OpenVMS VAX Patch Utility Manual

Order Number: AA–PS6DA–TE

May 1993

This document describes the VMS Patch Utility and explains how to examine and modify executable and shareable images.

Revision/Update Information: This document supersedes the *VMS Patch Utility Manual*, Version 5.0.

Software Version: OpenVMS VAX Version 6.0

Digital Equipment Copporation
Maynard, Massachusetts

May 1993

The postpaid Reader's Comments forms at the end of this document request your critical evaluation to assist in preparing future documentation.

ZK4554

This document was prepared using VAX DOCUMENT, Version 2.1.

Contents

Examples

Tables

Preface

Intended Audience

This manual is intended to be used by experienced system programmers who are writing or modifying executable images and shareable images. Familiarity with a patching utility is not required; this manual provides introductory information on PATCH and on PATCH concepts and terminology.

Document Structure

This document consists of the following five sections:

- Description—Provides a full description of the Patch Utility (PATCH).
- Usage Summary—Outlines the following PATCH information:
  - –Invoking the utility
  - –Exiting from the utility
  - –Directing output
- Qualifiers—Describes PATCH qualifiers, including format, parameters, and examples.
- Commands—Describes PATCH commands, including format, parameters, and examples.
- Examples—Provides additional PATCH examples.

Associated Documents

The following documents provide additional information:

- *OpenVMS User's Manual*
- *OpenVMS DCL Dictionary*
- *OpenVMS System Manager's Manual*
- *OpenVMS VAX Device Support Manual*

Conventions

In this manual, every use of VAX VMS means the OpenVMS VAX operating system.

The following conventions are used in this manual:

Ctrl/*x*	A sequence such as Ctrl/*x* indicates that you must hold down the key labeled Ctrl while you press another key or a pointing device button.

PF1 *x*	A sequence such as PF1 *x* indicates that you must first press and release the key labeled PF1, then press and release another key or a pointing device button.
Return	In examples, a key name enclosed in a box indicates that you press a key on the keyboard. (In text, a key name is not enclosed in a box.)
. . .	A horizontal ellipsis in examples indicates one of the following possibilities: • Additional optional arguments in a statement have been omitted. • The preceding item or items can be repeated one or more times. • Additional parameters, values, or other information can be entered.
. . .	A vertical ellipsis indicates the omission of items from a code example or command format; the items are omitted because they are not important to the topic being discussed.
( )	In format descriptions, parentheses indicate that, if you choose more than one option, you must enclose the choices in parentheses.
[ ]	In format descriptions, brackets indicate optional elements. You can choose one, none, or all of the options. (Brackets are not optional, however, in the syntax of a directory name in a VMS file specification, or in the syntax of a substring specification in an assignment statement.)
{ }	In format descriptions, braces surround a required choice of options; you must choose one of the options listed.
boldface text	Boldface text represents the introduction of a new term or the name of an argument, an attribute, or a reason. Boldface text is also used to show user input in online versions of the manual.
italic text	Italic text emphasizes important information, indicates variables, and indicates complete titles of manuals. Italic text also represents information that can vary in system messages (for example, Internal error *number*), command lines (for example, /PRODUCER=*name*), and command parameters in text.
UPPERCASE TEXT	Uppercase text indicates a command, the name of a routine, the name of a file, or the abbreviation for a system privilege.
-	A hyphen in code examples indicates that additional arguments to the request are provided on the line that follows.
numbers	All numbers in text are assumed to be decimal, unless otherwise noted. Nondecimal radixes—binary, octal, or hexadecimal—are explicitly indicated.
quotation marks apostrophes	The term quotation marks is used to refer to double quotation marks ("). The term apostrophe (′) is used to refer to a single quotation mark.

PATCH Description

After you have compiled or assembled and linked a program, you may need to make changes to the code. The VMS Patch Utility (PATCH) provides an extensive set of commands that allows you to make changes to an image file in the form of patches. After making changes, you can execute the patched program without recompiling or reassembling and relinking.

PATCH provides several features to help you find and correct erroneous code. These features include the following:

- Use of symbols to reference locations
- Patch area to store additional data and instructions
- Entry and display modes to control the environment in which PATCH accepts your commands and displays its output

Use PATCH to maintain executable image files, shareable image files, and device-driver image files in any language supported by the VMS operating system.

It is common procedure to use the VMS Debugger before using PATCH. Using the debugger, you can make temporary changes to your code; using PATCH makes any debugger changes permanent. The source code does not reflect these changes until you change it using a text editor. See the *OpenVMS Debugger Manual* for more information.

1 Using Patch

PATCH requires two types of input: the image file that you want to modify and the PATCH commands that perform the modifications. You can use the following two methods to make changes using PATCH commands:

- Enter commands during an interactive terminal session. When the PATCH commands are executed, the changes you have requested are made.
- Write a command procedure to be executed during interactive or batch mode processing.

When you use the command procedure method, enter the name of a PATCH command procedure that includes the name of the image file you want to patch and the PATCH commands to be applied. You can create a PATCH command procedure during a PATCH session using the CREATE command or with a text editor. See Section 1.4 for information on using command procedures with PATCH.

PATCH offers the following features:

- Because PATCH is a symbolic utility, you can pass symbol information or define new symbols during a PATCH session and use the symbolic information later as a reference tool. See Section 2 for information on using symbols.
- Entry and display modes for use in referencing locations in the image file. See Section 3 for more information on entry and display modes.
- Patch area, a storage area for additional instructions and data. See Section 4 for information on using patch area.

1.1 Applying Patches

To apply a patch to an image file, perform the following steps:

1. Invoke PATCH (See Format section).
2. Set the Engineering Change Order (ECO) level.
3. Enter PATCH commands to change the image file.
4. Apply the patch with the UPDATE command.
5. Exit from PATCH.

These steps are described below. See the Commands section for detailed information about each PATCH command.

After invoking PATCH, establish the ECO level of the patch using the SET ECO command. This command assigns a unique ECO level to your patch, allowing you to initiate, identify, and monitor each of your patches. You must set the ECO level in order to process patches using command procedures. Use the CHECK ECO and CHECK NOT ECO commands to see if ECO levels have been set on specified patches.

After setting the ECO level, use PATCH commands to examine and modify the image file. Use the UPDATE command to apply the patch to the image file. The SET ECO and UPDATE form the boundaries of each patch.

After completing all necessary patches, terminate the patch session with the EXIT command.

The following example shows a patch applied to the image file NEGATION.EXE;1:

```
PATCH>SET ECO 1
PATCH>EXAMINE/INSTRUCTION 606
00000606:  TSTL R4
PATCH>REPLACE/INSTRUCTION 606 = 'TSTL R4'
NEW>'TSTL R5'
NEW>EXIT
old:   00000606:   TSTL R4
new:   00000606:   TSTL R5
PATCH>UPDATE
%PATCH-I-WRTFIL, updating image file DB1:[GARTH]NEGATION.EXE;2
```

In the above example, the ECO level is defined as 1. The EXAMINE command requests that location 606 be displayed as a VAX MACRO instruction. The REPLACE command then changes the instruction in location 606 to a different instruction. The UPDATE command sets ECO level 1 and applies the patch to NEGATION.EXE.

1.2 Output Files

PATCH produces three output files: the output image file, the journal file, and the command procedure file (see Table PAT–1). The output image file and command procedure file are produced only if you request them. The journal file is created automatically.

Table PAT–1 Output Files

File	Description
Output image file	An updated image file. Create this file using the UPDATE command.
Journal file	An ASCII file containing a record of the PATCH session. This file is automatically created by PATCH and provides an easy way to keep track of the changes and attempted changes made to an image file.
Command procedure	A file containing all successful PATCH commands that can subsequently be used as input to PATCH. Create this file using the CREATE command.

1.3 Input Image File

The input image file is the file you update. Because PATCH is not a language-dependent utility, you can use it to update an image file written in any language supported by the VMS operating system. Note that the input image file must be created by the VMS Linker.

Note

PATCH is not intended for customer use on Digital software. Any revisions applied to this software, except those contained in Digital-supplied automatic update kits, may make it difficult for you to install future software updates. Furthermore, application of patches to VMS software that are not supplied by Digital invalidates the warranty on the Digital-supplied VMS software.

The input image file can be a shareable image, device driver image, or executable image.

An executable image is the most common type of image. An executable image is one that can be run by a process. Patching an executable image requires no special rules or considerations.

A shareable image is an image that does not have a starting address. It must be linked with one or more object modules to produce an executable image. The following restrictions apply when patching shareable images:

- Specify only universal symbols.
- Use only a user-defined patch area to patch position-dependent images. You can use the default patch area to patch position-independent shareable images. When default patch area is created while patching position-independent shareable images, PATCH transfers the image section characteristics to the patch area image section descriptor.

A device driver image is a set of routines and tables used by the operating system to process an I/O request for a particular device type. To patch a device driver image, use a user-defined patch area rather than the default patch area to store additional instructions and data.

PATCH alters the input image file only when the /NEW_VERSION qualifier is specified. See the description of this qualifier in the Qualifiers section. When you invoke PATCH, it creates a copy of the input image file and incorporates your changes into the copy. This allows you to test the new image file while retaining the original file for evaluation.

You can also patch files (including nonimage files) by specifying the /ABSOLUTE qualifier with the PATCH command. See the description of this qualifier in the Qualifiers section.

1.4 Creating Command Procedures

A command procedure is a sequence of commands (and, optionally, input data) that performs a task in a manner similar to a program.

Used with PATCH, a command procedure is a sequence of PATCH commands that, when executed, applies patches to a specified image file. Command procedures are frequently used to ease the task of patching multiple copies of an image file. Instead of manually patching each file, you can submit a command procedure for interactive or batch mode processing.

Following are two ways to create command procedures:

- Use the PATCH command CREATE.
- Use an interactive text editor.

Descriptions of each follow.

1.4.1 Using the CREATE Command

After you invoke PATCH, enter the CREATE command to create and open a command procedure. PATCH automatically inserts the name of the input image file as the first entry in the command procedure. Subsequent commands that affect the image are written into this file. Commands that are used only to display information are not written to the file; these include EXAMINE, HELP, EVALUATE, SHOW MODE, SHOW MODULE, SHOW PATCH_AREA, and SHOW SCOPE.

As PATCH writes commands into the command procedure, all symbolic references are changed to absolute values. This translation occurs because symbol information is not always passed to the image file during compiling or assembling and linking. For example, if an image file has been linked without the /DEBUG qualifier, neither local nor global symbol information appears in the image's symbol table. Using absolute values ensures that the command procedure executes properly.

PATCH automatically abbreviates all commands in command procedures created using CREATE to conserve space.

An example of creating a command procedure with the CREATE command appears in Section 1.4.3.

1.4.2 Using Text Editors to Create Command Procedures

When you use a text editor, such as EDT, to construct command procedures, the first entry in the command procedure must be the name of the input image file that is to incorporate the patches. Enter PATCH command lines to the file in the order in which you want them processed.

You can use symbolic references in your command procedures only if PATCH can resolve these references when the command procedure is run. If PATCH is unable to resolve a symbolic reference, the command procedure is prematurely terminated.

If you want to use local symbols that have been defined in the image file, enter the SET MODULE command at the beginning of the command procedure. SET MODULE enters these symbols into the PATCH symbol table.

The following example shows how to create a command procedure that modifies the image file TEST.EXE;2, which contains the local symbol LOOP. The command procedure TEST.COM contains the following lines:

```
TEST.EXE;2
SET MODULE/ALL
SET ECO 3
REPLACE/INSTRUCTION LOOP='ADDL2 #2,R3'
   .
   .
   .
UPDATE
EXIT
```

The first PATCH command, SET MODULE/ALL, places the local symbols from all modules of TEST.EXE;2 into the PATCH symbol table. The symbol LOOP then indicates the address of the instruction ADDL2 #2,R3.

1.4.3 Creating User-Defined Symbols in Command Procedures

PATCH command procedures can contain many patches. Each patch must start with a unique Engineering Change Order (ECO) level and end with the UPDATE command. If the patches within a command procedure contain user-defined symbols, initialize these symbols at the start of the command procedure. User-defined symbols are symbols defined with the DEFINE command.

To initialize user-defined symbols, place them at the beginning of the command procedure, and assign the value zero (0) to them. In the patch where the symbol is used, redefine the symbol to the value you want to use.

The following example shows how to initialize user-defined symbols when creating a command procedure with the CREATE command:

```
PATCH>CREATE PATCOM
PATCH>DEFINE FIRST=0
PATCH>DEFINE SECOND=0
PATCH>SET ECO 1
PATCH>DEFINE FIRST=600
PATCH>DEPOSIT/INSTRUCTION FIRST='MOVL R3, R4'
PATCH>UPDATE
PATCH>SET ECO 2
PATCH>DEFINE SECOND=700
PATCH>DEPOSIT/INSTRUCTION SECOND='MOVL R3, R6'
PATCH>UPDATE
PATCH>EXIT
```

In the preceding example, the command procedure PATCOM.COM can be used to patch additional copies of the image file. Because the user-defined symbols are initialized, PATCOM.COM will execute even if ECO level 1 or 2 is already set in the image file to be patched.

1.5 Output Image File

The output image file is a copy of the updated image file. An output image file is always created after you enter the UPDATE command. If you do not enter the UPDATE command, PATCH does not create an output image file.

During a single execution of PATCH, you can apply several patches to an image file. As you terminate each patch (by entering the UPDATE command), PATCH updates the output image file. The first UPDATE command creates the output image file; subsequent UPDATE commands (issued during the current PATCH session) overwrite this file.

By default, the output image file specification has the same file name and type as the input image file and a version number one greater than the highest version of the input image file. By default, the file is created in your process's current default device and directory.

To change the default file specification for the output image file, use the /OUTPUT command qualifier with the PATCH command. This qualifier is described in the Qualifiers section.

1.6 Journal File

The journal file provides a complete record of all PATCH commands entered during the editing of an input image file. A journal file entry is created every time a PATCH command is processed. Comments written during a PATCH session are recorded in the journal file. The journal file provides a record of changes actually made and unsuccessful attempts to edit the input image file.

A journal file is maintained for every PATCH session. If a journal file already exists for a particular file, the new journal entries are appended to it. If no journal file exists, PATCH creates a journal file with the same file name as the input image file name, a file type of JNL, and a version number of 1. By default, the file is created in your process's current default device and directory.

To change the default file specification for the journal file, use the /JOURNAL command qualifier with the PATCH command. This qualifier is described in the Qualifiers section.

Because entries in a journal file are written as ASCII text, you can examine the contents of the journal file by entering the DCL command PRINT or TYPE.

2 Using Symbols

When you use PATCH, you must identify the locations (virtual addresses) that contain the erroneous instructions or data before you can change the code. To do this, define symbolic names for these locations when you create the image file. To make a symbol available for your use, you must include specific information at compile or assembly and link time. For example, to pass local symbols in a MACRO program, include the /DEBUG qualifier when you assemble and link the program.

See the appropriate language user's guide to determine which qualifiers are required to pass symbol information to PATCH. Refer to the *OpenVMS Linker Utility Manual* to determine how to pass universal symbols to PATCH.

2.1 Symbols Recognized by PATCH

PATCH recognizes six types of symbols:

- Global symbols
- Local symbols
- Module names, program section names, and routine names
- Universal symbols
- Symbolic instruction labels
- Symbols defined with the DEFINE command

PATCH also recognizes patch area symbols. See Section 4 for more information about how to use a patch area.

PATCH cannot access local labels in VAX MACRO programs.

2.1.1 Global Symbols

Global symbols are symbols defined in one module that can be accessed in any number of other modules.

Indicate a global symbol in your source program by delimiting it with a special operator. Each language has defined its own special operators. For example, in VAX MACRO, define a global symbol by using double colons (::) or double equal signs (==). See individual language reference manuals for information on defining global symbols.

To pass global symbol information to PATCH, specify the /DEBUG qualifier when you link your program. See the appropriate language manual for complete information on passing global symbols.

2.1.2 Local Symbols

Local symbols are symbols that are defined in a particular module and can be accessed only by that module.

Indicate a local symbol in your source program by delimiting it with a special operator. Each language has defined its own special operators. For example, in VAX MACRO, define a local symbol by using a single colon (:) or an equal sign (=). See individual language reference manuals for information on defining local symbols. To pass local symbol information to PATCH, use the debugger when you assemble or compile and link your program. See the appropriate language manual and the *OpenVMS Linker Utility Manual* for more information on passing local symbols. When you run PATCH, use the SET MODULE command to enter local symbol information into the symbol table. This command is described in the Commands section.

2.1.3 Module Names, Program Section Names, and Routine Names

Symbols that are module names, program section names, and routine names are always passed to PATCH unless you specify the qualifiers /NODEBUG and /NOTRACEBACK at link time. When you specify /NODEBUG and /NOTRACEBACK, no symbol information is passed to PATCH.

To limit symbol information to module names, program section names, and routine names, link your program with the /TRACEBACK and /NODEBUG qualifiers, which are the default qualifiers.

2.1.4 Universal Symbols

Universal symbols are a subset of the global symbols of a shareable image. They are the only type of symbol PATCH can reference when patching a shareable image.

Any global symbol can be a universal symbol. To reduce system overload, define as universal only those symbols required for a particular application.

