEBUG - Displays the active breakpoints and datatrace variables and indicates with an asterisk which breakpoints and variables have an associated command list. To display any command list, enter any character in the box next to the associated breakpoint or variable and press the CMDLIST key.

EQUATES - Lists all the file equations which are in effect for the current workspace as a result of USE commands.

ACTIVITIES - Lists whether Editing, Reading, Listing, Running,and Compiling is currently taking place, and the names of the files associated with these activities. It also lists the workspace that is currently open.

LABELS [filename [#version designator]]
[#version range]

Gives the comments that were entered in the file label with the Workspace Management LABEL command. You may specify more than one version for which the label is displayed. If no version is specified, TOOLSET defaults to the LATEST version for owned files and the REFERENCE version for other files. If you do not specify a file-name, the label for the LATEST version of the edit file is shown, if there is an edit file.

Only the SHOW ACTIVITIES command is available while you are in batch mode.

XEQ

Allows input from a file.

Syntax

```
--------------------------------
-                              -
-  X[EQ] filename [#version]   -
-                              -
--------------------------------
```

Parameters

filename - either an ASCII or TSAM formatted file from which TOOLSET accepts commands.

#version - particular version of the input file.

Operation

The XEQ command directs TOOLSET to accept command input from either an ASCII or TSAM formatted file instead of the terminal. An ASCII file can have a record size of 132 bytes or less, but only the first 72 bytes are read from each record. A TSAM file can also have 132 bytes or less, but TOOLSET reads all the bytes in each record.

Visual mode is not accessible while an XEQ file is executing. If the EDIT, READ, or LISTING command is used in your XEQ file, the specified file is opened for that operation, but you will remain in command mode. If any of the commands used in the XEQ file require a menu, the menu is displayed, but any remaining commands in the XEQ file are ignored.

If an error occurs during execution, the remainder of the commands in the file are ignored.

Example

```
>>XEQ CMDFILE#2
```

SECTION 3

Developing a Program with TOOLSET

This section is designed to show you how to develop a COBOL program using many of the TOOLSET utilities. Step-by-step examples are provided to enable you to follow the development sequence and monitor its progression. You will define a Workspace, enter and edit a source file, compile the source into a USL file, prep the USL file into a Program file, and use Symbolic Debug functions. It is assumed that you you have read the first two sections of this manual. If you have not, do so now. After completing this section, you should be able to develop a program of your own using TOOLSET.

One of the features of TOOLSET is its parallel function keys and commands. The examples in this section use a mixture of commands and function keys. Most operations can be performed with either. All example screens use an HP264x terminal where both permanent and temporary key sets appear at the top of the screen. If you are using an HP262x terminal only active TOOLSET function keys appear at the bottom of your screen.

The operations that you perform are designated by greater than (>>) signs. Extra information is provided in *NOTES* throughout this section. Be sure to read them.

To Begin

Step 1

To access TOOLSET, log on to your group and account and run the TOOLSET program.

```
:HELLO user.account

:RUN TOOLSET.PUB.SYS
```

The double arrow prompt (>>) at the left of your screen means you are in command mode. You can either type commands or press available function keys. Modes and permanent function keys are discussed in Section 1.

NOTE

If you need or want more information, you can access the TOOLSET on-line HELP facility or reference the associated page(s) in other parts of the manual. To get help, either enter the HELP command after a command prompt (>>), press the HELP key (located in the permanent key set) or type a question mark (?) after a command. Help can be obtained in the form of subsystem overviews, command syntax, or interpretation of error messages.

Specifying a Workspace

Step 2

In order to manage your files, TOOLSET must associate them with a Workspace you either create or specify after accessing TOOLSET. Most TOOLSET operations cannot be performed until a Workspace is defined. To define a Workspace for your program:

>>>

Press the Workspace (Wrkspc) function key

When prompted for a name, call this workspace ONE. Type ONE followed by a carriage return. Since Workspace ONE does not yet exist, TOOLSET asks if you want to create a new Workspace. Answer YES.

>>

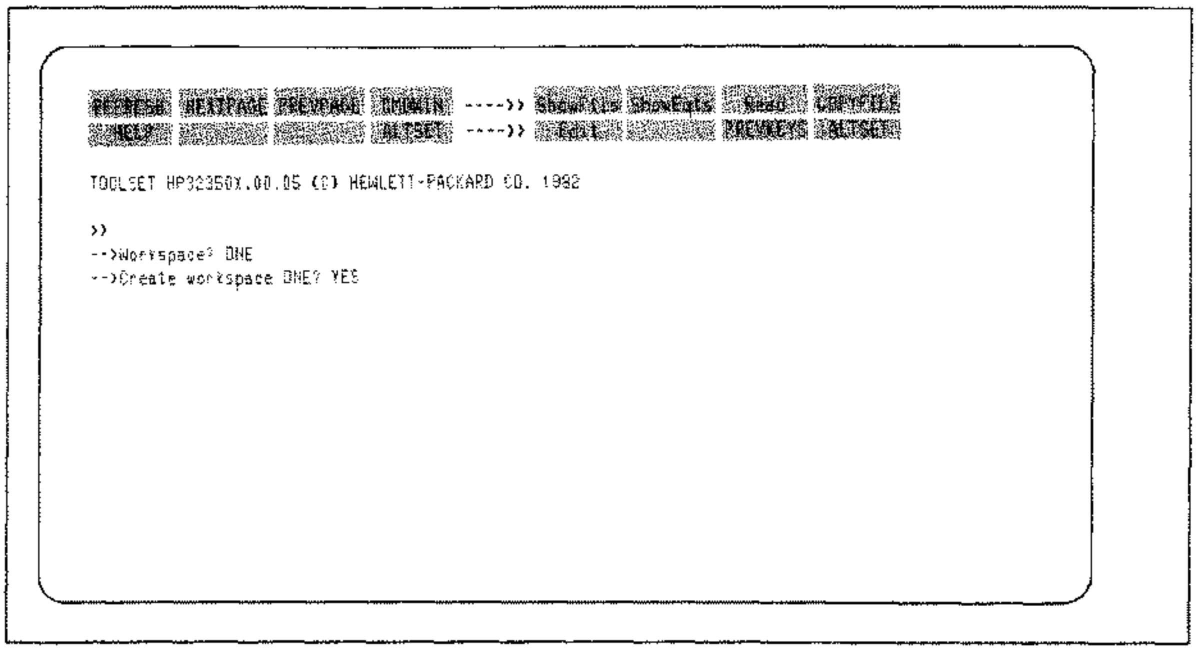


Figure 3-1. Creating a Workspace.

NOTE

The Workspace function key belongs to the first set of temporary keys that appear when you enter TOOLSET. If you are using an HP264x terminal the Workspace key appears as one of the temporary keys at the top right of your terminal. If you are using an HP262x terminal these keys are located at the bottom of your terminal.

When TOOLSET creates Workspace ONE, an options menu known as the SET PROGRAM menu is displayed. It allows you to define the USL and Program file names, the :PREP command options, and the Priority queue. Defaults for these options are automatically filled in except for the :PREP option which uses the MPE :PREP command defaults. You can specify options on TOOLSET menus by tabbing to the option and typing over the default. When the options are defined as you want them, press the SET OK function key to enter them into TOOLSET.

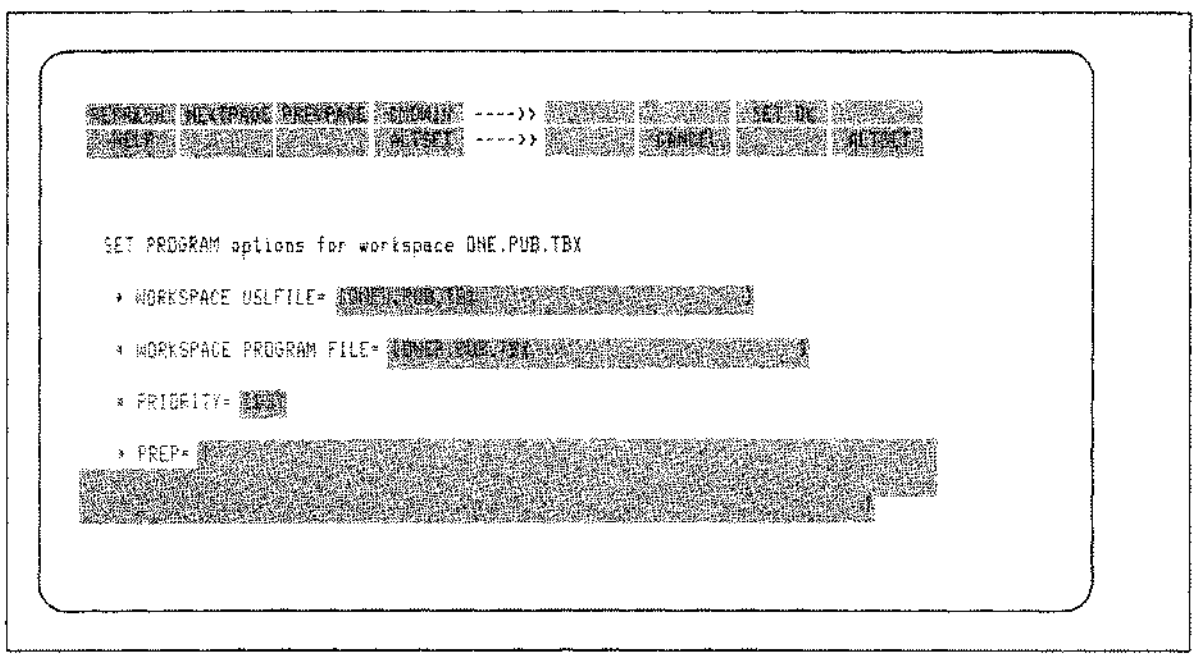


Figure 3-2. SET PROGRAM Options Menu.

>>>

For this menu, keep the defaults. Press the SET OK key.

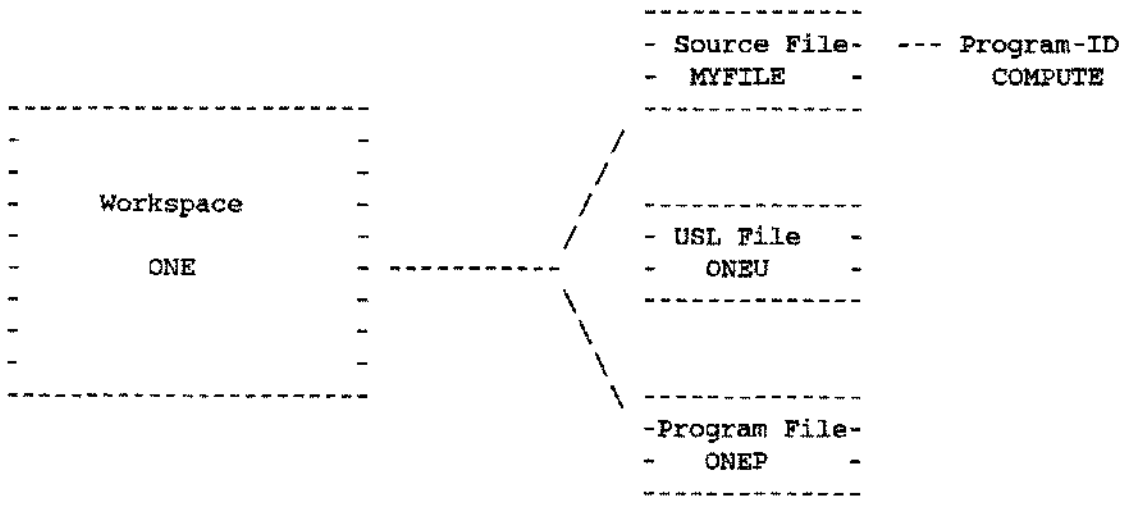
>>>

NOTE

(1) If you decide that you don't want to create a Workspace, press the CANCEL function key. Creation of the workspace is then cancelled.

(2) This menu can be viewed or changed at any time by entering the SET PROGRAM command after a >> prompt, or by pressing the SET PROG key when available. This command is explained in Section 2.

The Workspace you have just created is a directory of all the files you use to develop one program. Workspace ONE is the development environment for all the source, program, and USL files you use in creating a COBOL program. In this section, you are developing a program called COMPUTE that calculates total earnings using a simple formula and a weekly salary and $ Sales input. Notice that you will be dealing with two file names: the Workspace just created called ONE, and the source you will create next called MYFILE with a Program-ID called COMPUTE. Because you kept the defaults for the USL file and Program file when Workspace ONE was created, the USL and Program file names were defined as ONEU and ONEP respectively.



Creating Your Source File

Step 3

Once you have declared a Workspace to allow TOOLSET to associate your development files, you can begin creating the source file in the TOOLSET editor. Your screen should look this. Notice the function keys that are now available and also notice that you are in command mode (>>).

```
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

 Press the Edit function key to enter the TOOLSET editor.

>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
```

You are prompted for the name of the file you wish to edit.

```
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

Call this file MYFILE.

>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
```

Since this is a new file, that is, a file with this specified name does not already exist, you are asked if you wish to create a new file.

```
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

Respond YES to this prompt.

>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
```

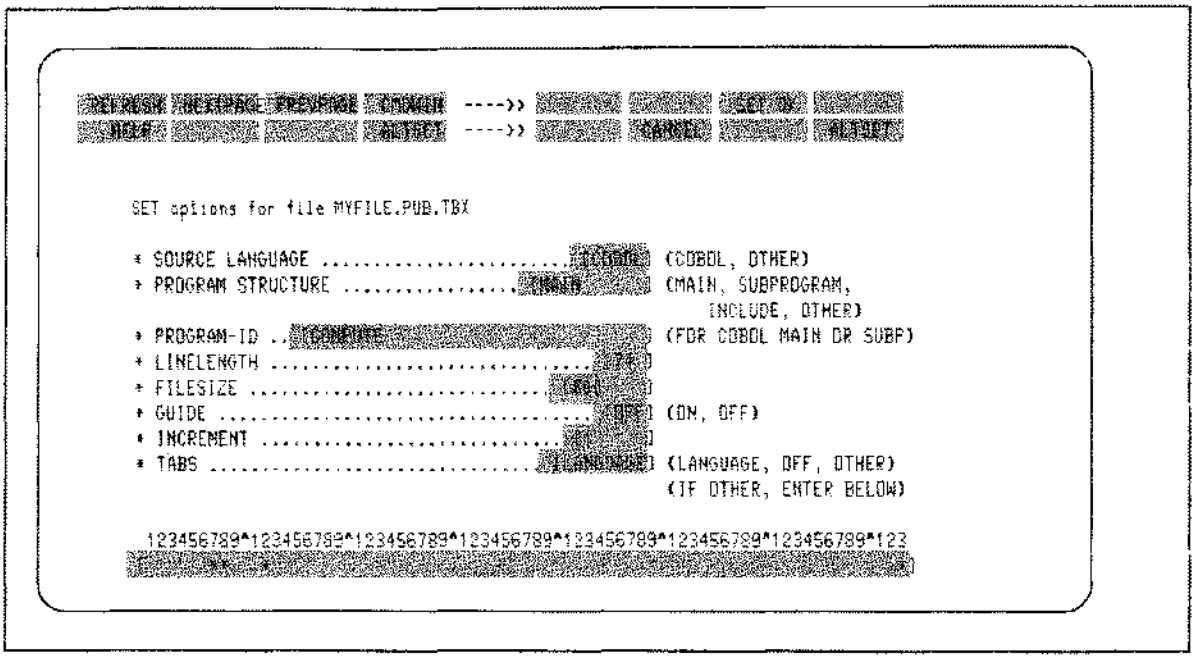


Figure 3-3. SET EDIT Options Menu.

When you confirm the creation of the new file MYFILE, an options menu is displayed that allows you to define your file.

'MYFILE' can be defined with all of the above options, and these options can be re-defined at any time except for LINELENGTH, FILE SIZE, and SOURCE LANGUAGE. These will remain as you define them here for as long as your file exists.

The defaults for these options are filled in except for the Program-ID. This you must define if you are creating a COBOL Main or Sub-program. This name should match the Program-ID in the IDENTIFICATION DIVISION of your program. For the program COMPUTE, use the Edit defaults shown on the SET EDIT menu except for FILESIZE.

>>

Tab to Program-ID and define it as COMPUTE. Next, tab to the FILESIZE option and change the default of 6000 to 500. When you have changed these two options, press the SET OK function key.

>>>

NOTE

(1) To view the Edit options while the file is open for editing, enter the SET EDIT command after a >> prompt or press the SET EDIT function key when available. SET EDIT is discussed in Section 2 of this manual.

(2) The default for the TABS option is LANGUAGE which aligns tabs to the COBOL language. Whenever you want to set your own tabs, designate the 'OTHER' option and type a character in the column where you want each tab placed.

Once you have entered the options, MYFILE is open for editing. TOOLSET automatically provides the first four lines of your source file. Note that the Program-ID you defined in the previous menu appears here. Also note that the $CONTROL statement contains a SYMDEBUG parameter. 'SYMDEBUG' must appear in this statement to use the Symbolic Debug feature of TOOLSET.

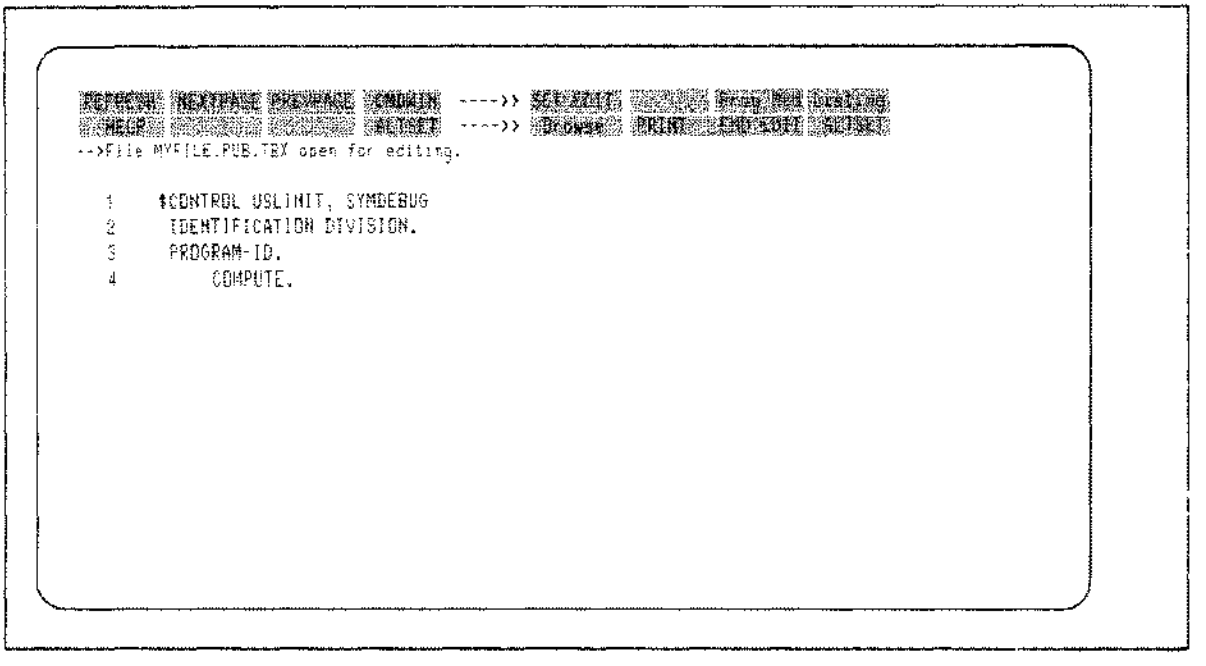


Figure 3-4. File Open For Editing.

NOTE

"USLINIT" appears if your file is MAIN. "SUBPROGRAM" appears if your file is a Subprogram.

Modes for Entering Your Source File

There are four modes in TOOLSET: Visual, Command, Menu, and Add.

Visual mode is the default mode when you enter the TOOLSET Editor. It is screen oriented and allows you to use the terminal cursor and edit keys to enter or modify text anywhere on your screen. This mode is most efficient when you want to make specific modifications to your edit file. You use it later on to make changes to MYFILE.

Commands can also be used to perform edit functions. To use commands, you must be in command mode; commands cannot be entered from Visual mode. To access Command mode, first access the Permanent key set with the ALTSET key, then press the CMDWIN key. Since most edit functions put you in visual mode, it is probably more efficient to use function keys instead of commands.

Menus are TOOLSET screens that show you various options and allow you to change them. You chose Workspace and file options from menus when you created Workspace ONE and the source file MYFILE. Menus also allow you to indicate various files and perform TOOLSET operations on them. For example, you can choose a source file from a Show Files menu and press the edit function key to edit the file.

ADD mode is line oriented and prompts you for each line. Cursor control keys are not recognized while you are editing in Add mode. This mode is best used when you are adding a lot of lines to your file. ADD mode is used in this section to enter the program COMPUTE.

Visual, Add, and Command modes are discussed in Section 5 of this manual.

NOTE

Visual Editor is the default mode when you enter TOOLSET Editor, but can be overridden with the EDITMODE option of the SET ENVIRONMENT command. This command is found in Section 2.

Entering Your Source File in Add Mode

Step 4

Since you want to enter multiple lines of source into the TOOLSET Editor, Add mode is the most efficient for your use. Remember, you are currently in Visual mode. To access Add mode to enter the Source file MYFILE:

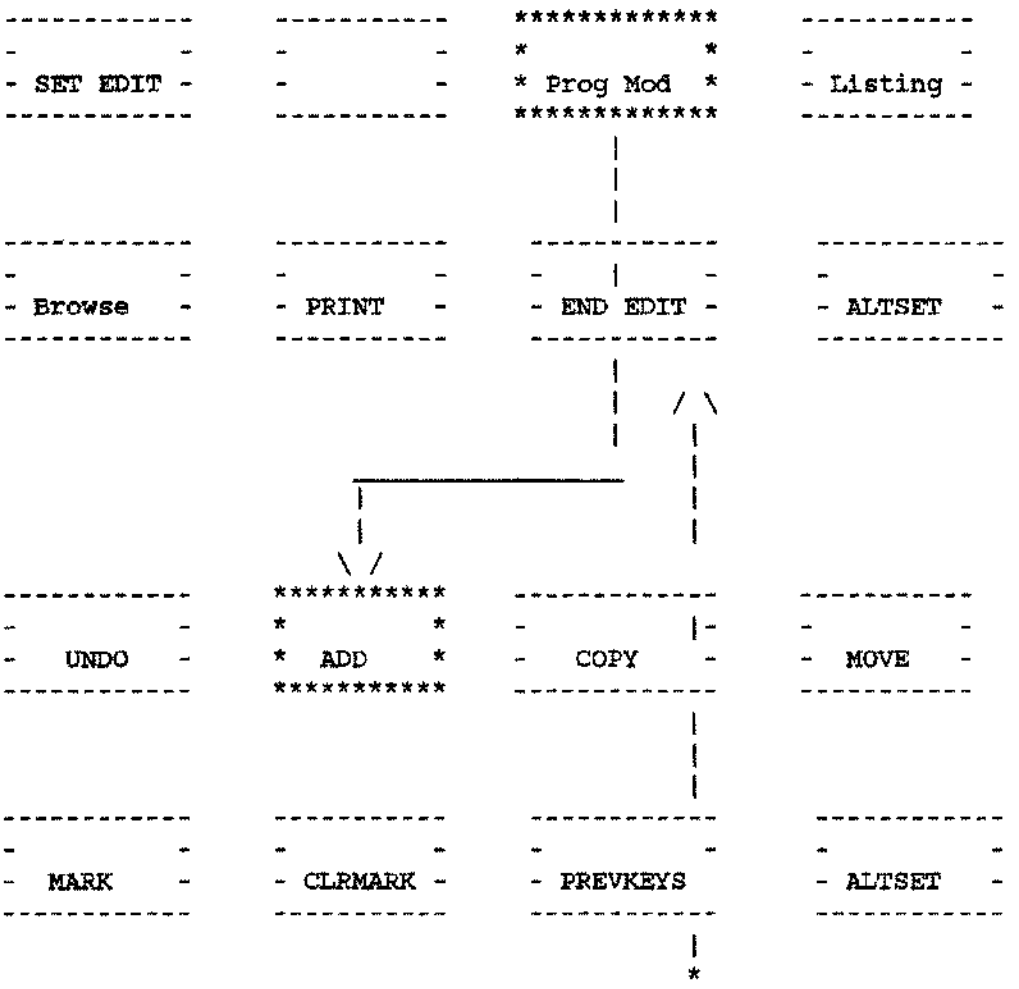
>>

Press the Prog Mod (program modification) key

>>

NOTE

The Prog Mod key, like the Wrkspc and Edit keys, are known as 'branch' function keys because they produce a secondary set of keys. Branch keys begin with a capital letter. Other function keys are printed in all capital letters.


SET EDIT
Prog Mod
Listing
Browse
PRINT
END EDIT
ALTSET
UNDO
ADD
COPY
MOVE
MARK
CLRMARK
PREVKEYS
ALTSET

>>

Next, press the ADD key.

>>

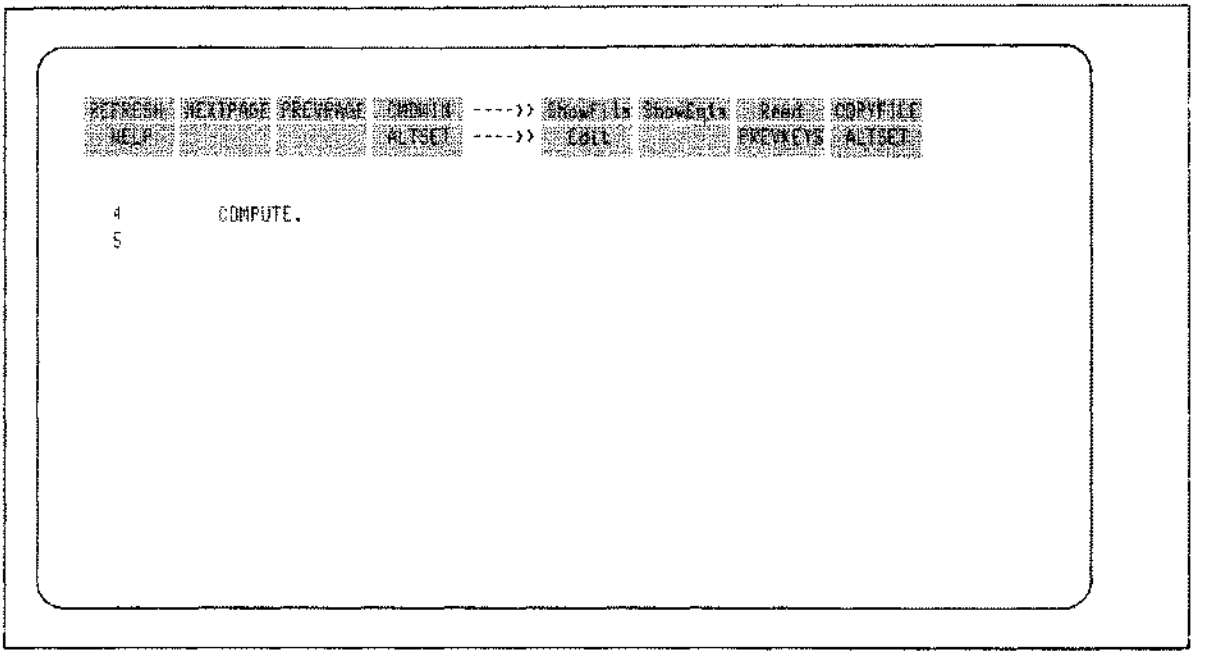


Figure 3-5. Entering Add Mode.

Once you have pressed the ADD key you are ready to type your program. Note that the program below has some mistakes in it which are marked with an asterisk. Type it in as it appears, but do not type the asterisks.

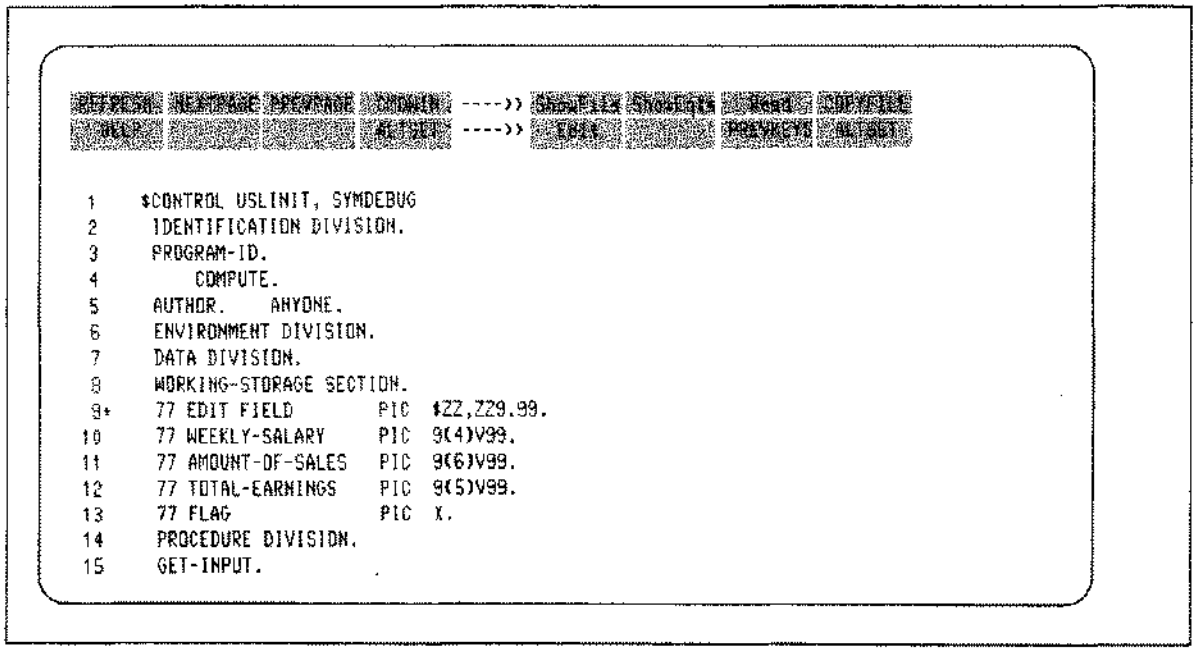
NOTE

In Add mode, the cursor is positioned at COBOL column 7 (for continuation lines, comment lines, or $ statements). Tab forward to column 8 to type COBOL statements.

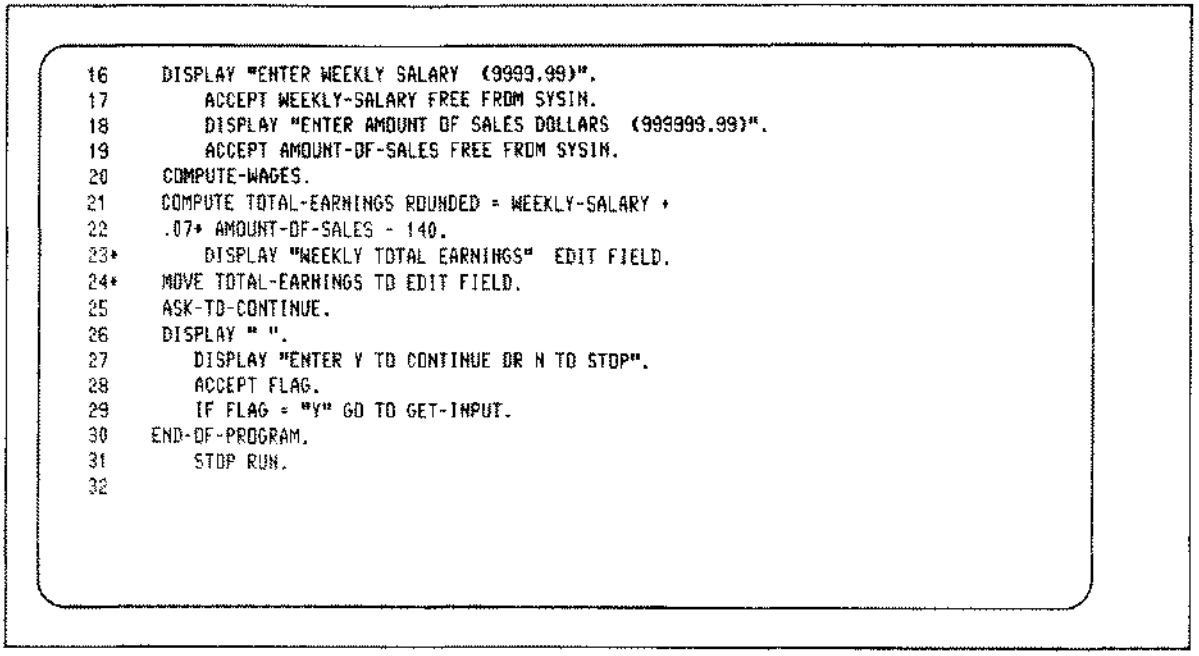
>>>

Type the program COMPUTE into the edit file MYFILE

>>>



```
REFRESH  NEXTPAGE  PREVPAGE  DOMAIN  ---->>  SHOWFILE  SHOWDICTS  READ  COPYFILE
 HELP                       ALTSET  ---->>  EDIT                   PREVCYC  ALTSET

 1    $CONTROL USLINIT, SYMDEBUG
 2     IDENTIFICATION DIVISION.
 3     PROGRAM-ID.
 4         COMPUTE.
 5     AUTHOR.    ANYONE.
 6     ENVIRONMENT DIVISION.
 7     DATA DIVISION.
 8     WORKING-STORAGE SECTION.
 9*    77 EDIT FIELD          PIC  $ZZ,ZZ9.99.
10     77 WEEKLY-SALARY       PIC  9(4)V99.
11     77 AMOUNT-OF-SALES     PIC  9(6)V99.
12     77 TOTAL-EARNINGS      PIC  9(5)V99.
13     77 FLAG                PIC  X.
14     PROCEDURE DIVISION.
15     GET-INPUT.
```



```
16     DISPLAY "ENTER WEEKLY SALARY  (9999.99)".
17         ACCEPT WEEKLY-SALARY FREE FROM SYSIN.
18         DISPLAY "ENTER AMOUNT OF SALES DOLLARS  (999999.99)".
19         ACCEPT AMOUNT-OF-SALES FREE FROM SYSIN.
20     COMPUTE-WAGES.
21     COMPUTE TOTAL-EARNINGS ROUNDED = WEEKLY-SALARY +
22     .07* AMOUNT-OF-SALES - 140.
23*        DISPLAY "WEEKLY TOTAL EARNINGS"  EDIT FIELD.
24*    MOVE TOTAL-EARNINGS TO EDIT FIELD.
25     ASK-TO-CONTINUE.
26     DISPLAY " ".
27        DISPLAY "ENTER Y TO CONTINUE OR N TO STOP".
28        ACCEPT FLAG.
29        IF FLAG = "Y" GO TO GET-INPUT.
30    END-OF-PROGRAM.
31        STOP RUN.
32
```

Figure 3-6. Source File MYFILE, Program COMPUTE.

