# VAX/VMS Patch Utility Reference Manual

Order Number: AA–Z426A–TE

**September 1984**

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

**Revision/Update Information:** This is a new manual.

**Software Version:** VAX/VMS Version 4.0

**digital equipment corporation**
**maynard, massachusetts**

**September 1984**

This document was prepared using an in-house documentation production system. All page composition and make-up was performed by TEX, the typesetting system developed by Donald E. Knuth at Stanford University. TEX is a registered trademark of the American Mathematical Society.

# PATCH Contents

## PATCH Contents

# 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.

## Structure of This Document

This document is composed of five major sections.

The Format Section is an overview of PATCH and is intended as a quick reference guide. The format summary contains the DCL command that invokes PATCH, listing all command qualifiers and parameters. The usage summary describes how to invoke and exit from PATCH, how to direct output, and any restrictions you should be aware of. The command summary lists all commands that can be used within PATCH.

The Description Section explains how to use PATCH.

The Qualifier Section describes each DCL command qualifier. Qualifiers appear in alphabetical order.

The Command Section describes each PATCH command. Commands appear in alphabetical order.

The Examples Section contains examples of common operations that you perform with PATCH.

## Associated Documents

Four other manuals that may be of use:

- *Guide to Using DCL and Command Procedures on VAX/VMS*
- *VAX/VMS DCL Dictionary*
- *Guide to VAX/VMS System Management and Daily Operations*
- *Guide to Writing a Device Driver for VAX/VMS*

## Conventions Used in This Document

| Convention | Meaning |
|---|---|
| $\boxed{\text{RET}}$ | A symbol with a one- to three-character abbreviation indicates that you press a key on the terminal, for example, $\boxed{\text{RET}}$ . |
| $\boxed{\text{CTRL/x}}$ | The phrase CTRL/x indicates that you must press the key labeled CTRL while you simultaneously press another key, for example, CTRL/C, CTRL/Y, CTRL/O. |
| PATCH>EXAMINE/ASCII 650<br>00000650: 'FE ' | Command examples show all output lines or prompting characters that the system prints or displays in black letters. All user-entered commands are shown in red letters. |
| .<br>.<br>. | Vertical series of periods, or ellipsis, mean either that not all the data that the system would display in response to the particular command is shown or that not all the data a user would enter is shown. Vertical ellipsis in coding examples indicate that lines of code not pertinent to the example are omitted. |
| file-spec,... | Horizontal ellipsis indicates that additional parameters, values, or information can be entered. |
| /JOURNAL[=file-spec] | Square brackets indicate that the enclosed item is optional. (Square brackets are not, however, optional in the syntax of a directory name in a file specification or in the syntax of a substring specification in an assignment statement.) |
| quotation marks<br>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. |

# New and Changed Features

This manual documents the Patch Utility in Version 4.0 of VAX/VMS. This section summarizes the main technical changes from Version 3.0:

- The /ABSOLUTE command qualifier has been included in the list of PATCH DCL command qualifiers. This qualifier allows the user to patch a file at absolute virtual addresses.
- The /NEW_VERSION command qualifier has been included in the list of PATCH DCL command qualifiers. This qualifier controls whether a new version of the patched file is created or the contents of the existing file are modified in place.
- Several new examples have been added to the manual including examples for the /ABSOLUTE and /NEW_VERSION qualifiers, and two, more-detailed examples in the Examples Section.

# PATCH

The VAX/VMS Patch Utility (PATCH) allows you make changes to an image file in the form of patches. You can then run the new version of the image without having to compile, assemble, or link the program source files.

## FORMAT

**PATCH** *file-spec*

| **Command Qualifiers** | **Defaults** |
|---|---|
| */ABSOLUTE* | *None.* |
| */JOURNAL[=file-spec]* | *None.* |
| */NEW_VERSION* | */NEW_VERSION* |
| */OUTPUT[=file-spec]* | *None.* |
| */UPDATE[=(eco-level [, . . . ])]* | *None.* |
| */VOLUME[=n]* | *None.* |

**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, the file specification 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