To define a symbol to be universal, specify the option UNIVERSAL= in the options file when you link your program. For example:

```
$ LINK/SHAREABLE PROGA,PROGB/OPTIONS

                 PROGB.OPT

                     .
                     .
                     .
          UNIVERSAL=A,B,C
                     .
                     .
                     .
```

In this example, the symbols A, B, and C are declared universal in the shareable image PROGA. Universal symbol declarations are stored in the linker options file named PROGB.OPT.

For more information about shareable images and universal symbols, see the *OpenVMS Linker Utility Manual*.

Note

If a symbol is in a transfer vector in a shareable image and has been declared universal by use of the .TRANSFER directive, the value of the symbol in the shareable image's user symbol table (UST) is the address of the transfer vector, not the actual routine address in the image. For PATCH to reference locations symbolically in a non-zero-based shareable image that is position independent, specify the symbol as:

symbol + <base>

2.1.5 Symbolic Instruction Labels

Symbolic instruction labels are 1- to 31-character symbols with the following characteristics:

- They must start with an alphabetic character (labels such as 10$ are invalid for use with PATCH).
- They can consist of alphanumeric characters, the dollar sign ($) character, the underscore (_) character, and the period (.).

A symbolic instruction label lets you associate a specific name with a particular location without knowing the numeric address of that location. The following are situations where symbolic instruction labels would be useful:

- You insert, replace, or deposit a series of instructions into the existing code.
- PATCH places the new instructions in the current patch area and generates branch instructions to and from the patch area to maintain the logical flow of program execution.

- You add new instructions, when one of the new instructions references another new instruction.

In the following example, a group of instructions is inserted after location 350 using a patch area:

```
PATCH>INSERT/INSTRUCTION 350
OLD>'TSTL R3'
NEW>'BEQL NONE'
NEW>'MOVL R4,R0'
NEW>'DIVL2 R0,R1'
NEW>'MULL2 R2,R1'
NEW>'NONE:  RET'
NEW>EXIT
```

In this example, the symbolic instruction label NONE is used in the first instruction to refer to the fifth instruction. The label NONE is used because the exact placement of these instructions is not known.

2.1.6 Impact of Using Symbolic Instruction Labels

Once a symbolic instruction label is assigned a location, the label always refers to that location, even if the contents of the location are moved. For example:

```
           TEST.EXE

address                contents
   .                       .
   .                       .
   .                       .
  500                  TSTL    R3
  502       XX:        BEQL    LBL3
  505       YY:        CLRL    -(R6)
   .                       .
   .                       .
   .                       .
  550                  CLRL    R5
  552       LBL3:      MOVL    R2,R3
   .                       .
   .                       .
   .                       .
  555                  ADDL    R3,R6
```

In this example, the symbolic instruction label LBL3 is assigned to location 552, and the instruction (BEQL LBL3) refers to LBL3. If you discover that the CLRL R5 instruction is wrong and you want to replace it with the instruction CMPL R5,R6, use the REPLACE/INSTRUCTION command.

In this example, the new instruction, CMPL R5,R6, requires more bytes of memory than the old instruction, CLRL R5. Therefore, PATCH automatically moves the CMPL R5,R6 instruction to the current patch area and replaces the CLRL R5 instruction with a branch to the patch area.

However, the branch instruction needed to reroute the logical flow of execution to the patch area also requires more bytes than the CLRL R5 instruction. To make room for the branch instruction, the MOVL R2,R3 instruction, which is the instruction following the CLRL R5 instruction, is also moved to the patch area. MOVL R2,R3 has the symbolic label LBL3 assigned to its location.

Because the MOVL R2,R3 instruction has a symbolic label, use the REPLACE /INSTRUCTION command to replace both the CLRL R5 and MOVL R2,R3 instructions. Then create a new symbolic label for the MOVL R2,R3 instruction, as shown in the example below:

```
PATCH>REPLACE/INSTRUCTION 550
OLD>'CLRL R5'
OLD>'LBL3: MOVL R2,R3'
OLD>EXIT
NEW>'CMPL R5,R6'
NEW>'NEW_LBL3: MOVL R2,R3'
NEW>EXIT
 old:    00000550:   CLRL R5
     old:    LBL3:   MOVL R2,R3
     new:    00000550:    BRW 00007800
     new:    00000553:    NOP
     new:    00000554:    NOP
     new:    00007800:    CMPL R5,R6
     new:    NEW_LBL3:    MOVL R2,R3
     new:    00007806:    BRW 00000555
```

To summarize the replacement operation:

- The CMPL R5,R6 instruction is moved to the current patch area.
- The MOVL R2,R3 instruction is assigned a new symbolic label and is moved to the current patch area.
- A branch instruction to the patch area is inserted into the area left blank by the replacement of the CLRL R5 instruction and removal of the MOVL R2,R3 instruction to the patch area.
- NOP instructions are used to fill the locations not used by the branch instruction.

As a result of this patch, the symbolic instruction label LBL3 still refers to location 552, which contains the following:

```
PATCH>EXAMINE/INSTRUCTION LBL3
LBL3:   MNEGD #01,#01
```

Because LBL3 now points to a different instruction, you need to change previous instructions that refer to LBL3 so that they refer to NEW_LBL3. The MOVL R2,R3 instruction now begins at NEW_LBL3 (location 7803 in the patch area). Change the reference to LBL3, as shown below:

```
PATCH>REPLACE/INSTRUCTION XX
OLD>'BEQL LBL3'
OLD>'CLRL -(R6)'
OLD>EXIT
NEW>'BNEQ LBL2'
NEW>'BRW NEW_LBL3'
NEW>'LBL2: CLRL -(R6)'
NEW>EXIT
```

To correct the reference made to LBL3, the BEQL LBL3 and CLRL -(R6) instructions were replaced with complementary instructions:

```
BNEQ LBL2          Branch if not equal to label LBL2

BRW  NEW_LBL3      Unconditional branch to NEW_LBL3
                   (location 7803)
```

In this example, you need to use the unconditional branch instruction, BRW, to direct the flow of execution to location 7803. The unconditional branch instruction uses a word displacement, which is large enough to hold the amount by which the program counter is increased. The conditional branch instruction, BEQL, uses only a byte displacement, which is not large enough to direct the flow of execution to location 7803.

2.1.7 Symbols Defined with the DEFINE Command

When you run PATCH, you can assign a symbolic name to a location or value by using the DEFINE command. This command lets you supplement or override existing symbols in your image for the duration of the PATCH session. See the DEFINE command in the Commands section for more information.

2.2 The PATCH Symbol Table

PATCH maintains a symbol table for global symbols, local symbols, universal symbols, module names, program section names, routine names, patch area symbols, and symbols defined with the DEFINE command. Use the SET MODULE or SET SCOPE commands to place global or local symbols in the PATCH symbol table after passing them during compiling or linking and assembling. If you did not pass these symbols, PATCH displays one or both of the following informational error messages when it is invoked:

```
%PATCH-I-NOGBL, some or all global symbols are not accessible
%PATCH-I-NOLCL, image file does not contain local symbols
```

To confirm that the symbol table contains the appropriate entries, use the SHOW MODULE command. See the Commands section for more information on the SET MODULE, SET SCOPE, SHOW MODULE, and SHOW SCOPE commands.

Pathnames

Some symbols can have several definitions within your image file. To differentiate the locations defined by these symbols, create unambiguous paths that direct PATCH to those locations. A pathname is a symbolic expression that uniquely defines a location within your file and has the following format:

```
scope\symbol-name±offset-value
```

scope — Identifies the module within your image file where PATCH searches for the symbol name supplied. The backslash character (\ ) is a required delimiter. Supplying a scope is unnecessary under any of the following conditions:

- All of the symbol names within your file are unique.
- The scope has already been set (using the SET SCOPE command) to the module you want to identify.
- The symbol being accessed is a global symbol.

symbol-name — Identifies a particular location within your image file. Symbol names can be global symbols, local symbols, or symbols you defined with the DEFINE command. You must supply the symbol name.

offset-value — Specifies a positive or negative numeric value, in the current radix, appended to the symbol name. The offset value is used to refer to unlabeled locations.

The following example demonstrates the use of a pathname:

```
PATCH>EXAMINE/INSTRUCTION REM_MOD1\CRM_STRT+1A
REM_MOD1\CRM_STRT+1A:  CLRQ R0
```

In this example, the EXAMINE/INSTRUCTION command requests that the contents of location CRM_STRT+1A, in the module named REM_MOD1, be displayed as a VAX MACRO instruction.

2.3 Translating Symbols and Values

PATCH follows a specific sequence of steps to translate an address value to a symbol or a symbol to an address value. The following sections explain the algorithms PATCH uses to compute these translations.

2.3.1 Translating Symbols into Address Values

In PATCH, the translation of symbolic entries into values is controlled by the GLOBALS-NOGLOBALS and SCOPE-NOSCOPE modes. The following steps outline the procedure PATCH uses to perform this translation:

1. PATCH compares the symbolic entry with all user-defined symbols, searching for an exact match.
2. If step 1 fails, PATCH checks the status of the GLOBALS-NOGLOBALS mode setting. If the mode is set to GLOBALS, PATCH compares the symbolic entry as you entered it with the symbols in the symbol table, looking for an exact match. When PATCH searches the symbol table for a match, it does not prefix the contents of the scope to the symbolic entry.
3. If step 2 fails, or if the mode is not set to GLOBALS, PATCH checks the status of the SCOPE-NOSCOPE mode setting. If the mode is set to SCOPE, PATCH prefixes the contents of the scope setting to the symbolic entry and searches the symbol table for an exact match.
4. If step 3 fails, or if the mode is not set to SCOPE, PATCH prefixes the contents of the scope (based on the current PC) to the symbolic entry and searches the symbol table for an exact match.
5. If step 4 fails, PATCH again checks the status of the GLOBALS-NOGLOBALS mode setting. If the mode is set to NOGLOBALS, PATCH compares the symbolic entry as you entered it with the symbols in the symbol table, looking for an exact match. When PATCH searches the symbol table for a match, it does not prefix the contents of the scope setting to the symbolic entry.

If the translation is successful, PATCH evaluates the expression in which the symbolic entry appears. If the translation fails, PATCH issues the following error message:

```
%PATCH-W-NOSYMBOL, no such symbol 'symbolic-entry'
```

2.3.2 Translating Address Values into Symbols

In PATCH, the translation of address values into symbolic entries is controlled by the SYMBOLS-NOSYMBOLS mode. The following steps outline the procedure PATCH uses to perform this translation:

1. PATCH compares the address value with all user-defined symbols for an exact match.
2. If step 1 fails, PATCH compares the address value with the global and local symbol definitions, looking for an exact match.
3. If step 2 fails, PATCH searches all symbol definitions, looking for the one whose corresponding address value is closest to, but less than, the specified value. PATCH rejects a symbol definition as being closest to the address value when the difference between the symbol and the address value is greater than or equal to hexadecimal 100.

If the translation is successful, PATCH evaluates the expression in which the address value appears and reports the address value as the corresponding symbol name. If the translation fails, PATCH still evaluates the expression in which the address value appears but reports the address value as a numeric address in terms of the current radix.

2.4 Commands That Affect Symbols and Pathnames

PATCH supplies the commands listed in Table PAT–2 for dealing with symbols and pathnames.

Table PAT–2 Commands That Affect Symbols and Pathnames

Command	Description
CANCEL MODULE	Removes local symbols and program section names defined in the specified module from PATCH's symbol table
CANCEL SCOPE	Cancels the current symbolic scope and reinstates the <null> scope
DEFINE	Equates a symbolic name with a value
EVALUATE	Displays the current symbolic definition or definitions of a value
SET MODULE	Enters local symbols and program section names defined in the specified module into PATCH's symbol table
SET SCOPE	Defines the current symbolic scope (by module name or routine name)
SHOW MODULE	Displays all the modules in the image file being patched and indicates whether the local symbols contained in those modules are entered in the PATCH symbol table
SHOW SCOPE	Displays the current symbolic scope

See the Commands section for more detailed information on each of these commands.

3 Using Entry and Display Modes

PATCH provides eleven entry and display modes to make referencing locations in an image file more convenient. These modes determine how PATCH interprets commands and displays output. The entry and display modes are grouped into the following categories:

- Context modes, which control whether PATCH accepts and displays information as instructions, ASCII text, or data. These modes also control whether addresses are represented symbolically or numerically.
- Length modes, which control the size, in bytes, of data entries and displays.
- Radix modes, which determine the base in which PATCH displays and interprets addresses and data entries.
- Symbol search modes, which control the way PATCH performs symbol translation.

When you invoke PATCH, the modes are set to NOASCII, NOINSTRUCTION, SYMBOLS, HEXADECIMAL, LONG, NOGLOBALS, and SCOPE. To change these modes, enter the SET MODE and CANCEL MODE commands, or enter an alternative mode qualifier to a PATCH command. Every mode setting has a corresponding mode qualifier. For example, both of the following examples delete an instruction:

Example 1

```
PATCH>SET MODE INSTRUCTION
PATCH>DELETE 400='MOVL (R6),(R5)'
```

Example 2

```
PATCH>DELETE/INSTRUCTION 400='MOVL (R6),(R5)'
```

In Example 1, the INSTRUCTION mode becomes a default setting. In Example 2, the INSTRUCTION mode is a temporary setting effective only for that command line. The next command line uses the current mode settings. Using an alternative mode qualifier on a command line overrides the current default setting.

To examine the status of the current modes, enter the SHOW MODE command.

3.1 Context Modes

The following context modes enable or disable a mode setting:

- INSTRUCTION—NOINSTRUCTION
- ASCII—NOASCII
- SYMBOLS—NOSYMBOLS

3.1.1 INSTRUCTION-NOINSTRUCTION Modes

The INSTRUCTION-NOINSTRUCTION modes control whether data can be entered and displayed as VAX MACRO instructions (see Section 5.2). The default setting is NOINSTRUCTION.

When INSTRUCTION mode is set, PATCH does the following:

- Accepts and displays data as VAX MACRO instructions.
- Accepts and displays full instructions, ignoring the current length mode setting. PATCH increments the current address by the number of bytes occupied by the instruction.
- When you use the REPLACE, INSERT, or DEPOSIT/PATCH_AREA command, PATCH invokes an automatic fitting mechanism that fits symbolic instructions into the locations indicated. If the new instructions are smaller than the available locations, unused bytes are replaced with no operation instructions (NOPs). If the new instructions are longer than the available location, PATCH moves them to the current patch area and inserts branch instructions to and from the patch area.

An instruction string entry must be delimited by quotation marks (") or apostrophes (’). The delimiter must not appear within the string.

When NOINSTRUCTION mode is set, PATCH accepts and displays data according to the current radix, length, and ASCII-NOASCII mode setting.

3.1.2 ASCII-NOASCII Modes

The ASCII-NOASCII modes control whether data can be entered and displayed as ASCII data. The default setting is NOASCII. Use the ASCII-NOASCII modes only when the NOINSTRUCTION mode is set.

When INSTRUCTION mode is set, PATCH ignores the ASCII-NOASCII setting and accepts and displays data as VAX MACRO instructions. When ASCII and NOINSTRUCTION modes are set, PATCH does the following:

- Displays ASCII output strings according to the current length mode.
- Truncates an ASCII input string if the string exceeds the limit imposed on it by the current length mode. PATCH truncates input strings by discarding excessive characters from the right.
- Ignores the current radix mode when accepting or displaying data.

ASCII data entries must be enclosed within quotation marks (") or apostrophes (').

When NOASCII and NOINSTRUCTION modes are set, PATCH accepts and displays data according to the current radix and length mode settings.

3.1.3 SYMBOLS-NOSYMBOLS Modes

The SYMBOLS-NOSYMBOLS modes control whether addresses are displayed by pathnames or symbols, rather than by numeric addresses. The default setting is SYMBOLS. When SYMBOLS mode is set, PATCH displays all locations by their pathnames or symbols and accepts pathnames as input.

When NOSYMBOLS mode is set, PATCH displays all locations by their numeric addresses. If you enter a pathname, PATCH displays that location by its numeric address.

3.2 Length Modes

The modes that specify length are BYTE, WORD, and LONG. They denote how many bytes in memory are available to store a given data item. The default setting is LONG.

PATCH uses the length modes when the NOINSTRUCTION mode is set. When the INSTRUCTION mode is set, PATCH ignores the current length mode.

When NOINSTRUCTION mode is set, PATCH performs the following operations:

- Truncates the most significant bit positions of numeric data that exceed the boundary imposed by the current length mode.
- Sets only the last length mode specified if you enter conflicting length modes on a command line.

3.3 Radix Modes

The modes that specify the radix are OCTAL, DECIMAL, and HEXADECIMAL. They refer to the base by which PATCH translates the entry or display of numeric data and addresses. The default setting is HEXADECIMAL. The following rules apply to the radix modes:

- The radix mode is ignored when the ASCII mode is set.
- Data is displayed according to the current length mode.
- Only the last radix mode specified is set if you enter conflicting radix modes on a command line.

You can change the radix modes by specifying a radix operator (^O, ^D, ^X, ^B) before you enter numeric data. A radix operator controls only the entry that it accompanies; it has no control over the radix where PATCH displays a value.