Editing Your Source File

Step 5

Add Mode to Visual Mode

If you copied the program COMPUTE exactly as you found it here, lines 23 and 24 are out of order. To place them in the correct sequence, the MARK and MOVE edit functions can be used. To use these functions you need to reaccess the Visual Editor.

>>

To access visual mode, press the Edit function key. Notice that pressing the Edit function key ends ADD mode.

>>

```
------------    -----------    -------------    -----------
-          -    -         -    -           -    -         -
- ShowFils -    - ShowEqts-    -  Read     -    - COPYFILE-
------------    -----------    -------------    -----------


************    -----------    ------------    ------------
*          *    -         -    -          -    -          -
*  Edit    *    -         -    - PREVKEYS -    - ALTSET   -
************    -----------    ------------    ------------
```

You are now back in Visual mode as the arrow and message at the top of your screen indicates. Note that the last line entered in your source appears also.

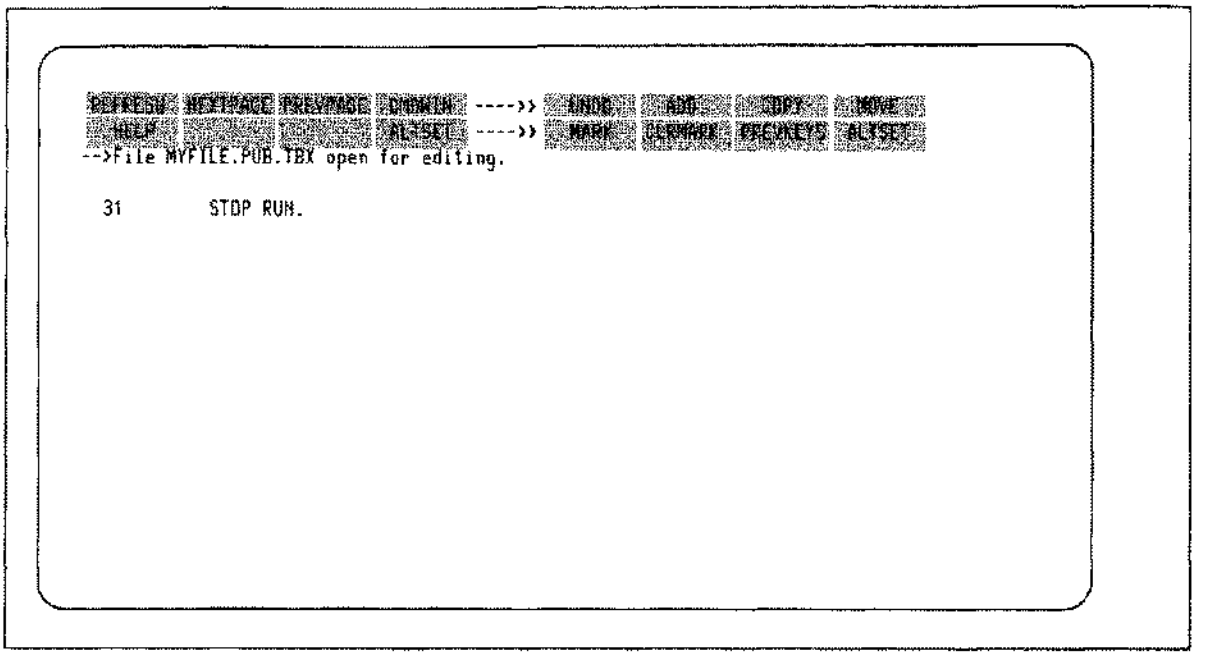


Figure 3-7. Visual Editor.

NOTE

Unless you have set the EDITMODE option of the SET ENVIRONMENT command to something other than Visual, visual mode is the default mode when you enter the TOOLSET Editor.

MARK Function

Step 6

The MARK temporary function key allows you to mark lines or characters on which you wish to perform an edit fucntion. In the program COMPUTE lines 23 and 24 are out of order. In order to move them into their correct sequence, you must first Mark them using the MARK key.

There are several steps to this process. Do the steps in order and check the resulting screen and keys with the sample screens you find here.

The MARK key is now available, but the lines in your source file that you need to mark are on a previous page. To access these,

>>>

Press the ALTSET key

>>>

Pressing this key activates the Permanent set of function keys. Notice that the Permanent set contains an ALTSET key also.

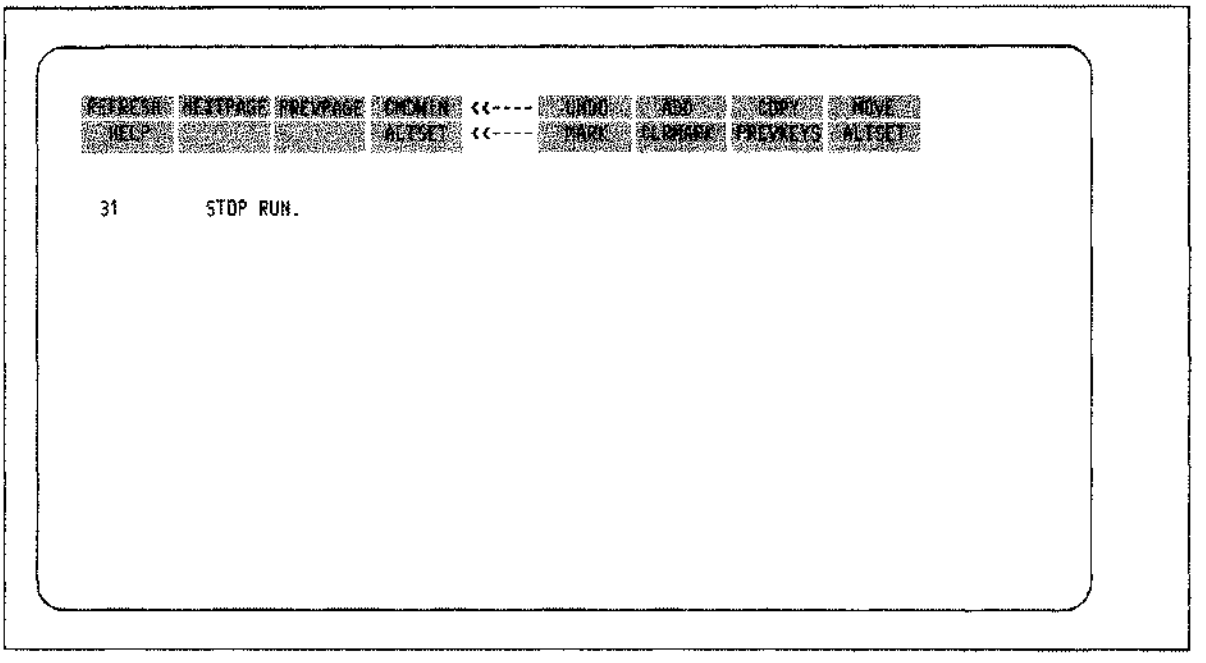


Figure 3-8. Accessing Permanent Function Keys.

Notice the PREVPAGE key located in the Permanent key set. This key displays the previous page of text while you are in visual mode.

>>>

Press the PREVPAGE key to display the previous page of the MYFILE source file.

>>>

The page of source containing lines 23 and 24 now appears on your screen. To activate the Temporary set of keys that contain the edit function keys:

```
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

   Press the ALTSET key

>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
```

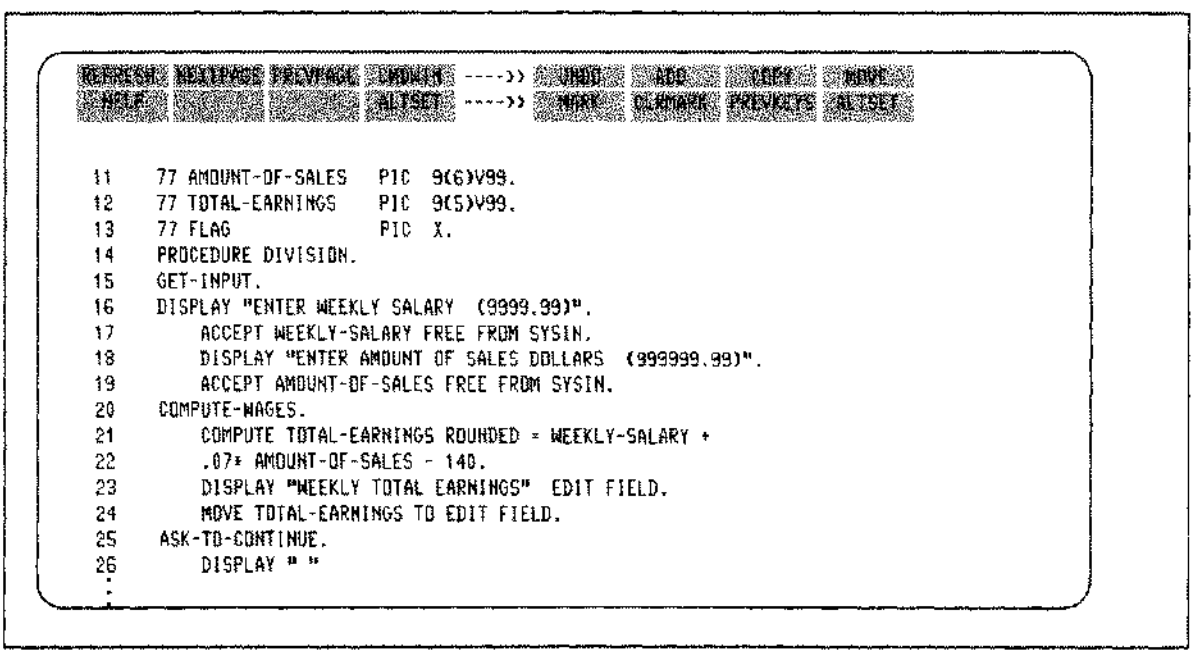


Figure 3-9. Accessing Previous Page of MYFILE.

>>>

To Mark line 23, move the terminal cursor to the beginning of the line and press the MARK key twice.

>>>

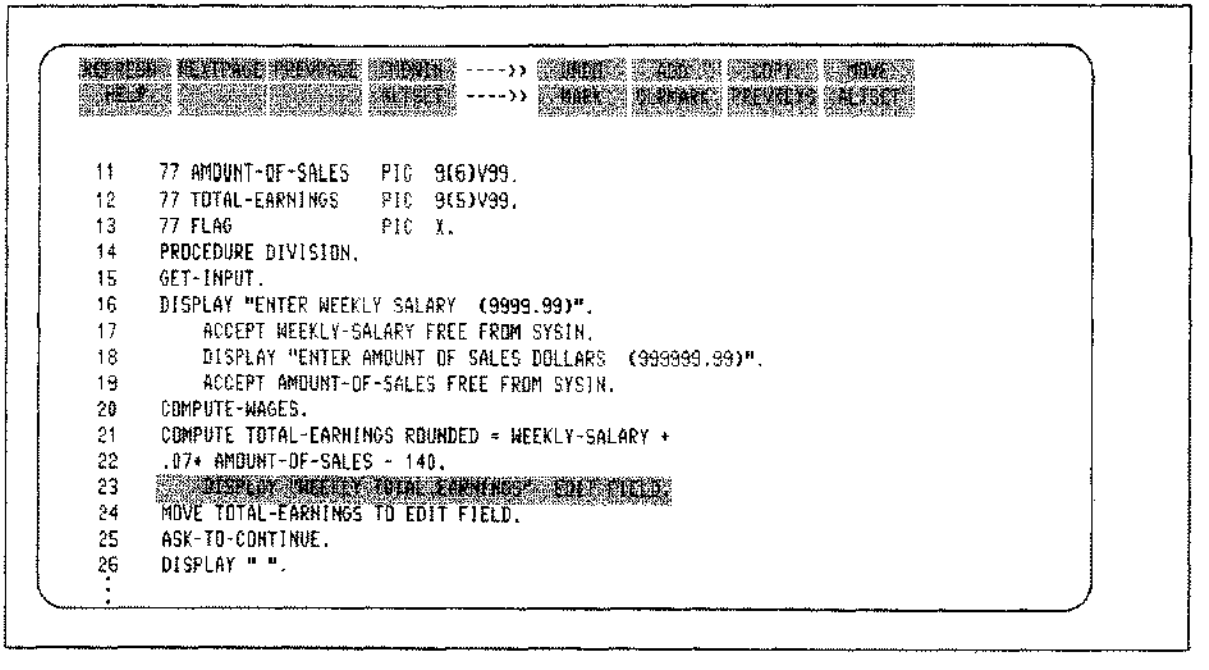


Figure 3-10. Mark Line Function.

These marks indicate to TOOLSET that you wish to perform an operation on the marked line. Notice that line 23 is highlighted after you have marked it.

To Mark the destination where you want to move line 23:

>>

Move the terminal cursor to the line above the point where line 23 is to be moved (line 24) and Mark that line by pressing the MARK key twice. Notice that it too is highlighted.

>>

MOVE Operation

Step 7

Both the line you want to move (23) and the place where you want it moved should be marked. Remember, the destination mark is one line above the point you want to insert line 23; in this case line 24.

To move line 23:

>>

Press the MOVE key

>>

Moving text removes it from the original location so that line 23 only exists at the destination you have indicated. In this example lines 23 and 24 are switched. Notice that line 23 becomes line 24.1.

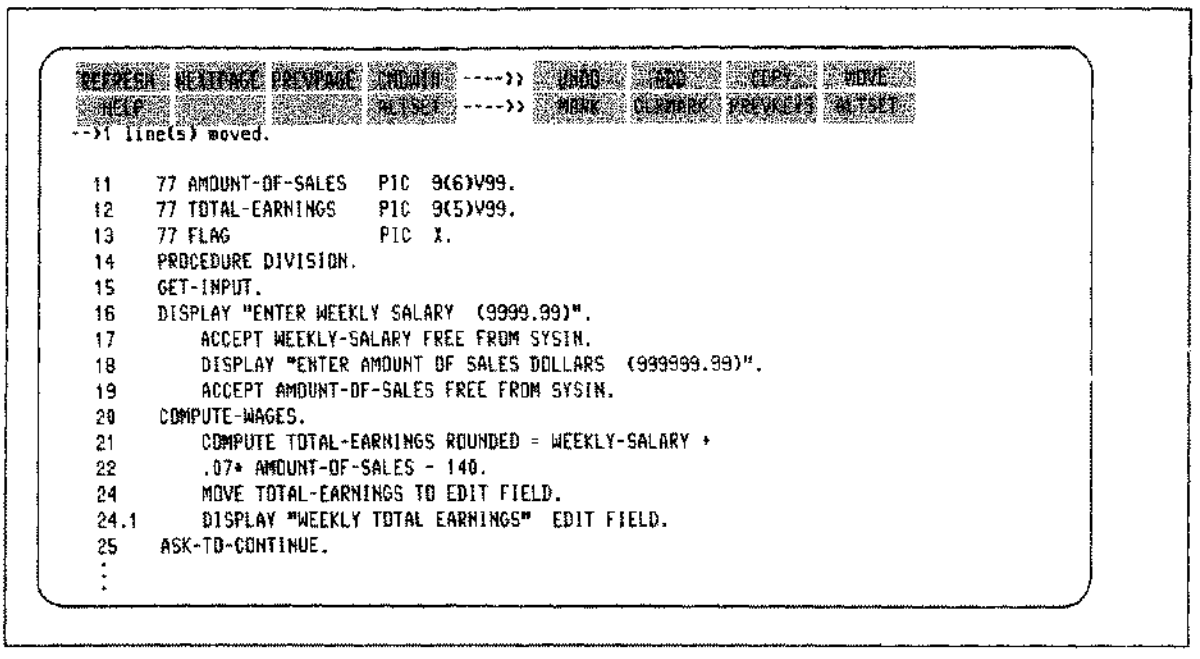


Figure 3-11. Move Function.

Step 8

All lines of COMPUTE are now in order, but the user-defined label EDIT FIELD is missing a hyphen and will not be recognized at compile time. To add these hyphens, use the TOOLSET FIND and CHANGE functions.

To access the FIND key:

>>>

Press the PREVKEYS key located on the current key set.

>>>

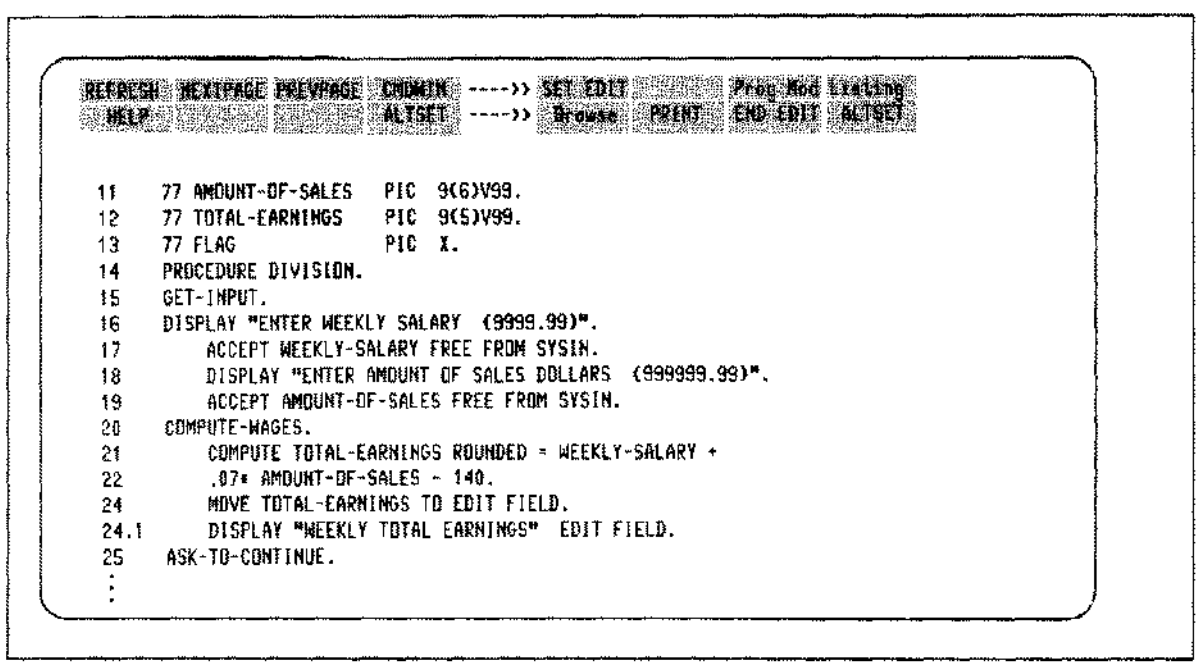


Figure 3-12. Accessing the FIND and CHANGE keys (1).

Next,

```
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
```

Press the Browse key

```
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
```

```
REFRESH  NEXTPAGE PRCVPAGE  CMDWIN  ---->>  SET N   FIND +N   FIND   FIND -N
  HELP                      ALTSET  ---->>  CHANGE          PREVKEYS ALTSET


 11     77 AMOUNT-OF-SALES   PIC  9(6)V99.
 12     77 TOTAL-EARNINGS    PIC  9(5)V99.
 13     77 FLAG              PIC  X.
 14     PROCEDURE DIVISION.
 15     GET-INPUT.
 16     DISPLAY "ENTER WEEKLY SALARY  (9999.99)".
 17         ACCEPT WEEKLY-SALARY FREE FROM SYSIN.
 18         DISPLAY "ENTER AMOUNT OF SALES DOLLARS  (999999.99)".
 19         ACCEPT AMOUNT-OF-SALES FREE FROM SYSIN.
 20     COMPUTE-WAGES.
 21         COMPUTE TOTAL-EARNINGS ROUNDED = WEEKLY-SALARY +
 22         .07* AMOUNT-OF-SALES - 140.
 24         MOVE TOTAL-EARNINGS TO EDIT FIELD.
 24.1       DISPLAY "WEEKLY TOTAL EARNINGS" EDIT FIELD.
 25     ASK-TO-CONTINUE.
  :
```

Figure 3-13. Accessing the FIND and CHANGE keys (2).

The FIND and CHANGE function keys are now available.

To find all occurrences of "EDIT FIELD";

>>>

Press the FIND key.

>>>

TOOLSET prompts you for the line number or string you want found in the message window on your screen.

>>>

Answer the prompt with [EDIT FIELD]

>>>

You can enter the find string in either single or double quotation marks, or brackets. If you indicate the string within quotes, it must be entered exactly as it appears in your program, that is, if it appears in all caps you must enter the string in all caps. If you enter the string within brackets TOOLSET performs any upshifting that is necessary.

When the string EDIT FIELD is found, the line in which it is located is moved to the top of the screen and the string is highlighed.

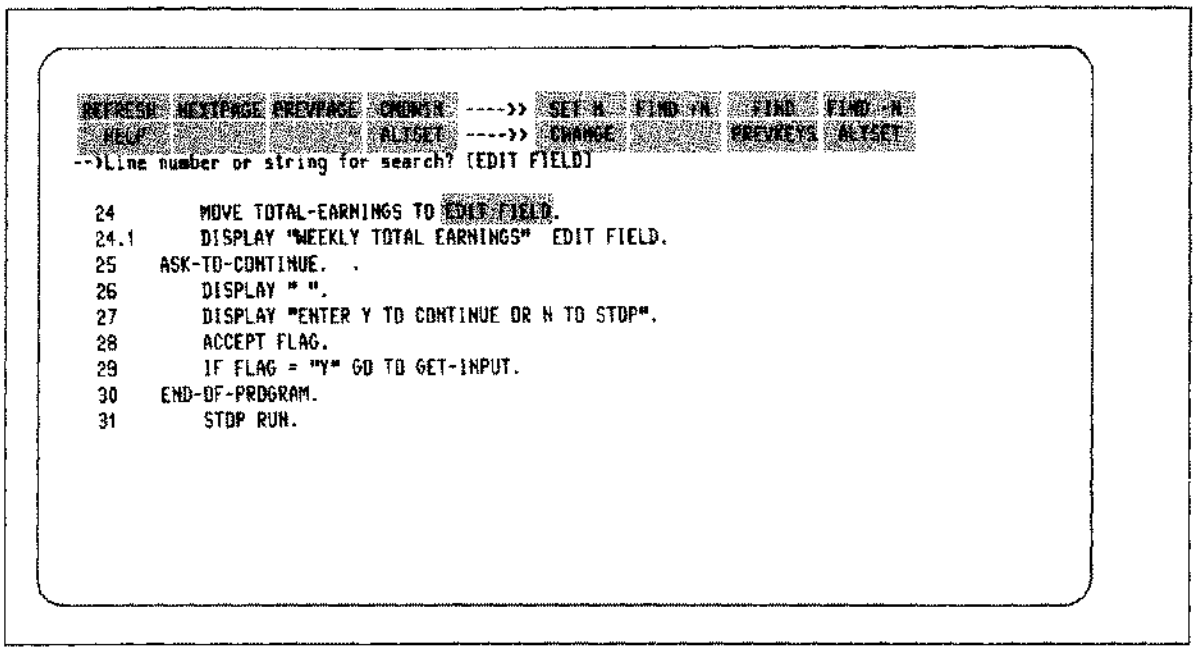


Figure 3-14. Entering Find-string.

When using the function key for the FIND operation, those lines containing EDIT FIELD are displayed one at a time. To view a subsequent line, you must press the FIND key again.

Change Function

Step 9

You should now have found the "EDIT FIELD" string on line 24 in the source file COMPUTE. To change this string to read correctly:

>>>

Press the CHANGE key.

>>>

A prompt for the Change string appears in the message window of your screen. This string should be entered exactly as you want it to appear in your source file enclosed in quotes.

>>>

Respond to the prompt by entering "EDIT-FIELD", and press the RETURN key on your terminal.

>>>

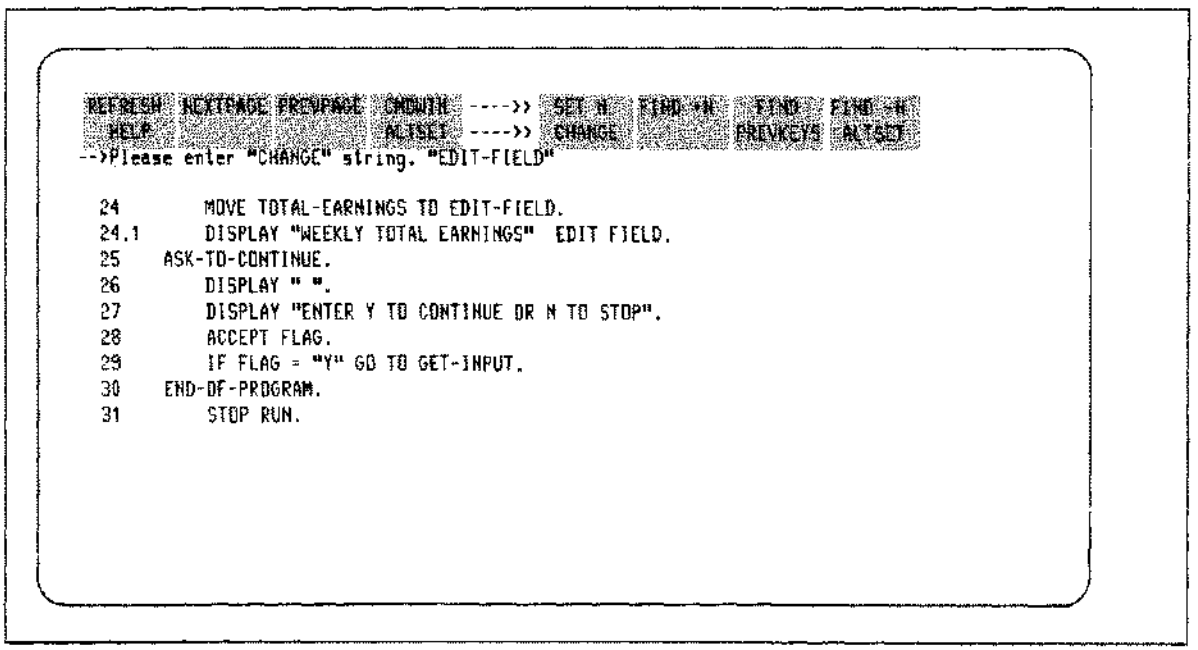


Figure 3-15. Change Function.

TOOLSET remembers the previous Find and Change strings you have entered, so the next time you press the FIND or CHANGE key, that string is indicated in the message window. There is a delay of a few seconds in which you can cancel the defined string by typing "NO" or pressing a carriage return. If you want to keep the same string, simply wait or type "YES" and TOOLSET locates or changes the already defined string.

>>>

FIND and CHANGE all EDIT FIELD strings to EDIT-FIELD.
There are three occurences.

>>>

NOTE

(1) The FIND+N and FIND-N keys allow you to view the next 'N' lines in your Edit file. The default number of lines is 10. You can set the value of 'N' with the SET N key or with the SET ENVIRONMENT command. See Section 2 for further information.

(2) The Edit changes above could also have been accomplished by using the terminal cursor keys to position the cursor between EDIT and FIELD and typing a dash.

Exiting the TOOLSET Editor

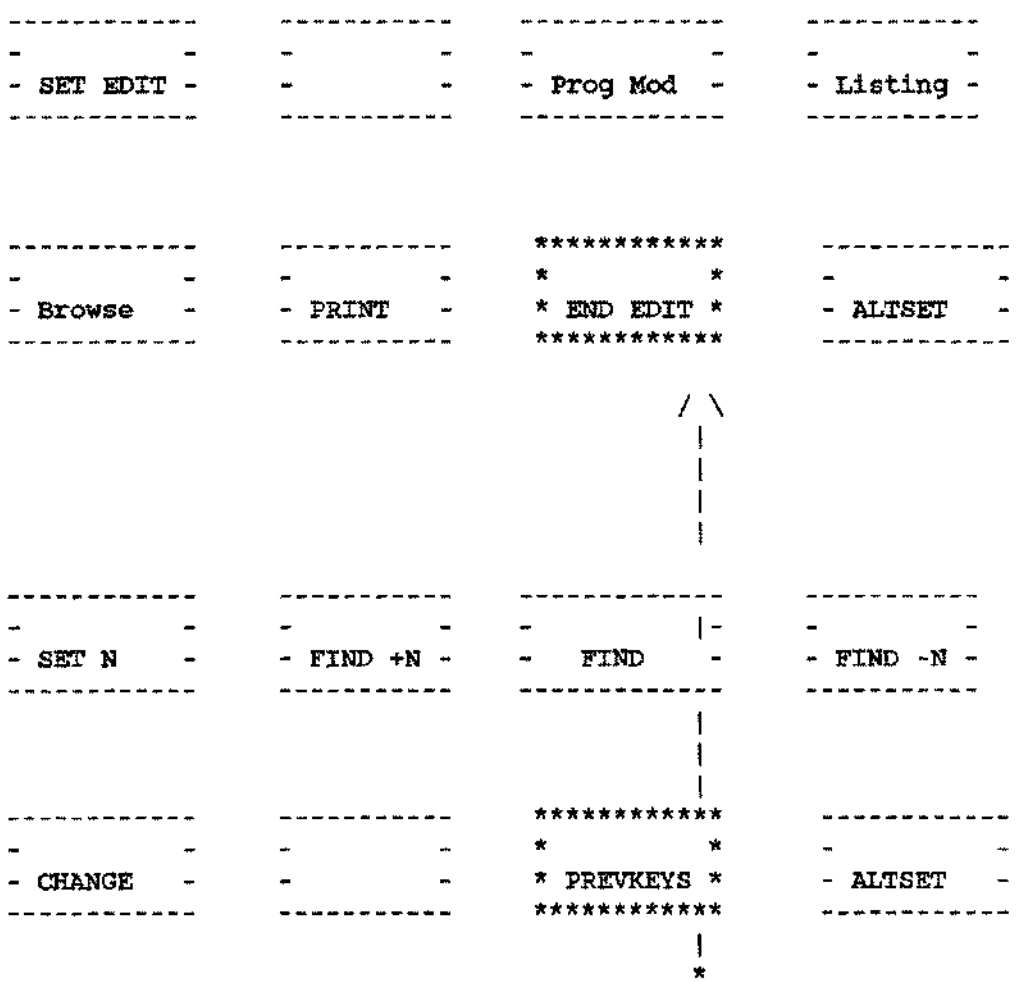
Step 10

To exit the TOOLSET Editor:

>>>

First, press the PREVKEYS key. Next, press the END EDIT key.

>>>



Pressing the END EDIT key returns you to the previous function key set before editing, in this case the Workspace key set that is also available when you press the Workspace key.

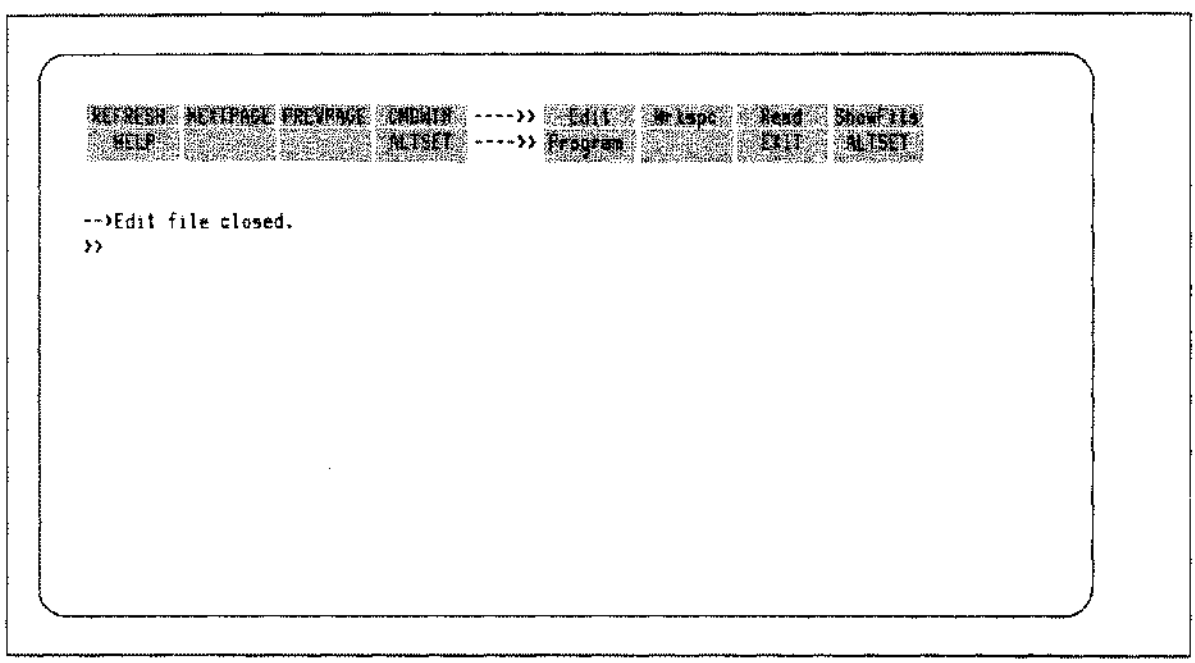


Figure 3-16. Closing the Edit File.

Step 11

This step is helpful when you wish to edit an ASCII formatted source file that you have already created in another of the HP text editors.

NOTE

If you are using a program that uses COPYLIB, the COPYLIB cannot be edited with the TOOLSET editor. You should use the COBEDIT program that is documented in Section XIV of the COBOL II Reference Manual. COBEDIT can be run from within TOOLSET by entering

```
:RUN COBEDIT.PUB.SYS
```

at a command prompt (>>). $INCLUDE files can be edited in TOOLSET.

Converting ASCII source files to TSAM source files.

Your TOOLSET Workspace can only manage files that are in TSAM (TOOLSET Access Method) format. To change your file from ASCII to TSAM:

```
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

Use the CONVERT command.

 CONVERT filename

>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
```

You can enter a new (destination) file name if you wish, but it is not necessary. If you do not specify a destination file, TOOLSET asks you if you wish to purge the old (ASCII) version of filename. During conversion, a SET EDIT options menu is displayed to allow you to define options for the new TSAM file you are creating. This menu is discussed in Step 3.

The file conversion is confirmed with the message

```
--> Convert finished successfully.
```

The ASCII file has now been replaced with its TSAM equivalent. A double arrow prompt is present indicating you are in command mode. Note that there is no Function key equivalent for the CONVERT command.