**Invoking**
To invoke PATCH, use the PATCH command.

**Exiting**
To end a PATCH session, enter the EXIT command or press CTRL/Z.

**Directing Output**
To direct output, use the /OUTPUT qualifier with the PATCH command.

**Privileges/Restrictions**
None.

## commands

**Syntax**
PATCH> command [parameter]

**PATCH Commands**

```
ALIGN
    /BYTE
    /WORD
    /LONG
    /QUAD
    /PAGE
CANCEL MODE
CANCEL MODULE
    /ALL
CANCEL PATCH_AREA
CANCEL SCOPE
CHECK ECO
CHECK NOT ECO
CREATE
DEFINE
DELETE
    /BYTE
    /WORD
    /LONG
    /OCTAL
    /DECIMAL
    /HEXADECIMAL
    /[NO]ASCII
    /[NO]INSTRUCTION
    /[NO]SYMBOLS
    /[NO]GLOBALS
    /[NO]SCOPE
DEPOSIT
    /BYTE
    /WORD
    /LONG
    /OCTAL
    /DECIMAL
    /HEXADECIMAL
    /[NO]ASCII
    /[NO]INSTRUCTION
    /[NO]SYMBOLS
    /[NO]GLOBALS
    /[NO]SCOPE
    /PATCH_AREA
EVALUATE
    /BYTE
    /WORD
    /LONG
    /OCTAL
    /DECIMAL
    /HEXADECIMAL
    /[NO]ASCII
    /[NO]INSTRUCTION
    /[NO]SYMBOLS
    /[NO]GLOBALS
    /[NO]SCOPE
```

```
EXAMINE
    /BYTE
    /WORD
    /LONG
    /OCTAL
    /DECIMAL
    /HEXADECIMAL
    /[NO]ASCII
    /[NO]INSTRUCTION
    /[NO]SYMBOLS
    /[NO]GLOBALS
    /[NO]SCOPE
EXIT
HELP
INSERT
    /OCTAL
    /DECIMAL
    /HEXADECIMAL
    /[NO]INSTRUCTION
    /[NO]SYMBOLS
    /[NO]GLOBALS
    /[NO]SCOPE
REPLACE
    /BYTE
    /WORD
    /LONG
    /OCTAL
    /DECIMAL
    /HEXADECIMAL
    /[NO]ASCII
    /[NO]INSTRUCTION
    /[NO]SYMBOLS
    /[NO]GLOBALS
    /[NO]SCOPE
SET ECO
SET MODE
SET MODULE
    /ALL
SET PATCH_AREA
    /INITIALIZE=size-expression
SET SCOPE
SHOW MODE
SHOW MODULE
SHOW PATCH_AREA
SHOW SCOPE
UPDATE
VERIFY
    /BYTE
    /WORD
    /LONG
    /OCTAL
    /DECIMAL
    /HEXADECIMAL
    /[NO]ASCII
    /[NO]INSTRUCTION
    /[NO]SYMBOLS
    /[NO]GLOBALS
    /[NO]SCOPE
```

## DESCRIPTION

After you have compiled or assembled and linked a program, you may need to make changes to the code. The VAX/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. You can then execute the patched program without recompiling or reassembling and relinking.

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

- 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

PATCH may be used to maintain executable image files, shareable image files, and device-driver image files written by users in any language supported by the VAX/VMS operating system.

In many cases, you will want to use the VAX/VMS Symbolic Debugger before using PATCH. The debugger lets you make temporary changes to your code; you can then use PATCH to make permanent any debugger sessions. (Of course, the source code will not reflect these changes until you change the source by using a text editor.) See the Symbolic Debugger in the *VAX/VMS Utilities Reference Volume* for more information.

## 1 Using the Patch Utility

PATCH requires two types of input: the image file to be modified, and the PATCH commands to be executed. You can use PATCH by entering commands during an interactive terminal session. When the PATCH commands are executed, the requested changes are made.

You can also submit a command procedure for PATCH to execute during interactive or batch mode processing. During patching by means of a command procedure, you enter the name of a PATCH command procedure that contains the name of the image file to be patched and the PATCH commands to be applied. A PATCH command procedure can be created during a PATCH session, or with a text editor. See Section 1.2 for detailed information on using a command procedure with PATCH.

The Patch Utility provides a number of commands that allow you to modify the input image file. Some of the commands include the following:

- The ECO commands to initiate, identify, and monitor the patches you apply to an image file. The SET ECO command assigns a unique ECO level to a patch. The CHECK ECO and CHECK NOT ECO commands test whether specified ECO levels have or have not been set.
- The CREATE command to create a PATCH command procedure.
- The UPDATE command to apply a patch to an image file.

All the PATCH commands are explained in detail in the Commands Section.

In addition to the PATCH commands, you can use other PATCH features when modifying the input image file. PATCH is a symbolic utility. You can pass symbol information to PATCH or define new symbols during a PATCH session, then use that symbolic information as a reference tool. See Section 3 for information on using symbols.

When referencing locations in the image file, you can use the entry and display modes that PATCH provides. See Section 4 for more information on entry and display modes.

PATCH provides a storage space called patch area when you need to add additional instructions or data entries. See Section 5 for information on using patch area.

When you patch an image file, PATCH can produce the following three output files:

| File | Description |
|---|---|
| Output image file | An updated image file. This file must be explicitly created 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. This file must be explicitly created using the CREATE command. |

The output image file and command procedure are optional files. You must request them for PATCH to create them. PATCH automatically creates a journal file.

## 1.1 Input Image File

The input image file is the file you want to 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 VAX/VMS operating system. The only restriction is that the input image file must be created by the VAX/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 VAX/VMS software that are not supplied by DIGITAL invalidates the warranty on the DIGITAL-supplied VAX/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 and must be linked with one or more object modules to produce an executable image. You must consider the following restrictions when patching shareable images:

- You can specify only universal symbols when patching a shareable image
- 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 propagates the image section characteristics to the patch area image section descriptor.
- You must, however, use a user-defined patch area to patch position-dependent images.

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, you should use a user-defined patch area rather than the default patch area to store additional instructions and data.

PATCH never alters the input image file. When you invoke PATCH, it creates a copy of the input image file and then incorporates your changes into the copy. You can thus test the new image file and, if it does not run correctly, still have the original file to reevaluate.

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 Command Qualifiers section for information on how to do this.

## 1.2 Command Procedure

You can use a command procedure to apply patches to an image file. 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 file of PATCH commands that, when the command procedure is executed, will apply patches to a particular image file. Usually, you create a command procedure 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.