3.4 Symbol Search Modes

The symbol search modes—SCOPE, NOSCOPE, GLOBALS, and NOGLOBALS—control how PATCH performs symbol translations.

The SCOPE-NOSCOPE modes control whether the contents of the scope setting are prefixed to a symbolic entry when PATCH tries to locate the numeric address represented by the symbolic entry. The default setting is SCOPE.

The GLOBALS-NOGLOBALS modes control the order in which PATCH searches the symbol table when performing symbol-to-value translations. If the GLOBALS mode is set, PATCH starts its search by trying to match the symbolic entry without prefixing the scope. If NOGLOBALS is set, the symbolic entry is used as the last pathname in a search, and the scope is added as a prefix to the symbolic entry. The default setting is NOGLOBALS.

Note that PATCH performs symbol-to-value translations differently, depending on the setting of the SCOPE-NOSCOPE and GLOBALS-NOGLOBALS modes.

4 Using a Patch Area

Applying a patch to an image file frequently requires adding lines of code to the file. To do this, PATCH stores additional lines of code in a storage space called a patch area.

A patch area is read/write virtual memory used to store additional instructions or data entries. There are two types of patch areas: default patch area and user-defined patch area.

A default patch area is created by PATCH and inserted into an image file following the last normal image section. PATCH automatically uses a default patch area unless you specify a user-defined patch area.

A user-defined patch area is a patch area that you define in the source code of your program. It can be located anywhere within the image file. PATCH then uses the area you have defined when you enter the SET PATCH_AREA command during a patch session.

If there is not enough space at the requested location to insert any additional code generated as a result of the patch operation, PATCH automatically inserts the code into the current patch area (either default or user-defined). PATCH then inserts branch instructions to and from the patch area to maintain the correct flow of program execution.

PATCH does not automatically insert new data into the current patch area. You can, however, place data into the current patch area by entering the DEPOSIT/PATCH_AREA command. Note that PATCH does not generate branch instructions to and from the patch area when you use the DEPOSIT/PATCH_AREA command; you must keep track of and document the correct flow of program execution.

4.1 Patch Area Descriptor

Every patch area has a descriptor associated with it. A patch area descriptor is a record of the current status of the patch area. The first longword contains the current size (in unused bytes) of the patch area, and the second longword contains the address of the first free byte of the patch area. You can observe the status of the current patch area by entering the SHOW PATCH_AREA command.

Any time a command that affects patch area is executed, the patch area descriptor is automatically updated to reflect the modifications.

4.2 Patch Area Symbols

PATCH uses the symbols PAA through PAZ to indicate the first free byte in each patch area used during a patch session. At the start of a patch session, PATCH assigns the symbol PAA to the first free byte of the default patch area. PATCH assigns the symbols PAB through PAZ to represent the first free byte of user-defined patch areas. PATCH assigns these symbols in the order in which the user-defined patch areas are accessed during a patch session.

The symbols PAA through PAZ are reserved for use by Digital. You can create your own patch area symbols using the ALIGN command. ALIGN lets you assign symbolic names to the starting address of patches in the default patch area and in all the user-defined patch areas. See the ALIGN command in the Commands section for more information.

4.3 Default Patch Area

The default patch area is automatically supplied by PATCH. Generally, the default patch area is expandable. It is limited by available disk space during PATCH execution and the value of the SYSGEN virtual page count parameter at run time. PATCH allocates the default patch area one page at a time.

When you patch an image file and the patches require the use of a patch area, the default patch is used. However, the default patch area cannot be used to store patches for device driver images and position-dependent shareable images that have been linked with other images.

If you need to patch a device-driver image or a position-dependent shareable image that has been linked with other images, use a user-defined patch area.

4.4 User-Defined Patch Area

A user-defined patch area is an area you define in the source code within the boundaries of the image file. PATCH does not extend the length of the image file to accommodate a user-defined patch area. A user-defined patch area is a fixed size; PATCH cannot increase the size of a user-defined patch area to accommodate additional patches.

Use a user-defined patch area to store additional code when patching device-driver images and position-dependent shareable images.

A device-driver image contains, at the start of the image, a location that defines the total length of the driver. When the device driver is loaded, the driver loader looks at this location to determine the amount of space to allocate to the device-driver. The fact that PATCH extends the device-driver image when default patch area is used means that the total length of the device-driver image increases. However, PATCH does not recognize the location that contains the length of the device-driver, nor can PATCH modify the contents of that location if it extends the device-driver image. Thus, a subsequent running of a device driver that uses default patch area causes the patch area to be truncated.

4.4.1 Creating and Accessing a User-Defined Patch Area

To create a user-defined patch area, define global symbols in your source program to indicate the size of the patch area, the location of the patch area, and, optionally, the location and contents of the patch area descriptor (see Section 4.1). When you enter the SET PATCH_AREA command during a patch session, patches are placed in the user-defined patch area.

The SET PATCH_AREA command can be used with or without the /INITIALIZE qualifier, which creates a patch area descriptor and locates it in the first eight bytes of the patch area. However, the way you define the patch area in your source program should be compatible with the form of the SET PATCH_AREA command you plan to use.

See the SET PATCH_AREA command in the Commands section for more information.

4.4.2 Terminating the Use of a User-Defined Patch Area

Enter the CANCEL PATCH_AREA command, and reset the current patch area to the default patch area to terminate the use of a user-defined patch area.

You can move from one user-defined patch area to another by entering the SET PATCH_AREA command.

4.5 Commands That Affect Patch Area

Table PAT–3 lists the PATCH commands that affect patch area.

Table PAT–3 Commands That Affect Patch Area

Command	Description
ALIGN	Aligns the first free byte of the current patch area on the specified boundary and defines a symbol for that address.
CANCEL PATCH_AREA	Cancels the use of the current user-defined patch area and reverts to the use of the default patch area.
DEPOSIT/PATCH_AREA	Deposits data or VAX MACRO instructions in the current patch area.
INSERT/INSTRUCTION	Inserts one or more VAX MACRO instructions into the current patch area.
REPLACE/INSTRUCTION	Stores new instructions in patch area if they occupy more bytes in memory than the instructions being replaced.
SET PATCH_AREA	Establishes a user-defined patch area as the current patch area. Used with the /INITIALIZE qualifier, this command creates a patch area descriptor and establishes a user-defined patch area as the current patch area.
SHOW PATCH_AREA	Displays the patch area descriptor of the current patch area.

Each of these commands is explained in the Commands section.

5 Rules of Syntax

The following section contains PATCH syntax rules.

5.1 Entering ASCII Data Strings

To enter ASCII data strings, specify either the /ASCII mode qualifier, or set the mode to ASCII.

Delimit ASCII data strings with quotation marks (") or apostrophes (′). The delimiter must not appear within the string. For example, if you want to insert the ASCII text 'AL', you would type the following:

```
PATCH>DEPOSIT/ASCII 500="'AL'"
```

PATCH truncates (from the right) a data string if it exceeds the length imposed on it by the current length mode setting.

5.2 Entering VAX MACRO Instructions

To enter VAX MACRO instructions, specify the /INSTRUCTION mode qualifier, or set the INSTRUCTION mode.

A VAX MACRO instruction string entry must be delimited by quotation marks or apostrophes and not a combination of both, and the delimiter must not appear within the string.

To enter operands with displacements, prefix the operand with B^, W^, or L^. For example:

```
PATCH>DEPOSIT 500='ADDL2 B^4(R0),R8'
```

In this example, PATCH deposits code for this instruction into sequential addresses starting with 500.

All two-operand instructions are displayed as xxx2. For example, ADDF is displayed as ADDF2.

When instructions assembled with forced-immediate-addressing mode (I^#) are displayed, they are displayed as #'xxxxx', where xxxxx is the size of a byte, word, or longword, depending on operand data type.

Table PAT–4 shows the difference between immediate-mode and forced-immediate-mode addressing.

Table PAT–4 Immediate Addressing Modes

Immediate Mode	Forced Immediate Mode
MOVW #1,R0	MOVW I^#1,R0
MOVL #1,R0	MOVL I^#1,R0
.END	.END

See the *VAX MACRO and Instruction Set Reference Manual* for more information.

5.2.1 VAX MACRO Instructions with the Same Opcodes

In VAX MACRO, different instructions can generate the same opcode. For example, MOVAL and MOVAF both generate the hexadecimal opcode DE. PATCH, however, displays only one instruction for any of the opcodes that can be produced by multiple instructions. For example, if you use MOVAF in your source program, PATCH displays MOVAL.

Table PAT–5 lists the instructions and their opcodes.

Table PAT–5 VAX MACRO Instructions with the Same Opcodes

Instructions with Equivalent Opcodes	Instruction Displayed by PATCH	Opcode
BCC BGEQU	BGEQU	1E
BCS BLSSU	BLSSU	1F
BEQL BEQLU	BEQL	13
BNEQ BNEQU	BNEQ	12
CLRD CLRG CLRQ	CLRQ	7C
CLRF CLRL	CLRL	D4
CLRH CLRO	CLRO	7CFD
MOVAD MOVAG MOVAQ	MOVAQ	7E
MOVAF MOVAL	MOVAL	DE
MOVAH MOVAO	MOVAO	7EFD
PUSHAD PUSHAG PUSHAQ	PUSHAQ	7F
PUSHAF PUSHAL	PUSHAL	DF
PUSHAH PUSHAO	PUSHAO	7FFD

5.3 Entering Numeric Data

When you enter numeric data to PATCH, the mode qualifiers /NOINSTRUCTION and /NOASCII must be specified, or the NOINSTRUCTION and NOASCII modes must be set. Do not enclose the data within quotation marks or apostrophes.

You can enter numeric data concurrently with instructions when you are using the assembler directives .BYTE, .WORD, or .LONG. For example:

```
PATCH>DEPOSIT/PATCH_AREA/INSTRUCTION
NEW>'TSTL R4'
NEW>'BEQL LBL3'
NEW>'MOVL B^LBL2,R0'
NEW>'LBL1: RET'
NEW>'LBL2: .LONG 1,0'
NEW>'LBL3: MOVL LBL2+4,R0'
NEW>'BRB LBL1'
```

Precede any alphabetic hexadecimal number (A through F) with a 0 or PATCH will interpret the hexadecimal number as a symbol. For example, enter the numeric string FFA as shown in this example:

```
PATCH>REPLACE 200=000000FA
NEW>0FFA
NEW>EXIT
old:     000000FA
new:     00000FFA
```

5.4 Delimiting Parameter Values

Use the equal sign (=) to separate an address expression from a data entry or to separate a symbol name from a value. The commands DEFINE, DELETE, DEPOSIT, INSERT, REPLACE, and VERIFY require an equal sign unless you enter their parameters after PATCH prompts.

Use a comma to separate parameter values in commands with multiple parameters.

5.4.1 Entering Comments

To insert comments, prefix them with the exclamation point (!) character. When PATCH reads an exclamation point, it ignores the remaining information on that line.

A comment cannot appear on the same line as a command string that uses a continuation character (-). Comments are entered into the journal file.

5.5 Special Operators for Arithmetic Expressions

Table PAT–6 lists the special operators that PATCH recognizes in arithmetic expressions.

Table PAT–6 Special Operators for Arithmetic Expressions

Operator	Name	Function
+	Addition operator	Unary plus sign or addition operator in arithmetic expressions
–	Subtraction operator	Unary minus sign or subtraction operator in arithmetic expressions
*****	Multiplication operator	Multiplication operator in arithmetic expressions
/	Division operator	Division operator in arithmetic expressions
@	Shift operator	Arithmetic shift operator
<>	Priority operators	Priority operators or bit field delimiters for the EVALUATE command
^H	Radix operator	Radix operator for hexadecimal notation
^D	Radix operator	Radix operator for decimal notation
^O	Radix operator	Radix operator for octal notation
^B	Radix operator	Radix operator for binary notation

5.6 Special Operators for Addressing Locations

Table PAT–7 describes the special operators that PATCH recognizes for locating addresses.

Table PAT–7 Special Operators for Addressing Locations

Operator	Name	Function
.	Current location operator	Refers to the location last specified (the current location). This value remains the same until you reference a new location.
^	Previous location operator	Refers to the location that precedes the last location specified. To calculate, subtract the number of bytes indicated by the length mode from the current location.
:	Range operator	Used to specify a range of locations for the CHECK ECO, CHECK NOT ECO, EVALUATE, and EXAMINE commands.
\	Backslash operator	Refers to either the contents of the location specified in a branch instruction or the value stored in an address specified in the previous EXAMINE command.

PATCH Usage Summary

The VMS Patch Utility (PATCH) allows you make changes to an image file and run the new version of the image without having to compile, assemble, or link the program source files.

Format

PATCH file-spec

Command Parameter

file-spec
Specifies the name of the image file to be patched or a command procedure that contains both the name of the image file to be patched and PATCH commands.

If the file specification denotes an image file, it must include the file name. If you omit the remaining fields (device, directory, file type, and version number), PATCH uses your default device and directory, assumes a file type of EXE, and uses the highest version of the specified file.

If the file specification denotes a command procedure, the file-spec parameter must be preceded by an at sign (@). Only the file name is required. If you omit the remaining fields (device, directory, file type, and version number), PATCH uses your default device and directory, assumes a file type of COM, and locates the highest version of the command procedure.

No wildcard characters are allowed in the file specification.

Usage Summary

To invoke PATCH, use the PATCH command. To end a PATCH session, enter the EXIT command or press CTRL/Z. To direct output, use the /OUTPUT qualifier with the PATCH command.

PATCH Qualifiers

This section describes qualifiers to the PATCH command that allow you to override default file specifications and process selected patches in command procedures.

/ABSOLUTE

Patches a file at absolute virtual addresses.

Format

/ABSOLUTE

Description

The /ABSOLUTE function allows a user to patch any file (not just image files) at virtual addresses relative to the beginning of the file. This feature allows replacement of existing data with new data needing the same or less space. If the data takes less space than that of the original data, PATCH uses the appropriate fill character for the mode in use. For example, if the current mode is instruction mode, a NOP is used for fill; if it is data (numeric or ASCII) mode, a NULL is used for fill. If the data to be patched requires more space than the data it is replacing, an error message is generated, and the command in progress terminates. You return to the PATCH or DCL level.

Note

There is no default patch area created because of the tendency to corrupt a file. Patch area is meaningless in all files other than an image file.

If you patch a file in absolute mode, remember that there are no symbols available to assist you in locating data locations. Make sure that you modify the data in the correct locations.

Although most PATCH commands work in the manner described, some precautions should be taken when performing certain functions. Use REPLACE and DEPOSIT for write operations; other commands are acceptable for read operations. Avoid commands that attempt to expand the file, such as ALIGN and INSERT, because they will probably corrupt the file. These commands are trapped by PATCH; an error message will be issued indicating that the replacement data must not exceed the length of the original data.

File attributes are propagated from the original input file to the output file. These include ALQ, TYPE, MRS, RAT, RFM, and RAC.

Example

```
$ PATCH/ABSOLUTE IMAGE.EXE
PATCH>EX/INS 604
00000604:  BBSS    #07,R1,00000608
PATCH>REPLACE/INS 604='BBSS #07,R1,0608'
NEW>  'BBSS #07,R1,0608'
NEW>  'CLRL R0'
NEW>  EXIT
old:        00000604:  BBSS    #07,R1,00000608
%PATCH-E-DATTOOLNG, length of new data may not exceed length of old data
PATCH>EX/INS 684
00000684:  MOVB    #01,(R5)+
```

```
PATCH>REPLACE/INS 684='MOVB #01,(R5)+'
NEW>  'MOVB #02,(R5)+'
NEW>  EXIT
old:        00000684:  MOVB    #01,(R5)+
new:        00000684:  MOVB    #02,(R5)+
PATCH>EX/INS 687
00000687:  MOVB    #00,(R5)+
PATCH>DEPOSIT/INS 687
NEW>  'CLRB (R5)+'
NEW>  EXIT
old:        00000687:  MOVB    #00,(R5)+
new:        00000687:  CLRB    (R5)+
PATCH>INSERT/INS 68D
OLD>  'MOVB #10,(R5)+'
NEW>  'MOVB #20,(R5)+'
NEW>  EXIT
old:        0000068D:  MOVB    #10,(R5)+
%PATCH-E-DATTOOLNG, length of new data may not exceed length of old data
PATCH>UPDATE
%PATCH-I-WRTFIL, updating image file DISK$STARWORK01:[NASR.PATCH]IMAGE.EXE;2
PATCH>EXIT
```

This example of the /ABSOLUTE qualifier uses /NEW_VERSION as the default. Note the error messages returned when the command tries to expand the file. The example for the /NEW_VERSION qualifier in the Qualifiers section shows the use of /ABSOLUTE with /NONEW_VERSION.

/JOURNAL

Overrides the default journal file specification.

Format

/JOURNAL [= file-spec]

Parameters

file-spec
Specifies the alternate journal specification. If you omit fields in the file specification, PATCH supplies the following default values:

Field	Default Value
Device and directory	Defaults of current process
File name	Name of input image file
File type	JNL
Version	1

No wildcard characters are allowed in the file specification.

Subsequent PATCH sessions append information to the journal file, rather than creating a new version of this file.

Description

By default, PATCH creates a journal file with a file specification that consists of the current defaults. Use the /JOURNAL qualifier when you want to specify an alternate file specification.