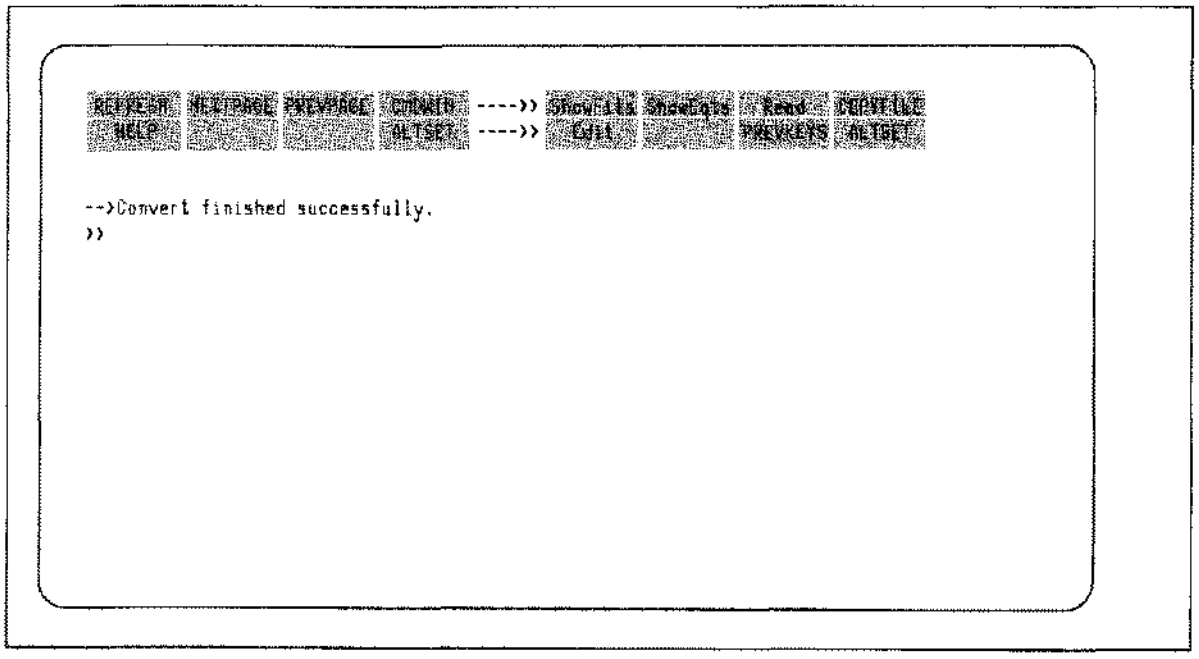


Figure 3-17. Converting an ASCII file.

The complete syntax and explanation of the CONVERT command can be found in Section 4.

Program Translation

Step 12

Compiling Your Source File

The next step in the program development cycle is to compile the source file MYFILE.

To access the COMPILE key:

>>

Press the PREVKEYS key, then press the Program key

>>

This results in a menu of the source files contained in Workspace ONE. This menu allows you to indicate the program that you want to compile. To compile COMPUTE,

>>

Tab to the program COMPUTE and place an X in the box next to it.

>>

To compile:

>>>

Press the Compile key.

>>>

Notice that the program COMPUTE is highlighed.

You are returned to command mode, and a message indicating that your file is compiling appears at the top of your screen.

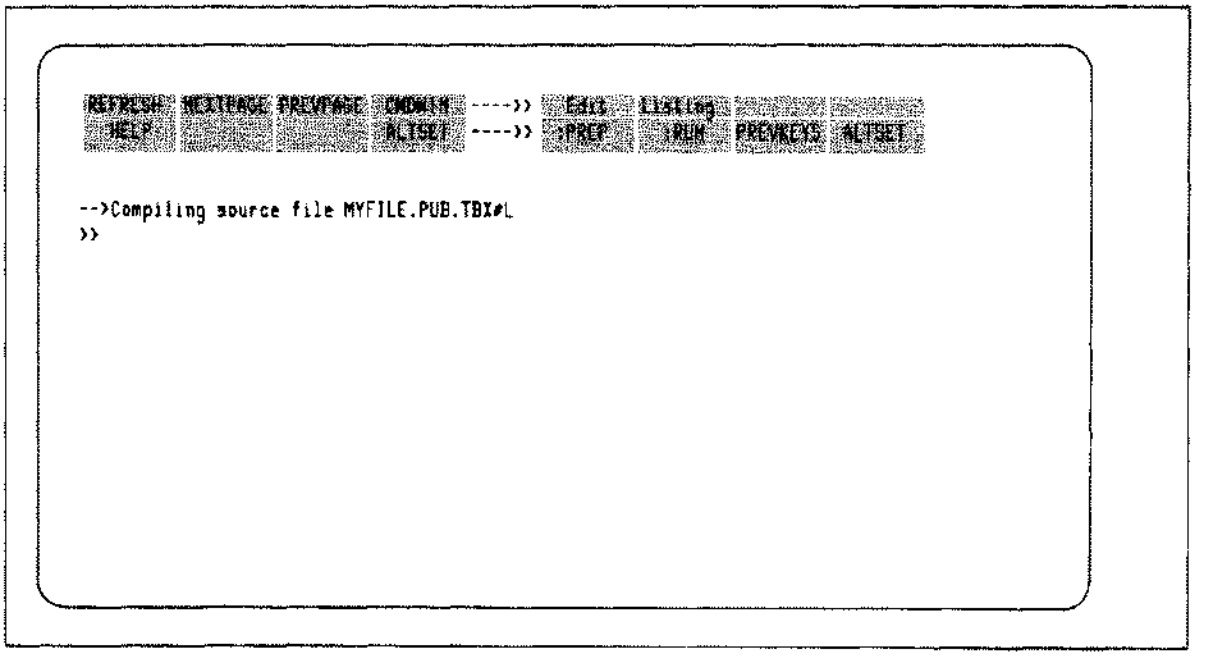


Figure 3-18. Compiling the Source file.

While your program is compiling, you can perform other operations as long as they do not access the USL or list file. A list file is the compiler listing. At this point, one does not exist for MYFILE, but is being created as it compiles. If you compile subsequent versions of MYFILE, the list file is not available to you while the compile is taking place.

NOTE

> You cannot edit the source file if the LATEST version is the version that is compiling. The LATEST version is the most recent copy of the source file in your workspace, and is the only version to which you can make changes. Versions are discussed in Section 4.

When your compilation has completed, the number of compile errors and warnings are displayed.

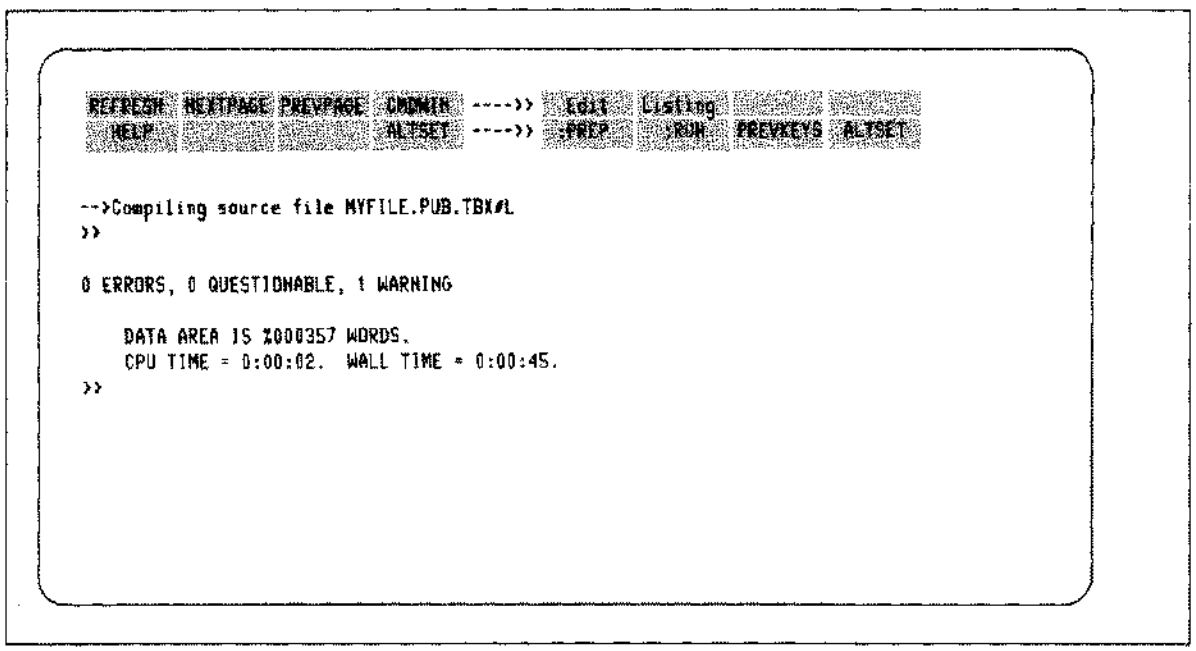


Figure 3-19. Completed Compile.

Compilation Listing and ERRORS Key

Step 13

Program COMPUTE compiled, but contained 1 warning. To look at the compilation listing:

>>>

Press the Listing key.

>>>

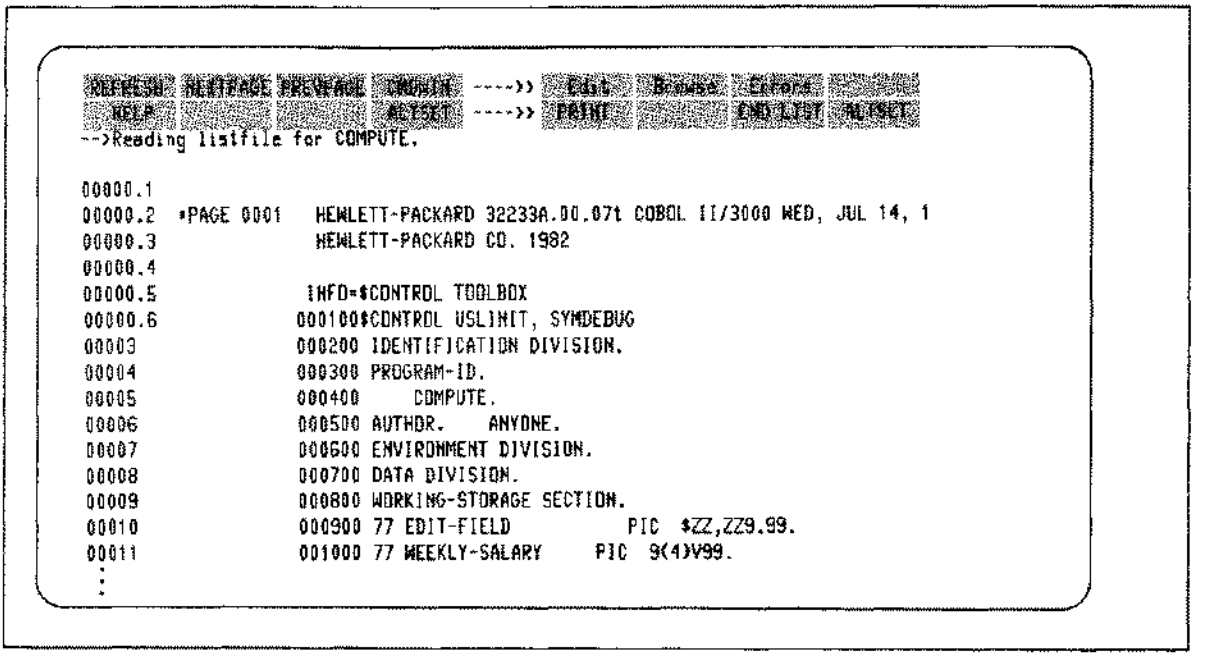


Figure 3-20. Compile Listing.

The first page of the listing is produced on your screen. To make a hard copy of the listing:

>>

Press the PRINT key. Pressing PRINT prompts

"Line range to be printed?"

Respond "ALL". The print is confirmed with:

--> offline listing complete

Keep this copy. It is used later on in this section.

>>

To view subsequent pages of the listing on your screen you can press the Browse key. This produces a secondary key set containing the FIND +N and FIND -N keys that allow you to go forward or backward by the number of lines to which 'N' is set. The default is 10.

To look at the Compile errors and warnings:

>>>

Press the Errors key.

>>>

```
REFRESH  NEXTPAGE PREVPAGE  CMDWIN   ---->>   Edit             Listing   PRINT
  HELP                      ALTSET   ---->>   MARK    CLRMARK  PREVKEYS  ALTSET

00036     COBOL ERRORS:
00037
00038     LINE # SEQ #   COL  ERROR  SVRTY  TEXT OF MESSAGE
00039     ------ -----   ---  -----  -----  ---------------
00040
00041     00023  002200  14    004     W    MISSING SPACE.
00042
00043     0 ERRORS, 0 QUESTIONABLE, 1 WARNING
00044
00045         DATA AREA IS %000357 WORDS.
00046         CPU TIME = 0:00:02.  WALL TIME = 0:00:45.
00001     5
00000     3$
```

Figure 3-21. Errors Screen.

To locate the warning that occurred in program COMPUTE:

>>

Mark the warning line by moving the terminal cursor to the error and pressing the MARK key twice. The line is highlighted.

>>>

```
REFRESH  NEXTPAGE PREVPAGE ENDQUIT  ---->> EDIT              LISTING  PRINT
 HELP                      ALTSET   ---->> MARK     SUMMARY  PREVKEYS ALTSET

00036     COBOL ERRORS:
00037
00038     LINE # SEQ #   COL  ERROR  SVRTY  TEXT OF MESSAGE
00039     ------ -----   ---  -----  -----  ---------------
00040
00041     00023  000230  14   004    W      MISSING SPACE.
00042
00043     0 ERRORS, 0 QUESTIONABLE, 1 WARNING
00044
00045         DATA AREA IS %000357 WORDS.
00046         CPU TIME = 0:00:02.  WALL TIME = 0:00:45.
00001     5
00000     3$
```

Figure 3-22. Marking Compile Warning.

To locate this error in the compilation listing:

\>>>

Press the Listing key again. The listing is displayed at the location of the warning line. The warning line is highlighted.

\>>>

```
REFRESH NEXTPAGE PREVPAGE CMDWIN ---->> Edit   Browse  Errors
  HELP                    OBJSET ---->> PRINT          END LIST OUTPUT

00023              002200     .07* AMOUNT-OF-SALES + 100.
00024               002400     MOVE TOTAL-EARNINGS TO EDIT-FIELD.
00025               002410     DISPLAY "WEEKLY TOTAL EARNINGS"  EDIT-FIELD.
00026               002500 ASK-TO-CONTINUE.
00027               002600     DISPLAY " ".
00028               002700     DISPLAY "ENTER Y TO CONTINUE OR N TO STOP".
00029               002800     ACCEPT FLAG
00030               002900     IF FLAG = "Y" GO TO GET-INPUT.
00031               003000 END-OF-PROGRAM.
00032               003100     STOP RUN.
00033
00034    PAGE 0002/COBTEXT  COMPUTE
00035
00036    COBOL ERRORS:
00037
00038    LINE # SEQ #   COL  ERROR  SVRTY  TEXT OF MESSAGE
00039    ------ -----   ---  -----  -----  ---------------
```

Figure 3-23. Locating Warning Line in Listing.

To locate the warning in your source file:

>>

Press the Edit key. The source line containing the warning appears as the first line on your screen.

>>

```
REFRESH  NEXTPAGE PREVPAGE CMDWIN  ---->> SET EDIT          Prog Mod Listing
  HELP                      ALTSET  ---->>  Browse   PRINT   END EDIT  ALTSET
-->File MYFILE.PUB.TBX open for editing.

 22        .07* AMOUNT-OF-SALES - 140.
 24        MOVE TOTAL-EARNINGS TO EDIT-FIELD.
 24.1      DISPLAY "WEEKLY TOTAL EARNINGS"  EDIT-FIELD.
 25     ASK-TO-CONTINUE.
 26        DISPLAY " ".
 27        DISPLAY "ENTER Y TO CONTINUE OR N TO STOP".
 28        ACCEPT FLAG.
 29        IF FLAG = "Y" GO TO GET-INPUT.
 30     END-OF-PROGRAM.
 31        STOP RUN.
```

Figure 3-24. Locating Warning Line in Source File.

NOTE

The compilation warning can be corrected by putting a space between the asterisk and the 7 in line 22 of the source file. COMPUTE will, however, prep and run without correcting this warning.

The Edit file must be closed if you wish to re-compile. To close the Edit file:

>>

Press the END EDIT key.

>>

Prepping the USL File

Step 14

To Prep the USL file you created in Step 13:

>>>

Press the :PREP key

>>>

When the file has been prepped, an End of Program message is printed on your screen.

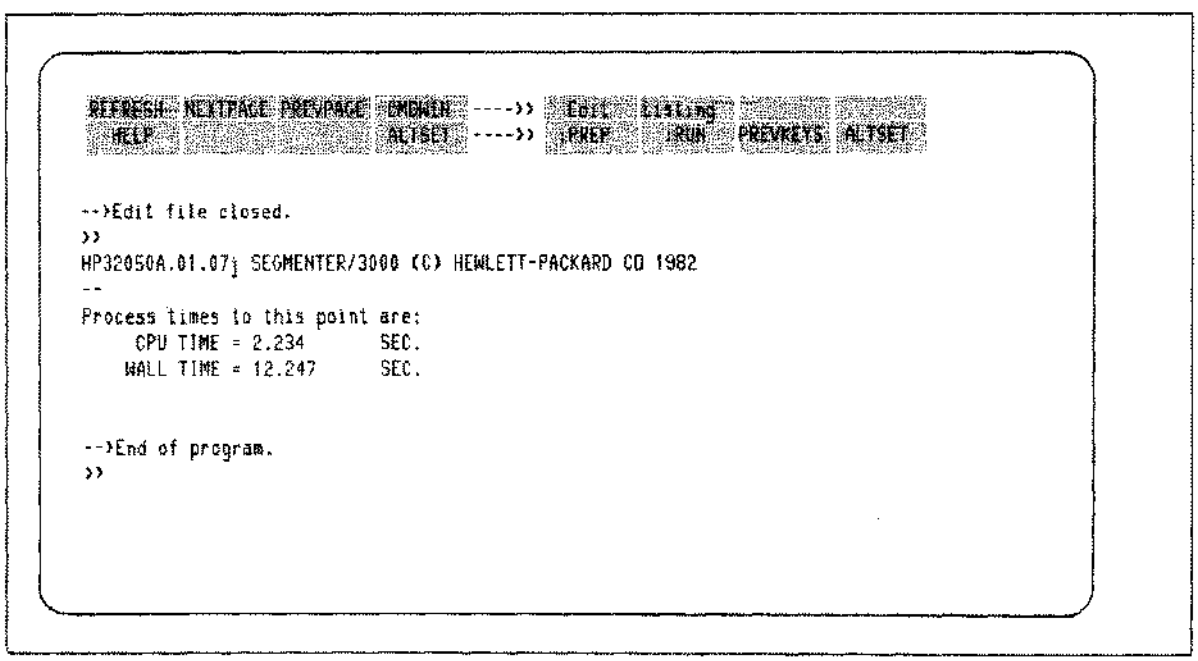


Figure 3-25. File Prep.

NOTE

To access the :PREP or :RUN key if it is not displayed, and you do not want to re-compile your sourcse file, enter the Compile command without parameters after a TOOLSET prompt (>>).

Unless you specify differently in the SET PROGRAM menu, the :PREP key uses the MPE :PREP command defaults. The SET PROGRAM menu is accessed when you create your Workspace and can be re-accessed by entering the SET PROGRAM command. A SET PROG key exists in the subset of keys available when the Program key is pressed. The SET PROGRAM command is explained in Section 2.

You can also specify prep options in the :PREP command to override the MPE defaults or the SET PROGRAM prep options.

Step 15

To run your program:

>>>

Press the :RUN key.

>>>

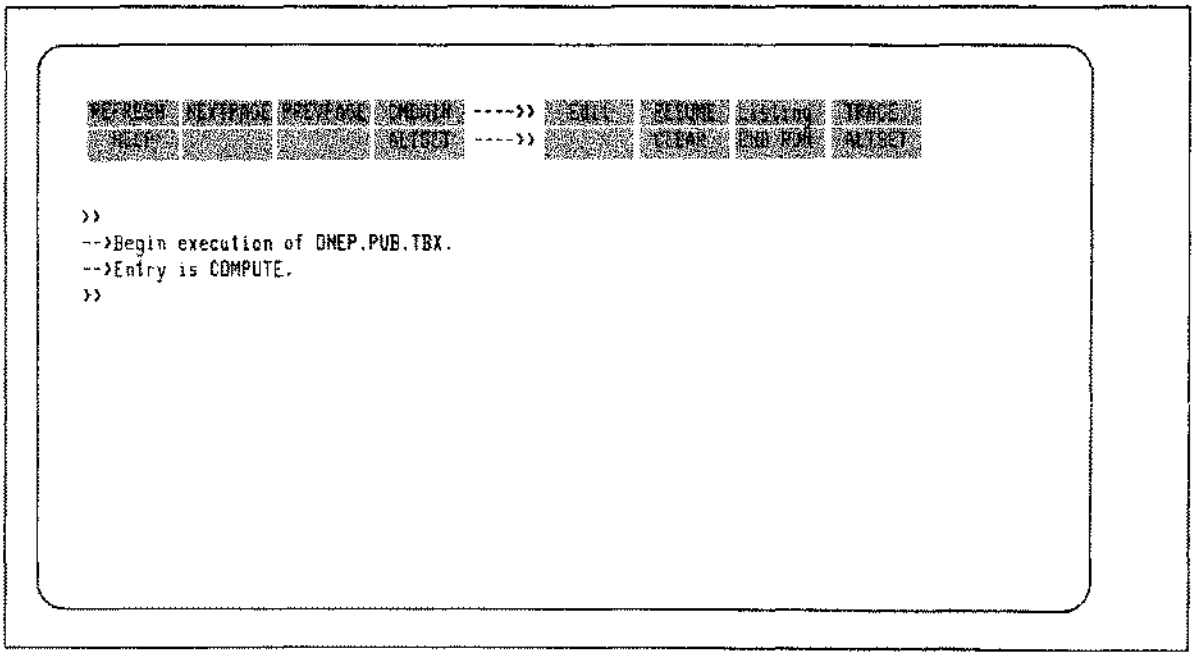
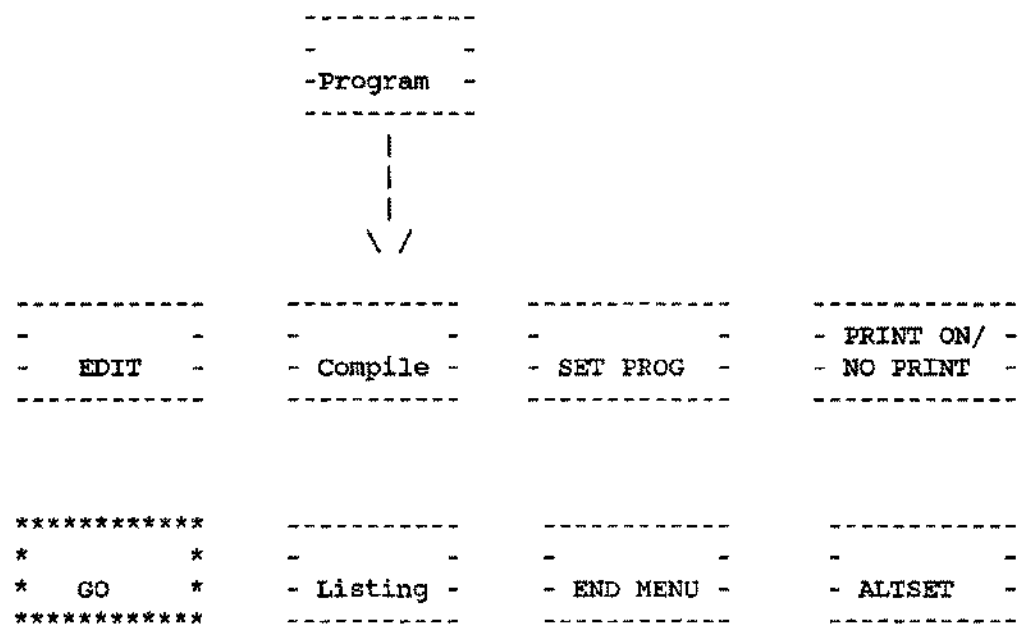


Figure 3-26. Run Screen.

If you have designated the SYMDEBUG parameter in the $CONTROL statement of your program, and have not prepped the program with the NOSYM (No Symbolic Debug) option, then the 'Begin Execution' message is displayed and control is transferred from MPE to TOOLSET. The 'Entry is' message indicates that your program is paused at the first executable statement. At this time Symbolic Debug functions can be performed.

NOTE

TOOLSET provides a GO function key that compiles, :PREPs and :RUNs your program in one step. It is available as one of the subset of keys available when you press the Program key.



RUNNING YOUR PROGRAM WITH SYMOBLIC DEBUG

Setting Breakpoints

Step 16

To set a breakpoint in program COMPUTE, you can enter the AT command following the >> prompt on your screen. Remember, when you run your program with Symbolic Debug, your program is paused at the first executable statement.

>>

Enter AT COMPUTE-WAGES after the >> prompt on your screen. TOOLSET confirms that your breakpoint is set and displays another >> prompt. Now press the RESUME key. Notice that the program COMPUTE prompts you to enter WEEKLY SALARY and SALES DOLLARS amounts. Enter an amount for each.

>>

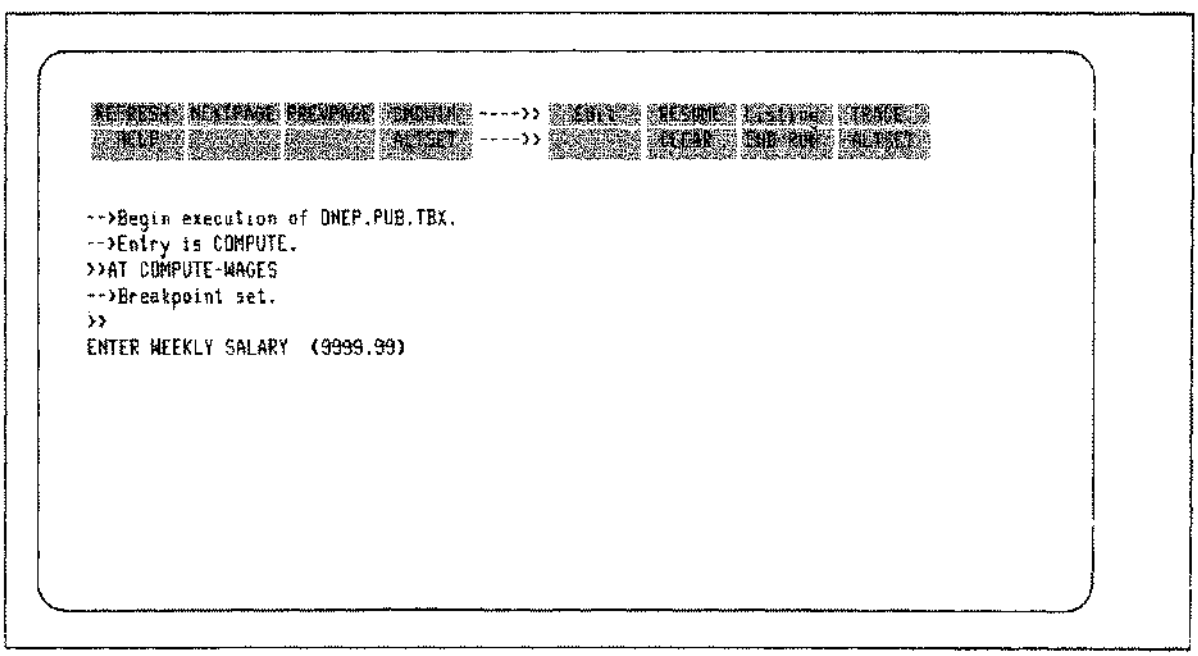


Figure 3-27. Setting a Breakpoint.

Your program executes until the breakpoint COMPUTE-WAGES is reached. You are then given the compiler generated line number where COMPUTE-WAGES is located.

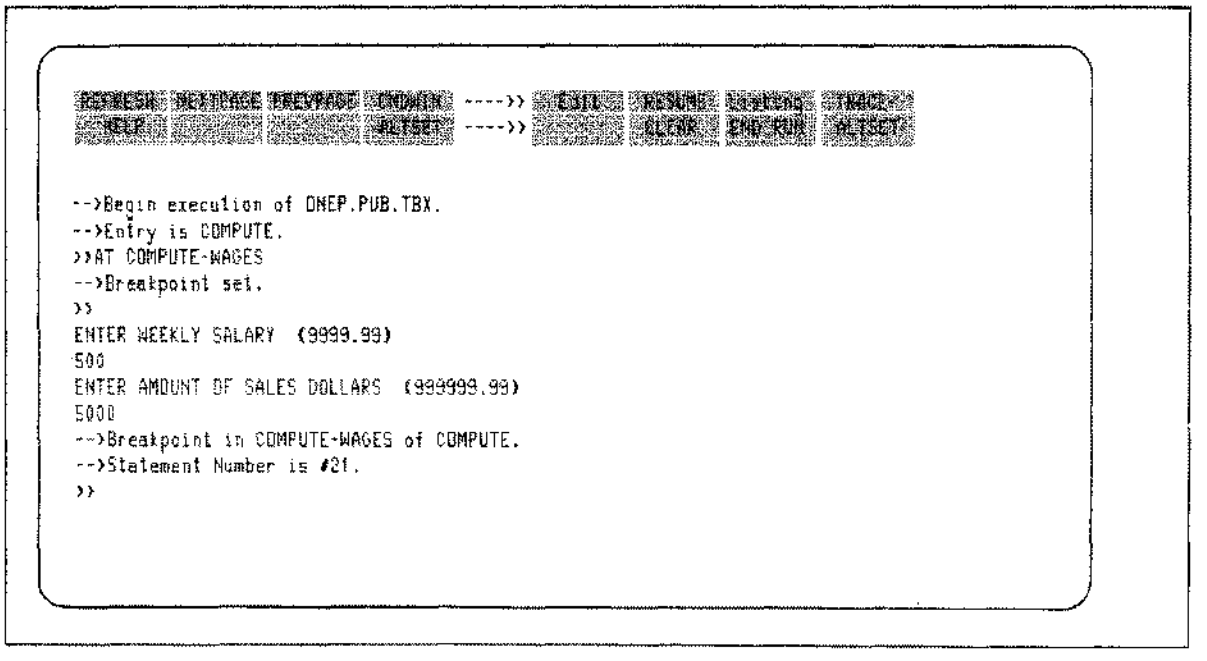


Figure 3-28. Program Stopped at Breakpoint.

Before resuming execution, enter the following after the >> prompt:

```
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

 AT ASK-TO-CONTINUE DO (DISPLAY TOTAL-EARNINGS)

>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>


 DO NOT RESUME YET.  Now enter


>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

  SHOW DEBUG following the >> prompt

>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
```

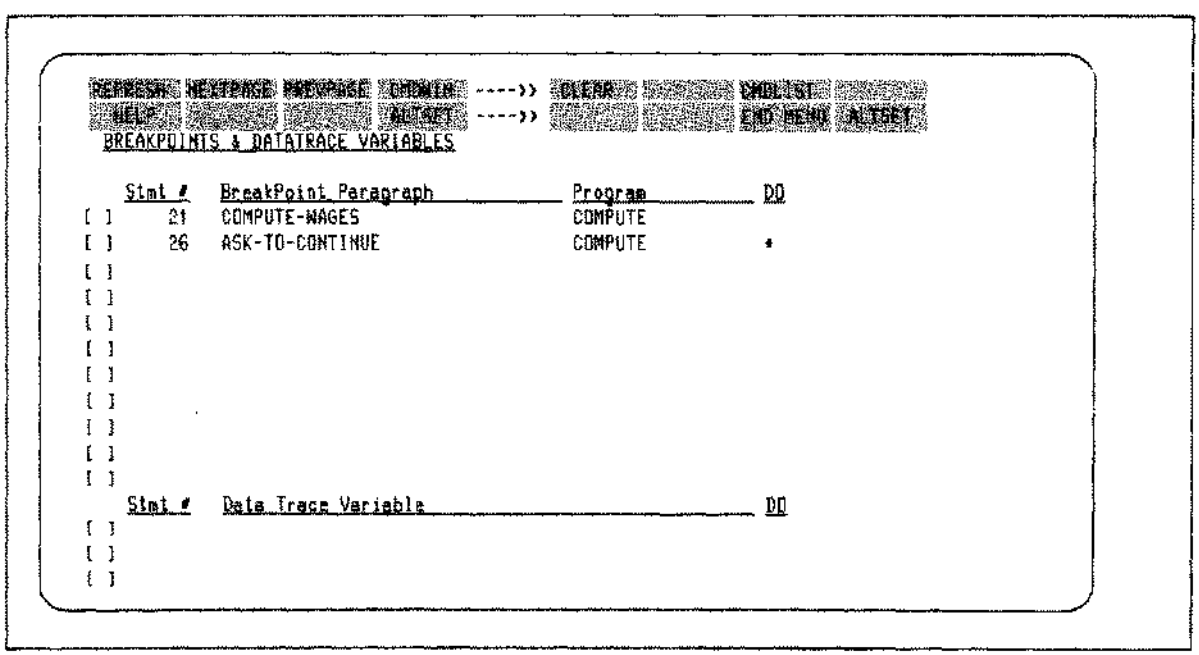


Figure 3-29. Show Debug Menu.

Both of the breakpoints you have set are displayed on the menu with their compiler listing statement number. Note the asterisk under the DO column for the breakpoint ASK-TO-CONTINUE. This indicates there is a command list associated with this breakpoint, and refers to the DO (DISPLAY TOTAL-EARNINGS) you entered when you set up the ASK-TO-CONTINUE breakpoint. The DISPLAY command causes the current contents of the data item TOTAL-EARNINGS to be displayed on your terminal.

>>

Press the END MENU key. When you see the >> prompt, type RESUME.

>>

Your program continues until the breakpoint ASK-TO-CONTINUE is reached. Notice that the current contents of TOTAL-EARNINGS are displayed also.