There are two ways to create command procedures. One way is to use the PATCH command CREATE. The second way is to use an interactive text editor. Both ways are described below.

### 1.2.1 Using the CREATE Command to Create Command Procedures

After you invoke PATCH, issue the CREATE command to request that a command procedure be created and opened. 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 then 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. Hence, when using the PATCH command CREATE to create a command procedure, PATCH changes all symbolic references to absolute values to ensure that the command procedure will execute properly.

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

Section 1.2.3 shows an example of creating a command procedure with the CREATE command.

### 1.2.2 Using Text Editors to Create Command Procedures

You can use a text editor, such as EDT, to construct command procedures. When you do so, the first entry in the command procedure must be the name of the input image file that is to incorporate the patches. Then enter PATCH command lines to the file in the order that you want PATCH to process them.

When you use a text editor to create command procedures, you can use symbolic references as long as PATCH is able to resolve these references when the command procedure is run. If PATCH is unable to resolve a symbolic reference, the command procedure will be prematurely terminated.

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

The following example shows how to create a command procedure that can be used to modify the image file TEST.EXE;2. TEST.EXE;2 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 in TEST.EXE;2 into the PATCH symbol table. After this, you can use the symbol LOOP to indicate the address of the instruction ADDL2 #2,R3.

### 1.2.3 Creating User-Defined Symbols in Command Procedures

PATCH command procedures can contain many patches if each patch starts with a unique ECO level and ends with the UPDATE command. If the patches within a command procedure contain user-defined symbols, you should initialize these symbols at the start of the command procedure. User-defined symbols are symbols defined with the DEFINE command.

The following describes why user-defined symbols should be initialized and shows how to initialize them.

A patch's ECO level is one factor that determines whether or not the patch is applied when the command procedure is executed. If a patch in the command procedure has an ECO level that is already set in the image file to be patched, that particular patch is not applied.

When the command procedure is executed, PATCH parses every command line regardless of whether the command belongs to a patch that is applied. During the parsing operation, PATCH examines each command line and attempts to resolve all symbolic references. If a patch belongs to an ECO level that has already been set in the image file, the commands in the patch are parsed but not applied.

If a DEFINE command occurs in a patch that is not applied, the symbol assignment attempted by that command is not made. Any further references to that symbol will generate an undefined symbol message, causing the command procedure to terminate.

PATCH can, however, resolve further references to the symbol if the symbol was previously initialized. 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
```

The command procedure PATCOM.COM can now 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.

For example, if PATCOM.COM is used to patch an image file in which ECO 1 is set, the first patch (identified by the command SET ECO 1) is not applied. However, PATCH still tries to resolve the symbol FIRST when parsing the DEPOSIT command line. Because FIRST was previously initialized at the start of the procedure, PATCH is able to resolve the symbolic reference. If FIRST had not been initialized, an error would have occurred, and the command procedure would have been terminated.

## 1.3 Output Image File

The output image file is a copy of the image file that has been updated. It consists of all the changes performed by the commands, and the original data that was not altered. An output image file is always created after you enter the UPDATE command. If you fail to 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 issuing 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 consists of the same file name and type as the input image file name and type, and a version number one greater than the highest version of the input image file. Also 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 Command Qualifier Section.

### 1.4 Journal File

The journal file provides a complete record of all PATCH commands issued 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 will also be recorded in the journal file. The journal file provides a record not only of changes actually made, but also of 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. Otherwise, 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. Also 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 Command Qualifier Section.

Entries in a journal file are written as ASCII text, thereby allowing you to examine the contents of the journal file simply by issuing the DCL command PRINT or TYPE.

## 2 Applying Patches

Applying a patch to an image file involves the following steps:

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

You can invoke PATCH to edit an image file from an interactive terminal session or by using a command procedure (see Section 1.2).

Whichever way you invoke PATCH, use the following sequence of PATCH commands for applying patches.

First, establish the ECO level of the patch by using the SET ECO command. Although the use of this command is not mandatory, DIGITAL recommends including it with your patch to make it easier for you and future users to identify the patches made to an image file. More importantly though, using the SET ECO command enables you to process selected patches using command procedures.

Use whichever PATCH commands you need to examine and patch the image file. Then use the UPDATE command to apply the patch to the image file at the specified ECO level. A patch is defined as the commands bounded by the SET ECO and UPDATE commands.

You can then either apply another patch to the image or terminate the patch session with the EXIT command.

The following example shows a patch applied to the image file named 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.

## 3 Using Symbols

When you use PATCH, you must identify the locations that contain the erroneous instructions or data before you can change the code. That is, you must identify the virtual addresses of the errors within the image file. You can do so by by means of a symbolic name defined in the image file when it was created. To ensure that symbol is available for your use, you must include certain 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 and refer to the VAX/VMS Linker in the *VAX/VMS Utilities Reference Volume* to determine how to pass universal symbols to PATCH.

## 3.1 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 5 for more information.

PATCH cannot access local labels in VAX MACRO programs. Local labels are symbols that consist of a number (0 through 65,535) followed by the dollar sign.

## 3.1.1 Global Symbols

Global symbols are symbols defined in one module to be accessed by any number of other modules.

You indicate a global symbol in your source program by delimiting it with a special operator. This operator depends on the language in which your program is written. For example, in VAX MACRO, you define a global symbol by using double colons (::) or double equal signs (==). (Individual language reference manuals contain 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.)

## 3.1.2 Local Symbols

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

You indicate a local symbol in your source program by delimiting it with a special operator. This operator depends on the language in which your program is written. For example, in VAX MACRO, you define a local symbol by using a single colon (:) or an equal sign (=). (Individual language reference manuals contain information on defining local symbols.) To pass local symbol information to PATCH, you must include the debugger when you assemble or compile and link your program. (See the appropriate language manual and the VAX/VMS Linker in the *VAX/VMS Utilities Reference Volume* 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.

## 3.1.3 Module Names, Program Section Names, and Routine Names

Module names, program section names, and routine names are always passed to PATCH unless you specify /NODEBUG and /NOTRACEBACK at link time. When these qualifiers are specified, no symbol information is passed to PATCH.

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

## 3.1.4 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. However, you should declare only a limited subset of global symbols to be universal, because not all symbols are required for a particular application and partial symbol declaration reduces system overhead.

To declare 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 the above example, the symbols A, B, and C are declared universal in the shareable image PROGA. The file named PROGB.OPT is the linker options file in which the universal symbol declarations are contained.

For more information about shareable images and universal symbols, see the VAX/VMS Linker in the *VAX/VMS Utilities Reference Volume*.

When using universal symbols in a shareable image, observe the following guideline: If a symbol is in a transfer vector 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.

Thus, for PATCH to reference locations symbolically in a non-zero-based shareable image that is position independent, you must specify the symbol as