Example

```
$ PATCH AVERAGE/JOURNAL=DB1:[JACKSON]TEST
```

In this example, PATCH is invoked for an interactive session with the image file AVERAGE.EXE. The /JOURNAL qualifier requests that the journal file be named TEST.JNL;1 and that it be created in the DB1:[JACKSON] directory.

/NEW_VERSION

Controls whether a new version of the patched file is created, or the contents of the existing file are modified in place.

Format

/[NO]NEW_VERSION

Description

The /NEW_VERSION qualifier is used in conjunction with the /ABSOLUTE qualifier to control whether a new version of the patched file is created, or the contents of the existing file are modified in place. /NEW_VERSION is the default. If /NONEW_VERSION is selected, the PATCH command UPDATE will act as a checkpoint operation. All modifications made to the file are written back to the file instead of waiting until image exit. If /ABSOLUTE is not specified with /NONEW_VERSION, /NONEW_VERSION is ignored. A new version of the file is created. Use /NONEW_VERSION when patching large data files, and there is not enough disk space to create a new version of the patched file.

Note

If you specify /NONEW_VERSION, your file is permanently overwritten. Make sure you have a backup copy of the file before you patch it.

PATCH always issues an informational message at image exit, indicating that the file is being overwritten.

Example

```
$ PATCH/ABSOLUTE/NONEW_VERSION  LOGIN.COM
PATCH>EX/ASCII 57
00000057:  'MANA'
PATCH>REPLACE/ASCII 57='MANA'
NEW>  'mana'
NEW>  'test'
NEW>  exit
old:        00000057:  'MANA'
%PATCH-E-REPLACEERR, replacement value too large for location
PATCH>replace/ascii 57='MANA'
NEW>  'mana'
NEW>  exit
old:        00000057:  'MANA'
new:        00000057:  'mana'
PATCH>EX/ASCII 24
00000024:  'F$MO'
PATCH>INSERT/ASCII 24='F$MO'
NEW>  'test'
NEW>  exit
%PATCH-E-INVCMD, invalid command
PATCH>UPDATE
%PATCH-I-OVERLAY, DISK$STARWORK01:[NASR.PATCH]LOGIN.COM;1 being overwritten
PATCH>EX 68:75
00000068:  4349544F
0000006C:  58542E45
00000070:  00010054
00000074:  00100024
```

```
PATCH>REPLACE 68
OLD>  4349544F
OLD>  58542E45
OLD>  00010054
OLD>  EXIT
NEW>  6369746F
NEW>  68642E65
NEW>  00010074
NEW>  EXIT
old:        00000068:  4349544F
old:        0000006C:  58542E45
old:        00000070:  00010054
new:        00000068:  6369746F
new:        0000006C:  68642E65
new:        00000070:  00010074
PATCH>EX/ASCII 68
00000068:  'otic'
PATCH>UPDATE
%PATCH-I-OVERLAY, DISK$STARWORK01:[NASR.PATCH]LOGIN.COM;1 being overwritten
PATCH>EXIT
%PATCH-I-OVERLAY, DISK$STARWORK01:[NASR.PATCH]LOGIN.COM;1 being overwritten
$
```

This is an example of a PATCH/ABSOLUTE/NONEW_VERSION command. Note the error messages that are returned when the command tries to expand the file and when the commands UPDATE and EXIT are performed. The example for the /ABSOLUTE qualifier in the Qualifiers section shows the use of /ABSOLUTE with /NEW_VERSION as the default.

/OUTPUT

Overrides the default output image file specification.

Format

/OUTPUT [= file-spec]

Parameters

file-spec
Indicates the output image file specification.

If you omit fields in the file specification, PATCH supplies the following default values:

Field	Default Value
Device and directory	Defaults of current process
File name	Name of input image file
File type	EXE
Version	One greater than the most recent copy of the input image file

No wildcard characters are allowed in the file specification.

Description

By default, PATCH creates an output file with a file specification that consists of the current defaults. Use the /OUTPUT qualifier when you want to specify an alternate file specification.

The output image file is created only when you enter the PATCH command UPDATE at the end of the PATCH session. You can enter multiple UPDATE commands in a single session. The first UPDATE command creates the output image file; subsequent UPDATE commands overwrite this file.

Example

```
$ PATCH PAYROLL/JOURNAL=TESTER/OUTPUT=TESTER
```

This command invokes PATCH for an interactive PATCH session with the image file PAYROLL.EXE. The journal file and the output image file created by this session are named TESTER.JNL;1 and TESTER.EXE respectively and reside in the current default device and directory.

/UPDATE

Processes only those patches associated with the specified ECO levels.

Format

/UPDATE [= (eco-level [, . . .])]

Parameters

eco-level
Specifies the ECO level of the patches. If you specify more than one ECO level, separate the levels with commas, and enclose the list in parentheses.

Description

When you use the /UPDATE command qualifier, only the patches corresponding to the ECO levels that you specify with /UPDATE are applied. Use the /UPDATE qualifier when processing a PATCH command procedure or when you invoke PATCH for an interactive patching session.

When you specify the /UPDATE qualifier, the PATCH command file specification denotes either a command procedure that contains the patches to be processed or an image file to which certain patches are to be applied. When the file specification denotes a command procedure, the /UPDATE qualifier must precede the file specification on the command line. When the file specification denotes an image file, the /UPDATE qualifier can precede or follow the file specification. The file specification is required in both cases.

If PATCH encounters an ECO level in a command procedure that does not match the ECO level that you specified on the /UPDATE qualifier, PATCH ignores the ensuing patch and displays a message. When you omit the optional ECO levels, PATCH processes all patches submitted.

Processing Selected Patches in Command Procedures

PATCH uses a file as a command procedure if the file specification is preceded by an at sign (@). PATCH reads the command procedure and parses every command. However, PATCH executes only those commands in patches corresponding to the ECO levels specified with the /UPDATE qualifier. The first entry in the command procedure is always the name of the image file to which the patches are to be applied.

Do not use the /UPDATE qualifier to apply selected patches if the command procedure contains user-defined symbolic references. If PATCH is unable to resolve symbolic references during parsing, PATCH terminates the command procedure.

Use ECO commands to apply selected patches in command procedures that contain user-defined symbols.

Note that when you execute command procedures, the /UPDATE qualifier must precede the name of the command procedures.

If you do not include the /UPDATE qualifier, PATCH applies all patches in the command procedure unless it encounters an ECO level that has already been applied to the image file.

When PATCH executes the command procedure, it displays the following information:

- Name of the image file to be patched
- PATCH introductory message
- Patches applied to the image file
- Relevant ECO error messages
  - ECO levels specified with /UPDATE, but already applied to the image file
  - ECO levels present in the command procedure, but not specified with /UPDATE

The first example below shows how PATCH executes selected patches in the command procedure TEST1.COM.

Examples

1.
```
$ PATCH/UPDATE=(10,12) @[JOHNSON.FORTRAN]TEST1.COM

PATCH   VERSION 4-00   15-Apr-1984

DRM1:  0EFDE4800
old:  DRM1:  0EFDE4800
new:  DRM1:  00000000
%PATCH-I-WRTFIL, updating image file DB1:[MATTHEWS.FORTRAN]PROGB.EXE;8
%PATCH-I-UPDATE, patch with eco 11 ignored due to update qualifier
AMSTR+50:  6B5C4005
old:  AMSTR+50:  6B504005
new:  AMSTR+50:  00000000
%PATCH-I-WRTFIL, updating image file DB1:[MATTHEWS.FORTRAN]PROGB.EXE;8
$
```

   In Example 1, the PATCH command executes the command procedure TEST1.COM, which resides in the directory [JOHNSON.FORTRAN]. The /UPDATE qualifier requests that only the patches identified by the ECO levels 10 and 12, contained in TEST1.COM, be applied to the image file PROGB.EXE in the directory [MATTHEWS.FORTRAN]. The output indicates that the patches specified with /UPDATE were successfully applied to PROGB.EXE and reports that the patch identified by ECO level 11 was not applied because that ECO level was not specified with the /UPDATE qualifier.

2.
```
$ PATCH/UPDATE=(2,4) CIRCLE
PATCH  VERSION 4-00 15-Apr-1984
PATCH>SET ECO 2
   .
   .
   .
PATCH>UPDATE
   .
   .
   .
PATCH>SET ECO 3
%PATCH-I-UPDATE, patch with eco level 3 ignored due to update qualifier
PATCH>SET ECO 4
```

When you include the /UPDATE qualifier with the input image file, PATCH processes only those patches represented by the ECO levels specified with /UPDATE. If, while editing the image file, you define an ECO level not specified with the /UPDATE qualifier, PATCH ignores the subsequent commands and displays an informational error message.

In Example 2, PATCH allows processing of the patch identified by the ECO level 2. When you try to define ECO level 3, PATCH displays a message indicating that the patch cannot be applied because it was not specified with /UPDATE.

Note that if you specify an ECO level with /UPDATE but do not set that ECO level during the PATCH session, PATCH issues an informational error message when you exit.

/VOLUME

Places the output file in a specified relative volume number of a multivolume set.

Format

/VOLUME =[n]

Parameters

n
Specifies the relative volume number of a multivolume set.

Description

If you specify /VOLUME without a number, the default is the relative volume number of the input image file. If you do not specify the /VOLUME qualifier, the output image file is placed in the first available position within the multivolume set.

Example

```
$ PATCH/VOLUME=2 PAYROLL.EXE
```

This command creates an output file in relative volume 2.

PATCH Commands

This section describes PATCH commands.

ALIGN

Allocates space for the default patch area if the image file has not been patched previously.

If the image file has been patched previously, the ALIGN command advances the starting address of the current patch area to the first free byte aligned on the requested boundary (byte, word, longword, quadword, or page) and equates a symbolic name to that address. Once you define the symbolic name, you can use it in place of the address it denotes.

Format

ALIGN symbol-name

Parameters

symbol-name
Specifies a 1- to 31-character symbol. It must start with an alphabetic character and consist of alphanumeric characters, dollar signs ($), underscores (_), or periods (.).

If you specify a symbol name for an address that has been previously assigned a symbol name, the newest symbol name takes precedence.

Qualifiers

/BYTE
Defines the symbol as the first free byte of the current patch area. If the current patch area has not been used previously, PATCH allocates the first block of default patch area.

/WORD
Advances the starting address of the current patch area to the first free word boundary.

/LONG
Advances the starting address of the current patch area to the first free longword boundary.

/QUAD
Advances the starting address of the current patch area to the first free quadword boundary.

/PAGE
Advances the starting address of the current patch area to the first free page boundary.

Description

When you specify ALIGN, none of the patch area between the old patch area address and the aligned patch area address is cataloged by PATCH. You must keep a record of the previous patch area if you might need to reference it later.

After an alignment has been made, the patch area string descriptor is updated to reflect the modifications. If the patch area is already aligned on the specified boundary, no update is performed.

Enter only one alignment qualifier for each ALIGN command.

Examples

1.
```
PATCH>ALIGN/QUAD MOD_1
old patch area size:          000001E3
old patch area address:       0000081D
new patch area size:          000001E0
new patch area address:       00000820
symbol "MOD_1" defined as     00000820
```

   In Example 1, the ALIGN command requests that the patch area starting address be advanced to the first free quadword boundary in the current patch area and that the symbol MOD_1 be equated with that address. The display shows the old and new patch area status.

2.
```
PATCH>ALIGN/BYTE PATAREA
old patch area size:          00000000
old patch area address:       00000000
new patch area size:          00000200
new patch area address:       00001800
symbol "PATAREA" defined as   00001800
```

   In Example 2, the ALIGN command allocates the first block for the default patch area and assigns the symbol PATAREA to its new starting address.

CANCEL MODE

Cancels the mode settings that you defined using the SET MODE command.

Format

CANCEL MODE

Parameters

None.

Qualifiers

None.

Description

Use CANCEL MODE to control the syntax of commands you enter and the values PATCH displays. CANCEL MODE cancels the current mode settings and reinstates the initial default mode settings NOINSTRUCTION, NOASCII, SYMBOLS, HEXADECIMAL, LONG, NOGLOBALS, and SCOPE.

Example

```
PATCH>SHOW MODE
modes:  nosymbols, instruction, ascii, scope, globals, decimal word
PATCH>CANCEL MODE
PATCH>SHOW MODE
modes:  symbols, noinstruction, noascii, scope, noglobals, hexadecimal long
```

The first SHOW MODE command displays the current mode status. The CANCEL MODE command requests that the initial default mode settings be reinstated. The second SHOW MODE command confirms that the initial default settings have been reestablished.

CANCEL MODULE

Removes local symbol information from the PATCH symbol table.

Format

CANCEL MODULE module-name [, . . .]

Parameters

module-name
Specifies the name of one or more modules whose local symbols are to be removed from the symbol table.

Do not specify a module name if you include the /ALL qualifier.

Qualifiers

/ALL
Removes all local symbol information from the symbol table.

Description

CANCEL MODULE does not remove global symbols, patch area symbols, universal symbols, or symbols you defined with the DEFINE command. Once the module's local symbols have been removed, they cannot be used to reference locations. To reenter local symbols into the PATCH symbol table, enter the SET MODULE command.

To remove all local symbol information from the symbol table, specify the /ALL qualifier.

If the scope is the name of the module that you specify with the CANCEL MODULE command, the contents of the scope is reset as an empty string (<null>).

Examples

1.
```
PATCH>SHOW MODULE
module name            symbols  size
LOVAT                  yes      180.

total modules:             1.
remaining size:            EEA4.
PATCH>CANCEL MODULE
NAM>LOVAT
NAM>EXIT
```

In Example 1, the SHOW MODULE command indicates that the local symbols contained in the module named LOVAT are entered in the symbol table. The CANCEL MODULE command purges the symbol table of the local symbols contained in LOVAT.

2.
```
PATCH>CANCEL MODULE/ALL
```

In Example 2, the CANCEL MODULE command removes all local symbol names entered in the symbol table.

3.
```
PATCH>SET SCOPE CIRCLE
    .
    .
    .
PATCH>CANCEL MODULE CIRCLE
PATCH>SHOW SCOPE
SCOPE: <null>
```

In Example 3, the SET SCOPE command establishes the module named CIRCLE as the current scope setting. Later, the CANCEL MODULE command removes all local symbols in CIRCLE from the symbol table and, at the same time, sets the scope to the empty string (<null>). The SHOW SCOPE command confirms the new scope setting.

CANCEL PATCH_AREA

Resets the current patch area from a user-defined patch area to the default patch area.

Format

CANCEL PATCH_AREA

Parameters

None.

Qualifiers

None.

Description

Use CANCEL PATCH_AREA when you have entered the SET PATCH_AREA command to establish a user-defined patch area as the current patch area. After you have inserted the necessary patch information into that area, type CANCEL PATCH_AREA to resume use of the default patch area. Required patch area is taken from the default patch area until the next SET PATCH_AREA command is issued.

You must specify the CANCEL PATCH_AREA command if you have defined, and are using, a user-defined patch area that is too small to store the patches you want to insert. Failing to use the CANCEL PATCH_AREA command causes an error message.

Example

```
PATCH>SET PATCH_AREA AREA1
    .
    .
    .
PATCH>CANCEL PATCH_AREA
```

In this example, the SET PATCH_AREA command establishes the user-defined patch area AREA1 as the current patch area. After entering the necessary data into AREA1, the CANCEL PATCH_AREA command is entered, and use of the default patch area resumes.

CANCEL SCOPE

Cancels the current symbolic scope.

Format

CANCEL SCOPE

Parameters

None.

Qualifiers

None.

Description

Use CANCEL SCOPE to cancel the current symbolic scope and revert to an empty string (<null>).

Example

```
PATCH>SHOW SCOPE
SCOPE: MOD1
PATCH>CANCEL SCOPE
PATCH>SHOW SCOPE
scope: <null>
```

In this example, the first SHOW SCOPE command indicates that the scope is set to the module named MOD1. The CANCEL SCOPE command cancels the scope setting and reverts to the empty string (<null>). The second SHOW SCOPE command confirms that the contents of the scope is an empty string.

CHECK ECO

Verifies that the patches represented by the specified ECO levels have been applied. Use this command before applying a patch.

Format

CHECK ECO eco-level [:eco-level] [, . . .]

Parameters

eco-level
Indicates one or more ECO levels that have been set. Enter ECO levels in lists separated by commas or ranges separated by colons.

You can specify lists and ranges in the initial command. However, only one ECO level or one range of ECO levels can be entered in response to an ECO level prompt (ECO>).

Qualifiers

None.

Description

Use CHECK ECO to check that one or more ECO levels have been set before applying a patch. If a specified ECO level has not been set in the image file, PATCH does not apply the current patch. PATCH does not execute any subsequent commands until it encounters the next SET ECO command.

Remember that an ECO level is not set until the patch that it represents is applied by the UPDATE command. For example, if you define an ECO level with the SET ECO command, then immediately check to see whether the ECO level is set, PATCH returns an error message indicating that the ECO level is not set.

When you enter the CHECK ECO command, you do not have to list all the ECO levels that have been set for a particular image file. However, specifying one or more ECO levels that have not been set will produce an error message.