>>

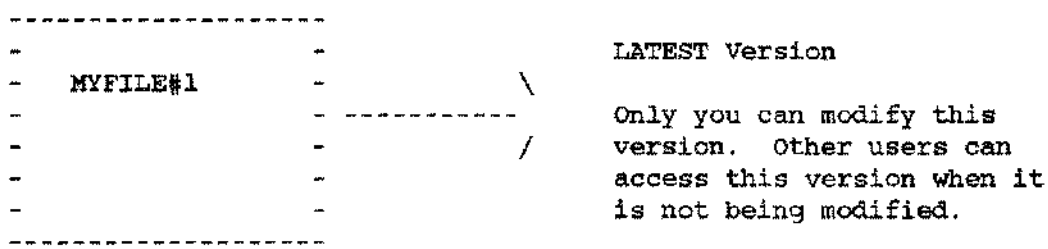
Press the RESUME key. When program COMPUTE completes running you are prompted with 'END OF PROGRAM'.

>>

File Versions

Typically source files go through many variations and modifications and a source file such as MYFILE may be only a part of a larger program. Instead of creating a new file each time you want to add to MYFILE or add a particular variation of it to another source file, TOOLSET allows you to keep versions of your source. These versions can be shared between users. You can have up to 32 active version of a file at one time. An active version is one that has not been purged.

When you created MYFILE, TOOLSET numbered it version 1. Since it is the version you have been modifying, it is also the LATEST version. The LATEST version of a file can only be modified by its owner. Other users cannot read the LATEST version while it is being modified.



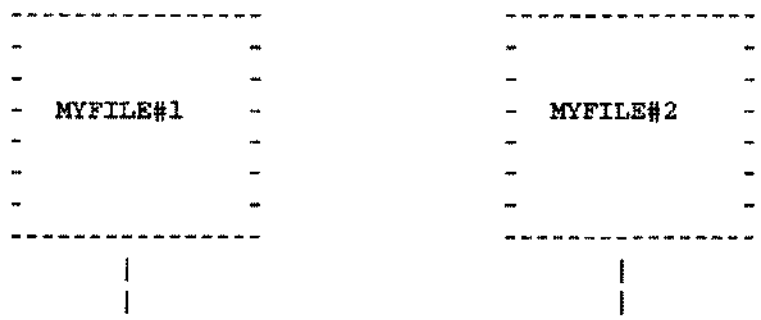
Once you are satisfied that the current version of MYFILE is stable and does not require further modification, you can 'freeze' version #1 with a SETVERSION command. This version can then be read by any user but can no longer be modified.

Step 17

>>>

Enter the command SETVERSION MYFILE

>>>



Can no longer be modified.

Version #2 is now the LATEST version, and can be modified by its owner. By default it is the version other users reference when it is not being modified.

A SETVERSION command freezes a file so that no further changes can be made to it, and creates a new LATEST version that can be modified and assigns it the next version number. It does not explicitly designate a reference version for other users to access. The default access version for other users will still be the LATEST version. To allow other users to share a stabilized version of MYFILE, you can designate a specific usable version with the SETREF command.

Step 18

```
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

Enter the command SETREF MYFILE#1

>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
```

Version #2 is still the LATEST and modifiable version, but is no longer the default access version for other users. Other users now reference version #1.

Step 19

>>>

Enter the SHOW FILES command.

>>>

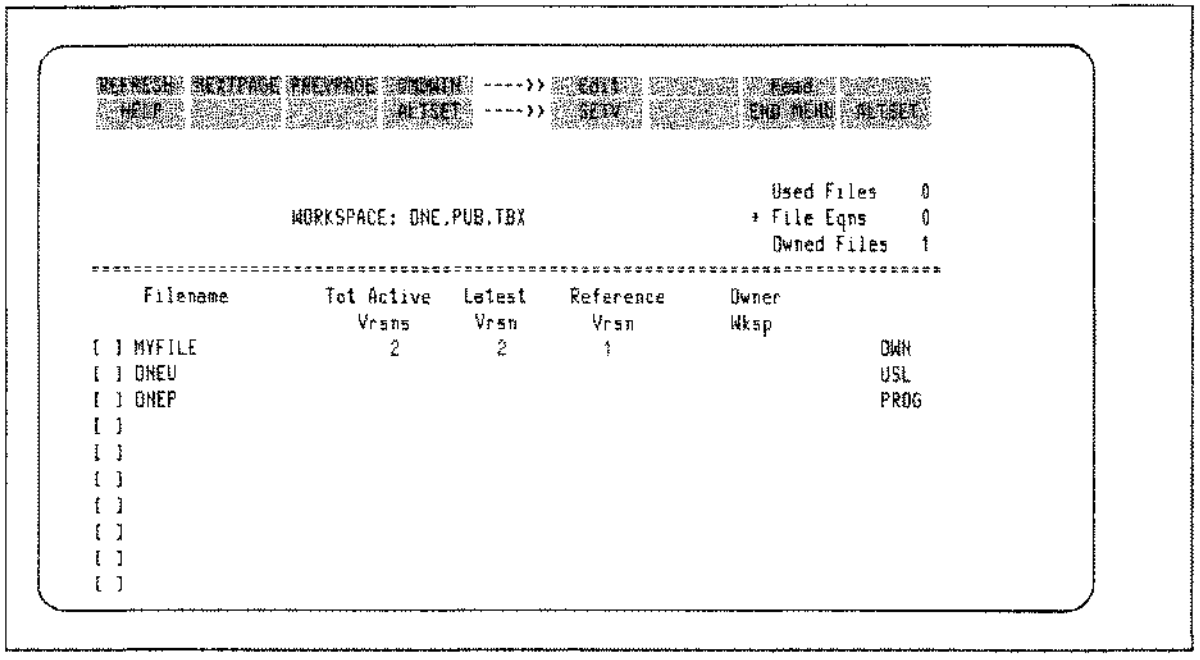


Figure 3-30. Show Files Menu.

NOTE

You can Edit, Read, or perform a SETVERSION command on any of the source files that appear on a Show Files menu by placing a character in the box next to that file. The File equations (*) refer to any USE commands that have been issued on a file. The SETVERSION, SETREF, and USE commands are discussed in Section 4.

EXITING TOOLSET

Step 20

To exit the TOOLSET utility, enter the EXIT command after the >> prompt or press the EXIT function key when available. From the SHOW FILES menu:

>>>

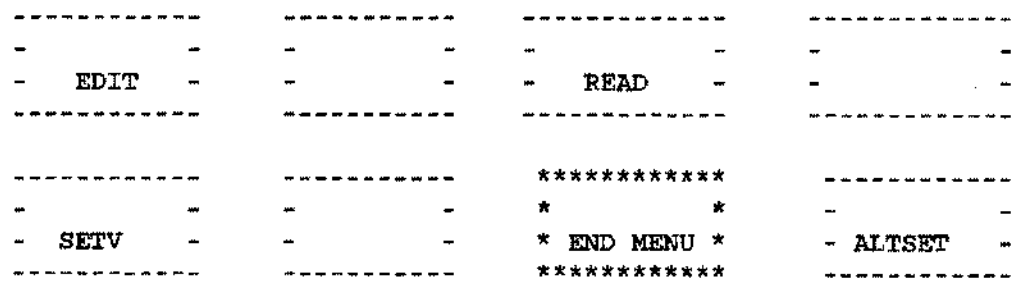
Press the END MENU key.

>>>

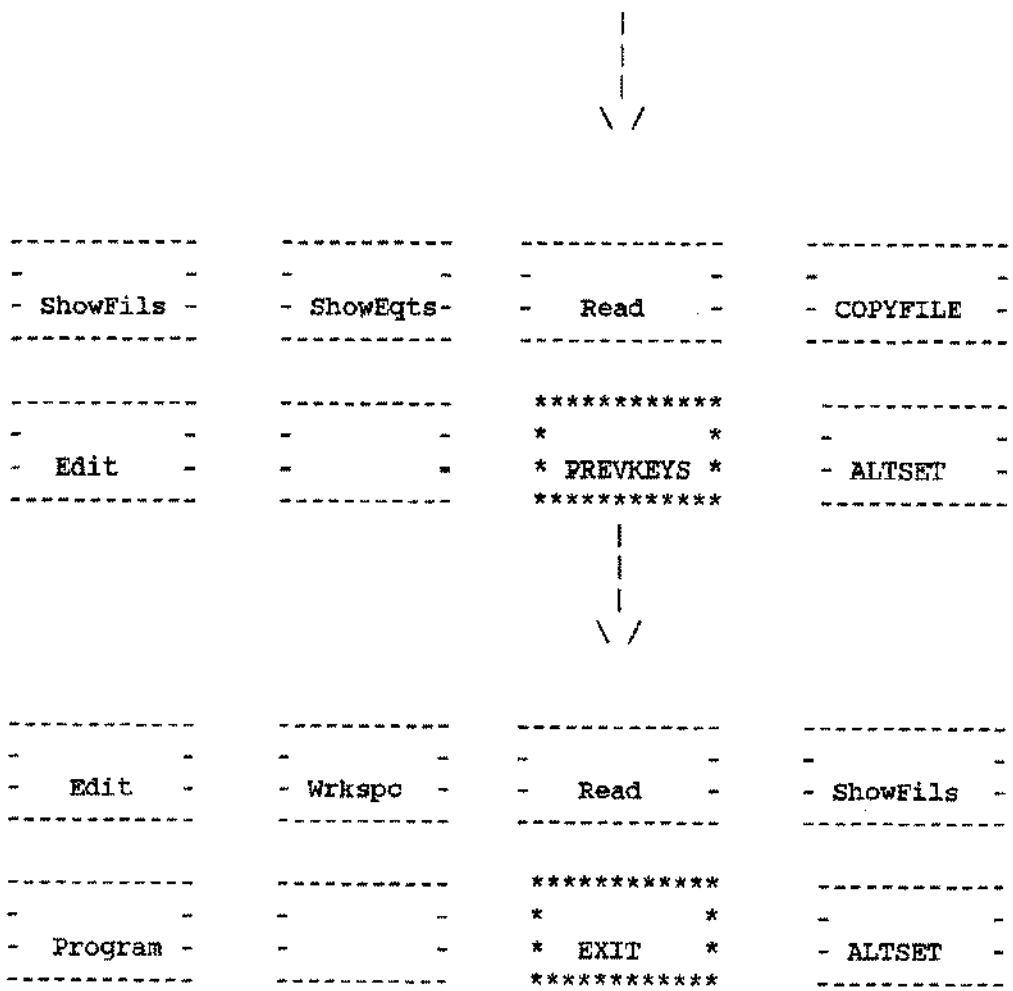
>>

Press the PREVKEYS key, then the EXIT key.

>>



Show Files Menu



SECTION 4

WORKSPACE and FILE MANAGEMENT

HPTOOLSET keeps track of your source files and their versions by associating each source file with a Workspace. Each workspace is a collection of files that go into the development of one program, including the source files, Program file, and USL file. In addition, the Workspace maintains a directory of which of these files it OWNs and which files it USEs, and contains information regarding the USL file, the Program file and the source file's compile priority.

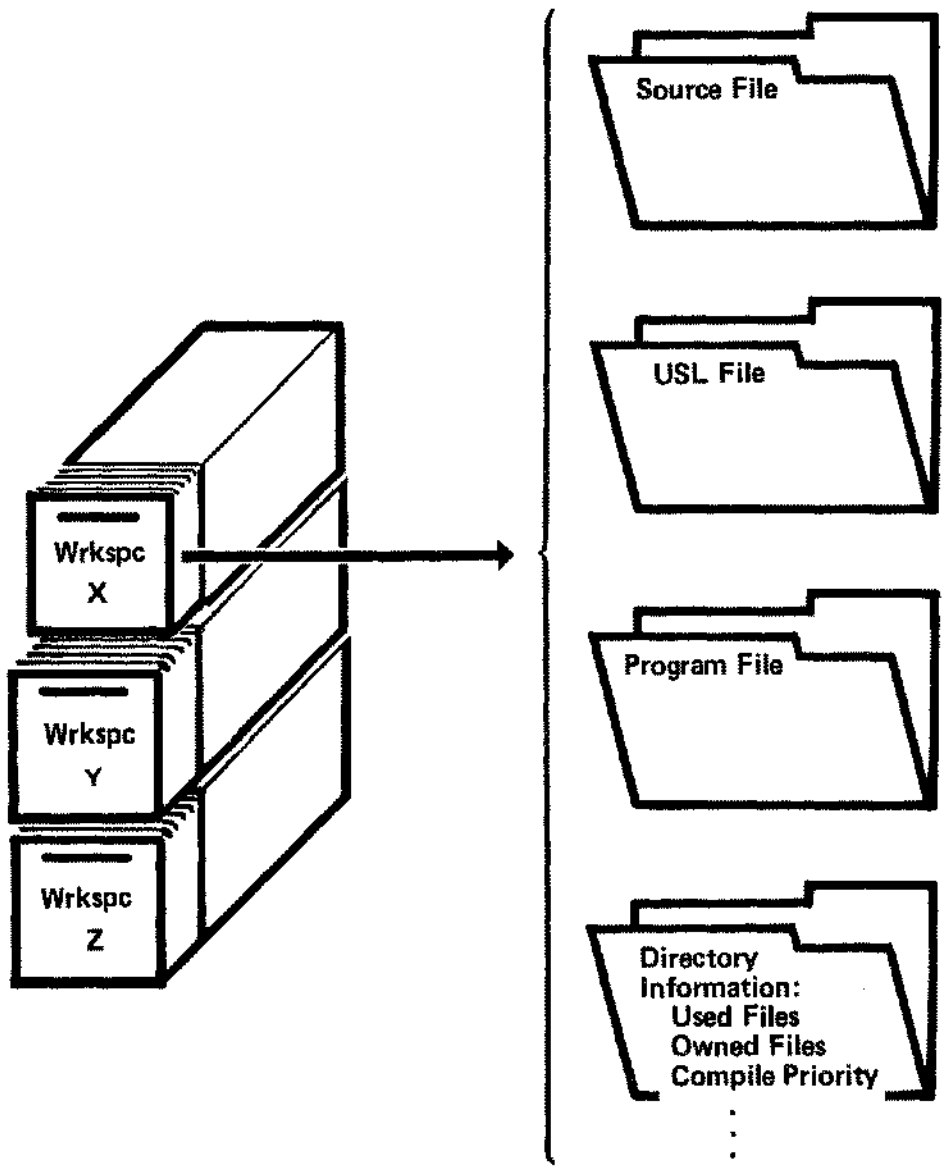


Figure 4-1. Workspace Management.

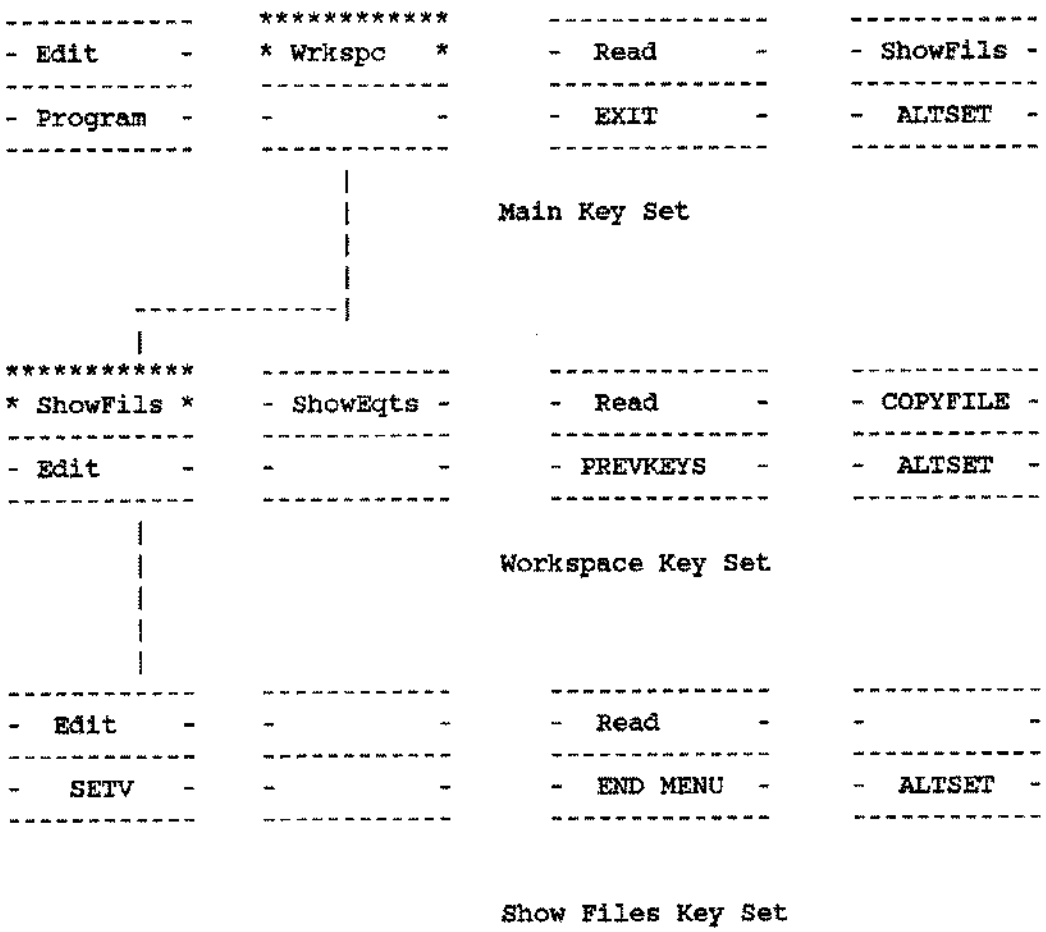


Figure 4-2. Workspace Key Set Flow.

TSAM files

All source files that you create within HPTOOLSET are stored in a random access TOOLSET internal format called the Toolset Access Method, or TSAM, that has its own intrinsics to perform file operations.

Since TSAM files are built and accessed with the dynamic locking access option, you must have lock access at the account, group and file levels. If you are accessing more than one file at a time, such as editing one file and reading another, TOOLSET must be built with Multiple RIN capability. (See the MPE INTRINSICS REFERENCE Manual for information on MR capability).

Only source files that are in TSAM format are managed by the TOOLSET Workspace Manager. Any operations performed on TSAM files directly from MPE such as renaming and purging are not recorded by the Workspace Manager even though the operation is actually performed. It is recommended that you use the TOOLSET PURGE and RENAME commands instead of their MPE equivalents so that the Workspace Manager is aware of those file changes.

Creating a Workspace

Almost all HPTOOLSET operations require a defined workspace, so upon logging on and accessing TOOLSET you need to create a workspace or declare an already defined one. To do this, either press the Wrkspc function key (1), or enter the WORKSPACE command (2) following the >> prompt on your screen. You can only be associated with one workspace at any given time.

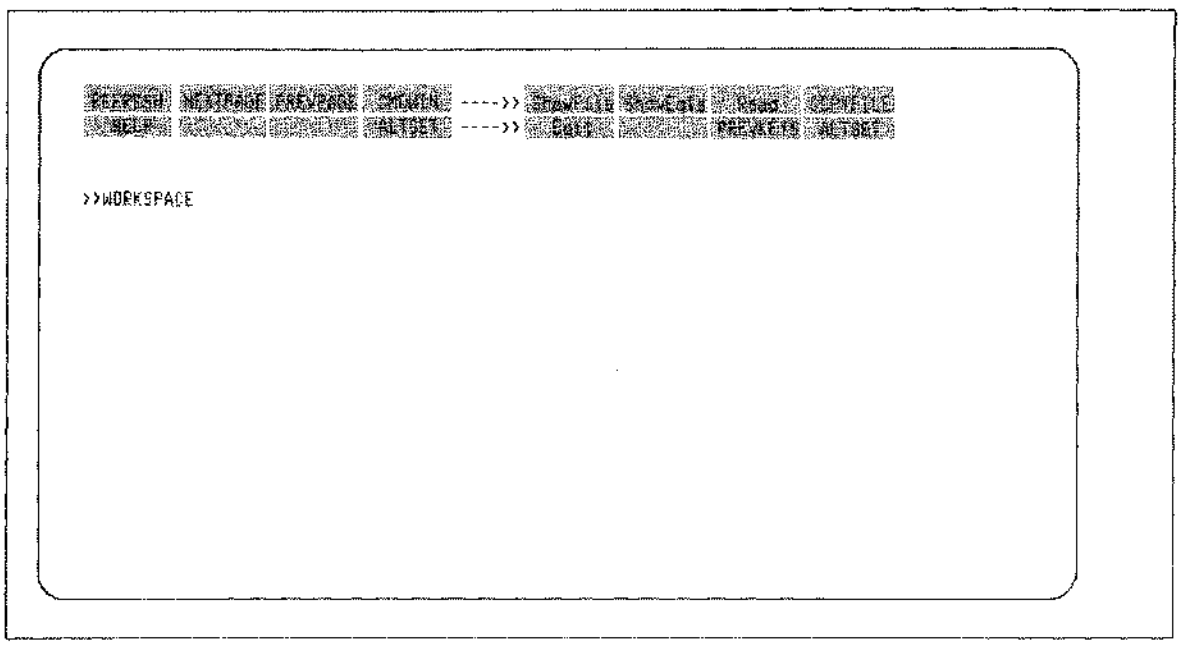


Figure 4-3.

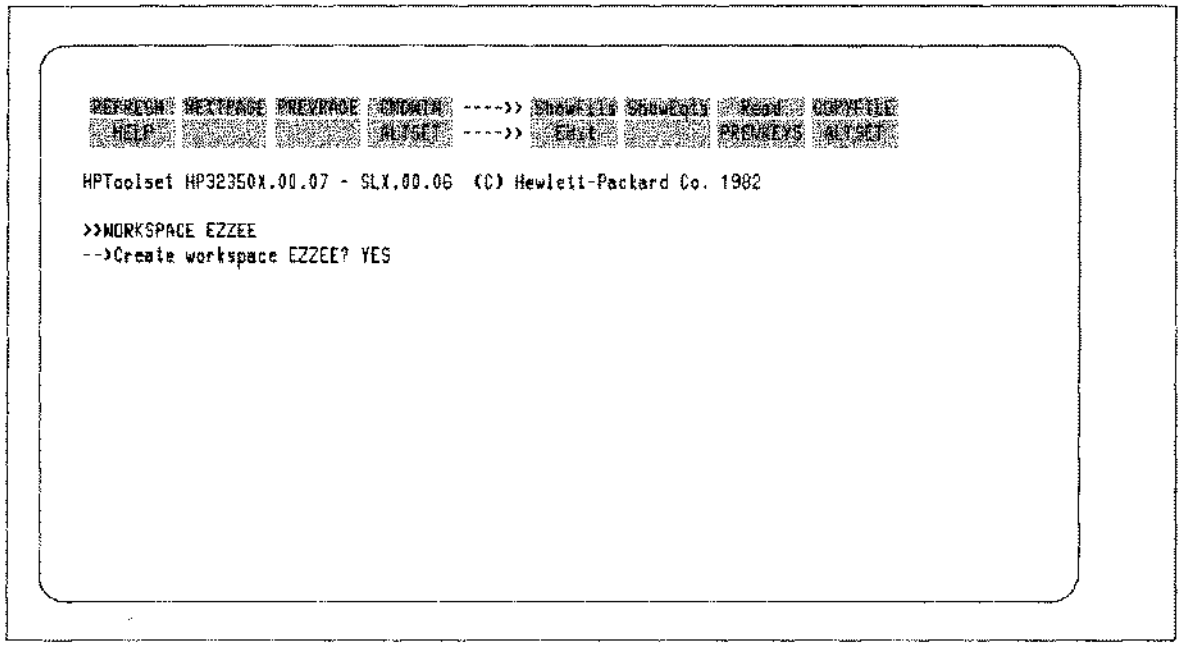


Figure 4-4.

The above dialogue confirms the creation of Workspace EZZEE. A 'Yes' response to the Create prompt displays a SET Menu for the workspace. Any other response cancels the creation of the Workspace and displays the message "***Operation cancelled."

Pressing the PREVKEYS key returns you to the main function key set in Figure 4-3.

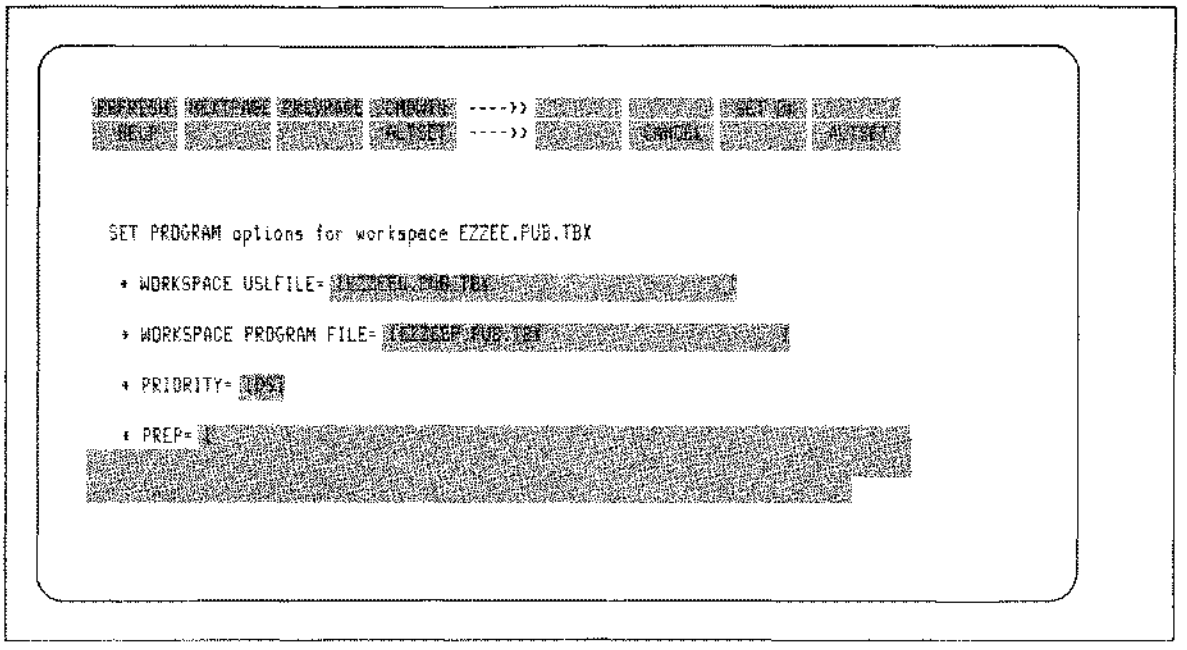


Figure 4-5. SET Options Menu for WORKSPACE

The default options are displayed between brackets. You can enter any changes you wish to make at this time by tabbing to that option and typing in the change(s). When the options are as you want them, press the SET OK function key or the ENTER key on your terminal.

Cancelling Workspace Creation

A CANCEL function key is available with this menu should you change your mind about creating the workspace. If you cancel the creation of the workspace, the message "***Operation Cancelled" is displayed.

```
**********
* CANCEL *
**********
     |
     |
```

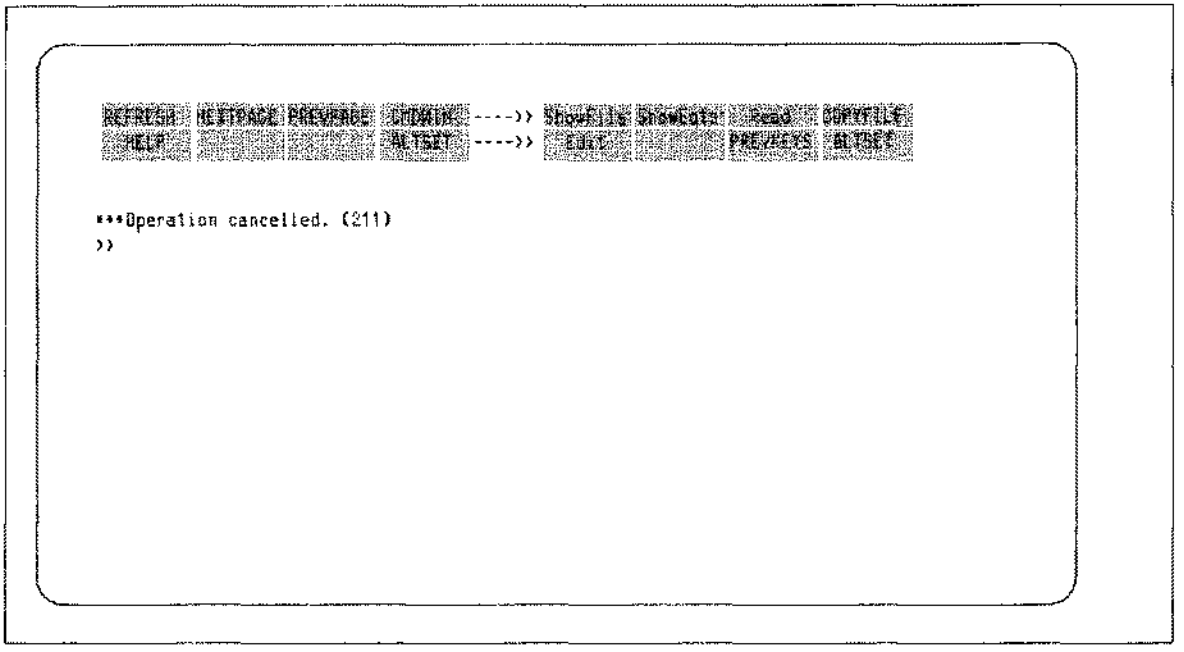


Figure 4-6.

WORKSPACE

Allows you to create or change workspaces.

Syntax

```
-----------------------------------
-                                 -
-  W[ORKSPACE]  [workspacename]   -
-                                 -
-----------------------------------
```

Parameters

workspacename	A filename representing the root file name. This name needs to be fully qualified if it does not reside in your log on group and account.

Usage

The WORKSPACE command allows you to either create a new workspace or move out of one workspace and into another. When you change workspaces, your current workspace is closed including all currently open files. All file equations generated through the USE command are reset, and the information maintained in the Workspace directory is updated. Note that you cannot change workspaces while a program is running. You must end the program run (TOOLSET prompts you) to change your current workspace.

Entering the command with no parameter, or pressing the Wrkspc function key, displays the associated Workspace function keys and TOOLSET prompts you for a workspace name. If only a carriage return is entered at this prompt, the current workspace remains active. If the name you enter is new, you are asked if you wish to create a workspace. You must enter a 'Y' or 'Yes' to continue. A 'No' response or carriage return cancels the operation.

Converting files

HPTOOLSET functions can only be performed on files with a TSAM format. To enable you to use ASCII or KSAM files TOOLSET provides a convert function. Through the use of the CONVERT command, file format can be changed from ASCII or KSAM to TSAM, or from TSAM to ASCII.

CONVERT

Converts files from ASCII or KSAM to TSAM format, or from TSAM to ASCII format.

Syntax

```
-------------------------------------------------------------------
-                                                                 -
-  CON[VERT]  filename1 [version designator]  [|TO|  filename2]   -
-                                                                 -
-------------------------------------------------------------------
```

Parameters

filename1	name of an ASCII, KSAM, or TSAM file to be converted.
version designator	Takes the form of: #nn - where n is an integer #LATEST (#L) #REFERENCE (#R) Allows you to specify a particular version of a TSAM file you are converting. If no version is specified, the LATEST version is assumed if the from file is owned. If the from file is not owned, the REFERENCE version is assumed. This parameter only applies to TSAM formatted files.
TO	Optional word that can be used for syntax clarity.
filename2	The destination file. TOOLSET creates this file if it does not exist. If it does exist, it is purged if you confirm the TOOLSET prompt, and a new file with the same name is created. If no destination file is specified your source file is converted in place with a TOOLSET generated temporary file. The old file is purged upon confirmation from you.

Usage

MPE File Conversion

If the MPE file you are converting has a COBOL format filecode, the TSAM file is created as a COBOL file with a record size of 74 bytes. If the filecode is other than COBOL, TOOLSET checks the last 8 bytes of the record. If the last 8 bytes contain only numbers, they are used as the edit sequence numbers. If the last 8 bytes do not contain just numbers, TOOLSET considers the file unnumbered and assigns sequence numbers.

Before the new TSAM file is built, a SET EDIT menu is displayed to enable you to override any of the default file attributes.

Record Size

The default record size for non-COBOL files is equal to the record size of the old file if it was not numbered and equal to the old record size -8 if the file was numbered. If the record size you specify is smaller than the original record size, the data is truncated, and you are warned with a message.

File Size

The default file size given is 1 1/2 times the number of records in the old file, rounded up to the nearest 100. You should not decrease the file size by more than 1/3 or the file will not be large enough to hold all of the converted records.

TSAM File Conversion

When converting a TSAM file to an ASCII file, you can specify a file equation for the ASCII file to define your own record size, blocking factor, file code, et cetra. The file equation must specify the fully qualified ASCII file name. If no file equation is specified, the new ASCII file is built with a record size equal to the TSAM record size, plus the sequence number length. The file size is then equal to the active records in the TSAM file, and the blocking factor is calculated to maximize the use of disc storage space.

NOTE: You should not use file equations when converting from an ASCII file to a TSAM file. Back referencing files with an asterisk is not permitted.

Example

(1) >>CONVERT SSFILE#4 TO TTFILE

This example converts version 4 of the TSAM file SSFILE to the ASCII file TTFILE.

(2) >>CONVERT THISFILE

This example converts the file THISFILE in place from one format to another so that only one copy exists. You are prompted by TOOLSET to purge the old THISFILE.

FILES

Naming Files

The files in each workspace are recognized by MPE even though they are in TSAM (Toolset Access Method) format and therefore adhere to the same security and access rules. TSAM files, including the Workspace directory file can, therefore, be secured with lockwords.

Filename Definitions

TOOLSET files are formatted and managed in a particular way, but are essentially MPE files,so that all rules that apply to naming MPE file are applicable to TOOLSET files as well. Filename therefore implies

filename [/lockword] [.groupname [.accountname]]

NOTE:

Files created in a workspace reside in the workspace's group and account. This may be different from your log on group and account.

HPTOOLSET Maintenance Commands

Several TOOLSET file maintenance commands are available such as RENAME and PURGE, that parallel the same MPE commands. The Workspace Manager can only keep track of file operations that take place within TOOLSET, and is not cognizant of file changes performed by MPE commands. Therefore, file maintenance must be done with TOOLSET commands for the operation to be recognized by the Workspace.

NOTE: A file is purged if you use the MPE :PURGE command, but it is not deleted from the Workspace directory.

FILE MANAGEMENT

Managing various program files through the TOOLSET Workspace Manager accommodates both the single and multi-programmer development environments. A single programmer typically maintains and updates the most recent copy of his program file until it is ready to be compiled, prepped and run. The single programmer owns all the files that make up his program.

When a project requires multiple programmers, each programmer must have his own files for his particular module (owned) and have access to the files of other project members (shared). The Workspace keeps track of which files are owned by whom, and which versions of various files are shared.