```
symbol + <base>
```

### 3.1.5 Symbolic Instruction Labels

Symbolic instruction labels are 1- to 31-character symbols having 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 period ( . ).

A symbolic instruction label lets you associate a specific name with a particular location without knowing the numeric address of the location. This situation generally arises under three circumstances:

- 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, and one of the new instructions references another new instruction.

Because you do not know the exact location of the instruction being referenced, you can use a symbolic instruction label to reference the location. For example:

```
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
```

Here, a group of instructions is inserted after location 350 by means of a patch area. Note that 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.

## 3.1.6 Side Effect 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, suppose you have the following code in your program:

```
                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 situation, the symbolic instruction label LBL3 is assigned to location 552, and a preceding instruction (BEQL LBL3) refers to LBL3. Later, you discover that the CLRL R5 instruction is wrong and you want to replace it with the instruction CMPL R5,R6. To do this, 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 will automatically move the CMPL R5,R6 instruction to the current patch area, and will replace 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 will also require more bytes than those occupied by the CLRL R5 instruction. To make room for the branch instruction, the instruction following the CLRL R5 instruction will also be moved to the patch area. The instruction that follows the CLRL R5 instruction is the MOVL R2,R3 instruction, which has the symbolic label LBL3 assigned to its location.

Because the MOVL R2,R3 instruction has a symbolic label, you should not let PATCH automatically move this instruction to the patch area. Instead, you should use the REPLACE/INSTRUCTION command to replace both the CLRL R5 and MOVL R2,R3 instructions. Then, you can create a new symbolic label for the MOVL R2,R3 instruction. This is shown 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
```

During the replacement operation, the following events occur:

- 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 is the middle of the branch instruction. If you were to examine LBL3, the following instruction would be displayed:

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

To correct the problem, you need to change previous instructions that refer to LBL3. These instructions should now refer to NEW_LBL3 because the MOVL R2,R3 instruction 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.

### 3.1.7 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.

## 3.2 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. Global and local symbols can be placed in the symbol table if you followed the rules for passing these symbols during compiling or assembling and linking. 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 symls
```

If you passed symbols from your program correctly, you can place them into PATCH's symbol table with one SET MODULE or SET SCOPE command. To confirm that the symbol table contains the appropriate entries use the SHOW MODULE command. This command reports the current symbol table status. See the Commands Section for more information on the SET MODULE, SET SCOPE, SHOW MODULE and SHOW SCOPE commands.

#### Pathnames

Some symbols may have several definitions within your image file. To differentiate the locations defined by these symbols, you must create unambiguous paths that direct PATCH to the locations to which you want to refer. A pathname is a symbolic expression that uniquely defines a location within your file. It has the following format:

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

| | |
|---|---|
| scope | Identifies a particular module within your image file in which PATCH searches for the symbolic name supplied. The backslash character (\) is a required delimiter.<br><br>Supplying a scope is unnecessary when (1) all the symbol names within your file are unique; (2) the scope is already set, by use of the SET SCOPE command, to the region to which you want to refer; or (3) 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. The symbol name is the only element that must always be supplied in order to find that location. |
| 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.

## 3.3 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.

### 3.3.1 Translating Symbols into Address Values

PATCH's 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 First, PATCH compares the symbolic entry with all user-defined symbols, searching for an exact match. User-defined symbols are symbols you create with the DEFINE command during a PATCH session.

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. That is, 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. That is, when PATCH searches the symbol table for a match, it does not prefix the contents of the scope setting to the symbolic entry. (Note that this is the same procedure as step 2, except the mode is set to NOGLOBALS.)

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 to indicate that it could not evaluate the symbolic entry:

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

### 3.3.2 Translating Address Values into Symbols

PATCH's translation of address values into symbolic entries is governed 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 in terms of the corresponding symbol name. If the translation fails, PATCH still evaluates the expression in which the address value appears; however, PATCH reports the address value as a numeric address in terms of the current radix.