Examples

1.
```
PATCH>SET ECO 18
PATCH>CHECK ECO   3:8,14,16
PATCH>
```

In Example 1, the CHECK ECO command checks that the patches associated with ECO levels 3, 4, 5, 6, 7, 8, 14, and 16 have been applied to the image file. The third PATCH prompt indicates that the specified patches have been set; PATCH will continue to execute commands. The patch associated with ECO level 18 will be applied after the UPDATE command is entered.

2.
```
PATCH>SET ECO 14
PATCH>CHECK ECO 12
%PATCH-E-ECONOTSET, eco level 12 not set in DB2:[HARINGTON]BIND_NOW.EXE;4
PATCH>CHECK ECO 12
PATCH>EXAMINE/INSTRUCTION 800
PATCH>SET ECO 13
PATCH>EXAMINE/INSTRUCTION 800
00000800: TSTL R5
```

In Example 2, the first CHECK ECO command checks that the patch associated with ECO level 12 has been applied to the image file. The error message indicates that ECO level 12 has not been set and that the patch it represents has not been applied to the image file BIND_NOW.EXE. Therefore, the current patch (represented by ECO level 14) will not be applied. No further commands are executed until the next SET ECO command.

CHECK NOT ECO

Verifies that the specified ECO levels have not been applied and are available for use.

Format

CHECK NOT ECO eco-level [:eco-level] [, . . .]

Parameters

eco-level
Indicates unset ECO levels. Enter ECO levels in lists separated by commas or in ranges separated by colons.

You can enter lists and ranges in the initial command. However, only one ECO level or one range of ECO levels can be entered in response to an ECO level prompt (ECO>).

Qualifiers

None.

Description

Use CHECK NOT ECO to check that one or more ECO levels is available for use in a particular image file.

The CHECK NOT ECO and the CHECK ECO commands determine opposite conditions. Both can be used to confirm whether or not a particular patch has been applied to an image file. Usually the CHECK NOT ECO command is used to confirm that the stated ECO levels have not been set and are available for use in the current image file.

If a specified ECO level is not available for use because it has already been set in the image, the current patch is not applied. PATCH does not execute any subsequent commands until it encounters the next SET ECO command.

Examples

1.
```
PATCH>CHECK NOT ECO   4:6,10
PATCH>
```

In Example 1, this command confirms that the ECO levels 4, 5, 6, and 10 are available for use. The second PATCH prompt indicates that the specified patches have not been applied, and PATCH will continue to execute commands.

2.
```
PATCH>CHECK NOT ECO
ECO>17
ECO>EXIT
%PATCH-E-ECOSET, eco level 17 already set in DB1:[REAVER]MYFILE.EXE;7
```

In Example 2, PATCH responds to the CHECK NOT ECO command by indicating that ECO level 17 has already been used in the image file MYFILE.EXE. No further commands are executed until the next SET ECO command.

CREATE

Creates a command procedure that contains all subsequent successfully executed PATCH commands that modify the image file.

Format

CREATE [file-spec]

Parameters

file-spec
Represents the file specification of the command procedure.

You can omit all or some of the fields in the command procedure file specification. PATCH uses the following default values for omitted fields:

Field	Default Value
Device and directory	The current default device and directory for the process
File name	The name of the input image file
File type	COM
Version	1 greater than the highest command procedure of the same name

Note

If you store a command procedure in a directory other than the one that contains the input image file to which the command procedure is applied, set your default to the directory that contains the input image file before you process the command procedure.

Qualifiers

None.

Description

Command procedures facilitate patching several copies of the same image file. Do the following to create command procedures:

- Specify the CREATE command when you invoke PATCH. All subsequent successfully executed commands are applied to the image file and are recorded in the command procedure.
- Use a text editor.

When you use CREATE, PATCH automatically inserts the name of the image file that uses the patches as the first entry in the command procedure. Symbolic names are converted to absolute values. Command names and qualifiers are truncated to their shorthand notation.

You can enter only one CREATE command for each PATCH session. To create another command procedure, close and reopen the input image file.

To process the patches contained in the command procedure, enter the following DCL command:

```
$ PATCH @file-spec
```

In the above command, the file-spec parameter represents the specification of the command procedure containing the patches.

Example

```
$ PATCH AVERAGE

PATCH VERSION 4-00 15-Apr-84

PATCH>CREATE PAT2
PATCH>SET ECO 1
PATCH>SET MODE NOSYMBOLS
PATCH>EXAMINE 600
00000600:     12345678
PATCH>DELETE 600 = 12345678
old:     00000600:     12345678
new:     00000600:     00000000
PATCH>UPDATE
%PATCH-I-WRTFIL, updating image file DBA2:[BRADLEY]AVERAGE.EXE;6
PATCH>EXIT
$ SET DEFAULT [NIMROD]
$ SHOW DEFAULT
DBA2:[NIMROD]
$ PATCH @[BRADLEY]PAT2
```

In this example of a command procedure, PATCH is invoked to patch the image AVERAGE.EXE in the DBA2:[BRADLEY] directory. The CREATE command creates the command procedure PAT2.COM in which all successful commands are stored.

After the PATCH session ends, the default directory is set to DBA2:[NIMROD], and the patches in the command procedure PAT2.COM are applied to the image file [NIMROD]AVERAGE.EXE.

DEFINE

Assigns a specific value to a symbolic name, and then places the symbolic name in the PATCH symbol table.

Format

DEFINE symbol-name = value [,symbol-name = value, . . .]

Parameters

symbol-name
Specifies a 1- to 31-character user-defined symbol to be associated with the specified value. The symbol name must start with an alphabetic character and can consist of alphanumeric characters, dollar signs ($), underscores (_), and periods (.). PATCH is not case sensitive.

The symbol name cannot be a pathname.

value
Specifies a numeric address or symbolic expression that is to be assigned the specified symbolic name.

Qualifiers

None.

Description

After an assignment has been performed using the DEFINE command, you can specify the symbolic name in place of the value it denotes for the duration of the PATCH session. At the end of the PATCH session, these user-defined symbols are deleted from the PATCH symbol table.

When you use the DEFINE command to create symbolic names, PATCH searches the symbol table for these symbolic names when it translates a symbol into a value.

More than one symbolic name can be assigned to a single value. Each symbolic name is recorded in the symbol table and can subsequently be used to reference the value it denotes.

You can redefine a symbolic name to represent a new value. When you specify that symbolic name, the most recent value is displayed.

You cannot specify the /INSTRUCTION or /ASCII mode qualifiers or set the INSTRUCTION or ASCII modes when equating a symbol name to a value.

Examples

1.
```
PATCH>DEFINE SWEDISH = CAR\VOLVO+144
Symbol "SWEDISH" defined as CAR\VOLVO+144
PATCH>EXAMINE SWEDISH
```

In Example 1, the DEFINE command creates a symbolic name for the value CAR\VOLVO+144. A subsequent EXAMINE command requests to see the contents of CAR\VOLVO+144 using the symbol name SWEDISH.

2.
```
PATCH>DEFINE
NAM>SECOND_CHOICE
VAL>406
NAM>EXIT
symbol "SECOND_CHOICE" redefined from 408 to 406
```

In Example 2, the DEFINE command reassigns the symbol SECOND_CHOICE from the value 408 to the value 406. The DEFINE command response shows the reassignment.

DELETE

Deletes an instruction or piece of data from one location or from several consecutive locations according to the current mode settings.

Format

DELETE location = current-contents [, . . .]

Parameters

location
Specifies either a single location whose contents are to be deleted or the starting address of a sequence of locations whose contents are to be deleted. The length of the sequence depends on the current mode settings.

current-contents
Specifies one or more data entry or instruction to be deleted. The data or instructions you specify must appear exactly as written in the instruction to be deleted.

Do not specify conflicting data types within a single DELETE command.

Qualifiers

/BYTE
Specifies that data is deleted in bytes. The length mode qualifiers affect only ASCII and numeric data. When you specify an instruction, the DELETE command ignores the current length setting and deletes the entire instruction.

/WORD
Specifies that data is deleted in words. The length mode qualifiers affect only ASCII and numeric data. When you specify an instruction, the DELETE command ignores the current length setting and deletes the entire instruction.

/LONG
Specifies that data is deleted in longwords. The length mode qualifiers affect only ASCII and numeric data. When you specify an instruction, the DELETE command ignores the current length setting and deletes the entire instruction.

The initial default setting is /LONG.

/OCTAL
Specifies that virtual addresses and numeric data are interpreted and displayed using octal radix. The radix mode qualifiers do not affect symbolic addresses or ASCII data.

/DECIMAL
Specifies that virtual addresses and numeric data are interpreted and displayed using decimal radix. The radix mode qualifiers do not affect symbolic addresses or ASCII data.

/HEXADECIMAL
Specifies that virtual addresses and numeric data are interpreted and displayed using hexadecimal radix. The radix mode qualifiers do not affect symbolic addresses or ASCII data.

The initial default setting is /HEXADECIMAL.

/[NO]ASCII
Controls whether the data to be deleted is interpreted and displayed as ASCII data.

When you specify /ASCII, delimit the data with quotation marks (") or apostrophes (′). The current radix setting has no effect on the ASCII data being deleted. The DELETE command truncates the data if it exceeds the length imposed on it by the current length mode.

The initial default setting is /NOASCII.

/[NO]INSTRUCTION
Controls whether the data to be deleted is interpreted and displayed as VAX MACRO instructions.

When you specify /INSTRUCTION, delimit the instruction with quotation marks (") or apostrophes (′). The current length setting does not affect the instruction being deleted.

The initial default setting is /NOINSTRUCTION.

[/NO]SYMBOLS
Controls whether locations are displayed as pathnames or symbols, rather than as numeric addresses.

The initial default setting is /SYMBOLS.

/[NO]GLOBALS
Controls whether the symbolic entry is used, exactly as entered, as the first or last pathname in a search.

The initial default setting is /NOGLOBALS.

/[NO]SCOPE
Controls whether scope's contribution to a pathname is used to find the location specified.

The initial default setting is /SCOPE.

Description

When you use the DELETE command to delete instructions, the instructions are replaced with NOP instructions. When you use the DELETE command to delete ASCII and numeric data, the data is replaced with zeros.

Examples

1.
```
PATCH>DELETE/INSTRUCTION   2112 = 'CMPB (R0),(R5)'
old: 00002112: CMPB (R0),(R5)
new: 00002112:   NOP
new: 00002113:   NOP
new: 00002114:   NOP
```

   In Example 1, the DELETE command replaces the instruction CMPB (R0),(R5) with NOP instructions.

2.
```
PATCH>DELETE/SYMBOLS
LOC>7A6
OLD>0E6ADDE39
OLD>EXIT
old: OTS$LINKAGE\OTS$LINKAGE+7C    0E6ADDE39
new: OTS$LINKAGE\OTS$LINKAGE+7C    00000000
```

   In Example 2, the DELETE command deletes the contents of location 7A6. Because /SYMBOLS is specified, location 7A6 is reported symbolically.

DEPOSIT

Deposits new data or instructions into one or more consecutive locations.

Format

DEPOSIT location = new-contents [, . . .]

Parameters

location
Specifies either a single location whose contents are to be overwritten or the starting address of a sequence of locations whose contents are to be overwritten. The length of the sequence depends on the current mode settings.

new-contents
Specifies one or more data entries or instructions to be inserted. Do not enter conflicting data types with a single DEPOSIT command.

Qualifiers

/BYTE
Specifies that data is deposited in bytes. The length mode qualifiers affect only ASCII and numeric data. When you specify an instruction, the DEPOSIT command ignores the current length setting and deposits the entire instruction.

/WORD
Specifies that data is deposited in words. The length mode qualifiers affect only ASCII and numeric data. When you specify an instruction, the DEPOSIT command ignores the current length setting and deposits the entire instruction.

/LONG
Specifies that data is deposited in longword lengths. The length mode qualifiers affect only ASCII and numeric data. When you specify an instruction, the DEPOSIT command ignores the current length setting and deposits the entire instruction.

The initial default setting is /LONG.

/OCTAL
Specifies that virtual addresses and numeric data are interpreted and displayed using octal radix. The radix mode qualifiers do not affect symbolic addresses or ASCII data.

/DECIMAL
Specifies that virtual addresses and numeric data are interpreted and displayed using decimal radix. The radix mode qualifiers do not affect symbolic addresses or ASCII data.

/HEXADECIMAL
Specifies that virtual addresses and numeric data are interpreted and displayed using hexadecimal radix. The radix mode qualifiers do not affect symbolic addresses or ASCII data.

The initial default setting is /HEXADECIMAL.

/[NO]ASCII
Controls whether the data to be deposited is interpreted and displayed as ASCII data.

When you specify /ASCII, delimit the data with quotation marks (") or apostrophes (′). The current radix setting has no effect on the ASCII data being deposited. The DEPOSIT command truncates the data if it exceeds the length imposed on it by the current length mode.

The initial default setting is /NOASCII.

/[NO]INSTRUCTION
Controls whether the data to be deposited is interpreted and displayed as VAX MACRO instructions.

When you specify /INSTRUCTION, delimit the instruction with quotation marks (") or apostrophes (′). The current length setting does not affect the instruction being deleted.

The initial default setting is /NOINSTRUCTION.

[/NO]SYMBOLS
Controls whether locations are displayed as pathnames or symbols, rather than as numeric addresses.

The initial default setting is /SYMBOLS.

/[NO]GLOBALS
Controls whether the symbolic entry is used, exactly as entered, as the first or last pathname in a search.

The initial default setting is /NOGLOBALS.

/[NO]SCOPE
Controls whether scope's contribution to a pathname is used to find the location specified.

The initial default setting is /SCOPE.

/PATCH_AREA
Signals PATCH to deposit the data or instructions into the current patch area, starting at the specified location.

Description

When depositing data or instructions, you can replace the contents of a location or of several consecutive locations according to the current mode settings.

Note

The DEPOSIT command does not request verification of the current contents before replacing the contents with new data or instructions. In cases when you want to confirm the data or instructions that will be overwritten, use the REPLACE command.

When you append the /PATCH_AREA qualifier to the DEPOSIT command, the data or instructions specified are inserted into the current patch area. The location you supply must be the first free byte in patch area. To determine the first free byte in patch area, enter the SHOW PATCH_AREA command or the ALIGN/BYTE command. After you deposit the data, PATCH updates the patch area string descriptor to reflect the modifications.

The DEPOSIT/PATCH_AREA command uses the patch area to store additional instructions or data; however, PATCH does not automatically generate branch instructions to and from the patch area when you deposit the new data. Unlike the INSERT and REPLACE commands, the DEPOSIT/PATCH_AREA command requires that you insert the branch instructions manually into the appropriate locations to maintain the logical flow of program execution. You must also recalculate the relative displacements of all branch-type instructions affected by the insertion of new data or instructions.

Examples

1.
```
PATCH>DEPOSIT/ASCII/BYTE 1111 = 'A'
old: 00001111:    'B'
new: 00001111:    'A'
```

In Example 1, the DEPOSIT command requests that a byte of ASCII data be deposited in location 1111. The output shows that the new data, 'A', successfully replaced the original data, 'B'.

2.
```
PATCH>DEPOSIT/INSTRUCTION
LOC>413
NEW>'CMPW (R1),R6'
NEW>EXIT
old: 00000413:    CMPW (R6),R1
new: 00000413:    CMPW (R1),R6
```

In Example 2, the command deposits an instruction into location 413. The /INSTRUCTION qualifier indicates that an instruction is being deposited. The output indicates that the instruction was successfully deposited.

3.
```
PATCH>SET PATCH_AREA NEW_PATCH
PATCH>ALIGN/BYTE PAT1
old patch area size:       0000018C
old patch area address:    000004A8
new patch area size:       0000018C
new patch area address:    000004A8
symbol "PAT1" defined as:  000004A8
PATCH>DEPOSIT/PATCH_AREA/ASCII
LOC>PAT1
NEW>'RUN'
NEW>'SKIP'
NEW>EXIT
old:  PAT1:  "
old:  000004AC:
new:  PAT1:   'RUN'
new:  000004AC:   'SKIP'
```

In Example 3, the SET PATCH_AREA command establishes the user-defined patch area named NEW_PATCH as the current patch area. The ALIGN /BYTE command realigns the starting address of NEW_PATCH on the first available byte address and defines the symbol PAT1 to that address. The DEPOSIT/PATCH_AREA command is entered to deposit the ASCII data RUN and SKIP into NEW_PATCH at PAT1.

EVALUATE

Displays values for arithmetic expressions, values, and variable-length bit fields.

Format

EVALUATE expression [, . . .]

Parameters

expression
Indicates an arithmetic expression, a value and corresponding bit field, or a literal value that is to be evaluated in terms of the current mode settings. When you evaluate a selected bit field in a value, the following is the format of the expression:

```
value <high-bit:low-bit>
```

Qualifiers

/BYTE
Specifies that data is evaluated in bytes. The length mode qualifiers affect only the evaluation of arithmetic expressions; they have no effect on the evaluation of bit fields.

/WORD
Specifies that data is evaluated in words. The length mode qualifiers affect only the evaluation of arithmetic expressions; they have no effect on the evaluation of bit fields.

/LONG
Specifies that data is evaluated in longwords. The length mode qualifiers affect only the evaluation of arithmetic expressions; they have no effect on the evaluation of bit fields.