Owned Files

A file is owned by a workspace if it is created by the TOOLSET Editor, copied with the COPYFILE command, or converted from an ASCII or KSAM file while that workspace is active. An owned file is created in the same group and account as the Workspace which owns it (the currently open Workspace). This workspace.group.acct may be different from your log on group and account. File ownership implies read, modify and lock access to the file.

Shared Files

TOOLSET accommodates file sharing between workspaces with the USE command. File sharing implies read and lock, but not modify access. Shared files are typically in their most stable state of development, and are referenced by other users by an owner designated REFERENCE version, unless an access version is specified in the USE command. File versions are discussed later in this section.

USE

Allows you to share a file owned by another workspace and specify which version of a file will be shared.

Syntax

```
(1) US[E] filename1[version designator ]
          [, filename2 [version designator] ]. .
(2) US[E] formal1 |FOR| filename1 [version designator]
          [, formal2 |FOR| filename2 [version designator]]. .
```

Parameters

Format 1 allows you to share a file that is owned, usually but not necessarily, by another workspace. A specific version of that file can be shared if you explicitly indicate a version number. If no version is indicated, a default access version is assigned. The default access version for a shared file is the REFERENCE version even if the file is owned by the workspace that issues the USE command.

filename1	name assigned to the file you wish to share.

version designator	Takes the form: #nn - where n is an integer #LATEST (#L) #REFERENCE (#R) Allows you to specify which version of a file you want to share. If a version designator is not specified, TOOLSET uses the default (REFERENCE) version even if the file is owned by the current workspace.

Format 2 is an extension of format 1 and allows you to specify a formal designator. It generates a file equation between formal and filename so that you can refer to the shared file by a different name.

formal	The formal file designator. It may be up to 8 alphanumeric characters long. Note that formal designators cannot be qualified with group and account, nor can you specify a version designator.
filename	Actual file designator for the file equation.
FOR	Optional word used for syntax clarification.
version designator	See Format 1 above.

Usage

The USE command can be used by both owners and users of files, and allows you to both specify the sharing of a specific version of a file and reference it by another name. TOOLSET automatically generates the file equations implied by format 2 every time you enter the workspace it is associated with. Therefore, the USE command need only be issued once per file.

A file version that is specified in the USE command can be overridden by explicitly specifying another version number in a different TOOLSET operation. For example, if you have entered the following USE command,

```
>>USE ZFILE#2
```

that references version 2 of ZFILE, you can change the version you want to reference by specifying a new version,

```
>>CONVERT ZFILE#4 TO NEWFILE
```

When the CONVERT command is entered above, it is version 4 of ZFILE rather than version 2 that is converted to NEWFILE, overriding the specified version in the USE command above.

Example

```
(1)   >>USE JFILE FOR KFILE.group
             |          |
             |          |
             |          |
        Workspace A   Workspace B
```

This example allows workspace A to share the file KFILE owned by Workspace B. Because Workspace B resides in a different group than Workspace A, KFILE must be qualified with its group. Since no version is specified for KFILE, Workspace A uses the default REFERENCE version.

TOOLSET sets up a file equation equating JFILE with KFILE.group JFILE is the formal file designator, and KFILE.group is the actual file designator.

```
(2)   >>USE GFILE#5
```

This example specifies that version 5 of GFILE is to be used by default when GFILE is opened. This command can be entered by either the owner of GFILE or a user sharing GFILE.

DISCARD

Negates the USE command.

Syntax

```
-------------------------------------------------------
-                                                     -
-  DISC[ARD]  filename1 [version designator]           -
-             [, filename2 [version designator] . . .   -
-                                                     -
-------------------------------------------------------
```

Parameters

filename	name of file or formal designator specified in the USE command.
version designator	Takes the form: #nn - where n is an integer #LATEST (#L) #REFERENCE (#R) Particular version specified for filename. If not specified, the first USE specified for that file is cancelled.

Usage

The DISCARD command disassociates the link set up to the file in the USE command. Any existing file equations generated by TOOLSET are deleted.

Example

```
>>DISCARD JFORMAL, GFILE#3
```

This example negates the file equation set up for JFORMAL and deletes the USE entries for both JFORMAL and version 3 of GFILE from the workspace directory.

RENAME

Changes the sytem file identification of a file.

Syntax

```
-------------------------------------
-                                   -
- RENA[ME] filename1 |,| filename2  -
-                                   -
-------------------------------------
```

Parameters

filename1	original name of file
filename2	new name of file

Usage

The RENAME command allows the file owner to change the system file identification by removing the file and filename1 and creating another file with identical contents. It is then known to the system and the workspace as 'filename2'. RENAME can only be used within the owner workspace.

Note that the USE command allows you to refer to a file by a different name, but does not change the identity of the file itself as does the RENAME command.

Example

```
>>RENAME OLDFILE, NEWFILE
```

This example renames 'OLDFILE' to 'NEWFILE'

VERSION MANAGEMENT

Whether you operate in a single or multiple programming environment, programs undergo multiple changes. TOOLSET recognizes and manages each changed text file as a 'version'. As you make changes to your text file in the way of additions, deletions and updates, these changes are kept as versions of the same file.

TOOLSET allows you to control the amount of changes to the text file made at one time and freeze a version with the SETVERSION command. This version is a logical copy of the text file that can be read, but can no longer be modified.

Versions are referred to by number and the most recent copy of a file is known as the LATEST version. When a file is created, it is the equivalent of version 1. The first SETVERSION command adds version 2 and subsequent SETVERSION commands add versions 3, 4 and so on. Each version logically includes records from previous versions.

Latest Version

The LATEST version of a file is the working or most recent version of a text file, and is the only version that may be modified. Modifications to the LATEST version can only be made by the owner workspace. Other users cannot read the LATEST version while it is being modified, nor can the LATEST version be modified while it is being read. The LATEST version can be simultanously read by multiple users if it is not being modified.

Each text file can have up to 32 Active versions. An active version is a version that has not been purged.

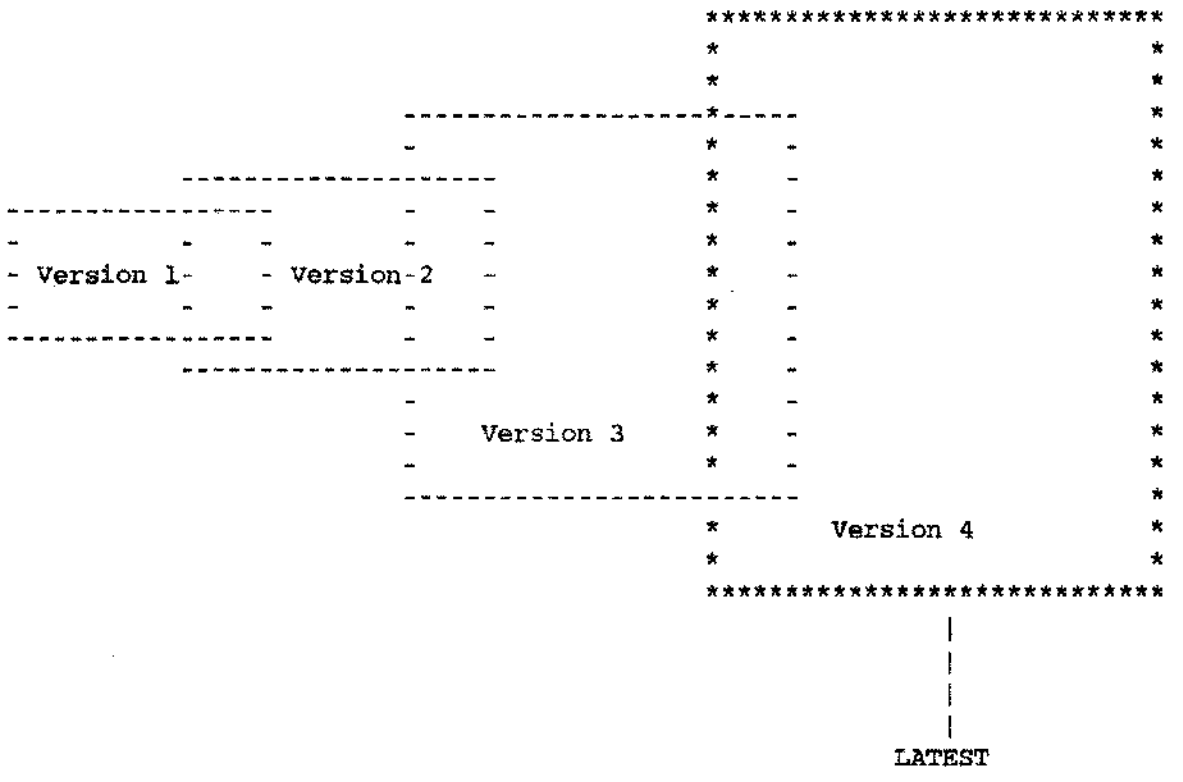


Figure 4-7. File ZIP, Versions 1-4.

Version Access

Versions are accessed according to the access pointer setting. The owner of a file that is shared can set the access pointer with the SETREF command to establish the version of the file that is referenced by other users. The version that is set with the SETREF command is called the REFERENCE version, and generally it is the most stable version. If no version has been set by the file owner, other users access the LATEST version by default. Only the LATEST version of a file can be modified.

If you own a file, you always access the LATEST version of the file unless you issue a USE or explicitly request a particular version. This allows you to end the editing on your file, perform other TOOLSET operations or even exit the TOOLSET program, and re-access the file at a later time to continue editing. While the LATEST version is being edited, it is locked and no other users can access this version of the file at this time. To enable other users to read the LATEST version, including the compiler, you must end the edit operation with the END EDIT command or function key. By default, you compile the LATEST versions of the files you own, and the most stable (REFERENCE) versions of the files you do not own or have issued a USE on.

A USE overrides an OWN. If you issue a USE command for a file you also own, you access the REFERENCE version of that file by default until a DISCARD is issued to negate the USE.

NOTE: If an abnormal termination occurs while you are accessing the Latest version of a file, this file remains locked, and must be recovered before it can be accessed. See the RECOVER command in this section of the manual.

A LIST CHANGE command is available that provides a listing of the changes from one version of a file to a higher version.

SETVERSION

Freezes changes to a file and increments the latest version number.

Syntax

```
-----------------------------
-                           -
- SETV[ERSION]  [filename]  -
-                           -
-----------------------------
```

Parameters

filename	Name of the file you want to assign a version number to. If filename is not specified, TOOLSET assigns a version number to the file you are currently editing.

Usage

The SETVERSION command freezes a version so that no further changes can be made and associates a new version number with the file. This command is valid only for files that are owned by the current workspace, and requires exclusive access to the file. Other users can not access the file while the command is executing.

New files are automatically assigned version 1, and further versions are assigned in ascending sequence starting with version

2. When a new version is assigned to a file, the contents of the previous version are frozen. You can have up to 32 active (non-purged) versions of a file at one time.

Each version logically includes active records from previous versions. For example, if Version 1 has 10 records and 3 records are added to Version 2, Version 2 logically contains 13 records.

Notice that once a version of a file is frozen, no more changes can be made to that particular version. Only the LATEST version can be modified.

Example

```
1  This              /
2  is               /
3  version         /  version 1
4  1               \
5  of               \
6  the               \
7  file
8  YOURFILE

>>SETVERSION YOURFILE
```

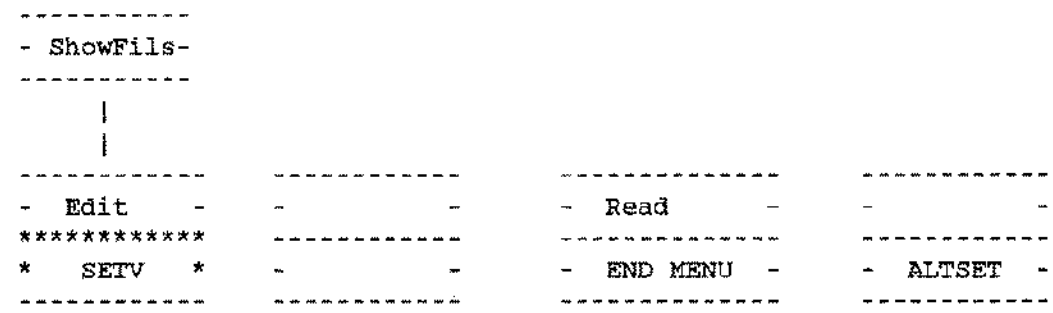
This command creates version 2 of YOURFILE. Version 1 above is now frozen and no further changes can be made to it. Only the Latest version, version 2, can be modified.

```
 1  This
 2  is
 3  version
 4  2              ----------record 4 modified
 5  of             ----------record #6 deleted
 7  file
 8  YOURFILE          /
 9  Notice           /
10  that            /
11  changes        /   version 2
12  can            \
13  be              \
14  made             \
15  to                \
16  it
17  because
18  it             ------------------records 9 through 22 have
19  is                                 been added
20  the
21  LATEST
22  version
```

```
>>SETVERSION YOURFILE

 Version 2 of YOURFILE has now been set, incrementing the Latest
  version to version 3.  It contains records 1 through 5 and 7 through
  22 since record #6 was deleted in version 2 above.  You can only
  make changes to version 3.
```

```
 1  This
 2  is
 3  version
 4  2
 5  of
 7  file
 8  YOURFILE          /
 9  Notice           /
10  that            /
11  changes        /   version 3  -  LATEST version of YOURFILE
12  can            \
13  be              \
14  made             \
15  to                \
16  it
17  because
18  it
19  is
20  the
21  LATEST
22  version
```

SETVERSION Function Key



The SETV function key is accessed by pressing the ShowFils key. To use this key,

(1) Position the cursor at any of the TSAM files in the SHOW FILES menu, and type a character in the box next to the file you choose.

(2) Press the SETV key

The current version of the file you indicated is now frozen and no further changes can be made to that version. A new LATEST version is assigned to the file. If you press the SETV key without marking any files in the SHOW FILES menu, the file currently being edited is acted upon.

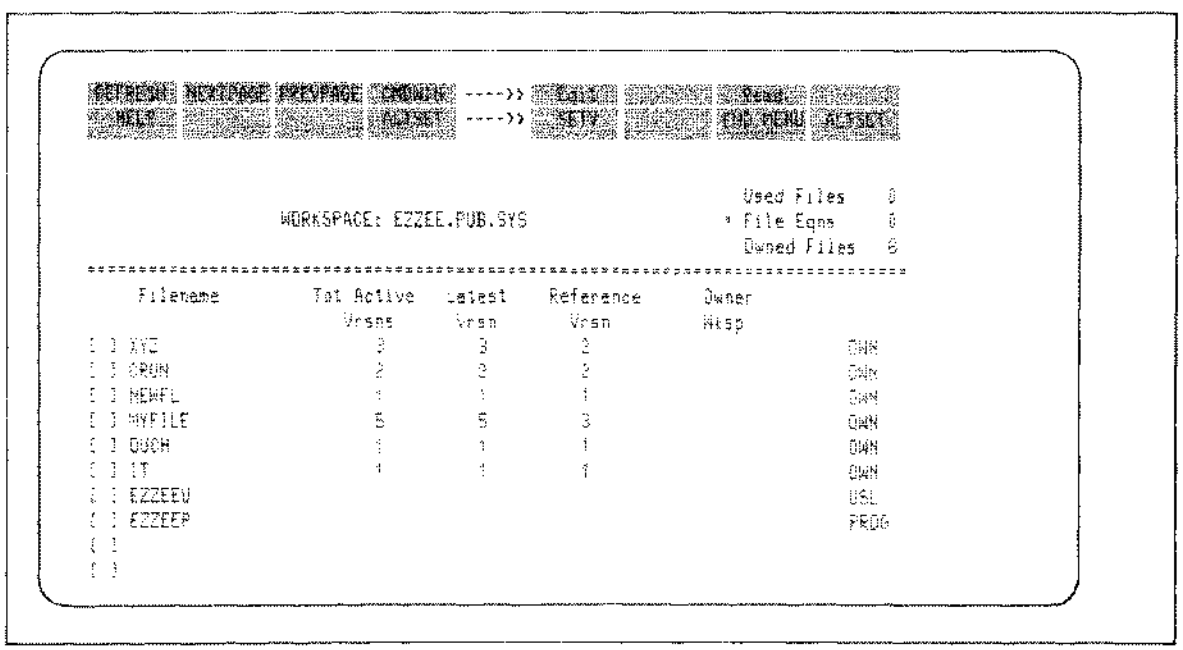


Figure 4-8. Show Files Menu

Example

To freeze the current version of COMPUTE, enter a character in the brackets next to the filename, and press the SETV key. The LATEST version will then be version #3.

SETREF

Designates the REFERENCE version of a file that becomes the default access version by users other than the file owner.

Syntax

```
-------------------------------------------------------
-                                                     -
-  SETR[EF]  filename1 [version designator1]          -
-            [, filename2 [version designator2]]. .   -
-                                                     -
-------------------------------------------------------
```

Parameters

filename	name of the file you wish to set a REFERENCE version for
version designator	Takes the form: #nn - where n is an integer #LATEST (#L) #REFERENCE (#R) Specifies a particular version of filename. If not specified, the LATEST version is referenced.

Usage

The SETREF command is used to set the access pointer to a REFERENCE version that becomes the default access version for users of the file. The reference version for users is, by default, the LATEST version until a SETREF command is performed. Only the owner of the file can set a reference version on his file for other users, and setting the version requires exclusive access to the file. Other users cannot access the file while this command is executing.

If no version designator is specified in this command, TOOLSET uses the LATEST version. You would typically use the SETREF command without the version designator parameter to reset the access pointer from a previously set value to the LATEST. Note that if you follow the SETREF command without the version designator parameter by a SETVERSION command, the Reference version does not move to the new Latest version. (See the example below).

Example

The current Latest version of AFILE is version #5.
If the following commands are performed on AFILE, the Reference version is no longer the Latest version.

(1) >>SETREF AFILE#3

This example sets the Reference version to #3. Users of AFILE reference version #3. The owner of AFILE references the LATEST version, version #5, by default.

(2) >>SETREF AFILE

This example sets the Reference version equal to the Latest, version #5. Users and the file owner both reference version #5 by default.

(3) >>SETVERSION AFILE

This command freezes version #5 so that no further changes can be made to #5 and sets the Latest version equal to #6. The Refernce version has not changed. It is still version #5.

(4) >>PURGE AFILE#R

This command purges the current Reference version, #5. This sets the Reference version equal to the Latest version which is #6 until the next SETREF command is performed.

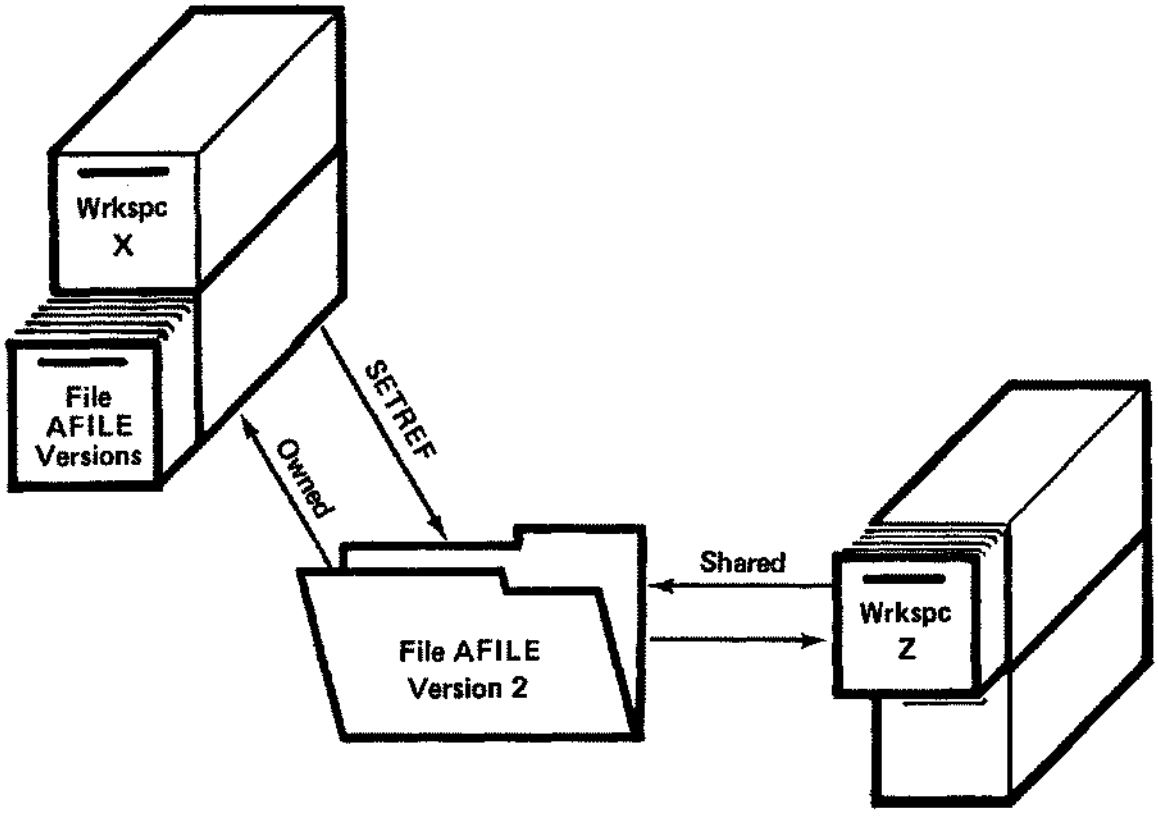


Figure 4-9. Referencing File Versions.

LIST CHANGE

Lists changes from the previous version to the specified version, or lists modifications that were made to a specific version while it was the Latest version.

Syntax

```
LIST CHANGE filename[version designator]  [|TO| LP ]
                    [version range     ]
                    [all versions      ]
```

Parameters

filename	name of file whose version changes you wish to list
version designator	Takes the form: #nn - n is an integer #LATEST (#L) #REFERENCE (#R) Specifies a particular version for filename. Changes made while this version was the LATEST version are listed. If no version is specified, changes made in the LATEST version for the owner, and REFERENCE version for other users are listed.

version range	Takes the form: #/# #/# Lists all changes for the active versions between the two designators. Changes are listed separately for each version in the range.
all versions	Takes the form: #@ Lists changes for all versions.
TO	Optional word used for syntax clarity.
LP	Generates a line printer listing of the version changes. The output is sent to TOOLLIST instead of your terminal ($STDLIST).

Usage

The LIST CHANGE command provides a list of changes from one active version of a file to the next higher active version. If a range of versions is specified, changes are listed for each version separately. If neither a version designator or range is specified, changes are listed for the LATEST version if you own the file, or the REFERENCE version if you are sharing the file. Records that have been deleted in a version are listed and designated with a "D", and records that have been added are designated with an "A".

Example

(1) >>LIST CHANGE AFILE#@

This example provides a list of changes for all active versions of AFILE.

```
-->Version #1

A       1       one
A       2       two
A       3       three
A       4       four
A       5       five
A       6       six
A       7       seven
A       8       eight
A       9       nine
A      10       ten

-->Version #2

        2       2
A       4.01    4.5
        5       5
D       7       seven
        9       nine
D      10       ten
A      11       eleven
A      12       twelve
>>
```

(2) >>LIST CHANGE AFILE#3/#LATEST

This example provides a separate list of changes for each active version of AFILE in the range #3 through the LATEST version. If the LATEST version is version 5, the lists will include changes to 3, 4, and 5.

(3) >>PURGE AFILE #2
>>LIST CHANGE AFILE #1/#6

This example provides a separate list of changes for each active version of AFILE in the range #1 through #6. Changes are listed for versions 1, 3, 4, 5, and 6. Version #2 was purged, and will not be listed, but the changes made in #2 are included in the listing for version #3.

LABEL

Allows you to put a comment on a file version.

Syntax

```
-------------------------------------------------------------------
-                                                                 -
-                      [version designator]                       -
-  LA[BEL]  filename   [version range     ]   "comment string"    -
-                      [all versions      ]                       -
-                                                                 -
-------------------------------------------------------------------
```

Parameters

filename	Name of file the label is added to. It must be an owned file.
version designator	Takes the form: #nn - where n is an integer #LATEST (#L) #REFERENCE (#R) Specifies a particular version you wish to label. If no version is specified, the LATEST version is labeled.

version range	Takes the form: #/# All active versions between designators are labeled.
All versions	Takes the form: #@ Specifies that all active versions for the specified file be labeled.
"comment string"	The label or comment you wish to put on a version or versions of the specified file. The string must be enclosed in quotes and can be up to 80 characters (bytes) long.

Usage

The LABEL command allows you to add a comment string to the file user label of a specific version, version range, or all versions of a file. The file must be owned by the current workspace. If no version is specified, the comments are added to the LATEST version of the file. If a specifed version already contains a label, the new label replaces the old one.

Example

```
>>LABEL AFILE#R   "Reference Version 2/24/82"
```

This example adds the date comment to the Reference version of AFILE.

SHOW LABEL

Displays any comments associated with a version of a file.

Syntax

```
-------------------------------------------------
-                                               -
-  SHO[W] L[ABEL] filename [version designator]  -
-                          [version range     ]  -
-                          [all versions      ]  -
-                                               -
-------------------------------------------------
```

Parameters

version designator	Takes the form: #nn - where n is an integer #LATEST (#L) #REFERENCE (#R) Specifies the file version whose label you wish to see.
version range	Takes the form: #/# The labels for the active versions between designators are displayed.
All versions	Takes the form: #@ Displays labels for all versions of the specified file that contain labels.

Usage

This command allows you to see what comments, if any, are associated with versions of a specified file. If no version is specified for filename, the label for the LATEST version is displayed if you own the file, and the label for the REFERENCE version is displayed if you do not own the file or if a USE has been issued for that file.

Example

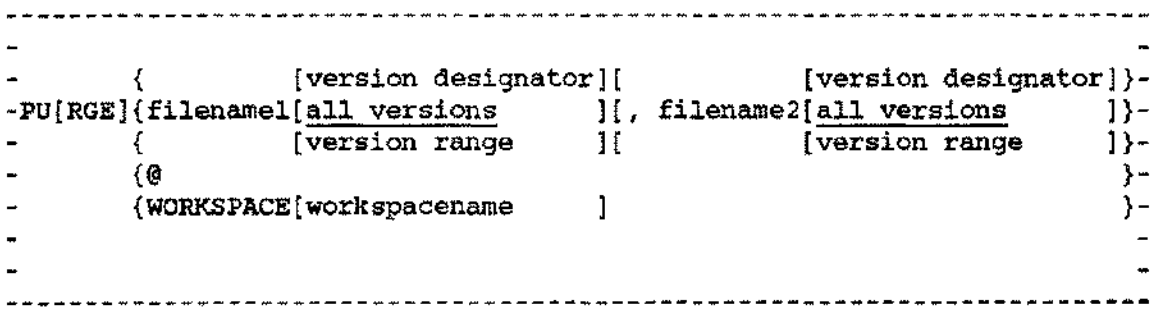
```
>>SHOW LABEL XFILE#3/#R
```

This example displays the comments for all the active versions between version 3 and the REFERENCE version of XFILE.

PURGE

Purges one or more files, or versions of a file, owned by the current workspace.

Syntax



Parameters

filename	Name of the file you wish to purge. If there is more than one file, they should be separated with a comma.
version designator	Takes the form: #nn - where n is an integer #LATEST (#L) #REFERENCE (#R) Specifies a particular version of filename you wish to purge

All versions	Takes the form: #@ The entire file is purged. This is the default if specific versions are not designated.
version range	Takes the form: #/# Specifies that all active versions between, and including, the two designators be purged.
@	Purges all owned source files in the current workspace.
WORKSPACE	Specifies that an entire workspace is to be purged. All versions of all owned source files, all listing files, the USL file, Program file, and Workspace directory of the current or specified workspace are purged. You are prompted to confirm the purge if this parameter is used. Note that you must enter the entire word 'workspace'.
workspacename	Allows you to specify the particular workspace you wish to purge. If a workspacename is not specified, the current workspace is purged. Do not specify this parameter if the workspace you are purging is your current workspace.

Usage

The PURGE command allows you to purge one or more files, or versions of files, from the currently active or specified workspace and account. If any active and open workspace is accessing the file you wish to purge, you must wait until it closes the file.

When a version is purged, that version is unavailable for any future referencing. If you reference a purged version of a file, you receive an error message. Later versions of a file are unaffected by the purging of earlier versions, so that cumulative changes remain. If the LATEST version is purged, the records added in this version are deleted since no future versions are dependent on them. Records that were changed are reset to the state previous to the change. The LATEST version number remains the same, however, to prevent any previous versions from being unfrozen.

Note that the number of active (non-purged) versions in a file may be different from the highest version number since some of the lower version may be purged. For example, if there were originally 10 versions of a file, and versions 2, 4, and 5 have been purged, there are 7 active versions, though the highest version is #10.

Example

(1) >>PURGE MYFILE#3

This example purges version 3 of MYFILE

(2) >>PURGE MYFILE#@

This example purges all versions of MYFILE and deletes it from the Workspace directory. It is the same as not specifying any version.

(3) >>PURGE @
-->Purge All owned files? YES

This example purges all the owned files in the current workspace except the USL file and the program file. Notice you are prompted to verify the purge before the files are actually deleted.

(4) >>PURGE WORKSPACE WW
--> PURGE WORKSPACE WW?

In this example all owned files, the USL, and program file for Workspace WW are purged. Any file equations for this Workspace are deleted as is any directory information.

FILE SYSTEM UTILITY COMMANDS

COPYFILE

Copies a named TSAM file to a new file in the current workspace.

Syntax

```
-------------------------------------------------------------------------
-                                                                       -
-                        [version designator]                           -
-COPYF[ILE]  [filenamel [all versions      ] |TO| [filename2 [NEWVERSION]]-
-                        [version range    ]                            -
-                                                                       -
-------------------------------------------------------------------------
```

Parameters

filenamel	Name of the file to be copied.
version designator	Takes the form: #nn - where n is an integer #LATEST (#L) #REFERENCE (#R) Indicates a particular version of filenamel you wish to copy. The default is that all versions of filenamel are copied.

all versions	Takes the form: #@ Copies all active versions in the file. Note that unless versions have been purged from the old file, it is significantly faster to COPYFILE all versions than it is to copy only a range of versions.
version range	Allows you to copy several versions of filename1. Specified as #/# and includes all active versions between and including the two designators.
filename2	Name of file you want to copy to. Filename2 must be a new file, so if it already exists, it is purged and recreated following a confirmatory response from you.
TO	Optional word used for syntax clarity.
NEWVERSION	Optional parameter that renumbers the active versions in filename2 sequentially starting with version 1.

Usage

The COPYFILE command copies a named file to a new file in the current workspace. The 'current' workspace is the workspace TOOLSET associates you with when you issue the workspace command. You can either copy specific versions of a file, or copy the entire file. If the complete file is copied, identical files exist at both the source and destination locations when the copy operation is completed.

The COPYFILE command is a way of establishing ownership of a file that belongs to another workspace. COPYFILE is a function that 'creates' a file in your workspace, thus allowing you to 'own' that file. Any file that is copied must be in TSAM (TOOLSET Access Method) format. Note that if you wish to establish ownership of an ASCII formatted file, you must use the CONVERT command. (See this section of the manual)

The COPYFILE command is also useful for cleaning up files that have had several versions purged. Although versions have been purged, the records that were modified or deleted in those versions still remain in the file. When a COPYFILE command is performed to a new file, all records that are not relevant to any active version are eliminated reducing the amount of disc space required for the file.

MPE rules regarding file name and access must be regarded, that is two files of the same name cannot co-exist in the same group and account. You must have at least read access and supply any necessary lockwords to the file being copied.

Example

```
>>COPYFILE OLDFILE#5 TO NEWFILE   NEWVERSION
```

This example copies the contents of version 5 of OLDFILE to a new file named NEWFILE. The new file has one active version, and because the NEWVERSION parameter is specified, that version is version #1.

STORE

Stores the entire workspace onto tape.

Syntax

```
-----------------------------
-                           -
-  STORE  [workspacename]   -
-                           -
-----------------------------
```

Parameters

workspacename	The name of the workspace to be stored onto tape. If no name is specified, the current active workspace is stored.

Usage

The STORE command stores all owned files, the USL file, Program file, listing files, and directory information for the specified or currently active workspace in the normal STORE format. If you are storing the current workspace, all ongoing activities are terminated before the STORE is performed.

Example

```
>>STORE WRKSPCX
```

This example stores all files associated with WRKSPCX onto tape.

RESTORE

Restores all files from tape to the specified workspace.

Syntax

```
-----------------------------------
-                                 -
-  RESTORE  workspacename  [KEEP] -
-                                 -
-----------------------------------
```

Parameters

workspacename	Name of the workspace to be restored.
KEEP	Prevents files which already exist on the system from being restored. Note that the Workspace directory file is always restored, whether or not KEEP is specified.

Usage

This command restores all owned files, the USL file, Program file, listing files, and Directory file information for the workspace you specify. Files must be restored into the group and account in which the workspace was created.

If you specify the KEEP parameter, those workspace files that are also contained on the system are not restored, with the exception of the directory file information. This information is always restored. If KEEP is not specified, all files are restored, and any files for the specified workspace that currently exist on the system are purged.

Restoring requires you to answer two tape mount requests at the system console. The first request restores the workspace directory and the second request restores the workspace owned files, the USL file, Program file, and listing files.

Example

```
>>RESTORE WXYZ
```

This example restores all stored files for the Workspace WXYZ, purging any of WXYZ's files that currently exist on the system.

SHOW FILES

Displays a list of all owned and used files for a particular workspace.

Syntax

```
-------------------------------------
-                                   -
-  SHO[W] F[ILES] [workspacename]   -
-                                   -
-------------------------------------
```

Parameters

workspacename	Identifies the workspace for which the owned and used files are displayed. If no workspace is specified, the display is for the current workspace.

Usage

The SHOW FILES command produces a display of all the owned and used files in a defined workspace. Entering the command is the same as pressing the ShowFils key. If your workspace contains more files than one screen can contain, use the PREV PAGE and NEXT PAGE permanent function keys to view all the files.

ShowFils Function Key

The ShowFils (Show Files) function key is available from the Main, Workspace and Show Equates key sets and when pressed provides a menu display of the owned and used files for the current workspace.