## 3.4 Commands That Affect Symbols and Pathnames

PATCH supplies eight commands used specifically for dealing with 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.

## 4 Using Entry and Display Modes

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

- Context modes 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 control the size, in bytes, of data entries and displays.
- Radix modes determine the base in which PATCH displays and interprets addresses and data entries.
- Symbol search modes 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, issue the SET MODE and CANCEL MODE commands or issue 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:

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

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

The difference between the two is that in the first example INSTRUCTION mode becomes a default setting; in the second example INSTRUCTION mode is a temporary setting effective only for that command line. The next command line adheres to 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, issue the SHOW MODE command.

### 4.1 Context Modes

The context modes, which enable or disable a mode setting, are listed below.

| | |
|---|---|
| • INSTRUCTION | NOINSTRUCTION |
| • ASCII | NOASCII |
| • SYMBOLS | NOSYMBOLS |

### 4.1.1 INSTRUCTION-NOINSTRUCTION Modes

The INSTRUCTION-NOINSTRUCTION modes control whether data can be entered and displayed as VAX MACRO instructions (see Section 6.2). The default setting is NOINSTRUCTION.

When INSTRUCTION mode is set, PATCH does all of the following:

- Accepts and displays data as VAX MACRO instructions.
- Accepts and displays instructions in their entirety, and ignores the current length mode setting. PATCH increments the current address by the number of bytes occupied by the instruction.
- Invokes, when you use the REPLACE, INSERT, or DEPOSIT/PATCH_AREA command, an automatic fitting mechanism that takes symbolic instructions and fits them 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.

### 4.1.2 ASCII-NOASCII Modes

The ASCII-NOASCII modes control whether data can be entered and displayed as ASCII data. The default setting is NOASCII. You can 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 all of 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.

### 4.1.3 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 respective pathnames or symbols and accepts pathnames as input.

When NOSYMBOLS mode is set, PATCH displays all locations by their numeric addresses. You can still enter a pathname as input to PATCH; however, PATCH displays that location by its numeric address.

## 4.2 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 two operations:

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

## 4.3 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 when you enter conflicting radix modes on a single command line.

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

## 4.4 Symbol Search Modes

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

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. The default setting is NOGLOBALS.

Note that PATCH follows a certain procedure when performing symbol-to-value translations. This procedure is based on the current setting of the SCOPE-NOSCOPE and GLOBALS-NOGLOBALS modes.

## 5 Using Patch Area

In many cases, applying a patch to an image file requires adding lines of code to the file. A convenient technique for adding code is to insert such code into 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 area: default patch area and user-defined patch area.

A default patch area is created by PATCH and is inserted into an image file following the last normal image section. PATCH will automatically use a default patch area unless you specify otherwise.

A user-defined patch area, on the other hand, is a patch area that you define in the source code for your program. It can be located anywhere within the image file. PATCH will use a user-defined patch area if you issue the SET PATCH_AREA command during a patch session.

PATCH automatically inserts additional instructions into the current patch area (either default or user defined) when there is not enough space at the requested location to insert the code. PATCH then inserts branch instructions to and from the patch area to maintain the correct flow of program execution.

PATCH will not automatically insert new data into the current patch area. You can, however, place data into the current patch area by issuing the DEPOSIT/PATCH_AREA command. Keep in mind 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 when you use that command.

### 5.1 Patch Area Descriptor

Every patch area, whether a user-defined or default 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 issuing the SHOW PATCH_AREA command. For example:

```
PATCH>SHOW PATCH_AREA
current patch area size:       00000200
current patch area address:    00007800
```

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

### 5.2 Patch Area Symbols

PATCH uses the symbols PAA through PAZ to indicate the first free byte of different patch areas used during a patch session. At the start of a patch session, PATCH assigns the label PAA to the first free byte of the default patch area. If user-defined patch areas are also used, PATCH assigns the labels PAB through PAZ to represent the first free byte of these 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 called patch area symbols and are reserved for use by DIGITAL. You are advised not to use these symbols. However, you can create your own patch area symbols using the ALIGN command. This command 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.

## 5.3 Default Patch Area

The default patch area is area 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, as necessary. For example, if you are inserting additional instructions into the image file, PATCH allocates another page of the default patch area if the existing default patch area is too small to accommodate the new instructions.

By convention, whenever you patch an image file and the patches require the use of a patch area, those patches are placed in the default patch area. 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, you should use a user-defined patch area.

## 5.4 User-defined Patch Area

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

You should 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.

### 5.4.1 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. When you issue the SET PATCH_AREA command during a patch session, subsequent patches will be placed in the user-defined patch area, rather than in the default patch area.

The SET PATCH_AREA command can be used with or without the /INITIALIZE qualifier. 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.

### 5.4.2 Terminating the Use of a User-Defined Patch Area

To terminate the use of a user-defined patch area, issue the CANCEL PATCH_AREA command that resets the current patch area to the default patch area.

You can also change the patch area from one user-defined patch area to another user-defined patch area by issuing the SET PATCH_AREA command.

## 5.5 Commands That Affect Patch Area

The PATCH commands that affect patch area are described below.

| 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.