The initial default setting is /LONG.

/OCTAL
Specifies that data is interpreted and displayed using octal radix. The radix mode qualifiers affect the evaluation of arithmetic expressions and of bit fields for specific values.

/DECIMAL
Specifies that data is interpreted and displayed using decimal radix. The radix mode qualifiers affect the evaluation of arithmetic expressions and of bit fields for specific values.

/HEXADECIMAL
Specifies that data is interpreted and displayed using hexadecimal radix. The radix mode qualifiers affect the evaluation of arithmetic expressions and of bit fields for specific values.

The initial default setting is /HEXADECIMAL.

/[NO]ASCII
Controls whether the data is interpreted and displayed as ASCII data. You cannot set the ASCII mode or use the /ASCII qualifier when you are evaluating variable-length bit fields.

The initial default setting is /NOASCII.

/[NO]INSTRUCTION
Controls whether the data is interpreted and displayed as VAX MACRO instructions. You cannot set the INSTRUCTION mode or use the /INSTRUCTION qualifier when you are evaluating variable-length bit fields or ASCII data.

The initial default setting is /NOINSTRUCTION.

/[NO]GLOBALS
Controls whether the symbolic entry is used, exactly as entered, as the first or last pathname in a search.

The initial default setting is /NOGLOBALS.

/[NO]SCOPE
Controls whether scope's contribution to a pathname is used to find the location specified.

The initial default setting is /SCOPE.

Description

The EVALUATE command interprets expressions and displays results in the current length and radix modes. Use the EVAUATE command to perform the following operations:

- Perform binary and unary arithmetic functions.
- Determine the value associated with a symbol or pathname. The values are displayed according to the current length and radix mode setting.
- Display the current contents of a specific bit field in a value.

Use the following syntax for the EVALUATE command:

```
PATCH>EVALUATE value <high-bit:low-bit>
```

The bit position delimiters (high-bit and low-bit) are specified as decimal integers.

Bit positions range from 0 (least significant bit) to 31 (most significant bit). PATCH extracts the contents of the bit positions and reports the contents in longword representation and in terms of the current radix setting. The current length mode is ignored. Note that ASCII mode and INSTRUCTION mode cannot be set when you evaluate selected bit positions.

Examples

1.
```
PATCH>EVALUATE/DECIMAL 101*123
12423
```

   In Example 1, the EVALUATE command performs the requested arithmetic operation. The values are interpreted and displayed in decimal representation.

2.
```
PATCH>SET MODE DECIMAL
PATCH>EVALUATE ^X200 + <^D65/^O12>
518
```

   In Example 2, the EVALUATE command performs the specified arithmetic operation according to the rules of precedence. PATCH first divides decimal 65 by octal 12, then adds the quotient to hexadecimal 200. The result is displayed in decimal representation.

3.
```
PATCH>EVALUATE ^O70 <5:3>
00000007
```

   In Example 3, the EVALUATE command calculates the specified bit field value for the octal number 70. The result is displayed in the current radix setting (in this case, hexadecimal).

4.
```
PATCH>EVALUATE FDR$PETE
00000800
```

   In Example 4, the EVALUATE command determines that the value assigned to the symbol FDR$PETE is 800. The result is displayed in the current radix setting (in this case, decimal).

EXAMINE

Displays the contents of the specified locations according to the current mode settings.

Format

EXAMINE location [:location] [, . . .]

Parameters

location
Specifies one or more locations whose contents you want to be displayed. Specify multiple locations in lists separated by commas, or in ranges separated by colons. You can specify both lists and ranges in a single command.

The location parameter can also be represented by the backslash operator (\ ).

If you do not supply a location, the contents of the next sequential location are displayed. The next sequential location depends on the current mode settings.

Qualifiers

/BYTE
Specifies that data is displayed in bytes. The length mode qualifiers affect only ASCII and numeric data. When you specify an instruction, the EXAMINE command ignores the current length setting and displays the entire instruction.

/WORD
Specifies that data is displayed in words. The length mode qualifiers affect only ASCII and numeric data. When you specify an instruction, the EXAMINE command ignores the current length setting and displays the entire instruction.

/LONG
Specifies that data is displayed in longwords. The length mode qualifiers affect only ASCII and numeric data. When you specify an instruction, the EXAMINE command ignores the current length setting and displays the entire instruction.

The initial default setting is /LONG.

/OCTAL
Specifies that virtual addresses and numeric data are interpreted and displayed using octal radix. The radix mode qualifiers do not affect symbolic addresses or ASCII data.

/DECIMAL
Specifies that virtual addresses and numeric data are interpreted and displayed using decimal radix. The radix mode qualifiers do not affect symbolic addresses or ASCII data.

/HEXADECIMAL
Specifies that virtual addresses and numeric data are interpreted and displayed using hexadecimal radix. The radix mode qualifiers do not affect symbolic addresses or ASCII data.

The initial default setting is /HEXADECIMAL.

/[NO]ASCII
Controls whether data is displayed as ASCII data.

When you specify /ASCII, the EXAMINE command truncates the data if it exceeds the limit imposed on it by the current length setting.

The initial default setting is /NOASCII.

/[NO]INSTRUCTION
Controls whether data is displayed as VAX MACRO instructions.

When you specify /INSTRUCTION, the EXAMINE command ignores the current length setting when displaying an instruction.

The initial default setting is /NOINSTRUCTION.

[/NO]SYMBOLS
Controls whether locations are displayed as pathnames or symbols, rather than as numeric addresses.

The initial default setting is /SYMBOLS.

/[NO]GLOBALS
Controls whether the symbolic entry is used, exactly as entered, as the first or last pathname in a search.

The initial default setting is /NOGLOBALS.

/[NO]SCOPE
Controls whether scope's contribution to a pathname is used to find the location specified.

The initial default setting is /SCOPE.

Description

Using the backslash operator (\ ), you can use the EXAMINE command to examine the contents of a branch instruction or the contents of an address displayed in response to the previous EXAMINE command.

Examples

1.
```
PATCH>EXAMINE/INSTRUCTION/NOSYMBOLS 600
00000600:  BNEQ 634
PATCH>EXAMINE/INSTRUCTION/NOSYMBOLS \
00000634:  TSTL R4
```

In Example 1, the response to the first EXAMINE command indicates that memory location 600 contains a branch instruction to location 634. Because the backslash character (\ ) is specified in the second EXAMINE command, the contents of location 634 are displayed. The /NOSYMBOLS qualifier requests that the locations be displayed by virtual addresses.

2.
```
PATCH>EXAMINE/ASCII 650:.
00000650:  'FE  '
00000654:  'FI  '
00000658:  'FO  '
0000065C:  'FUM '
```

In Example 2, the EXAMINE command requests an ASCII display of all data between location 650 and the current location (represented by the dot character).

3.
```
PATCH>SET MODE SCOPE,SYMBOLS
PATCH>SET SCOPE NEGATION
   .
   .
   .
PATCH>SET MODE NOSCOPE
   .
   .
   .
PATCH>EXAMINE/INSTRUCTION/SCOPE RO_CODE
NEGATION/RO_CODE:  PUSHL L^NEGATION/RW_DATA+14
```

In Example 3, the modes SCOPE and SYMBOLS are set, and the scope is established as the module named NEGATION. After PATCH performs a series of commands, the SCOPE-NOSCOPE mode is turned off, although the scope is still set to NEGATION.

Later, the EXAMINE command is specified to request the display of the instruction in location RO_CODE. The /SCOPE qualifier requests that the scope be prefixed to RO_CODE and that the symbol table be searched for a value that matches the scope.

4.
```
PATCH>EXAMINE/INSTRUCTION   .
00000600:   BNEQ  634
PATCH>EXAMINE/INSTRUCTION
00000602:   BRW   LPLIST
PATCH>EXAMINE/INSTRUCTION  \
LPLIST:  MOVL  R3,R4
```

In Example 4, the first EXAMINE command requests that the contents of the current location (represented by the dot character) be displayed as a VAX MACRO instruction. The second EXAMINE command requests that the next sequential location be displayed as a VAX MACRO instruction. The third EXAMINE command requests that the contents of the destination of the previous branch instruction (BRW) be displayed.

EXIT

Ends a PATCH session and passes control back to the command interpreter. Also ends a repetitive prompt such as NEW>, OLD>, or ECO>.

Format

EXIT

Parameters

None.

Qualifiers

None.

Description

When using the DEFINE command, do not type EXIT in response to the value prompt (VAL>). In DEFINE, only the name prompt (NAM>) recognizes the EXIT command.

Note

You can also specify CTRL/Z to terminate a PATCH session. Press CTRL/Z in response to the PATCH prompt (PATCH>).

Examples

1.
```
PATCH>SET MODE
NEW>INSTRUCTION
NEW>NOSYMBOLS
NEW>NOSCOPE
NEW>EXIT
PATCH>
```

In Example 1, the SET MODE command continuously prompts for new mode settings until you enter the EXIT command.

2.
```
PATCH>EXIT
$
```

When you specify the EXIT command in response to a PATCH prompt (PATCH>), the PATCH session is terminated, and control is passed back to the command interpreter. The DCL prompt ($) indicates that you are no longer in PATCH.

3.
```
PATCH>DEFINE
NAM>TEST1
NEW>500
NAM>TEST2
NEW>800
NAM>EXIT
symbol "TEST1" defined as 00000500
symbol "TEST2" defined as 00000800
PATCH>
```

In Example 3, the DEFINE command assigns the symbols TEST1 and TEST2 to 500 and 800, respectively. The EXIT command is entered in response to the name prompt to terminate this level of prompting.

HELP

Displays information about PATCH commands.

Format

HELP topic [subtopic . . .]

Parameters

topic
Specifies the name of the command for which you want help.

subtopic
Specifies a particular qualifier or parameter about which you want further information, or a command keyword that gives you information about a range of qualifiers and parameters.

If you want information about a particular qualifier or parameter, specify it as a subtopic. Note that when you specify a qualifier, you must include a preceding slash (/). If you want information about all command qualifiers, specify "qualifier" as a subtopic. If you want information about all parameters, specify "parameter" as a subtopic.

Qualifiers

None.

Description

HELP displays the following information about each PATCH command: description, format, qualifiers that can be specified with the command, and parameters that can be specified with the command. The HELP command also provides information about modes and expressions.

Example

```
PATCH>HELP CANCEL
CANCEL

The CANCEL commands allow the user to reinstate initial
defaults for the various display and addressing characteristics
of PATCH.

Additional information available:

MODE     MODULE     PATCH_AREA     SCOPE

PATCH>
```

The HELP CANCEL command displays information about the CANCEL commands.

INSERT

Inserts VAX MACRO instructions into specific locations within an image file.

Format

INSERT location = current-instruction new-instruction . . .

Parameters

location
Specifies the address after which one or more new instructions is to be added.

current-instruction
Specifies the instruction currently occupying the specified location.

new-instruction
Specifies one or more new instructions to be inserted into the image file following the current instruction.

Qualifiers

/OCTAL
Specifies that virtual addresses are interpreted and displayed using octal radix.

/DECIMAL
Specifies that virtual addresses are interpreted and displayed using decimal radix.

/HEXADECIMAL
Specifies that virtual addresses are interpreted and displayed using hexadecimal radix.

The initial default setting is /HEXADECIMAL.

/[NO]INSTRUCTION
Controls whether the data to be inserted is interpreted as VAX MACRO instructions.

To use the INSERT command, you must either specify the /INSTRUCTION mode qualifier or explicitly set mode to INSTRUCTION.

The initial default setting is /NOINSTRUCTION.

[/NO]SYMBOLS
Controls whether locations are displayed as pathnames or symbols, rather than as numeric addresses.

The initial default setting is /SYMBOLS.

/[NO]GLOBALS
Controls whether the symbolic entry is used, exactly as entered, as the first or last pathname in a search.

The initial default setting is /NOGLOBALS.

/[NO]SCOPE
Controls whether scope's contribution to a pathname is used to find the location specified.

The initial default setting is /SCOPE.

Description

When you use the INSERT command, you must specify the /INSTRUCTION qualifier or set the INSTRUCTION mode. To insert additional data into a patch area, use the DEPOSIT/PATCH_AREA command.

Before inserting the new instruction, the INSERT command confirms the contents of the location preceding the insertion (the current instruction). New instructions are inserted after the current instruction.

When the INSERT command is executed, it replaces the current instruction with a branch instruction and places the current instruction and the new instructions in the current patch area. The most recent instruction is followed by a branch instruction that redirects the flow of execution back to the inline code. The INSERT command automatically generates branch instructions.

After the insertion of new instructions, the patch area string descriptor is updated to reflect the modifications. The INSERT/INSTRUCTION command stores new instructions in the patch area and generates unconditional branch instructions to reroute the logical flow of execution to the patch area, then back to the inline code. You can use the INSERT command only for instructions, not for data.

When you insert one or more new instructions into your code, the following events can occur:

- A branch-type instruction is moved to the current patch area.
- The patch area is far enough away to prevent the branch-type instruction from reaching its destination.

If these events occur, the INSERT/INSTRUCTION command recalculates the relative displacement of the branch-type instruction to allow the branch to reach its destination. For example, the insertion of a new instruction causes the BRB 200 instruction to be moved to the patch area. Once the BRB 200 instruction is in the patch area, the byte displacement is not large enough to reach location 200. The INSERT/INSTRUCTION command then recalculates the byte displacement to allow the branch to succeed. If necessary, the BRB instruction is changed to a BRW or JMP instruction.

Note that PATCH always completes a patch with a branch or a jump instruction to the inline code, even if the last instruction of the patch is an unconditional branch or jump instruction.

The INSERT/INSTRUCTION command may cause an instruction identified by a symbolic label to be moved to the patch area. If this occurs, the symbolic label still points to the original location of the instruction. Therefore, you must change any instructions in the program that use the symbolic label to refer to the instruction that was moved. (See Section 2.1.6.)

Calculating the Location for the Branch Instruction

If a branch instruction to the patch area is longer than the current instruction, additional instructions following the current instruction are also moved to the patch area, and the branch instruction is deposited in the vacated memory locations. Unused memory locations are filled with NOP instructions.

Calculating Relative Displacements for Branch Instructions

PATCH calculates the relative displacements for the branch instructions it generates and recalculates the relative displacements for all branch-type instructions moved to the patch area. Instructions and data moved to the patch area may, however, be referenced by instructions not affected by the move. Note that PATCH does not recalculate any relative displacements in the unaffected instructions.

Note also that if PATCH moves an instruction with a current address defined by a symbolic instruction label, you must check and correct any references made to that label.

Example

```
PATCH>SET MODE INSTRUCTION
PATCH>EXAMINE 600
00000600:     MOVL R1,R4
PATCH>EXAMINE
00000602:     BSBB LP_NAME
PATCH>INSERT 602 = 'BSBB LP_NAME'
NEW>'CVTFW R3,R2'
NEW>EXIT
old:    00000602:     BSBB    LP_NAME
old:    00000604:     CMPB    R2,R8
new:    00000602:     BRW     PAA
new:    00000605:     NOP
new:    00000606:     NOP
new:    PAA:          BSBW    LP_NAME
new:    00030374:     CVTFW   R3,R2
new:    00030377:     CMPB    R2,R8
new:    00030740:     BRW     00000607
```

This example shows the procedure used to insert an additional instruction into the existing code. After confirming the contents of location 602 and specifying the new instruction to be inserted, PATCH performs the following steps:

1. Determines how many instructions from the existing code must be moved to the patch area to make room for a branch instruction.
2. Moves the instructions calculated in step 1 to the current patch area, and inserts a branch instruction in their place. PATCH calculates the relative displacement value for the branch instruction and generates a patch area symbol name to identify the destination of the branch.
3. Fills the unoccupied locations in the existing code with NOP instructions.
4. Recalculates the relative displacements for all branch-type instructions moved to the patch area.
5. Inserts a branch instruction into the patch area to reroute the program's flow of execution back to the inline code.

REPLACE

Replaces the contents of one or more locations with new instructions or data according to the current mode settings.

Format

REPLACE location = current-contents [, . . .] new-content . . .

Parameters

location
Specifies either a single location whose contents are to be replaced or the starting address of a sequence of locations whose contents are to be replaced. The length of the sequence depends on the current mode settings.

current-contents
Specifies one or more data entries or instructions to be replaced. The data or instructions you specify must appear exactly as written in the instruction to be replaced.

Do not specify conflicting data types within a single REPLACE command.

new-contents
Specifies one or more data entries or instructions that are to replace the current contents.

Do not specify conflicting data types within a single REPLACE command.

Qualifiers

/BYTE
Specifies that data is replaced in bytes. The length mode qualifiers affect only ASCII and numeric data. When you specify an instruction, the REPLACE command ignores the current length setting and replaces the entire instruction.

/WORD
Specifies that data is replaced in words. The length mode qualifiers affect only ASCII and numeric data. When you specify an instruction, the REPLACE command ignores the current length setting and replaces the entire instruction.

/LONG
Specifies that data is replaced in longwords. The length mode qualifiers affect only ASCII and numeric data. When you specify an instruction, the REPLACE command ignores the current length setting and replaces the entire instruction.

The initial default setting is /LONG.