- Edit -	- Wrkspc -	- Read -	* ShowFils *
- Program -	- -	- EXIT -	- ALTSET -

MAIN

* ShowFils *	- ShowEqts -	- Read -	- COPYFILE -
- -	- -	- PREVKEYS -	- ALTSET -

WORKSPACE

* ShowFils *	- -	- -	- -
- -	- -	- END MENU -	- ALTSET -

SHOW EQUATES

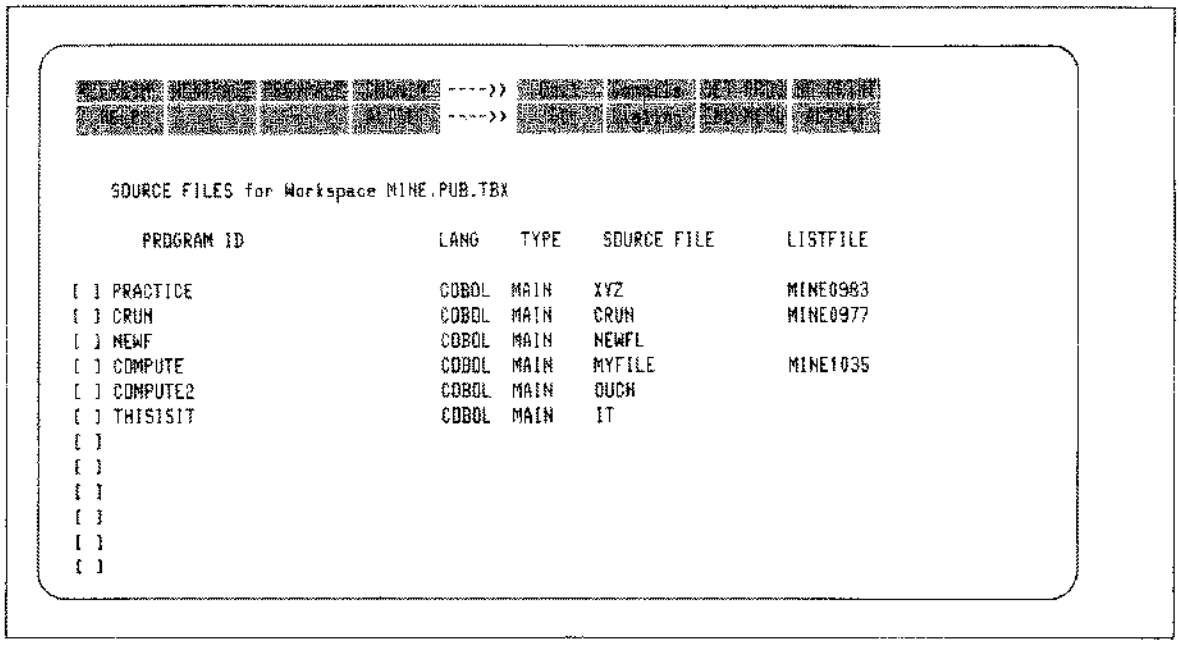


Figure 4-10. Show Files Menu

(1) Workspace	The name of the workspace for which files are being listed.
(2) Name of owner workspace	The name of the workspace that owns the used files.
(3) Number of Active Versions	The number of versions not purged in the file.
(4) LATEST version	The version number assigned with the last SETVERSION command.
(5) REFERENCE version	The file version that users other than the file owner use by default.
(6) OWN, USE, USL, PROG	Type of file. The USL and Program file names are defined when your workspace is created. The default is to append a U or P to the workspace name as in Figure 4-9.
(7) *	Designates that there is a file equation associated with the USE file.

Pressing the ShowFils key or entering the SHOW FILES command produces a secondary function key set containing the Edit, Read, and SETV (SETVERSION) operations. Any of these operations can be performed by typing a character in the box next to any file and pressing the appropriate function key.

SHOW EQUATES

Displays the file equations that have been set up by the USE command.

Syntax

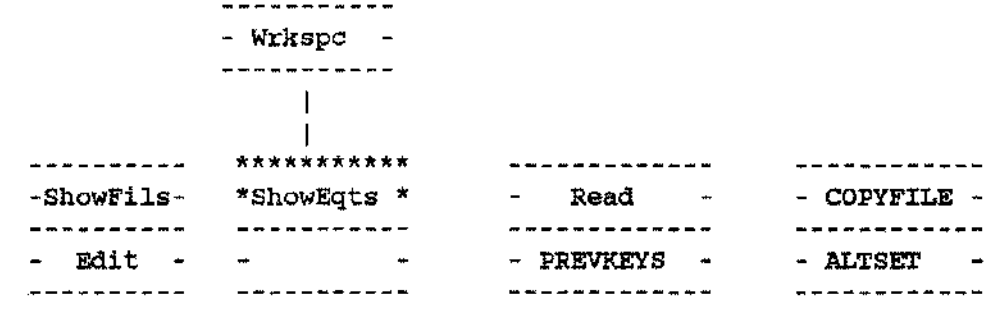
```
--------------------
-                  -
-  SHOW E[QUATES]  -
-                  -
--------------------
```

Usage

This command displays the file equations that have been set up by the USE commands that are in effect for your current workspace. There is one file equation for each USE command issued that includes a formal designator. Entering the command is the same as pressing the ShowEqts function key.

Show Equates (ShowEqts) Function Key

The ShowEqts (SHOW EQUATES) key is available from the Workspace key set, and when pressed displays all of the file equations that have been set up by the USE commands that are in effect for the current workspace.



Example

```
>>USE FNAME FOR XYZ
```

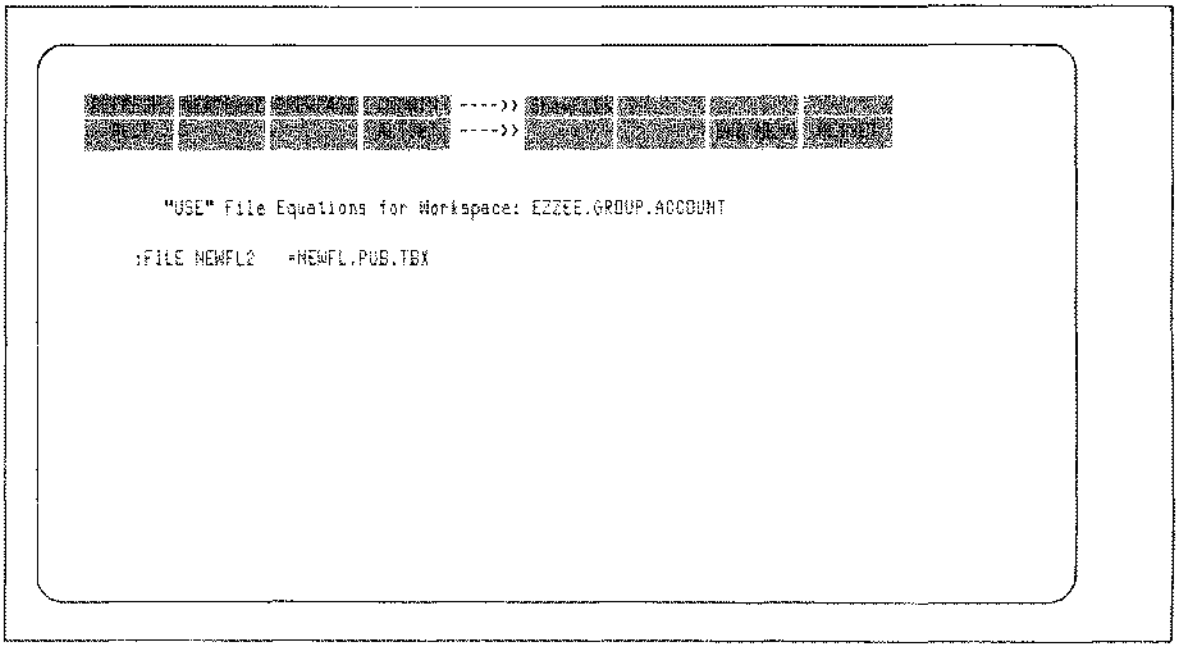


Figure 4-11. SHOW EQUATES Display

System Failure and Recovery

In the event of a 'soft crash' where the disc is not physically damaged, any changes made to the file in most cases remain intact. If an abnormal termination should occur, such as a compile abort, while you are accessing the LATEST version of your source file, this version remains locked. You must recover the file in order to unlock it and access it for editing. Refer to the RECOVER command in this section of the manual.

If your system crashes while you are modifying a file, your file is automatically recovered the next time you access that file in HPTOOLSET. Note that you may lose data that was in memory when the crash occurred. To check on data that may have been lost, list the contents of your file.

RECOVER

Recovers all versions of a file.

Syntax

```
-------------------------------------
-                                   -
- REC[OVER]  [filename [,U[NLOCK]]]  -
-                                   -
-------------------------------------
```

Parameters

filename	Name of the file you wish to recover. If this parameter is not specified, the current Edit file is recovered.
UNLOCK	Option you may use for a quick recovery if you were not making modifications to your file when the system crashed or an abnormal termination occurred.
	NOTE: If you use this option and your file is still inconsistent, do a full recovery.

Usage

The RECOVER command allows you to recover your entire file in the event of an abnormal termination of TOOLSET where file recovery is not automatic. This command requires that you have exclusive access to the file being recovered, so that no other users can read or modify the file during recovery.

During recovery, TOOLSET creates a permanent 'scratch' file with the name TBRxxxxx where 'x' is a digit. TOOLSET renames this scratch file to its filename previous to the file loss. If an error occurs during this renaming process, this file (TBRxxxx) will be in your current workspace group and account, and may be used in place of the file to be recovered.

There are some conditions under which a file may not be recoverable, such as an unreadable TSAM file directory. If this occurs, you will be advised to reload an old copy of the file.

Example

```
>>RECOVER  Myfile
```

Section 5

EDITOR

The HPTOOLSET Editor is a screen oriented editing tool that enables you to create and modify source files using the editing functions of TOOLSET supported terminals. Editing functions can be performed by typing TOOLSET commands while in command mode or by positioning the cursor with the terminal control keys in visual mode , and pressing one of the Edit function keys. To edit a file, the file must be in TSAM (Toolset Access Method) format, and exist in the current workspace. Only one file can be open for editing at a time.

Since edit files are 'owned' by the current workspace, all edit files within a workspace belong to the same group and account as the workspace. The workspace group and account may be different from your log-on group and account.

To open a file for editing, either type the EDIT command following the TOOLSET prompt (>>), or press the EDIT function key. If you do specify the filename, you are prompted for one. If the file does not already exist, you will be asked if you wish to create one. For new files, and for files being CONVERTed to TSAM format, a SET Options menu is displayed from which the file's dimensions such as LINELENGTH can be defined.

Edit files are closed when either the END EDIT, END ALL, EXIT, WORKSPACE, or another EDIT command is entered or their corresponding function keys are pressed. You can also enter the END command without parameters, in which case you are prompted before an END EDIT is performed and your file closed. For details on the END and EXIT commands, see Section II of this manual under 'ALL Purpose Commands'.

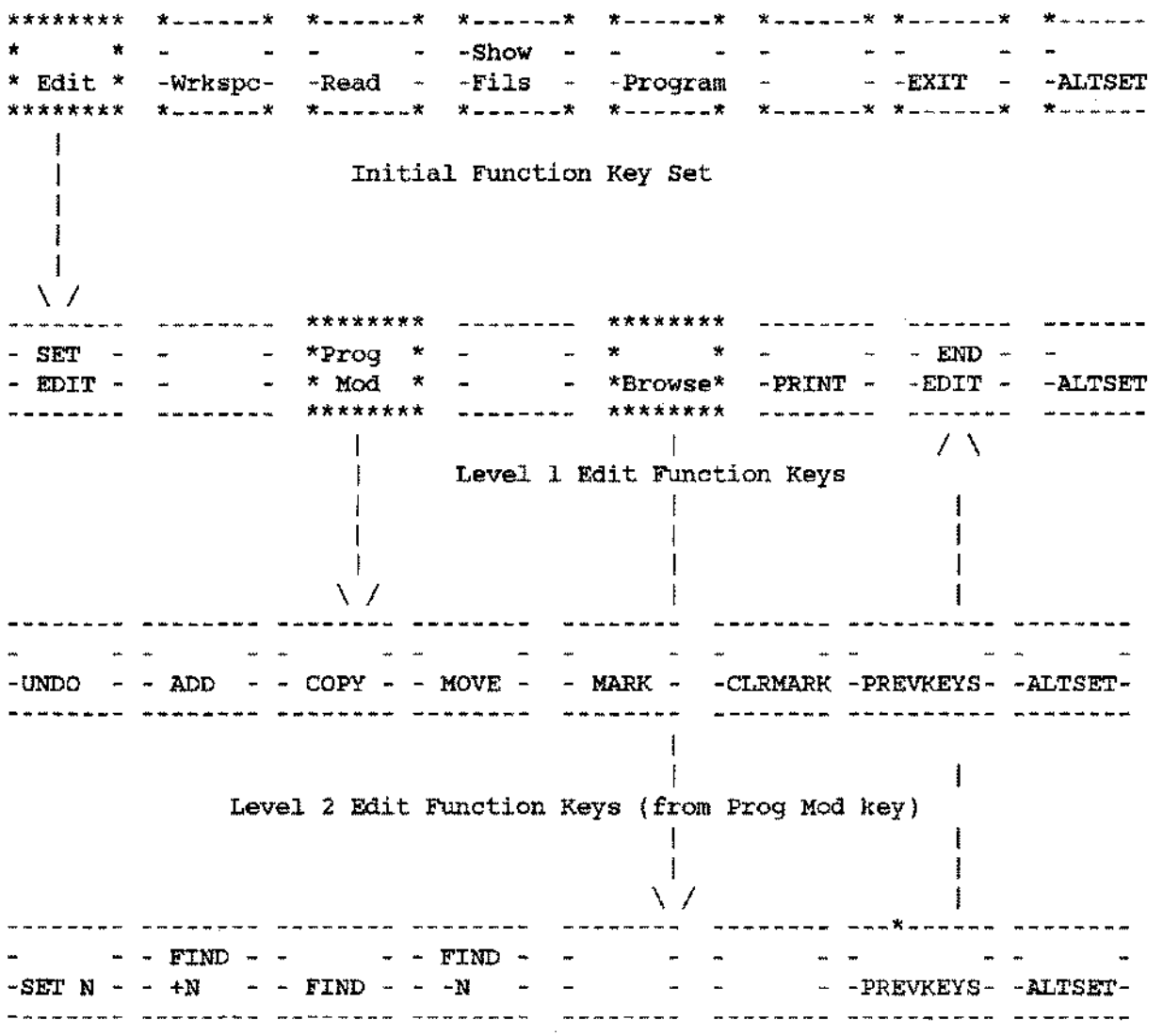


Figure 5-1. Edit Function Key Tree

EDIT

Opens a file for editing.

To open a file for editing, press the EDIT function key, or enter the EDIT command if in command mode (>>). The level 1 edit function keys are displayed and you are in Visual Mode.

Syntax

```
-------------------------------------
-                                   -
-  ED[IT] [filename  [linepos] ]    -
-                                   -
-------------------------------------
```

Parameters

filename	Name of the file you wish to edit. If no file name is indicated, TOOLSET redisplays the currently open edit file. If no file is currently open for editing, TOOLSET opens the file associated with the current context or prompts you for a name.
linepos	The number of the line at which you want your edit file displayed. If no line position is specified, the beginning of your file is displayed. If the file is already open for editing, it is redisplayed from the point it was last viewed.

Usage

Only one file may be open for editing at one time, and you may only edit files that belong to your current workspace. An edit file can be: (1) a file named in a command or marked in the SHOW FILES or PROGRAM menu, (2) the file associated with the line marked or the top line of your screen when a listing is currently displayed or (3) the file associated with the current breakpoint when running a program with Symbolic Debug.

Note that the default group and account of your file is the same as the group and account of the current workspace; this may be different from your log-on group and account.

You are automatically in the Visual editor upon opening an edit file unless the EDITMODE option of the SET ENVIRONMENT command is set equal to COMMAND, or unless the EDIT command was entered from inside a command list or XEQ file. If either of these is the case, you remain in Command mode instead of entering Visual mode.

Since you can only edit the Latest version of your file, the filename parameter does not provide for version designation. When you are editing a file, the Latest version is locked and no one can edit or read that version while the file is open.

For more information on Workspaces and file Versions, see the Workspace and File Management section of this manual.

Example

```
>>EDIT MYFILE
--> File MYFILE.group.account open for editing
```

Edit Options

An Edit options menu is automatically displayed when you create a file or when you convert an ASCII formatted file to a TSAM format. You can re-define most options for an existing edit file with the SET EDIT command following a TOOLSET command prompt (>>) or by pressing the SET EDIT function key located at level one. TOOLSET defaults are filled in with the exception of PROGRAM-ID. This must be filled in if you are creating a COBOL MAIN or SUBPROGRAM, and it should match the Program ID given in your source file. Use the cursor control keys or the TAB key on your terminal to change any of these options. When you have defined the options as you want them, press the SET OK function key.

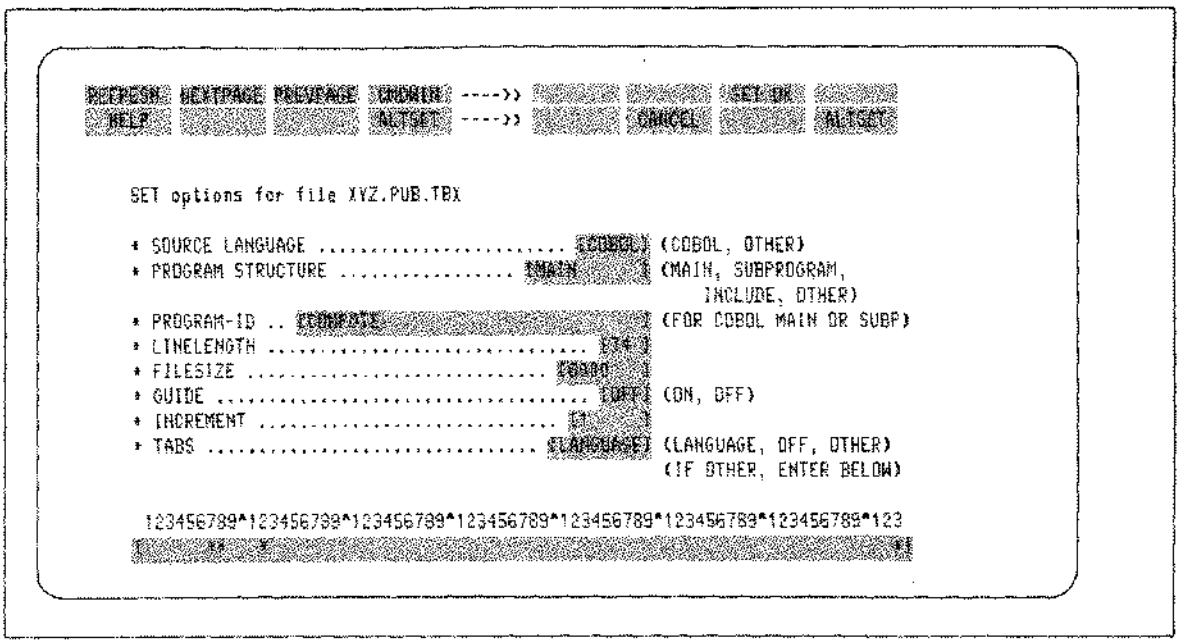


Figure 5-2. Sample SET Menu for COBOL Files

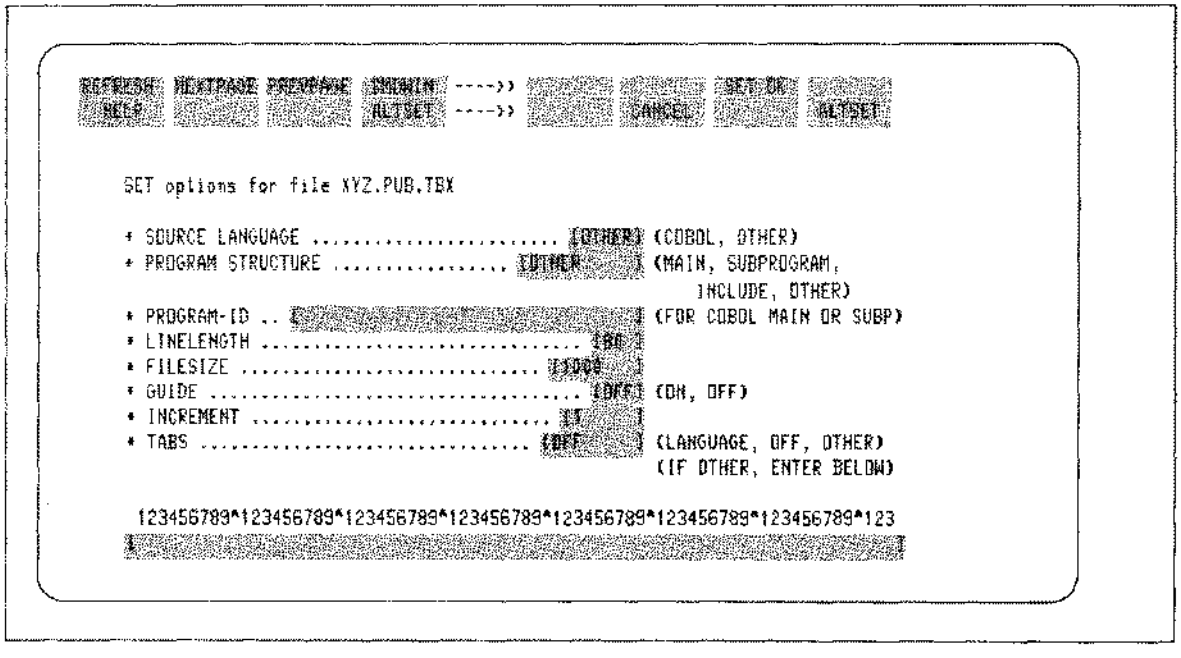


Figure 5-3. Sample SET Menu for NON-COBOL files

Changing Edit Options

All of the set options for an edit file can be changed after the file is created except for SOURCE LANGUAGE, LINELENGTH, and FILESIZE. These options remain as you initially define them for the life of your file. The rest of the options may be changed by redisplaying the menu with the SET EDIT command or function key while the file is open for editing.

SET EDIT

Allows you to view and change editing options.

Syntax

```
---------------------------
-                         -
-  SET [EDIT          ]   -
-      [PROGRAM       ]   -
-      [ENVIRONMENT   ]   -
-                         -
---------------------------
```

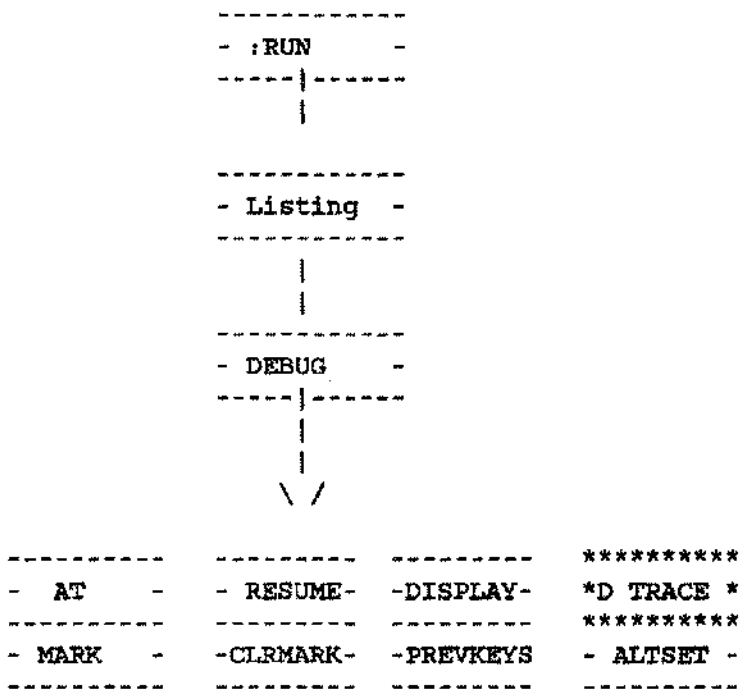
EDIT- This parameter gives you a menu of the following options:

```
[SOURCE LANGUAGE = [COBOL] [OTHER] ]
[STRUCTURE = [MAIN] [SUBPROGRAM] [INCLUDE] [OTHER] ]
[PROGRAM ID]
[LINELENGTH = [74] ]
[FILESIZE = [6000] ]
[GUIDE = [ON] [OFF] ]
[INCREMENT =[ 1 ]
[TABS = [OFF] [LANGUAGE] [OTHER]
```

Where

SOURCE LANGUAGE	Indicates which language is being used in your source program. This selection affects other Editor and Program Key options, such as the format of sequence numbers. For COBOL files, the sequence number can have up to 4 digits on the left side of the decimal point, and 2 digits on the right. For other Language files, 5 digits can be used to the left of the decimal point, and 3 to the right. This option is only available in the initial Set Menu and cannot be reset.
PROGRAM STRUCTURE	Informs TOOLSET of the source language structure contained in your edit file. You can change this option at any time.
PROGRAM-ID	What you enter for this option becomes your COBOL Program ID, and is needed if your program language is COBOL and your Program Structure is MAIN or SUBPROGRAM. This option should match the Program ID you put in your source file. TOOLSET automatically writes the Program ID to your file when it is created, but if you later change your source file ID, you must update the SET EDIT menu accordingly.
LINELENGTH	Sets the length of the logical records in your file. This should be set to the maximum number of characters you want per record, excluding the sequence number. The default is 74. You cannot reset this option.
FILESIZE	The number of logical records in your file. The default is 6000 records. This option cannot be reset.

DTRACE Function Key



To use the D TRACE (DATATRACE) function key,

(1) Display the compile listing by pressing the Listing key.

(2) Press the DEBUG key to access the D TRACE and MARK keys. (Your listing is still displayed).

(3) Position the cursor at the data-item whose value you want monitored, and Mark the beginning and end with the MARK key. (Character marks are made by pressing the MARK key once).

(4) Press the D TRACE key. A message confirming the Data trace is displayed at the top of your screen.

Using the D TRACE function key is the same as entering the DATATRACE command without any of the optional parameters.

Displaying Contents

DISPLAY

Displays contents of data-items to your terminal.

Syntax

```
-------------------------------------------------------------------------
-                                                                       -
-                              [ O[CTAL]      ]                         -
-                                                                       -
-                              [ I[NTEGER]    ]  [|FOR| n |ITEMS|]  }   -
-  DI[SPLAY]   { data-item     [ C[HARACTER]  ]                         -
-              {"literal"  }   [ H[EXADECIMAL]]                         -
-                                                                       -
-------------------------------------------------------------------------
```

Parameters

data-item	Any COBOL data-name or index-name defined in the COBOL source program. A data-item may be qualified with: (1) OF (2) IN (3) subscript (4) A period followed by a subordinate data-name and group data-name If data-item is subscripted, the subscript must be an integer value, and not a variable. If no subscript has been used, a subscript of 1 is assumed. You cannot reference a data-item in a program or subprogram other than the one that is currently executing.

literal	A character string that defines itself rather than representing some other value. It must be enclosed in double quotes. Quote marks cannot be used within the literal.
OCTAL INTEGER CHARACTER HEXADECIMAL	These keyword options override the implied format of the specified data-item. If the CHARACTER option is used, the contents of the entire item is displayed as a continuous string of ASCII characters. Non-printing characters are replaced by a period.
FOR n ITEMS	Used to List a specified number of table elements. This clause can only be used if the data-item contains an OCCURS clause in its DATA DIVISION declaration. "n" is assigned the value of 1 if this clause is not specified.

Usage

The DISPLAY command displays the current contents of data-items you specify to your terminal according to formats implied by the descriptions in the DATA DIVISION of your source file.

Data Division Formats

(1) Data with PICTURE X or A is output as ASCII characters.

(2) Picture 9 data is output in a format consistent with the COBOL DISPLAY statement.

(3) COMPUTATIONAL-3 packed data is output as decimal digits. The sign bits are interpreted and output before the number, and leading zeros are displayed.

(4) COMPUTATIONAL binary data is output as decimal integers. Negative signs are output preceding the number.

Table items

If the data-item is a table item with an actual or assumed subscript (i), the [FOR n ITEMS] clause has the effect of issuing a DISPLAY command for each item in the table with subscript (i) to (i+n-1) inclusive.

Group items

The default format for a group item is to display the entire item in character format.

Example

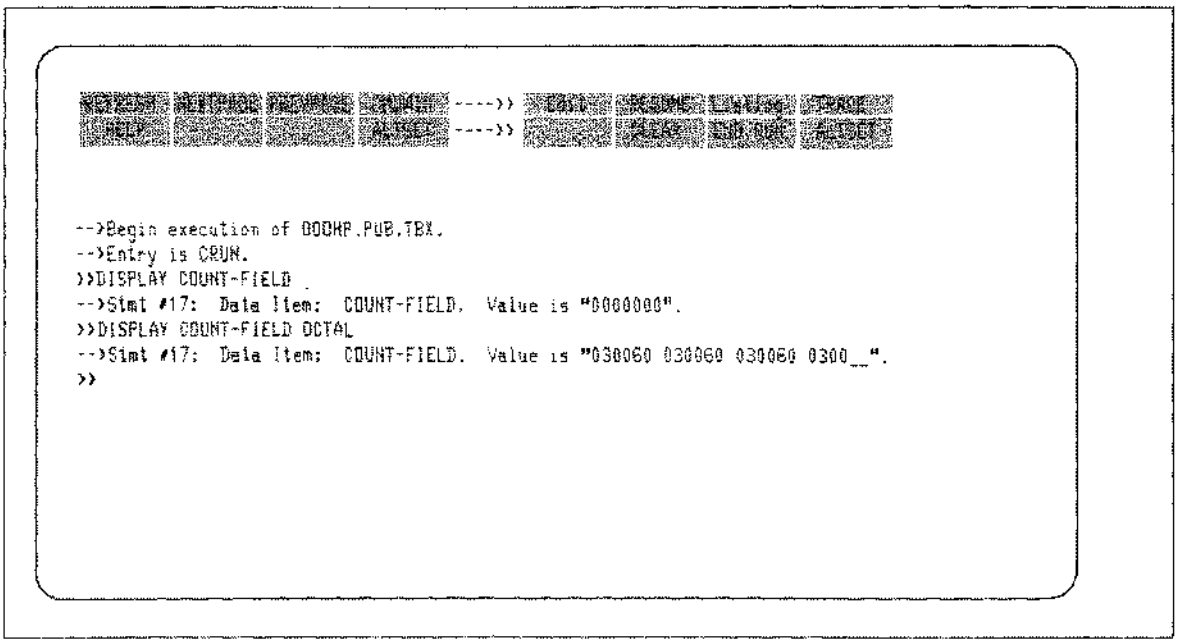
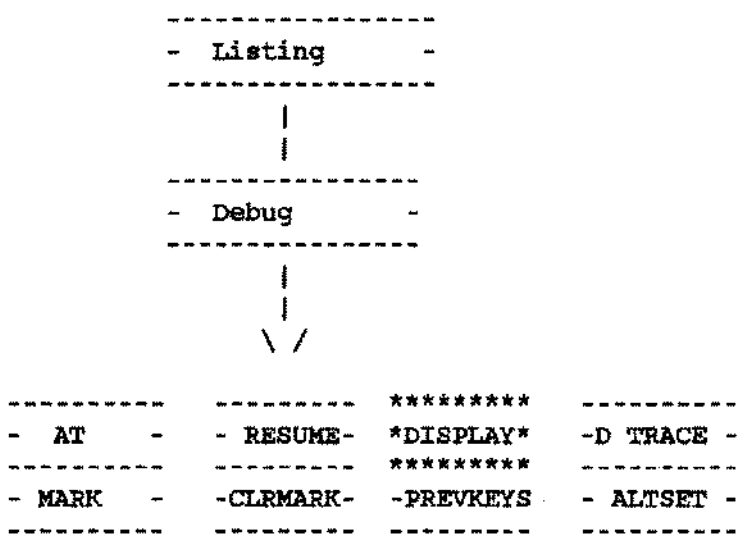


Figure 7-11. DISPLAY Command Screen.

DISPLAY Function Key



To use the DISPLAY function key,

(1) Display the compile listing by pressing the Listing key.

(2) Access the DISPLAY key by pressing the Debug key.

(3) Position the cursor at the data-item whose value you want displayed, and Mark the beginning and end of that item with the MARK key.

(4) Press the DISPLAY key.

Notice that your screen is cleared if the display is greater than one line.

Changing the Contents of a Data-Item

MOVE

Transfers the value of a literal, figurative constant or data-item to another data-item.

Syntax

```
                                                                    
                                                                    
          {literal     }                                            
 MOV[E]   {data-item-1 }  TO  data-item-2  [|FOR| n |ITEMS| ]       
          {fig-constant}                                            
                                                                    
```

Parameters

literal	A character string that defines itself rather than representing some other value. The value being transferred may be numeric, signed or unsigned, or nonnumeric. Note that nonnumeric values require double quotes. This is a sending field.

data-item-1	The sending field for the MOVE command. A data-item is any COBOL data-name or index-name defined in your source program. If subscripted, the subscript must be an integer value. If the data-item is a table item and unsubscripted, the subscript (1) is assumed. NOTE: Level 88 items are not allowed.
figurative constant	A value that has been assigned a fixed data-name in COBOL. ZEROS and SPACES are the only figurative constants allowed in TOOLSET. This is a sending field.
data-item-2	The receiving field. See data-item-1. When the value has been moved, this field is automatically displayed.
FOR n ITEMS	Number of times data-item-1 is to be moved. "n" is a positive or unsigned integer. This clause can only be used if data-item-2 contains an OCCURS clause in its Data Division declaration. A value of 1 is always assumed for n if this clause is not specified. NOTE: The Lower Bound of a table is always assumed to be one, and the Upper Bound is always assumed to be the maximum value regardless of any possible DEPENDING ON clause.

Usage

The Symbolic Debug MOVE command allows you to transfer the value of a literal, figurative constant, or data-item to another data-item. When the sending field or literal is shorter than the receiving field, data-item-2 is padded with zeros or spaces as required. If data-item-2 is shorter, the value is truncated. The permissible types for data-item-1 and data-item-2, and conversion, are defined according to the COBOL MOVE statement.

Subscripts

If data-item-2 is a table item with actual or assumed subscript (i), the FOR n ITEMS clause has the same effect as issuing a MOVE command for each item of the table with subscript (i) to (i+n-1) inclusive.

If, in addition, data-item-1 is a table item with actual or assumed subscript (j), the subscript of data-item-1 takes the values (j) to (j+n-1) inclusive when functioning as the source item for the MOVE command.

Decimal Points

Symbolic Debug aligns decimal points (".") in the sending and receiving fields for you.