### 5.5.1 DEPOSIT/PATCH_AREA Command

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. You must insert the branch instructions manually. You must also recalculate the relative displacements of all branch-type instructions affected by the insertion of new data or instructions.

### 5.5.2 INSERT/INSTRUCTION Command

When you use the INSERT/INSTRUCTION command, PATCH automatically stores the 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.

Occasionally, when you insert one or more new instructions into your code, the following events may 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.

When 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, suppose the insertion of a new instruction causes the BRB 200 instruction to be moved to the patch area. However, 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.

Be aware 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.

### 5.5.3 REPLACE/INSTRUCTION Command

When you use the REPLACE/INSTRUCTION command, PATCH 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.)

As with the INSERT/INSTRUCTION, the REPLACE/INSTRUCTION command can also 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.

Be aware 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.

## 6 Rules of Syntax

The following section includes the syntax rules of which you must be aware when using PATCH.

### 6.1 Entering ASCII Data Strings

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

When you enter ASCII data strings to PATCH, they must be enclosed within matching quotation marks (") or apostrophes ('). By allowing either quotation marks or apostrophes to be string delimiters, you can include one or the other delimiter as part of the input string. For example, if you want to insert the ASCII text 'AL', you would type

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

You cannot delimit an ASCII data string with the same delimiter as the one you include as part of the string.

When entering an ASCII data string, PATCH will truncate the string if it exceeds the length imposed on it by the current length mode setting. PATCH truncates the string by discarding excessive characters from the right.

### 6.2 Entering VAX MACRO Instructions

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

The rule for the use of quotation marks (") and apostrophes (') described for entering ASCII data strings also applies to entering VAX MACRO instructions. An 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, you must prefix the operand with B^, W^, or L^. For example:

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

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

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.

The following entries show the difference between immediate-mode and forced-immediate-mode addressing:

```
        MOVW    #1,RO
        MOVW    I^#1,RO
        MOVL    #1,RO
        MOVL    I^#1,RO
        .END
```

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

### 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.

The instructions in the table below have the same opcodes.

| Instructions with Equivalent Opcodes | Instruction Displayed by PATCH | Opcode |
|---|---|---|
| BCC<br>BGEQU | BGEQU | 1E |
| BCS<br>BLSSU | BLSSU | 1F |
| BEQL<br>BEQLU | BEQL | 13 |
| BNEQ<br>BNEQU | BNEQ | 12 |
| CLRD<br>CLRG<br>CLRQ | CLRQ | 7C |
| CLRF<br>CLRL | CLRL | D4 |
| CLRH<br>CLRO | CLRO | 7CFD |
| MOVAD<br>MOVAG<br>MOVAQ | MOVAQ | 7E |

| Instructions with Equivalent Opcodes | Instruction Displayed by PATCH | Opcode |
|---|---|---|
| MOVAF<br>MOVAL | MOVAL | DE |
| MOVAH<br>MOVAO | MOVAO | 7EFD |
| PUSHAD<br>PUSHAG<br>PUSHAQ | PUSHAQ | 7F |
| PUSHAF<br>PUSHAL | PUSHAL | DF |
| PUSHAH<br>PUSHAO | PUSHAO | 7FFD |

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

## 6.3 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. In addition, do not enclose the data within matching quotation marks or apostrophes.

You can, however, enter numeric data concurrently with instructions provided you are using the assembler directives listed below:

```
.BYTE
.WORD
.LONG
```

The following example illustrates such an insertion:

```
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 hexadecimal number starting with an alphabetic character (that is, A through F) with a 0. Otherwise, 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
```

## 6.4 Delimiting Parameter Values

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

Use a comma to separate parameter values in commands that can take multiple parameters.

### Entering Comments

To insert comments, prefix them with the exclamation point (!) character. When PATCH reads an exclamation point, it ignores all following information on that line. The only restriction on entering comments is that a comment cannot appear on the same line as a command string that uses a continuation character (-). Comments will be entered into the journal file.

## 6.5 Special Operators For Arithmetic Expressions

This section describes the special operators that PATCH recognizes in 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 | 1. Priority operators<br>2. Bit field delimiters for the EVALUATE command |
| ^H | Radix operator | Radix operator for hexadecimal notation |

| Operator | Name | Function |
|---|---|---|
| ^D | Radix operator | Radix operator for decimal notation |
| ^O | Radix operator | Radix operator for octal notation |
| ^B | Radix operator | Radix operator for binary notation |

## 6.6 Special Operators For Addressing Locations

This section describes the special operators that PATCH recognizes for locating addresses and gives examples of how to use these operators.

| Name | Operator | Function |
|---|---|---|
| Current location operator | . | Refers to the location last specified; that is, 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; subtract, from the current location, the number of bytes indicated by the length mode |
| 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 |

## COMMAND QUALIFIERS

This section describes the optional 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 absolute virtual addresses relative to the beginning of the file. This feature allows replacement of existing data with new data of the same length. If the data is smaller 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. Any PATCH operation that results in a data replacement longer than the length of the original data generates an error message and terminates the command in progress; either the PATCH or DCL prompt is then displayed, whichever is appropriate.

Also, note that there is no default patch area, and none will be created, because of the tendency to corrupt a file. Patch area is meaningless in other than an image file.