/OCTAL
Specifies that virtual addresses and numeric data are interpreted and displayed using octal radix. The radix mode qualifiers do not affect symbolic addresses or ASCII data.

/DECIMAL
Specifies that virtual addresses and numeric data are interpreted and displayed using decimal radix. The radix mode qualifiers do not affect symbolic addresses or ASCII data.

/HEXADECIMAL
Specifies that virtual addresses and numeric data are interpreted and displayed using hexadecimal radix. The radix mode qualifiers do not affect symbolic addresses or ASCII data.

The initial default setting is /HEXADECIMAL.

/[NO]ASCII
Controls whether the data to be replaced and the data to be deposited are interpreted and displayed as ASCII data.

When you specify /ASCII, delimit the data with quotation marks (") or apostrophes (′). The current radix setting has no effect on ASCII data being replaced or deposited; however, the REPLACE command truncates ASCII data if it exceeds the length imposed on it by the current length setting.

The initial default setting is /NOASCII.

/[NO]INSTRUCTION
Controls whether the data to be replaced and the data to be deposited are interpreted and displayed as VAX MACRO instructions.

When you specify /INSTRUCTION, delimit the instructions with quotation marks (") or apostrophes (′). The current length setting does not affect the instruction being replaced.

The initial default setting is /NOINSTRUCTION.

[/NO]SYMBOLS
Controls whether locations are displayed as pathnames or symbols, rather than as numeric addresses.

The initial default setting is /SYMBOLS.

/[NO]GLOBALS
Controls whether the symbolic entry is used, exactly as entered, as the first or last pathname in a search.

The initial default setting is /NOGLOBALS.

/[NO]SCOPE
Controls whether scope's contribution to a pathname is used to find the location specified.

The initial default setting is /SCOPE.

Description

Before performing the replacement, the REPLACE command confirms the contents of the specified locations.

When you replace instructions, and the new instructions occupy more bytes in memory than the current instructions, the new instructions are moved to the patch area. PATCH generates branch instructions to maintain the program's logical flow of execution. PATCH generates branch instructions for the REPLACE command the same way that it does for the INSERT command. (See the INSERT command description for a description of the mechanics of generating branch instructions.) All unused bytes are filled with NOP instructions.

If patch area is used to accommodate the new contents, the patch area string descriptor is updated to reflect the modifications.

When you replace ASCII or numeric data, the number of replacement entries cannot exceed the number of existing entries. If the number of replacement entries is less than the number of existing entries, the remaining locations are filled with zeros.

PATCH truncates replacement entries if they exceed the limit imposed on them by the current length mode. For ASCII characters, the right-most characters are discarded. For numeric data, the left-most digits are discarded.

PATCH calculates the relative displacements for the branch instructions it generates and recalculates the relative displacements for all branch-type instructions moved to the patch area. Instructions and data moved to the patch area may be referenced by instructions not affected by the move. Note that PATCH does not recalculate the relative displacement values in the unaffected instructions.

The REPLACE/INSTRUCTION command automatically stores the new instructions in the patch area only if the new instructions occupy more bytes in memory than the instructions being replaced. If the patch area is used, PATCH generates branch instructions to and from the patch area to maintain the correct flow of program execution.

When you replace numeric or ASCII data, the new data is truncated if it exceeds the length of the data that it is to replace. (That is, the patch area is not used.)

The REPLACE/INSTRUCTION command can cause a branch-type instruction to be moved to the patch area. If the new location of the branch-type instruction is too far away to allow the branch to reach its destination, the REPLACE /INSTRUCTION command recalculates the relative displacement of the branch-type instruction to allow the branch to reach its destination.

Note that PATCH always completes a patch with a branch or a jump instruction to the inline code, even if the last instruction of the patch is an unconditional branch or jump instruction.

The REPLACE/INSTRUCTION command may cause an instruction identified by a symbolic label to be moved to the patch area. If this occurs, the symbolic label still points to the original location of the instruction. Therefore, you must change any instructions in the program that use the symbolic label to refer to the instruction. (See Section 2.1.6.)

Examples

1.
```
PATCH>SET MODE INSTRUCTION, NOSYMBOLS
PATCH>REPLACE 600 = 'TSTL R4'
NEW>'CMPB R2,R4'
NEW>EXIT
old:     00000600: TSTL R4
old:     00000602: BEQL 00000610
new:     00000600: BRW 0000784A
new:     00000604: NOP
new:     0000784A: CMPB R2,R4
new:     0000784D: BNEQ 00007852
new:     0000784F: BRW 00000610
new:     00007852: BRW 00000605
```

In Example 1, the instruction occupying location 600 is replaced with the instruction CMPB R2,R4. This instruction occupies more bytes than the instruction TSTL R4. To make room for CMPB R2,R4, the instructions CMPB R2,R4 and the instruction that follows TSTL R4 are moved to the

current patch area. A branch instruction is deposited in place of TSTL R4 to direct program execution to the patch area. The last instruction deposited in the patch area is a branch instruction back to the inline code.

2.
```
PATCH>SET SCOPE MOD1
PATCH>REPLACE/SCOPE RO_CODE+20 = 1000
NEW>100
NEW>EXIT
old:     MOD1\E:   00001000
new:     MOD1\E:   00000100
```

In Example 2, the SET SCOPE command establishes the current symbolic scope as MOD1. A subsequent REPLACE command requests that the contents of location RO_CODE+20 in the module named MOD1 be replaced with new data. The display indicates that the replacement was successful. Note that the location is reported by the pathname MOD1\E because the symbol E is the closest symbol to location RO_CODE+20.

SET ECO

Assigns an ECO level to the patch that you are creating.

Format

SET ECO eco-level

Parameters

eco-level
Specifies a decimal integer of 1 through 128. ECO levels outside this range of integers are disallowed.

Qualifiers

None.

Description

When you apply a patch to an image file, first type a SET ECO command. This command provides you with a way to identify your patches easily and enables you to process selected patches using a command procedure. When PATCH processes a command procedure, it searches the file for ECO levels and processes only those patches represented by the specified ECO levels.

You can enter as many SET ECO commands as you need during a PATCH session. Once you specify an ECO level, that level cannot be used again for that particular image file. When you begin a patch with a SET ECO command, you must also terminate the patch with the UPDATE command. If you do not enter the UPDATE command, the ECO level specified with the SET ECO command is not set, and the patch is not applied to the image file. This means that you cannot enter another SET ECO command.

Example

```
PATCH>SET ECO 15
   .
   .
   .
PATCH>UPDATE
%PATCH-I-WRTFIL, updating image file DBA3:[HOWARD]NEWFILE.EXE;2
PATCH>CHECK NOT ECO 15
%PATCH-E-ECOSET, eco level 15 already set in DBA3:[HOWARD]NEWFILE.EXE;2
```

In this example, the SET ECO command defines the ECO level as 15 for the subsequent patch. When the UPDATE command is entered, the ECO level is set, and the image file is updated to include the effects of the new commands. The CHECK NOT ECO indicates that ECO level 15 is set in the image file NEWFILE.EXE.

SET MODE

Controls the syntax of the PATCH commands that you enter as well as the values that PATCH displays.

Format

SET MODE mode [, . . .]

Parameters

mode
Specifies one or more modes from the context, radix, length, and symbol search mode categories to be established as the current modes. These modes determine how PATCH interprets entries and displays output.

The commands that are affected by the entry and display modes are DELETE, DEPOSIT, EVALUATE, EXAMINE, INSERT, REPLACE, and VERIFY. Note that you can set the modes by using mode qualifiers with these commands.

Mode	Description
BYTE	Specifies that data is interpreted in bytes. The length mode qualifiers affect only ASCII and numeric data.
WORD	Specifies that data is interpreted in words. The length mode qualifiers affect only ASCII and numeric data.
LONG	Specifies that data is interpreted in longwords. The length mode qualifiers affect only ASCII and numeric data. The initial default setting is /LONG.
OCTAL	Specifies that virtual addresses and numeric data are interpreted and displayed using octal radix. The radix mode qualifiers do not affect symbolic addresses or ASCII data.
DECIMAL	Specifies that virtual addresses and numeric data are interpreted and displayed using decimal radix. The radix mode qualifiers do not affect symbolic addresses or ASCII data.
HEXADECIMAL	Specifies that virtual addresses and numeric data are interpreted and displayed using hexadecimal radix. The radix mode qualifiers do not affect symbolic addresses or ASCII data. The initial default setting is HEXADECIMAL.
[NO]ASCII	Controls whether data is to be interpreted and displayed as ASCII data. When you specify ASCII, delimit the data with quotation marks (") or apostrophes (′). The initial default setting is NOASCII.

Mode	Description
[NO]INSTRUCTION	Controls whether the data is interpreted and displayed as VAX MACRO instructions. When you specify INSTRUCTION, delimit the instruction with quotation marks (") or apostrophes (′). The initial default setting is NOINSTRUCTION.
[NO]SYMBOLS	Controls whether locations are displayed as pathnames or symbols, rather than as numeric addresses. The initial default setting is SYMBOLS.
[NO]GLOBALS	Controls whether the symbolic entry is used, exactly as entered, as the first or last pathname in a search. The initial default setting is NOGLOBALS.
[NO]SCOPE	Controls whether scope's contribution to a pathname is used to find the location specified. The initial default setting is SCOPE.

Qualifiers

None.

Description

Use SET MODE to control the syntax of the commands you enter and the values PATCH displays.

Examples

1.
```
PATCH>SET MODE INSTRUCTION,GLOBALS,NOSYMBOLS
```

In Example 1, the SET MODE command requests that the modes INSTRUCTION, GLOBALS, and NOSYMBOLS be established as the current modes.

2.
```
PATCH>SET MODE INSTRUCTION
PATCH>DEPOSIT 78B = 'BEQL NEWPATCH'
```

In Example 2, the SET MODE command establishes INSTRUCTION mode as the current mode setting. This enables a subsequent DEPOSIT command to deposit an instruction without specifying the /INSTRUCTION mode qualifier.

3.
```
PATCH>SET MODE
NEW>OCTAL
NEW>WORD
NEW>EXIT
```

In Example 3, the SET MODE prompts continuously for new modes until EXIT is typed.

SET MODULE

Enters local symbol information from the specified modules into the PATCH symbol table.

Format

SET MODULE module-name [, . . .]

Parameters

module-name
Specifies the name of one or more modules whose local symbols are to be entered in the symbol table.

Do not specify a module name if you include the /ALL qualifier.

Qualifiers

/ALL
Requests that local symbol information from all the modules in the image file be entered in the symbol table.

Description

To use SET MODULE, make sure you passed local symbol information to PATCH correctly.

If the symbol table is too small to accommodate the local symbol information, PATCH displays an error message.

Note that if you are patching a shareable image, no local or global symbol information is passed to PATCH. Only universal symbols can be accessed.

Examples

1.
```
PATCH>SET MODULE    GREAT_APES
PATCH>EXAMINE/ASCII ORANGUTANS:ORANGUTANS+7
GREAT_APES\ORANGUTANS:  'AMAZ'
GREAT_APES\ORANGUTANS+4:  'ING '
```

In Example 1, the SET MODULE command enters all local symbol information in the module GREAT_APES into the symbol table. A subsequent EXAMINE command can use the local symbol ORANGUTANS to reference particular locations.

2.
```
PATCH>SET MODULE/ALL
```

In Example 2, the SET MODULE command requests that all symbol information from all modules be added to the symbol table.

SET PATCH_AREA

Overrides the use of the default patch area to use the patch area that you defined at assembly or compile time.

Format

SET PATCH_AREA address-of-descriptor

Parameters

address-of-descriptor
Defines the address of the patch area descriptor. The address of the patch area descriptor can be represented as a symbol or a numeric constant.

If you enter the SET PATCH_AREA command without the /INITIALIZE qualifier, the address-of-descriptor is the address of the patch area descriptor defined in the source program.

If you use the /INITIALIZE qualifier, the address-of-descriptor is the address where PATCH locates the patch area descriptor. When you use the /INITIALIZE qualifier, PATCH creates the descriptor and locates it in the first eight bytes of the user-defined patch area.

Qualifiers

/INITIALIZE=size-expression
Creates a patch area descriptor and locates it in the first eight bytes of the patch area. /INITIALIZE must precede the address-of-descriptor.

The size-expression defines the size of the patch area in bytes. In order to accommodate the descriptor and PATCH entries, the size-expression must be eight bytes larger than the area needed for patches.

If the value of the size-expression is specified as zero (0) or is greater than the number of bytes contained in the patch area, PATCH uses a default size value. The default size is the number of unused bytes in the patch area image section, excluding eight bytes for the descriptor. PATCH issues an informational message when it uses the default size. The default size must be at least 12 bytes.

Description

A user-defined patch area, like the default patch area, must have a patch area descriptor.

You can set up a user-defined patch area in the following ways:

- Initialize the patch area at assembly time. Do not use the /INITIALIZE qualifier if the patch area was initialized at assembly time.
- Initialize the patch area at patch time. Use the /INITIALIZE qualifier if you access a patch area initialized at patch time. PATCH creates the descriptor and locates it in the first eight bytes of the patch area.

If you intend to use the SET PATCH_AREA command without the /INITIALIZE qualifier, define the contents and location of the patch area descriptor within the source program. When you assemble and link your program, the patch area code is not shareable and incurs additional overhead during image activation.

If you intend to use the SET PATCH_AREA command with the /INITIALIZE qualifier, you do not need to define the patch area descriptor in your program. Declare the size and starting address of the patch area as global symbols. When you enter the SET PATCH_AREA/INITIALIZE command during a patch session, PATCH creates the patch area descriptor and places it in the first eight bytes of the patch area. By letting PATCH build the descriptor, you can avoid extra overhead during image activation and retain the shareable characteristics of the image.

The SET PATCH_AREA Command Without /INITIALIZE

The SET PATCH_AREA command establishes a user-defined patch area as the current patch area. To create a user-defined patch area that will be accessed with the SET PATCH_AREA command, follow these steps:

1. Declare a size for the patch area.
2. Create a patch area descriptor that specifies the size and address of the first free byte of the patch area. Define a global symbol to represent the start of the descriptor.
3. Declare storage for the patch area.

Enter the SET PATCH_AREA command using the following format:

```
SET PATCH_AREA address-of-descriptor
```

The address-of-descriptor is the address of the patch area descriptor.

The following example shows a source program with global symbols for the size of the patch area, the starting address of the patch area, and the patch area descriptor:

```
                .ENTRY  BEGIN,M<>
                        .       ;User-written code
                        .
                        .       ;End of useful code
;
;Declare Patch Area and its associated descriptor
;
PATCH_SIZE ==   512
PATCH_DESC::    .LONG    PATCH_SIZE
                .ADDRESS PATCH_AREA
PATCH_AREA::    .BLKB    PATCH_SIZE
                .END BEGIN       ;End of program
```

After invoking PATCH, you can access the patch area by entering the SET PATCH_AREA command followed by the address of the patch area descriptor. In the example above, you would access user-defined patch area as follows:

```
PATCH>SET PATCH_AREA PATCH_DESC
```

Make sure you passed local symbol information to PATCH correctly and that symbols appear in the symbol table before you attempt to use the global symbol PATCH.DESC.

The SET PATCH_AREA Command With /INITIALIZE

When you use the /INITIALIZE qualifier, PATCH creates a patch area descriptor and establishes the user-defined patch area as the current patch area. To create patch area that will be accessed with SET PATCH_AREA/INITIALIZE, follow these steps:

1. Declare a size for the patch area. The size must include eight bytes for the patch area descriptor.

2. Define a global symbol to represent the starting address of the patch area. PATCH places the patch area descriptor in the first eight bytes of the patch area.

Enter the SET PATCH_AREA command with the /INITIALIZE qualifier, as follows:

```
SET PATCH_AREA/INITIALIZE=size-expression address-of-descriptor
```

The size-expression indicates the size, in bytes, of the patch area and should include eight bytes for the descriptor. The address-of-descriptor is the address where PATCH locates the patch area descriptor.

Both the size-expression and the address-of-descriptor can be expressed as numeric constants, as global symbols defined within the image, or as expressions using combinations of numeric constants and symbols. The /INITIALIZE qualifier must precede the address-of-descriptor argument.

The following example shows a source program with global symbols for the size and starting address of a user-defined patch area:

```
                .ENTRY BEGIN,M<>
                        .       ;User-written code
                        .
                        .       ;End of useful code
;
;Declare Patch Area              ;Add eight bytes for the
descriptor
;
PATCH_SIZE ==   512+8
PATCH_AREA::    .BLKB  PATCH_SIZE
                .END BEGIN      ;End of program
```

After invoking PATCH, access the user-defined patch area as follows:

```
PATCH>SET PATCH_AREA/INITIALIZE=PATCH_SIZE PATCH_AREA
```

This command initializes the patch area descriptor and locates it in the first eight bytes of PATCH_AREA. The first longword of the descriptor contains the size of the patch area available for patches (512 bytes). The second longword contains the location of the first free byte of the patch area. In the example above, the first free byte is the byte immediately following the descriptor.