Example

(1) Figurative Constant

```
>>MOVE ZERO TO TABLE(5) FOR 20
```

This example moves the value ZERO starting at element 5 of TABLE for 20 times.

(2) Data-Item

```
>>MOVE KOUNT TO TOT-KOUNT
```

This example moves the value of data-item KOUNT to the data-item TOT-KOUNT.

(3) Numeric Literal

```
>>MOVE 25 TO KOUNT
```

This example moves the value 25 to the data-item KOUNT which is defined as a picture 99.

(4) Table-Item

```
>>MOVE MY-TABLE TO THEIR-TABLE(2,4,3) FOR 3
```

This example moves the items in MY-TABLE to THEIR-TABLE in the following manner where MY-TABLE is defined as a (3*3) dimension table, and THEIR-TABLE is defined as a (5*5*5) dimension table.

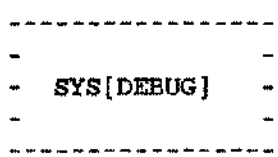
MY-TABLE	(1,1)	(1,2)	(1,3)	(2,1)	(2,2)
THEIR-TABLE	(2,4,3)	(2,4,4)	(2,4,5)	(2,5,1)	(2,5,2)

Since MY-TABLE has no explicit subscript, it defaults to the first element in the table. Notice that the subscript that is nested the deepest varies the quickest.

SYSDEBUG

Accesses the MPE DEBUG utility.

Syntax



The SYSDEBUG command allows you to access MPE DEBUG from HPTOOLSET. Exiting MPE DEBUG does not automatically return control to TOOLSET. Your program resumes execution and control is returned to symbolic debug only when a TOOLSET interrupt is encountered, or when execution of your program is terminated.

MPE DEBUG cannot be accessed from TOOLSET when your program is suspended at an active breakpoint, until that breakpoint is cleared. You are asked if you wish to clear the present breakpoint.

Example

```
>>SYSDEBUG
*BREAK* 0.535
 ?
```

END RUN

Terminates execution of your program.

Syntax

Usage

The END RUN command causes program execution to terminate. All currently set breakpoints and traces are cleared.

Example

```
 >>END RUN
-->End of program.
```

CONTROL-Y

Entering a Control-Y returns control to TOOLSET unless you code it to perform a different function, in which case that function takes priority.

Process Handling

Symbolic Debug handles up to 3 processes simultaneously when process handling is employed. If 3 processes are using Symbolic Debug, any other processes created are run without Symbolic Debug regardless of how they are compiled and prepped. The user is responsible for the the timing and interaction of his programs.

Using Symbolic Debug on Files Outside Your Workspace

If you wish to run Symbolic Debug on a program file other than the one in your current workspace, you can use one of two options,

(1) Enter the workspace of that program file.

(2) Issue a USE command for the program file so that it is known to your workspace. (See Section 4 for information on the USE command).

(3) Issue a :RUN progfilename

Using Symbolic Debug on Files Edited Outside TOOLSET

The TOOLSET Symbolic Debug utility can be used for programs you have created and modified with an Editor other than the TOOLSET Editor. To run Symbolic Debug on these programs, use the following steps.

(1) Use a $CONTROL SYMDEBUG statement in your source file.

(2) Compile your source file outside of TOOLSET, that is, use the MPE Compile command.

(3) Prep your USL file outside of TOOLSET using the MPE Prep command.

(4) Enter HPTOOLSET

(5) Do a USE command on the Program file name used when you prepped the USL file.

(6) Run your program using the program file name

```
:RUN progname
```

NOTE: No list file is generated when source files are executed in this way, so you will not be able to peform debug operations referencing line numbers, or by Marking. You can perform debug operations through symbolic referencing with Debug commands.

```
>>AT PARA-TWO

  NOT

>>AT #125
```

APPENDIX A

TOOLSET Command Summary

Following is a quick summary of all TOOLSET commands and the TOOLSET function with which they are associated. If you need additional information, refer to that section of the manual that discusses that particular operation. An asterisk (*) next to the command indicates that there is a corresponding function key.

A[DD]	*	Allows you to add text to the file you are currently editing.	Editor
AT	*	Allows you to set breakpoints in your program.	Symbolic Debug
[B]REAK		Returns control to the TOOLSET Command Interpreter.	Symbolic Debug
CA[LLS]		Displays the currently executing programs and subprograms.	Symbolic Debug
CH[ANGE]	*	Allows you to change specific strings or column ranges in the edit field.	Editor

CL[EAR]	*	Removes a breakpoint in the the user program.	Symbolic Debug
COM[PILE]	*	Causes TOOLSET to call the COBOL compiler for your COBOL source file.	Program Key
CON[VERT]		Converts files from ASCII or KSAM to TSAM format, or from TSAM to ASCII format.	Workspace
COP[Y]	*	Copies text within, or to and from an edit file.	Editor
COPYF[ILE]	*	Copies a named TSAM file to a new file in the current workspace.	Editor
DA[TATRACE]	*	Monitors the value change of a data-item.	Symbolic Debug
D[ELETE]		Deletes text from an edit file.	Editor
DISC[ARD]		Negates the USE command	Workspace
DI[SPLAY]	*	Displays the contents of data-items to your terminal	Symbolic Debug
ED[IT]	*	Opens a file for editing.	Editor

EN[D]	*	Terminates a TOOLSET function or activity.	All-Purpose
EX[IT]	*	Exits TOOLSET and returns you to MPE.	All-Purpose
F[IND]	*	Locates a specific occurrence of text in the file you are currently editing or reading.	Editor
H[ELP]	*	Accesses the TOOLSET Help facility.	All-Purpose
LA[BEL]		Allows you to put a comment on a file version.	Workspace
L[IST]		Allows you to list a specific portion of your edit file while you are in command mode. There is an associated PRINT key.	Editor
L[IST] C[HANGE]		Lists changes from one version of a file to a higher version.	Workspace
M[ODIFY]		Allows you to make changes to a given line in your edit file.	Editor
MOV[E]	*	Moves text from a source location to a destination location.	Editor

MOV[E]		Transfers the value of a literal, figurative constant or data-item to another data-item.	Symbolic Debug
:PREP	*	Prepares a USL file and creates a program file.	Program Key
PRO[GRAM]	*	Displays workspace source files.	Program Key
PU[RGE]		Purges one or more files owned by the current workspace.	Workspace
REA[D]		Allows you to read a file without opening it for editing.	Editor
REC[OVER]		Recovers all versions of a file.	Workspace
RED[O]		Enables correction and re-execution of the last command or command list entered at the command prompt.	All-Purpose
RENA[ME]		Changes the system file identification of a file.	Workspace
RENU[MBER]		Renumbers all lines in your text file.	Editor
RES[UME]	*	Starts or restarts program execution	Symbolic Debug

RET[RACE]	Lists the most recently executed paragraph(s) name.	Symbolic Debug
;RUN *	Executes a program file.	Program Translation
SET ED[IT]*	Allows you to define Edit options. These options are initially defined when you create your file.	All-Purpose and Editor
SET EN[VIRONMENT]	Allows you to define general user options such as pagesize. Each option can be entered as a command.	All-Purpose
SET PR[OGRAM]*	Allows you to define program options. These options are initially defined when you create your workspace.	All-Purpose and Program Tranlation
SETR[EF]	Designates the Reference version of a file.	Workspace
SETV[ERSION] *	Freezes changes to a file.	Workspace
SHO[W] A[CTIVITIES]	Lists the current TOOLSET operations, their associated files, and the current Workspace.	All-Purpose

SHO[W] D[EBUG]		Displays the active breakpoints and datatrace variables.	Symbolic Debug
SHO[W] E[QUATES]	*	Lists all the file equations which are in effect for the current workspace.	Workspace
SHO[W] F[ILES]	*	Displays all owned and used files in a designated workspace.	Workspace
SHO[W] L[ABEL]		Displays comments that were entered in the file label.	Workspace
SYS[DEBUG]		Accesses the MPE DEBUG utility.	Symbolic Debug
T[RACE]	*	Identifies each subprogram, section and paragraph before it executes	Symbolic Debug
UN[DO]		Allows you to undo the last CHANGE, DELETE, MOVE, SHIFT or MODIFY command just entered.	Editor
US[E]		Allows you to share a file owned by another workspace and specify which version of a file will be shared	Workspace
W[ORKSPACE]	*	Allows you to create or change workspaces.	Workspace
X[EQ]		Allows input from a file.	All-Purpose

APPENDIX B

GLOSSARY OF TOOLSET TERMS

Access Pointer	Used to access different versions of a file. The access pointer can be set to any particular version of a file. If it is not set explicitly, it defaults to the Latest version for the file owner and the Reference version for the user. Versions are discussed in Section 4.
Active Function Keys	The key set which is currently defined for TOOLSET. For 264x terminals, the key sets appear at the top of your terminal and the active set is pointed to by the arrow located between the two sets. For 262x terminals, the active key set is located at the bottom of your terminal.

Active Version	A version of a file that has not been purged. The number of active versions may be different from the highest version number. File versions are discussed in Section 4.
Batch Mode	TOOLSET mode used with terminal types other than 2645, 2647, 2622 and 2624. It is also used when TOOLSET is invoked in a job. Certain commands are not available such as SET, SHOW, and REDO. Visual mode editing is not available.
Branching Function Key	Batch terminals: 2640, 2621, HP125 A function key that when pressed produces a new set of active function keys. A Branching key has only the first letter capitalized. Other function keys are fully capitalized. Function keys are discussed in Section 1.
Command List	A string of TOOLSET commands. Each command must be delimitted by a semi-colon (;). You may use a command list in the AT and DATATRACE commands.
Command Mode	Refers to typing TOOLSET commands following a >> prompt rather than pressing function keys.

Currently Executing Program	That program or subprogram which has current control of the user process. If several programs are executing by virtue of nested CALL statements, the most recently called program without a matching EXIT PROGRAM is the currently executing program. If no subprograms are executing, the main program is the one considered currently executing.
Execute (XEQ) Files	A TSAM or ASCII file used to input commands to TOOLSET. It is accessed by entering the XEQ command.
Function Keys	The eight programmable keys f1 through f8 on your terminal that TOOLSET re-defines according to which TOOLSET operations are currently taking place.
Function Key Tree	The structure that shows all of the TOOLSET function keys and the relationship between the different levels.
GO Function Key	Part of the function key set that results from pressing the Program key. This key compiles, preps and runs your source file in one step. See Section 6 for more information.

Latest Version	The working version of a file that is still subject to change. Only latest versions can be modified.
line-no (number)	The compiler generated integer that appears on your listing in front of each line of the COBOL source program. In Symbolic Debug commands it is designed with a # sign followed by 1 to 5 digits. A line number can also be qualified by appending it to a program name: MYPROG#100. When not qualified, a line number is assumed to reference the currently executing program.
Listfile	A TOOLSET named file containing the most recent compile for a given source file. It is accessed by using the Listing command.
Offset	The number of compiler listing line numbers from the line containing the paragraph name given. It is prefaced by a pound (#) sign: PARA-1 + #10
Owned File	A TSAM file that belongs to the Workspace by being created, converted or copied.

Para-name

Any paragraph name in the Procedure Division. Para-name cn be qualified in the following ways:

(1) para-name OF section-name
IN

section-name.para-name

(2) Para-name and Section-name may reference a program explicity:

prog-name. .para-name

prog-name.section-name
prog-name.section-name.para-name

or with OF or IN:

para-name OF section-name Of prog-name

If prog-name is not specified, the currently executing program or subprogram is assumed.

EXAMPLE:

(1) PARA1 IN SECT1 IN PROG1

(2) PROG1.SECT1.PARA1

Permanent Function Keys

The function keys whose meanings remain the same throughout the TOOLSET session. For 264x terminals this set is always located at the top left of your terminal. For 262x terminals, this set appears at the bottom of your terminal when they are active.

Program Function Key

A Branching function key available from the Main key set. This key produces a menu of all source files and a key

	set associated with program translation operations such as prepping and compiling.
Reference Version	The version of a file that is referenced by its users. The reference version is set by the file owner with the SETREF command. See Section 4 for further information.
Reserved Words	TOOLSET commands and connector words such as 'TO' and 'FROM' that can be optionally used for syntax clarity.
Source File	A primary text input file from which the compiler initially reads the source program records. This input file may actually expand to a set of files through the use of the $INCLUDE command or the COBOL COPY statement.
Symbolic Debug	A feature of TOOLSET that allows you to interactively debug a program without knowing memory locations or code addresses. It enables you to display and modify data structures on-line using names you defined in the source program. See Section 7 for further information.
Temporary Function Keys	Function keys whose meanings change according to the current TOOLSET operation. On 264x terminals these keys these keys appear at the top right of your terminal. On 262x terminals these keys appear on the bottom when they are active.
Token	TOOLSET commands, parameters, filenames or reserved words.

TSAM	The TOOLSET Access Method. Files must be in this format for TOOLSET to manage your program development files.
Used File	A TSAM file shared with the Workspace. A used file is made known to the Workspace by entering a USE command.
Version	A logical copy of a file. The copy is always available regardless of further modifications to the file. A logical copy or version is created with the SETVERSION command. The version concept allows multiple users to share the same file without making multiple copies.
Visual Mode	A TOOLSET editing mode that allows full screen editing with the terminal field edit and cursor control keys.
Workspace	A collection of files that is used in the development of one program. This collection includes the source files, USL file, and program file, and also a directory that keeps track of these files and their versions.

APPENDIX C

Error Messages

For more information on error messages you encounter and their recoveries, type help or press the HELP function Key immediately after the error is displayed. The cause and corrective action for that error is displayed. To obtain a hard copy of the error catalog TSETCAT.PUB.SYS, FCOPY it or list it offline. TSETCAT is an 80 byte ASCII file.

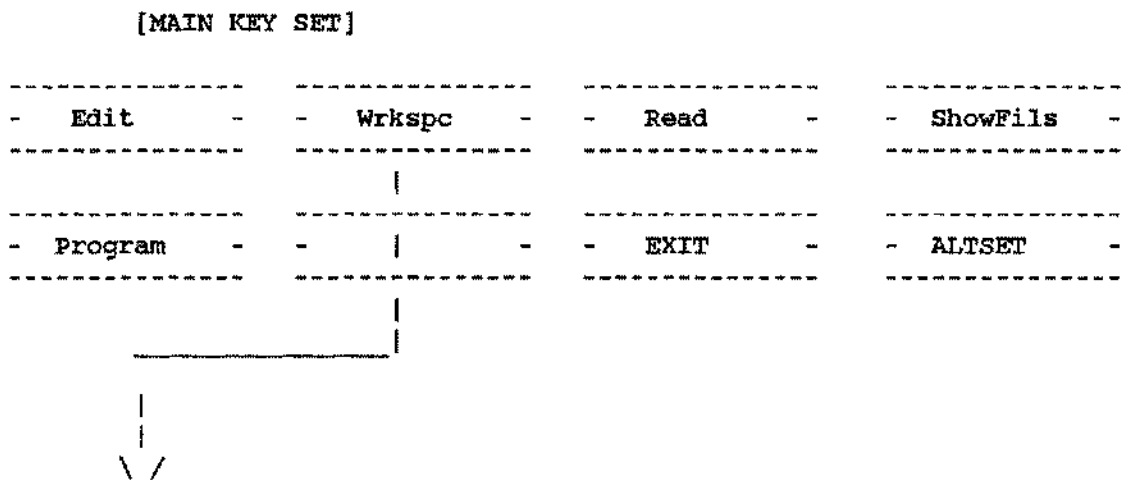
EXAMPLE:

```
>>add 100
***No file currently open for editing.  (85)
>>help
 Cause:  A command was entered which is only valid while
         an EDIT file is open, and there was not one open.
 Action:  Do an EDIT command before reentering the command.
>>
```

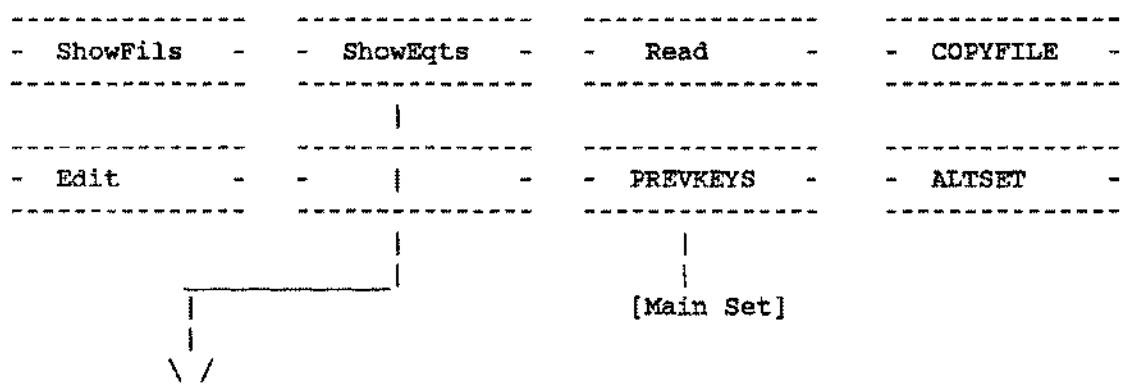
APPENDIX D

TOOLSET Function Key Tree

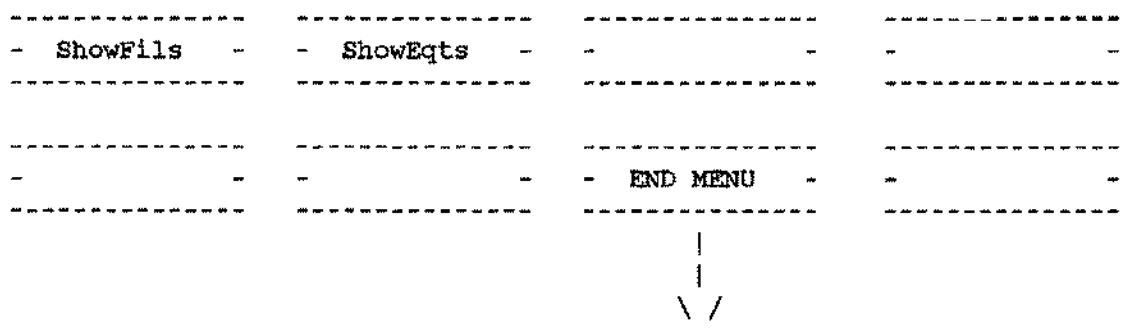
The following diagrams shows the path flow of the TOOLSET temporary function keys.



[Workspace Set]

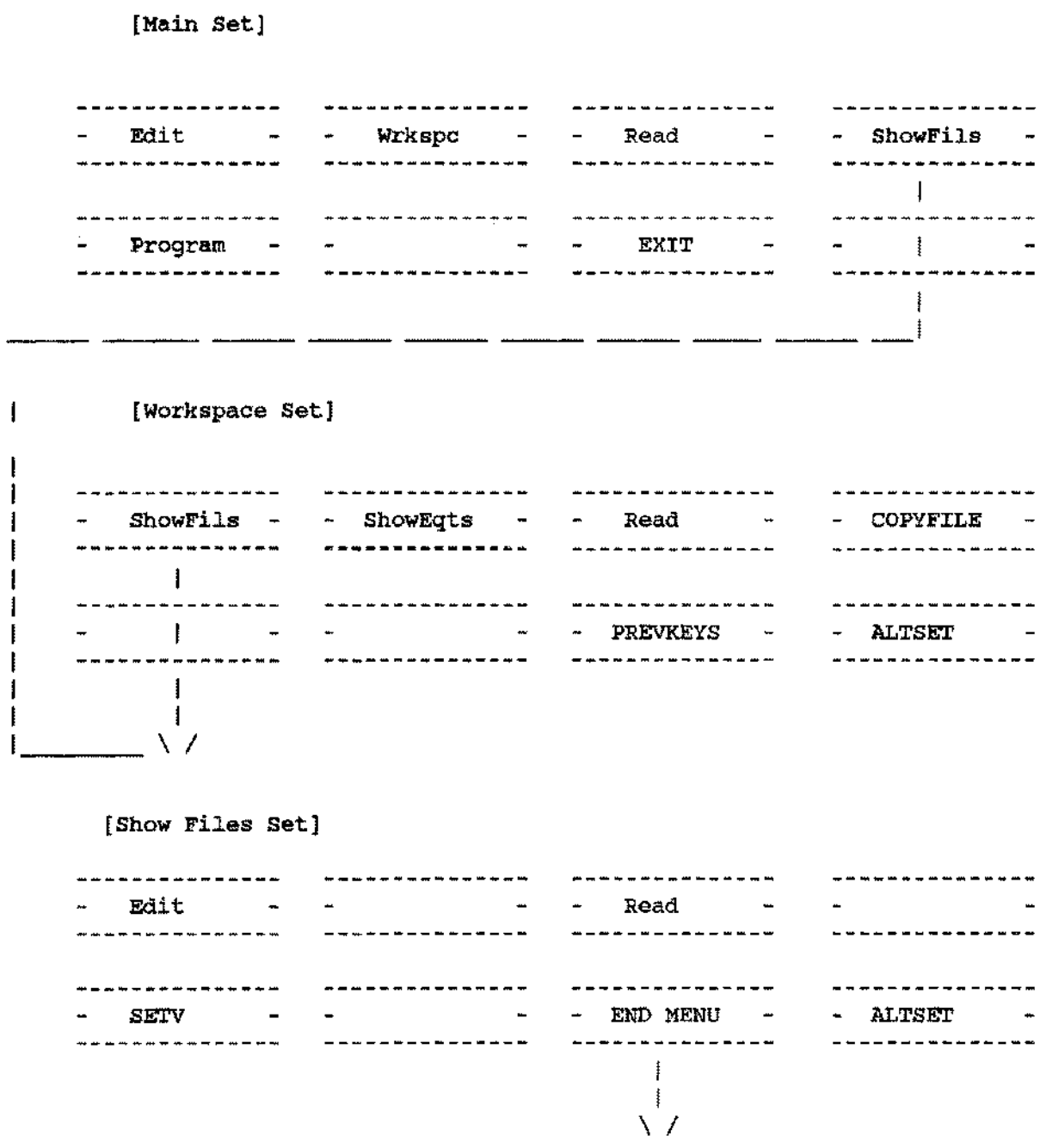


[Show Menu Set]



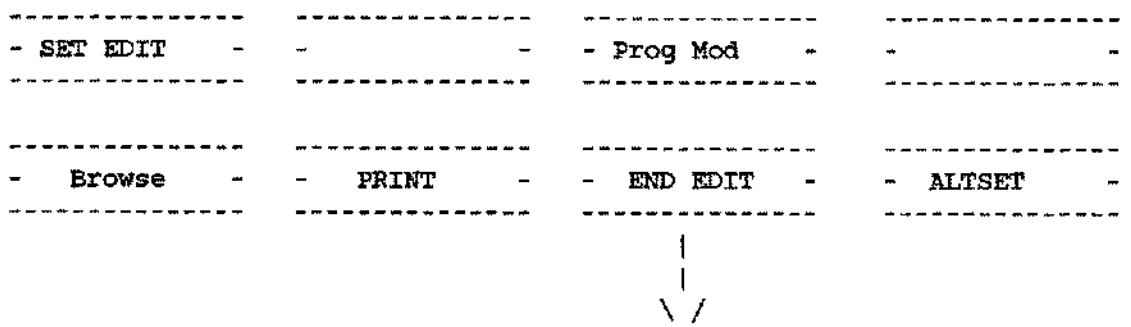
[Workspace set if softkey used.
Function key set before the menu
if the command was used.]

SHOWFILS Key Set



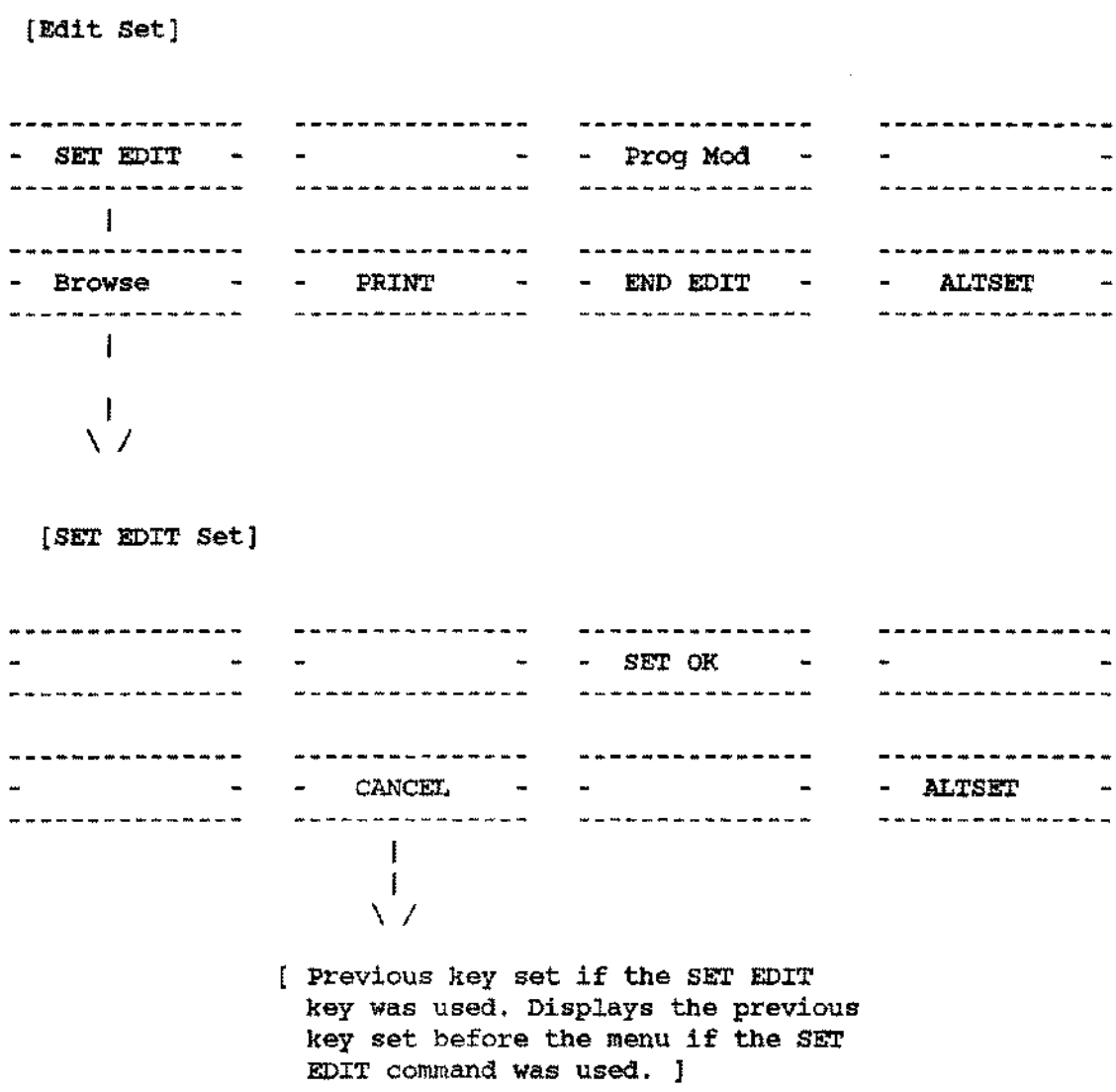
Edit Key Set

```
Edit key or command

From the following sets:

    Main Set
    Workspace Set
    Show Files Set
    Program Set
    Compile Set
    Listing from Compile
    Listing from Run

 [Edit Set]
```

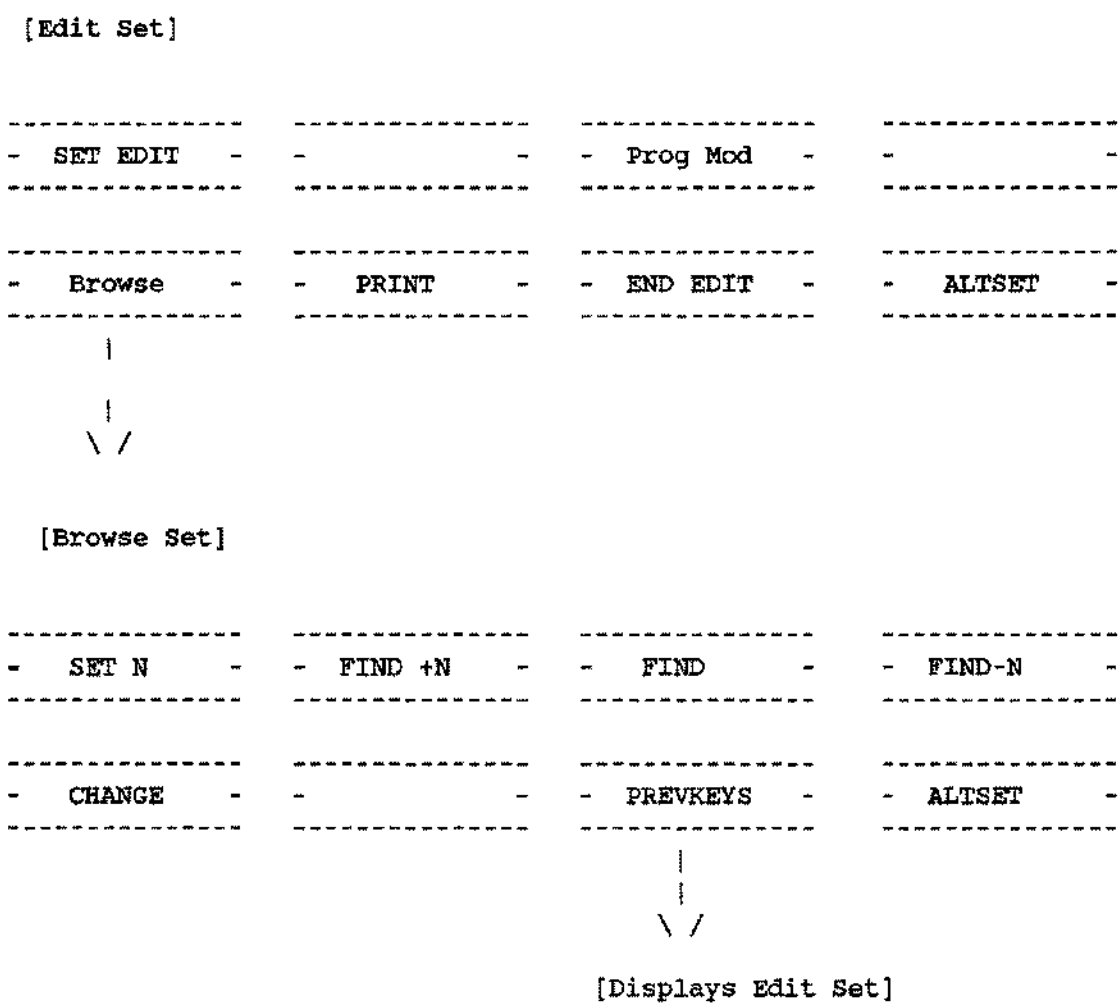


[Returns you to the previous key set if the Edit function key was used. Also returns you to the previous set if the command was used and not a menu or Set was done before the command.]

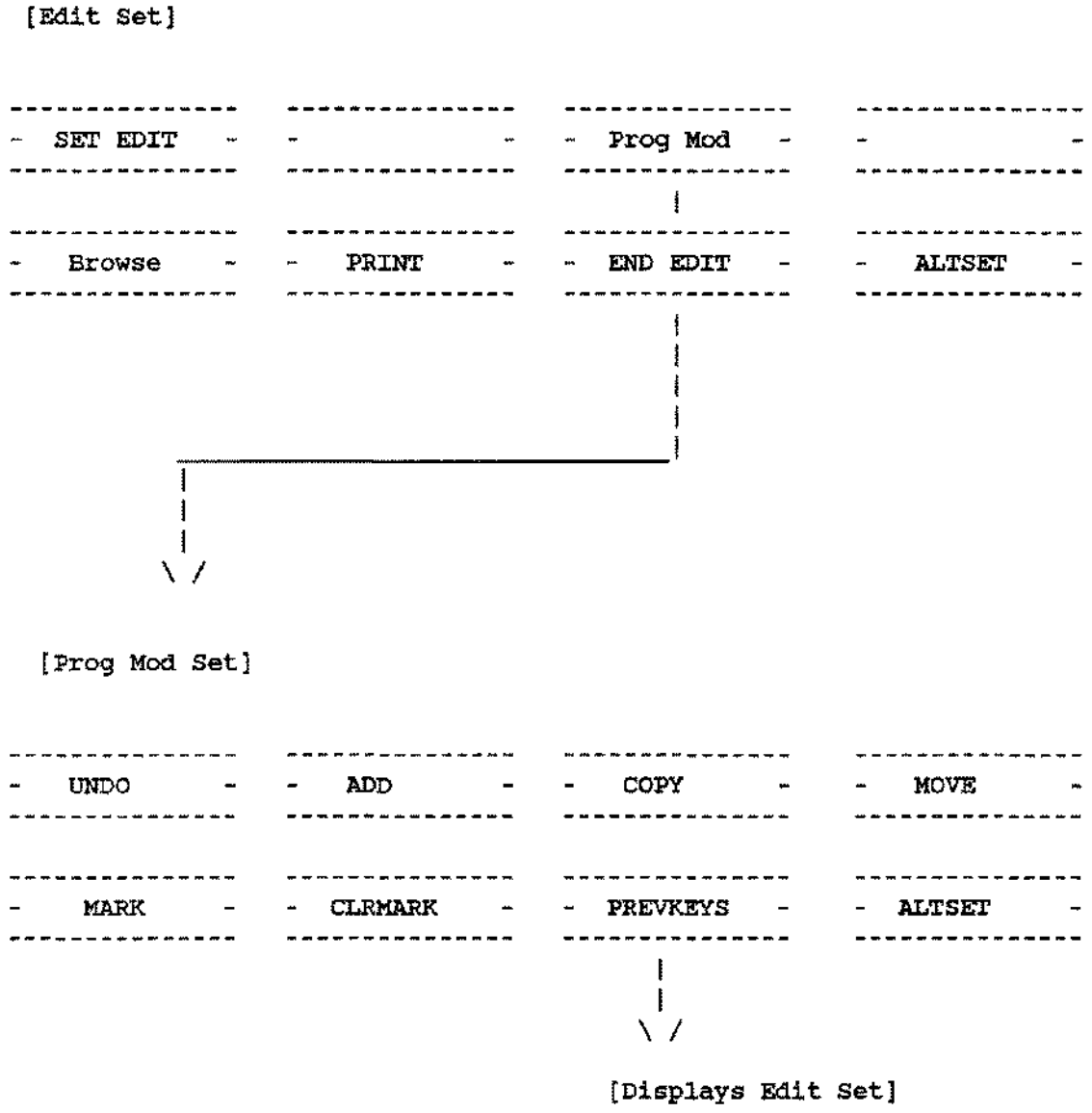
The Edit Set is also displayed when you create a new edit file.