If you patch a file in absolute mode, remember that that there are no symbols available to assist you in locating data locations. You must exercise great care to ascertain that the correct locations are modified.

Most PATCH commands will work in their normal fashion. However, only REPLACE and DEPOSIT should be used for write operations; other commands are acceptable for read operations. Commands that attempt to expand the file, such as ALIGN and INSERT, should be avoided because they will probably corrupt the file. (These commands will be trapped by PATCH and 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
```

An example of the /ABSOLUTE qualifier with /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 Command Qualifiers section shows the use of /ABSOLUTE with /NONEW_VERSION.

## /JOURNAL

Overrides the default journal file specification.

### FORMAT

**/JOURNAL** *[=file-spec]*

### qualifier value

***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 create 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
```

This command invokes PATCH for an interactive PATCH 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; that is, 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, that is, a new version of the file will be created. /NONEW_VERSION will need to be used when patching large data files when there is not enough disk space to create a new version of the patched file.

Note: **If /NEW_VERSION is specified, the file will be overwritten. No attempt on the part of the user, including pressing CTRL/Y, will prevent this result. Therefore, you should have a back up copy of the file before making any attempt to patch it.**

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

### EXAMPLE

```
$ PATCH/ABSOLUTE/NONEW_VERSION  LIN.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]GIN.COM;1 being overwritten
$
```

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, that is, (file) "being overwritten." The example for the /ABSOLUTE qualifier in the Command 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]*

### qualifier value

***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 issue the PATCH command UPDATE at the end of the PATCH session. You can issue 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 [, . . . ])]*

**qualifier value**

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

**DESCRIPTION**

The /UPDATE command qualifier lets you apply to an image file only the patches that correspond to the ECO levels specified with /UPDATE. Typically, you use the /UPDATE qualifier when processing a PATCH command procedure. However, you can also use it when invoking 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. In either case, the file specification is required.

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

### Processing Selected Patches in Command Procedures

PATCH assumes that the file is a command procedure if the file specification is preceded by an at sign (@). PATCH reads the command procedure and parses every command in the command procedure. 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. When PATCH parses each command line to look for the patches specified by the /UPDATE qualifier, PATCH may be unable to resolve symbolic references. If this occurs, PATCH terminates the command procedure.

You can, however, use ECO commands to apply selected patches in command procedures that contain user-defined symbols.

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

- The name of the image file to be patched
- The PATCH introductory message
- The patches applied to the image file
- The 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 [MATTHEWS.FORTRAN]PROGB.EXE
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
$
```

This 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 display 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.

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.

### Processing Selected Patches Interactively

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. For example:

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
```

In the above example, PATCH allows processing of the patch identified by the ECO level 2; however, 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 also 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 from PATCH.

## /VOLUME

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

### FORMAT

**/VOLUME** *=[n]*

### qualifier value

*n*
Specifies the relative volume number of a multivolume set.

### DESCRIPTION

The /VOLUME command qualifier lets you specify a relative volume number where the output file will be placed.

If you specify /VOLUME without a number, the number defaults to 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.

## COMMANDS

This section describes the 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*

### command parameter

#### *symbol-name*

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

### command 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 disjointed patch area for it to be used.

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.

You must enter only one alignment qualifier for each ALIGN command.

## EXAMPLES

1

```
PATCH>ALIGN/QUAD MOD_1
old patch area size:           0001E3
old patch area address:        0000081D
new patch area size:           000001E0
new patch area address:        00000820
symbol " MOD_1"  defined as      00000820
```

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:        000000
symbol " PATAREA"  defined as  00001800
```

This 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**

**command parameters**

*None.*

**command 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 [, . . . ]*

### command parameter

***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.

### command qualifier

***/ALL***

Removes all local symbol information from the symbol table.

### DESCRIPTION

Use CANCEL MODULE to remove local symbol information from the PATCH symbol table; this command 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, issue 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 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
```

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
```

This 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>
```

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 back to the default patch area.

### FORMAT

**CANCEL PATCH_AREA**

**command parameters**

*None.*

**command qualifiers**

*None.*

### DESCRIPTION

Use CANCEL PATCH_AREA to reset the current patch area from a user-defined patch area to the default patch area. Any patch area needed thereafter will be taken from the default patch area until the next SET PATCH_AREA command is issued.

The CANCEL PATCH_AREA command is used primarily when you have issued 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.

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 PATCH to return an error message indicating that it cannot insert the patch into the patch area.

### EXAMPLE

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

The SET PATCH_AREA command establishes the user-defined patch area AREA1 as the current patch area. After depositing the necessary data into AREA1, the CANCEL PATCH_AREA command is issued to resume use of the default patch area.

## CANCEL SCOPE

Cancels the current symbolic scope.

### FORMAT

**CANCEL SCOPE**

#### command parameters

*None.*

#### command 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>
```

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] [, . . . ]*

### command parameter

***eco-level***

Indicates one or more ECO levels that have been set. ECO levels can be entered in either lists, which are separated by commas, or ranges, which are separated by colons.

Both lists and ranges can be specified with 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> ).

### command 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 previously set in the image file, PATCH will not apply the current patch. PATCH will 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 issue 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>
```

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 following PATCH prompt indicates that the specified patches have been previously set, so PATCH will continue to execute commands. The patch associated with ECO level 18 will be applied after the UPDATE command is issued.