If you specify an invalid size expression, PATCH computes and uses a default size for the patch area. To use the /INITIALIZE qualifier, the default size must be at least 12 bytes. If the default size is less than 12, PATCH issues the following error message and does not initialize the descriptor:

```
%PATCH-E-NOPATAREA, Insufficient patch area of xxxxxxxx size=dddddddd
```

PATCH computes the default size of the patch area in the following order:

1. Computes the number of bytes in the image section containing the patch area. This is computed by multiplying the number of pages in the image section by 512.
2. Subtracts eight bytes for the patch area descriptor.
3. Subtracts the number of bytes in the image section that have already been used by the program code that precedes the patch area. The number of used bytes is computed by multiplying the section base page number by 512 to find the virtual address of the start of the image section. This value is then subtracted from the virtual address of the start of the patch area.

This is shown in the following formula:

```
Size = [(#pages_in_section_containing_patch_area * 512) - 8]-
       [patch_area_address - (section_base_page_number * 512)]
```

If you do not know the correct patch area size, use the value zero (0) as the size argument. PATCH substitutes the default size when it builds the descriptor, as long as the default size is at least 12. PATCH issues the following informational message:

```
%PATCH-I-BADINITSZ, illegal size value, defaulting patch size to XXXXXXXX bytes
```

If you enter the SET PATCH_AREA command with the /INITIALIZE qualifier, and the descriptor has already been built, PATCH displays the following message:

```
%PATCH-I-PREVINIT, patch area has previously been initialized
```

PATCH establishes the specified user-defined patch area as the current patch area and assumes that if the contents of the descriptor are nonzero, the descriptor was previously built.

Examples

1.
```
PATCH>SET PATCH_AREA LRDPATCHES
PATCH>SHOW PATCH_AREA
current patch area size:       0000004F
current patch area address:    000005F2
```

In Example 1, the SET PATCH_AREA command establishes the user-defined patch area as the current patch area. The patch area descriptor is located at LRDPATCHES. The SHOW PATCH_AREA command reports the current status of LRDPATCHES. With this information, you can deposit instructions or data into the user-defined patch area.

2.
```
PATCH>SET PATCH_AREA/INIT=PATSIZ UPATCH_AREA
PATCH>SHOW PATCH_AREA
current patch area size:        00000200
current patch area address:     00000408
```

In Example 2, the SET PATCH_AREA/INITIALIZE command establishes the user-defined patch area UPATCH_AREA as the current patch area. The size of this area is PATSIZ. The command also initializes a patch area descriptor, which is stored in the first eight bytes of UPATCH_AREA. The symbols PATSIZ and UPATCH_AREA were defined in the source program. The SHOW PATCH_AREA command displays the current patch area size and the address of the first free byte of UPATCH_AREA.

SET SCOPE

Establishes the specified module name as the explicit scope to be used for translating pathnames and symbols into values.

Format

SET SCOPE module-name [\routine-name [\ . . .]]

Parameters

module-name
Specifies the name of the module to which the scope is to be set.

routine-name
Specifies the name of a routine contained in the module to which the scope is to be set.

Qualifiers

None.

Description

When you enter the SET SCOPE command, PATCH inserts local symbol information associated with the specified module into the symbol table. If the symbol table is too small to accommodate this information, PATCH issues an error message.

Use the SHOW MODULE command to determine the modules to which the scope can be set.

If the local symbols in a specific module have not been entered in the symbol table by the SET MODULE command, the SET SCOPE command forces those symbols into the table.

Examples

1.
```
PATCH>SET SCOPE CARP_LPT
```

   In Example 1, the SET SCOPE command establishes the scope as CARP_LPT.

2.
```
PATCH>SET SCOPE
NEW>LIP_SIGN
%PATCH-W-NOSUCHMODU, no such module name "LIP_SIGN"
```

   In Example 2, the SET SCOPE command fails to find the module LIP_SIGN, so the previous scope setting is retained.

SHOW MODE

Displays the current modes.

Format

SHOW MODE

Parameters

None.

Qualifiers

None.

Description

The SHOW MODE command is used primarily with the SET MODE and/or CANCEL MODE commands.

When you enter the SHOW MODE command, the mode values are displayed in lowercase letters.

Example

```
PATCH>SHOW MODE
modes: symbols, noinstruction, noascii, scope, noglobals, hexadecimal long
```

In this example, the SHOW MODE command requests that the current modes be displayed.

SHOW MODULE

Displays all of the modules in the image file and indicates whether the symbols contained in the modules area are available for use.

Format

SHOW MODULE

Parameters

None.

Qualifiers

None.

Description

The SHOW MODULE COMMAND indicates the amount of symbol table space required by each module and routine and the total amount of unused space that remains in the symbol table.

The SHOW MODULE command reports an informational error message if no local symbol information was passed to PATCH.

Example

```
PATCH>SHOW MODULE
module name     symbols     size

AVERAGE            no         52.
OTS$LINKAGE        no         128.

total modules:     2.
remaining size:    64052.
PATCH>SET MODULE/ALL
PATCH>SHOW MODULE
module name     symbols   size

AVERAGE            yes     52.
OTS$LINKAGE        yes     128.

total modules:     2.
remaining size:    63880.
```

In this example, the first SHOW MODULE command indicates that there are two modules in the current image file, and that neither has its local symbols entered into the symbol table. The SET MODULE command requests that all local symbol information be entered into the symbol table. The second SHOW MODULE command confirms the presence of the local symbols in the symbol table.

SHOW PATCH_AREA

Reports the size and starting address (in hexadecimal) of the current patch area.

Format

SHOW PATCH_AREA

Parameters

None.

Qualifiers

None.

Description

Use this command with the ALIGN command or the DEPOSIT/PATCH_AREA command to determine the status of the patch area before realigning it or depositing data in it.

Example

```
PATCH>SHOW PATCH_AREA
current patch area size:     00000030
current patch area address:  00000308
PATCH>DEPOSIT/PATCH_AREA/ASCII   308= 'AT'
old:  00000308:
new:  00000308: 'AT'
```

In this example, the SHOW PATCH_AREA display reports that the current patch area size is 30 bytes, and the starting address is memory location 308. A subsequent DEPOSIT command can deposit data into that patch area.

SHOW SCOPE

Displays the current scope setting.

Format

SHOW SCOPE

Parameters

None.

Qualifiers

None.

Description

Use SHOW SCOPE to display the current scope setting.

Example

```
PATCH>SHOW SCOPE
scope:  TEMP1
PATCH>EXAMINE DEGREES+8
```

In this example, the SHOW SCOPE command indicates that the current scope setting is TEMP1. A subsequent EXAMINE command can reference a location using local symbol information contained in TEMP1.

UPDATE

Applies a patch to an image file.

Format

UPDATE

Parameters

None.

Qualifiers

None.

Description

The UPDATE command is a patch terminator. It applies the last patch you created to the image file and creates an output image file. If you do not enter the UPDATE command, no output image file is created.

You can specify the UPDATE command more than once during a single execution of PATCH. The first UPDATE command that you specify creates a new output image file. Subsequent UPDATE commands specified during the same PATCH session overwrite the output image file; a new version of the output image file is not created.

If you enter a SET ECO command, your next UPDATE command automatically sets the ECO level specified. (See the description of the SET ECO command.)

Example

```
PATCH>UPDATE
%PATCH-I-WRTFIL, updating image file DB1:[MASON]HOROSCOPE.EXE;4
```

In this example, the UPDATE command applies the last patch you created to the image file HOROSCOPE.EXE. The UPDATE display indicates that the image file has been successfully updated.

VERIFY

Confirms that one location or several consecutive locations contain the specified contents. Contents can be either data or instructions.

Format

VERIFY location = current-contents [, . . .]

Parameters

location
Specifies either a single location whose contents are to be checked or the starting address of a sequence of locations whose contents are to be checked. The length of the sequence depends on the current mode settings.

current-contents
Specifies one or more data entries or instructions to be verified. The data or instructions you verify must appear exactly as written in the instruction to be verified.

Do not specify conflicting data types within a single VERIFY command.

Qualifiers

/BYTE
Specifies that data is verified in bytes. The length mode qualifiers affect only ASCII and numeric data. When you specify an instruction, the VERIFY command ignores the current length setting and verifies the entire instruction.

/WORD
Specifies that data is verified in words. The length mode qualifiers affect only ASCII and numeric data. When you specify an instruction, the VERIFY command ignores the current length setting and verifies the entire instruction.

/LONG
Specifies that data is verified in longwords. The length mode qualifiers affect only ASCII and numeric data. When you specify an instruction, the VERIFY command ignores the current length setting and verifies the entire instruction.

The initial default setting is /LONG.

/OCTAL
Specifies that virtual addresses and numeric data are interpreted and displayed using octal radix. The radix mode qualifiers do not affect symbolic addresses or ASCII data.

/DECIMAL
Specifies that virtual addresses and numeric data are interpreted and displayed using decimal radix. The radix mode qualifiers do not affect symbolic addresses or ASCII data.

/HEXADECIMAL
Specifies that virtual addresses and numeric data are interpreted and displayed using hexadecimal radix. The radix mode qualifiers do not affect symbolic addresses or ASCII data.

The initial default setting is /HEXADECIMAL.

/[NO]ASCII
Controls whether the data to be verified is interpreted and displayed as ASCII data.

When you specify /ASCII, delimit the data with quotation marks (") or apostrophes (′). The current radix setting has no effect on the ASCII data being verified; however, the VERIFY command truncates the data if it exceeds the length imposed on it by the current length mode.

The initial default setting is /NOASCII.

/[NO]INSTRUCTION
Controls whether the data to be verified is interpreted and displayed as VAX MACRO instructions.

When you specify /INSTRUCTION, delimit the instruction with quotation marks (") or apostrophes (′). The current length setting does not affect the instruction being verified.

The initial default setting is /NOINSTRUCTION.

[/NO]SYMBOLS
Controls whether locations are displayed as pathnames or symbols, rather than as numeric addresses.

The initial default setting is /SYMBOLS.

/[NO]GLOBALS
Controls whether the symbolic entry is used, exactly as entered, as the first or last pathname in a search.

The initial default setting is /NOGLOBALS.

/[NO]SCOPE
Controls whether scope's contribution to a pathname is used to find the location specified.

The initial default setting is /SCOPE.

Description

Use the VERIFY command with the DEPOSIT command to verify the contents of the locations to be modified. The DEPOSIT command does not indicate the entries before overwriting them.

The VERIFY command is also useful for checking particular locations before attempting to modify them when you are patching an image file using a command procedure. If the VERIFY command fails, that patch is not applied, and PATCH skips to the next SET ECO command. If no other SET ECO command exists, the command procedure is terminated.

Example

```
PATCH>VERIFY/ASCII/NOSYMBOLS 6FF='RAIN'
old:    000006FF:   'RAIN'
```

In this example, the VERIFY command confirms that the ASCII text RAIN resides in location 6FF. The /NOSYMBOLS mode qualifier requests that location 6FF be displayed as a virtual address in terms of the current radix.

PATCH Example

Example PAT–1 illustrates an interactive PATCH session. The following VAX MACRO program prompts for a decimal integer and then displays the negation of the integer. The source program contains an error, so the program does not run properly.

Example PAT–1 PATCH Example

```
0000     1         .TITLE  NEGATION - Calculates negation of decimal integer
0000     2         .IDENT /01/
0000     3         .SUBTITLE - Pure Data
0000     4         .PSECT RO_DATA,NOWRT,NOEXE,LONG
0000     5 PROMPT: .ASCII  /Enter decimal integer: /
000C
0017     6 PROMPT_LEN = .-PROMPT
0017     7 RESULT_FAO:
0017     8         .ASCID /The negation is !SL./
0025
0031
0033     9         .SUBTITLE - Impure Data
0000    10         .PSECT RW_DATA,NOEXE,LONG
0000    11 BUFFER_SIZE     =       132             ;Buffer size
0000    12 BUFFER_DESC:
0000    13         .LONG   BUFFER_SIZE,BUFFER      ;Descriptor for buffer
0008    14 BUFFER:
0008    15         .BLKB   BUFFER_SIZE             ;Buffer to get line
008C    16 INTEGER:
008C    17         .BLKL   1                       ;Input integer
0090    18 RESULT:
0090    19         .BLKL   1                       ;Result of operation
0094    20
0094    21         .SUBTITLE - Define RAB and FAB for terminal I/O
0094    22 TRMFAB: $FAB    FNM=TT:,-       ;FAB for terminal
0094    23                FAC=<PUT,GET>,- ;Use for input and output
0094    24                RAT=CR
00E4    25
00E4    26 TRMRAB: $RAB    FAB=TRMFAB,-    ;RAB for terminal
00E4    27                UBF=BUFFER,-
00E4    28                USZ=BUFFER_SIZE,-
00E4    29                ROP=PMT-
00E4    30                PBF=PROMPT,PSZ=PROMPT_LEN
0128    31
0128    32         .SUBTITLE - Program Entry Point
0000    33         .PSECT RO_CODE,EXE,NOWRT
0000    34
0000    35         .ENTRY  ENTRY_POINT, ^M<>
0002    36         $OPEN   FAB=TRMFAB              ;Open terminal file
000F    37         $CONNECT        RAB=TRMRAB      ;Connect record stream
001C    38         $GET    RAB=TRMRAB              ;Get input from terminal
0029    39         PUSHAL  INTEGER                 ;Push integer address on
002F    40         PUSHL   RAB$L_RBF+TRMRAB        ;Push starting address of
0035    41                                         ;input on stack
0035    42         MOVZWL  RAB$W_RSZ+TRMRAB,-(SP)  ;push length of input on stack
003C    43         CALLS   #3,G^LIB$CVT_DTB                ;Convert to binary
0043    44         MOVL    INTEGER,RESULT          ;Calculate negation
004E    45
004E    46 ;determine length and contents of result
004E    47         $FAO_S  CTRSTR=RESULT_FAO,-             ;Use system service to
```

(continued on next page)

Example PAT–1 (Cont.) PATCH Example

```
004E    48                OUTLEN=TRMRAB+RAB$W_RSZ,-       ;convert binary result
004E    49                OUTBUF=BUFFER_DESC,-            ;return ASCII character
004E    50                P1=RESULT                       ;in output string
006D    51
006D    52 ;output result
006D    53         MOVAB   BUFFER,TRMRAB+RAB$L_RBF
0078    54         $PUT    RAB=TRMRAB
0085    55         MOVZBL  #SS$_NORMAL,R0
0089    56         RET
008A    57         .END    ENTRY_POINT
```

The program assembles and links, but produces the following erroneous results when executed:

```
$ MACRO/DEBUG NEGATION.MAR;1
$ LINK/DEBUG NEGATION.OBJ;1
$ RUN/NODEBUG NEGATION.EXE;1
Enter decimal integer 25
The negation is 25.
$
```

To fix the problem, locate the incorrect code using the VMS Debugger or by examining the listing. Then invoke PATCH to make a permanent change in the image file. The following example shows how to perform the patch. The numbers to the right of the example are keyed to the explanations that follow the example.

```
$ PATCH NEGATION.EXE;1 1
PATCH VERSION 4-00 15-Apr-1984
PATCH>SET ECO 1 2
PATCH>SET MODULE/ALL 3
PATCH>SET SCOPE NEGATION 4
PATCH>SET MODE INSTRUCTION 5
PATCH>EXAMINE RO_CODE+43
NEGATION\RO_CODE+43     MOVL L^NEGATION\INTEGER,L^NEGATION\RESULT
PATCH>REPLACE NEGATION\RO_CODE+43  6
OLD>'MOVL L^NEGATION\INTEGER,L^NEGATION\RESULT'
OLD>EXIT
NEW>'MNEGL L^NEGATION\INTEGER,L^NEGATION\RESULT'
NEW>EXIT
old: NEGATION\RO_CODE+43:  MOVL  L^NEGATION\INTEGER,L^NEGATION\RESULT
new: NEGATION\RO_CODE+43:  MNEGL L^NEGATION\INTEGER,L^NEGATION\RESULT
PATCH>UPDATE 7
%PATCH-I-WRTFIL, updating image file DB1:[GARTH]NEGATION.EXE;2
PATCH>EXIT 8
$
```

1 Invoke PATCH for an interactive terminal session by typing the DCL command line PATCH NEGATION.EXE;1. PATCH displays an introductory message indicating the version of PATCH that you are using and its release date.

2 The SET ECO command defines the ECO level for the ensuing patch. Note that the ECO level is not set until you enter the UPDATE command.

3 The SET MODULE/ALL command inserts all local symbol information into the symbol table, provided that you passed local symbols correctly at compile or assembly and link time.

4 The SET SCOPE command changes the contents of the scope. In this case, the scope has been set to the module named NEGATION.

5 The SET MODE command sets the INSTRUCTION mode, causing PATCH to display the contents of the specified locations as VAX MACRO instructions.

6 Several PATCH commands can be used to modify code. Each command alters the code differently. In this example, the REPLACE command was used to overwrite the existing instruction with a new, correct instruction.

7 After you correct the code, enter the UPDATE command. This sets the specified ECO level and applies the patch to a new version of the image file.

8 Type the EXIT command to end the PATCH session and return control to the VMS command interpreter.

The program NEGATION.EXE;2 now executes properly.

```
$ RUN/NODEBUG NEGATION.EXE;2
Enter decimal integer 37
The negation is -37.
$
```

Index

A

B

C

D

E

F

G

H

I

J

L

M

N

O

P

Q

R

S

T

U

V

W