Browse Key Set



Prog Mod Key Set

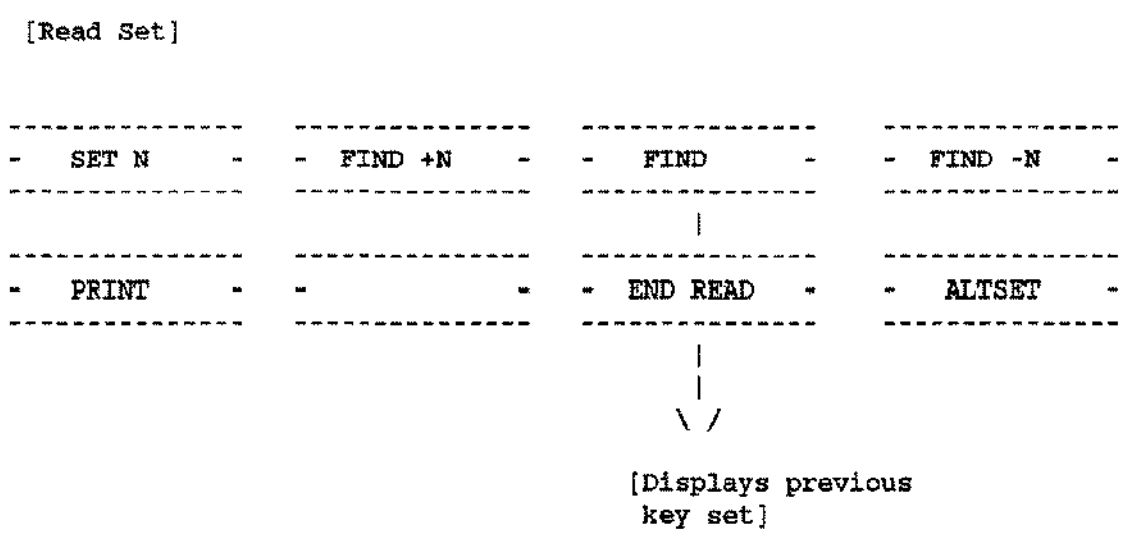


Read Key Set

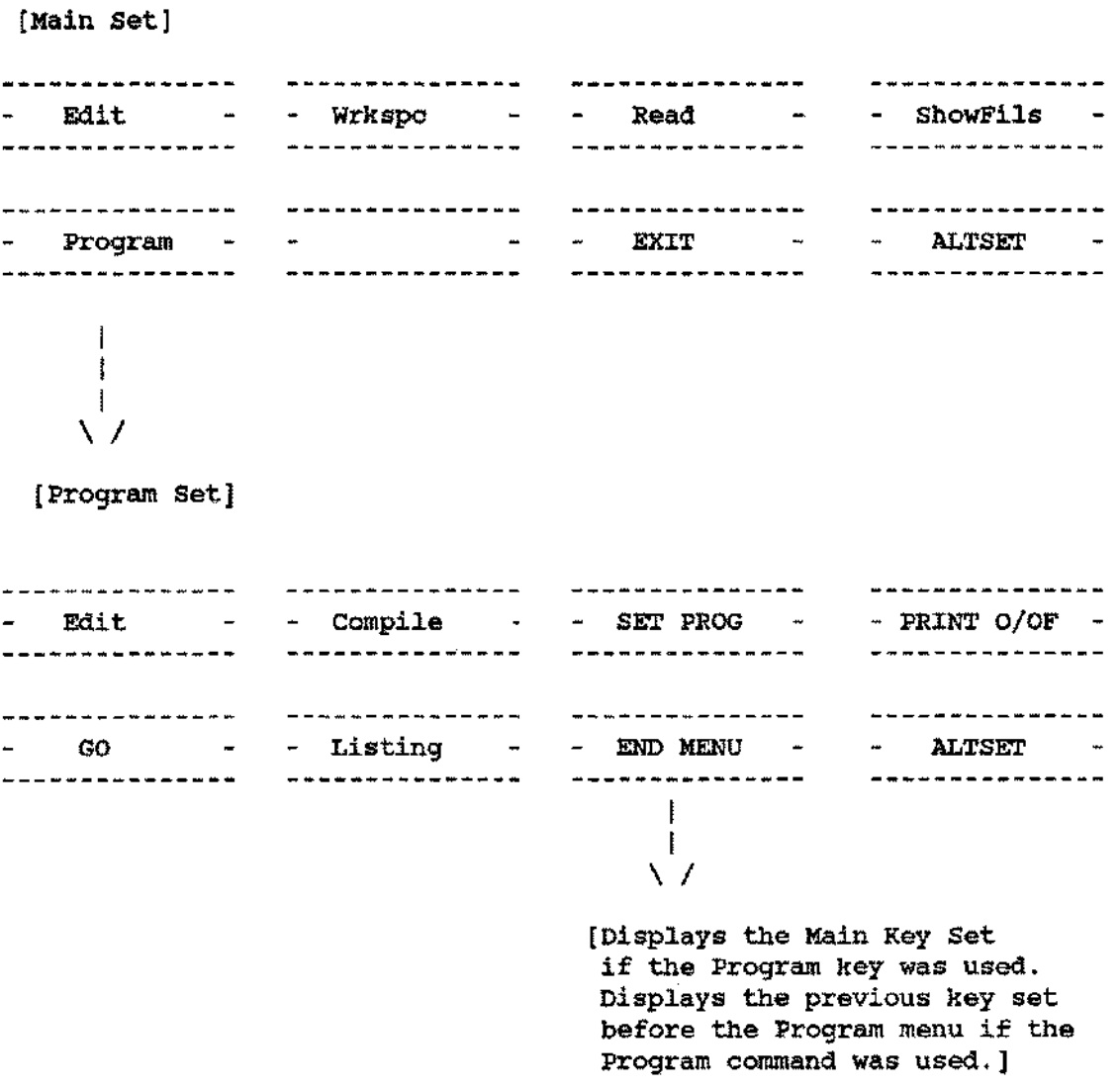
Using the Read function key or command
from the following key sets

Main Set
Workspace Set
Show Files Set

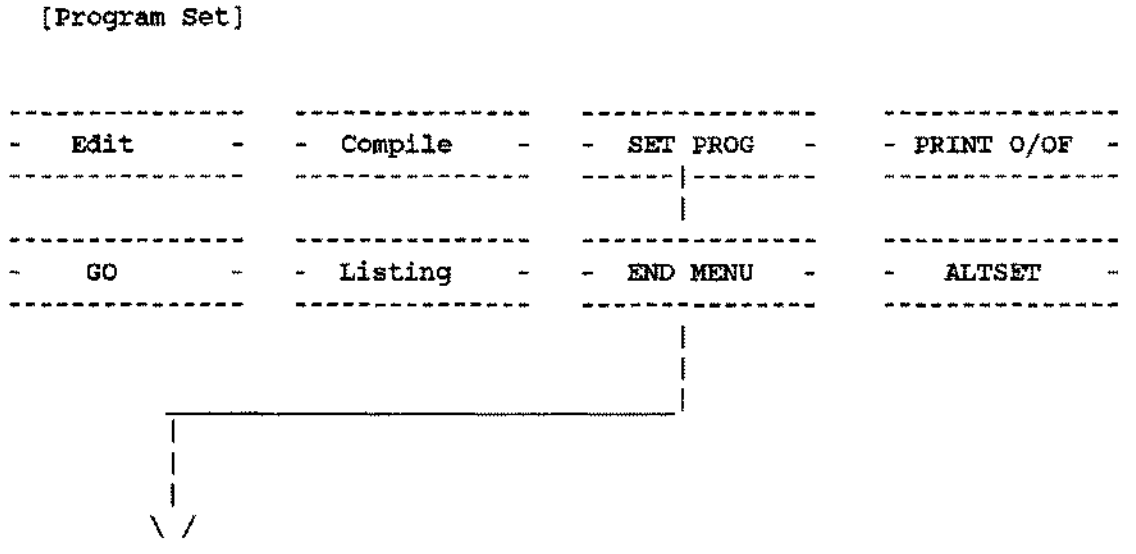
Produces :

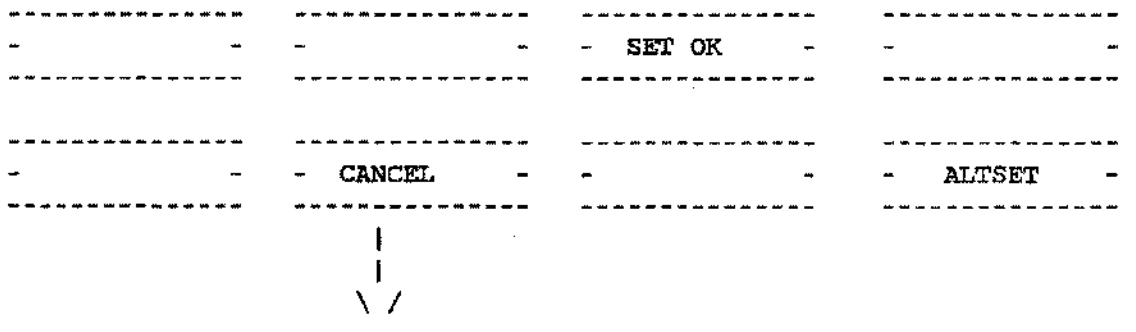


Program Key Set

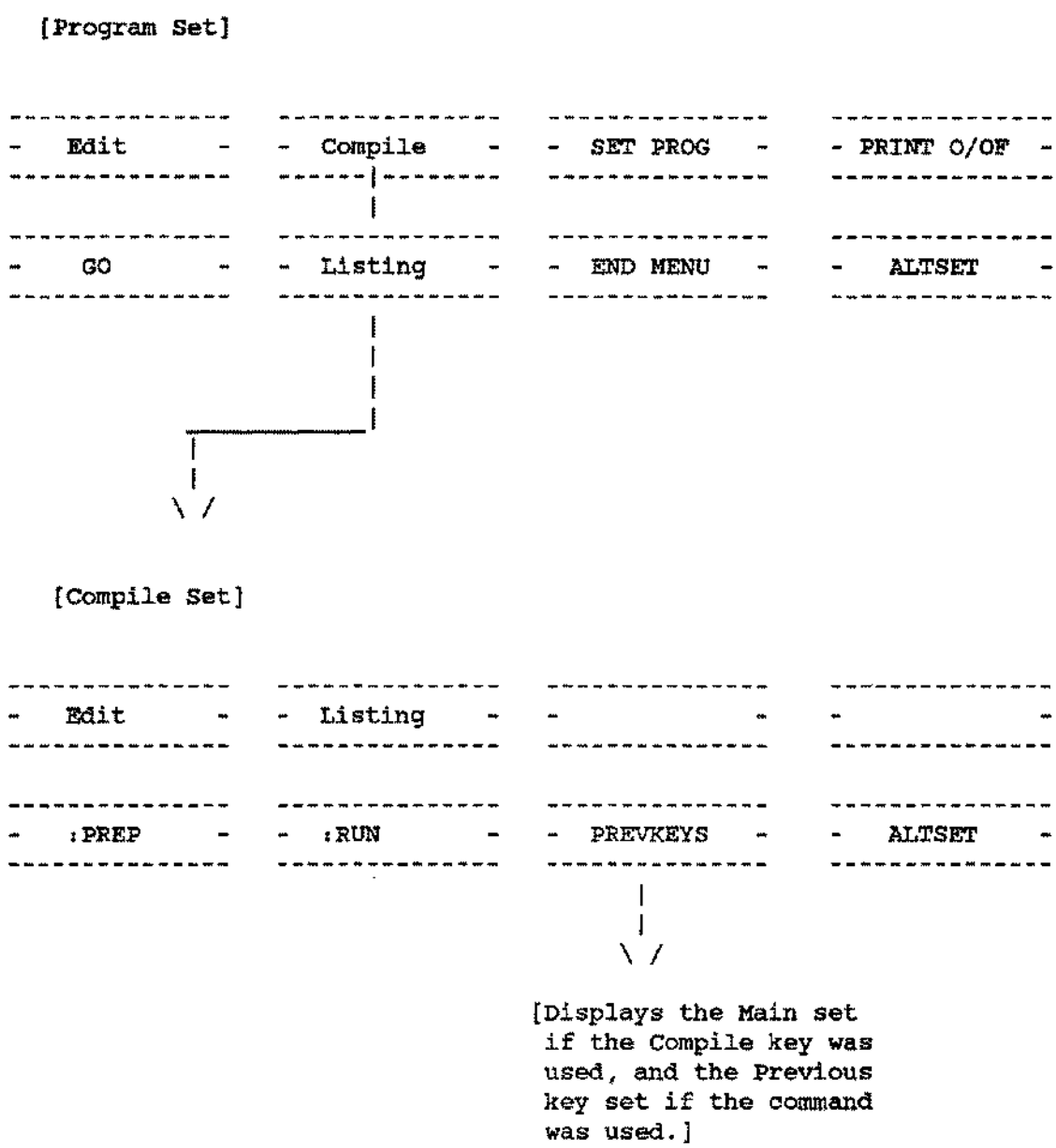


Set Prog Key Set

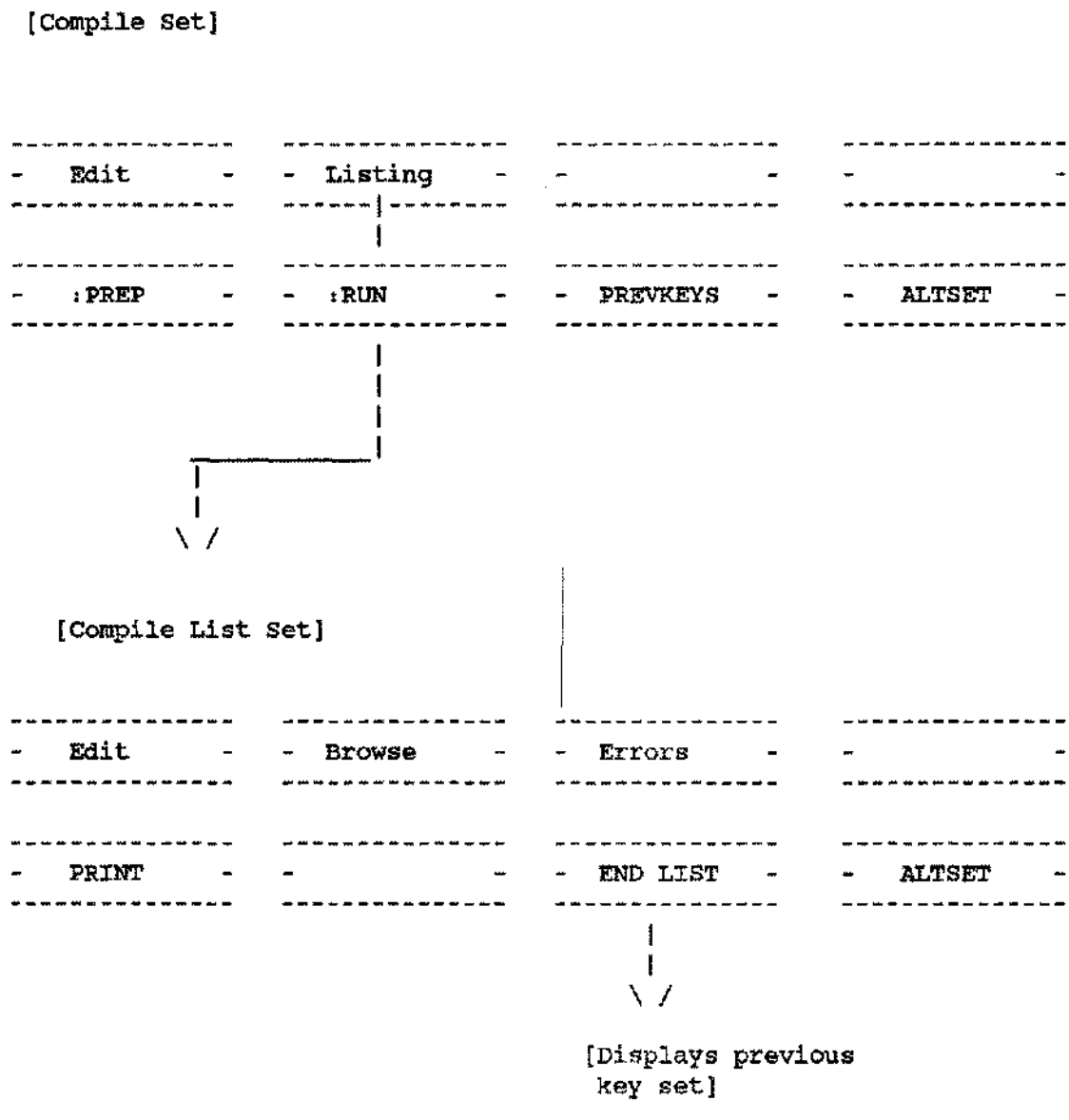




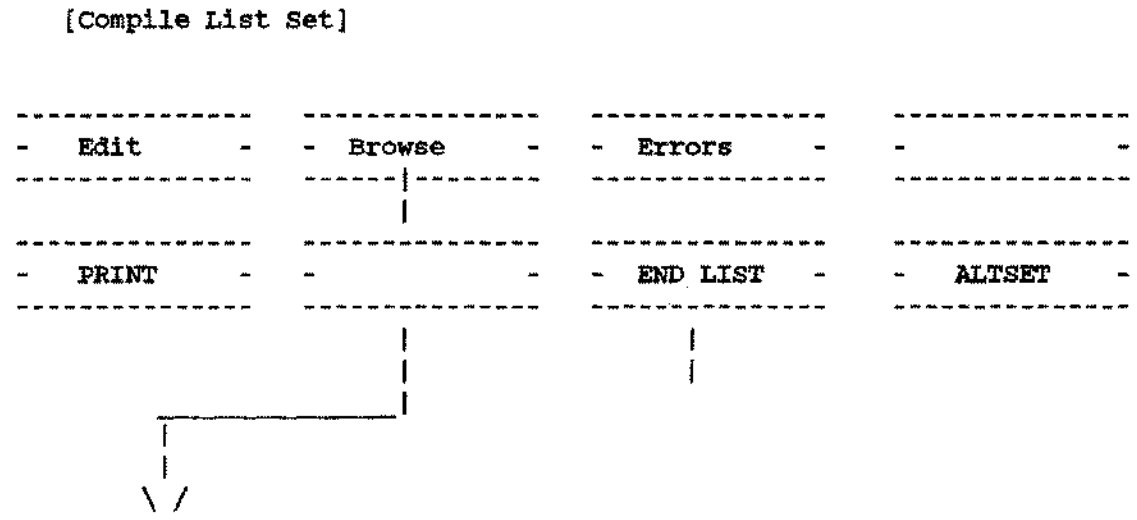
Compile Key Set



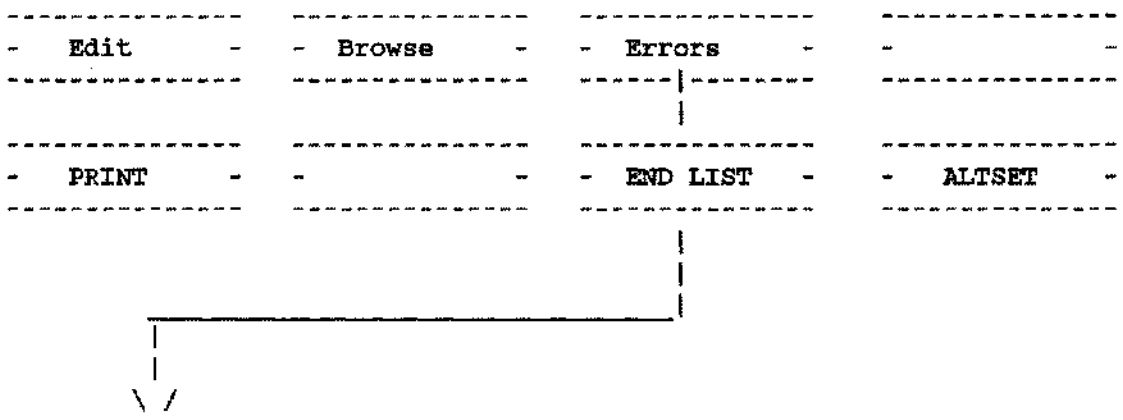
Compile List Set



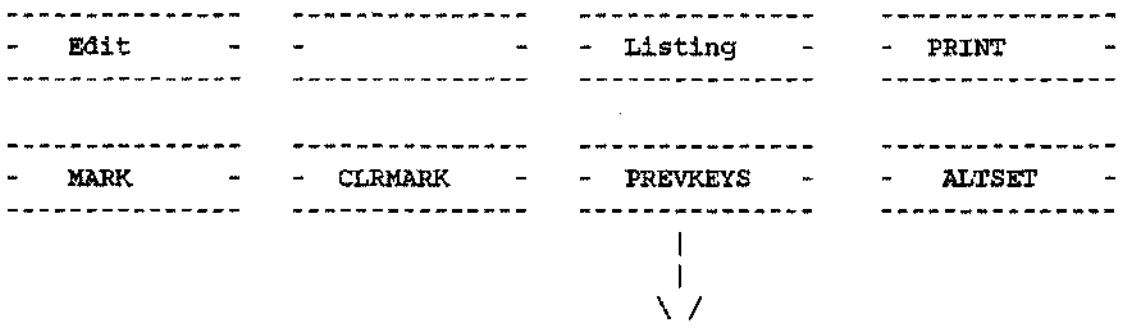
Compile Listing Browse Set



Compile List Errors Set



[Compile List Error Set]

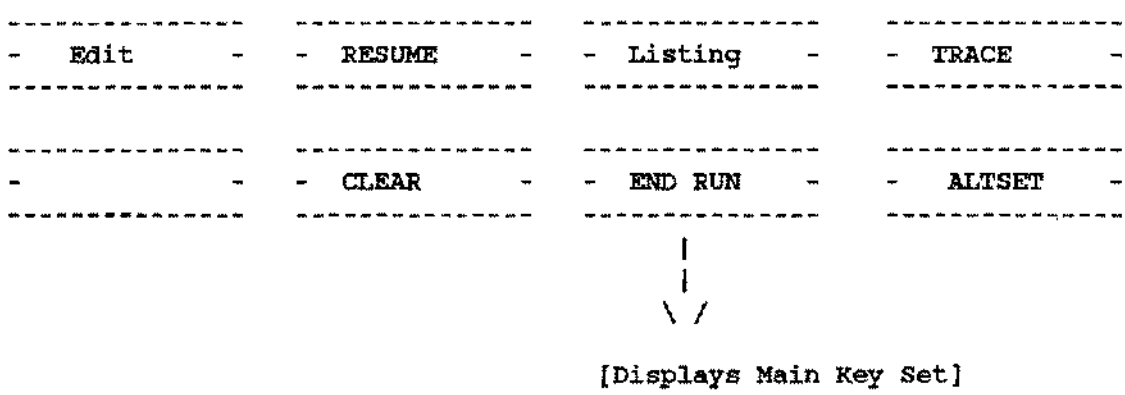


[Displays Compile List Set]

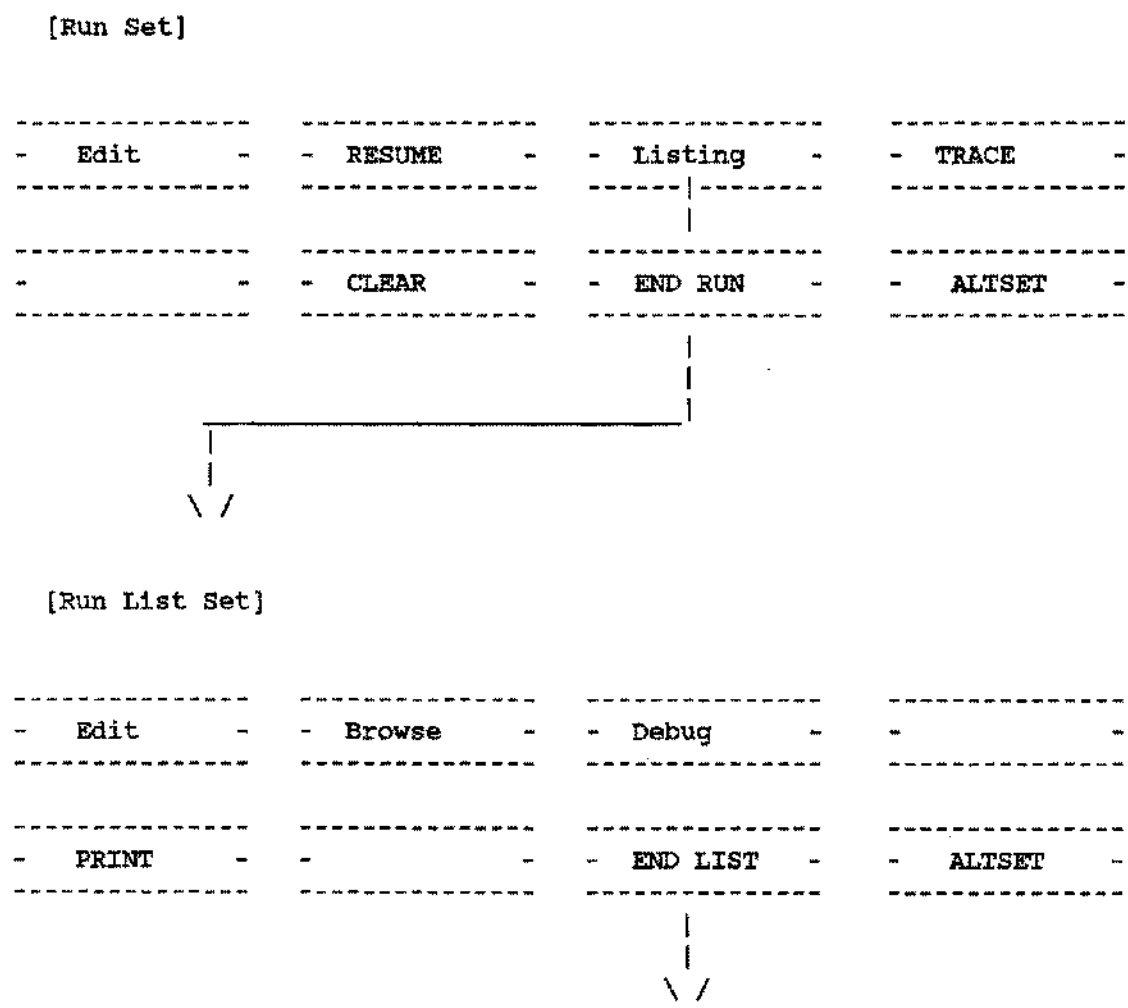
:RUN Key Set

From the :RUN command or key, or the GO function key

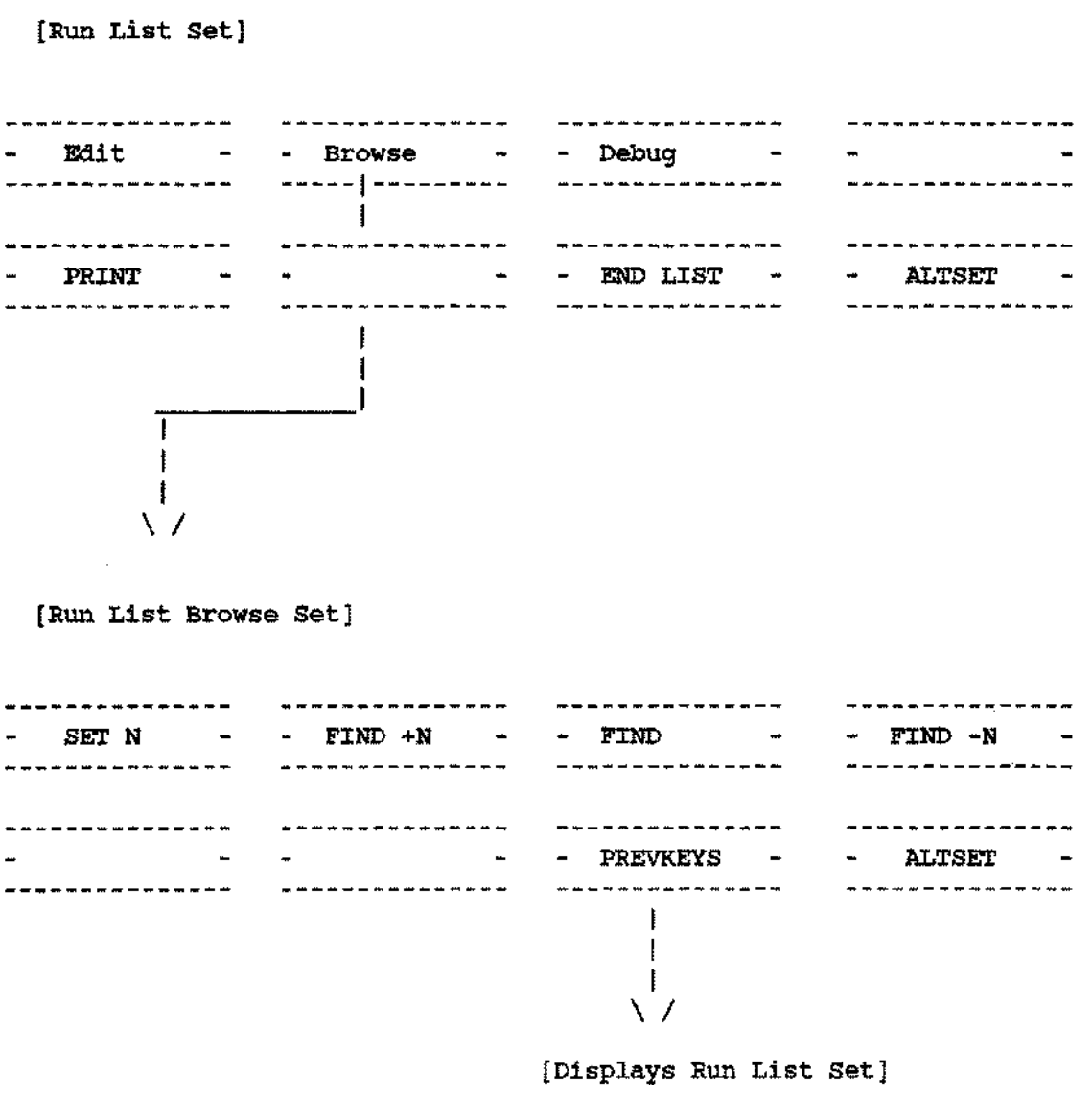
[Run Set]



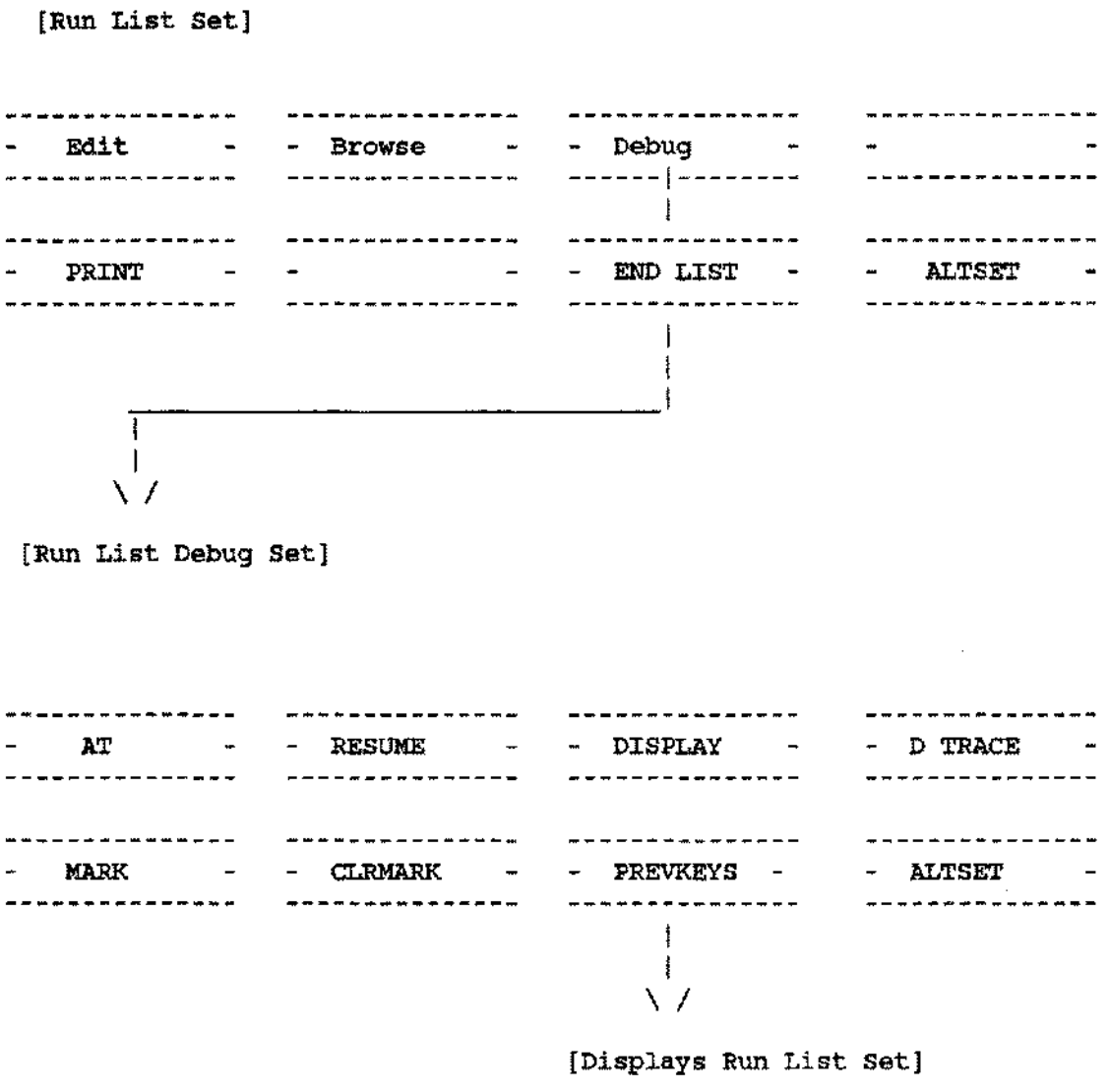
RUN Listing Key Set



Run List Browse Set



Run List Debug Set



INDEX

A

B

C

D

E

F

G

H

K

L

M

O

P

R

S

T

U

V

W

X