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
```

The first CHECK ECO command checks that the patch associated with ECO level 12 has been applied to the image file. The display 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] [, . . . ]*

### command parameter

*eco-level*
Indicates the ECO levels that are not set. ECO levels can be entered in either lists, which are separated by commas, or ranges, which are separated by colons.

Both lists and ranges can be entered with 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> ).

### command qualifiers

*None.*

### DESCRIPTION

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

The CHECK NOT ECO command is the negation of the CHECK ECO command. It too 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 (that is, it has already been set in the image), the current patch will not be applied. PATCH will not execute any subsequent commands until it encounters the next SET ECO command.

### EXAMPLES

**1**

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

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 response to the CHECK NOT ECO command, PATCH indicates 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 command procedures of PATCH commands. All subsequent successful PATCH commands that modify the image file will be written to this command procedure.

### FORMAT

**CREATE** *[file-spec]*

### command parameter

***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 default values listed below 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 that 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, you must set your default to the directory that contains the input image file before you process the command procedure.

### command qualifiers

*None.*

### DESCRIPTION

Use CREATE to create a command procedure that contains all PATCH commands successfully executed after the CREATE command.

Command procedures facilitate patching several copies of the same image file. There are two ways to create command procedures. One way is to specify the CREATE command when you invoke PATCH. All subsequent, successful commands are applied to the image file and recorded in the command procedure. A second way is to use a text editor.

When you use CREATE to create a command procedure, PATCH automatically inserts, as the first entry in the command procedure, the name of the image file that will incorporate the patches. All symbolic names are converted to absolute values, and all command names and qualifiers are truncated to their shorthand notation.

You can issue only one CREATE command per PATCH session. To create another command procedure, close the input image file and then reopen it.

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

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

In the above command, the file-spec 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 the above 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, . . . ]*

### command 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/or periods (.). (PATCH does not distinguish between uppercase and lowercase letters; that is, the value ABC is equivalent to the value abc.)

The symbol name cannot be a pathname.

#### *value*

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

### command qualifiers

*None.*

### DESCRIPTION

Use DEFINE to equate a symbolic name to a specific value and place the symbolic name in the PATCH symbol table. Once the assignment has been performed, 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 always searches the symbol table for these symbolic names first 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. Then, when you specify the symbolic name, the most recent value that the symbol denotes is displayed.

One restriction applies to the use of the DEFINE command. You cannot specify the /INSTRUCTION or /ASCII mode qualifiers, nor can you 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
```

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
```

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 a piece of data.

### FORMAT

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

### command 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 entries or instructions to be deleted. The data or instructions you specify must be the actual contents.

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

### command qualifiers

#### */BYTE*

Specifies that data is deleted in byte lengths. 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 word lengths. 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 longword lengths. 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, enclose the data within matching quotation marks (") or apostrophes ('). The current radix setting has no effect on the ASCII data being deleted; however, 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, enclose the instruction within matching 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 (exactly as entered) is used 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 DELETE to delete an instruction or piece of data from one location or from several consecutive locations in terms of the current mode settings.

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
```

The DELETE command replaces the instruction CMPB (R0),(R5) with NOP instructions.

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

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 [, . . . ]*

### command 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.

### command qualifiers

#### */BYTE*

Specifies that data is deposited in byte 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.

#### */WORD*

Specifies that data is deposited in word 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.

#### */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, enclose the data within matching quotation marks (") or apostrophes ('). The current radix setting has no effect on the ASCII data being deposited; however, 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, enclose the instruction within matching 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 (exactly as entered) is used 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
Use DEPOSIT to deposit new data or instructions into one or more consecutive locations.

When depositing data or instructions, you can replace the contents of a location or of several consecutive locations in terms of the current mode settings. The DEPOSIT command does not request verification of the current contents before replacing the contents with new data or instructions (that is, this command assumes that you know what you are doing). In cases when you want to confirm the data or instructions that will be overwritten, use the REPLACE command.

When you are adding instructions to an image file, it is easier to use the INSERT command. This command performs automatic branching to and from the patch area.

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, issue 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.

Unlike the INSERT and REPLACE commands, the DEPOSIT/PATCH_AREA command requires that you insert the branch instructions into the appropriate locations to maintain the logical flow of program execution to and from the patch area.

## EXAMPLES

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

The DEPOSIT command requests that a byte of ASCII data be deposited in location 1111. The DEPOSIT display indicates that the data was successfully deposited and that the old contents was B.

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

The DEPOSIT command deposits an instruction into location 413. The /INSTRUCTION qualifier is specified to indicate that an instruction is being deposited. The DEPOSIT display 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'
```

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 then issued to deposit into NEW_PATCH at PAT1 the ASCII data RUN and SKIP.

## EVALUATE

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

### FORMAT

**EVALUATE** *expression [, . . . ]*

### command parameter

#### *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. As explained above, when you evaluate a selected bit field in a value, the format of the expression is

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

### command qualifiers

#### */BYTE*

Specifies that data is evaluated in byte lengths. 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 word lengths. 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 longword lengths. 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 (exactly as entered) is used 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

You can use the EVALUATE command to perform binary and unary arithmetic operations. The EVALUATE command interprets expressions and displays results in the current length and radix modes.

You can use the EVALUATE command to determine the value associated with a symbol or pathname. The values are displayed in terms of the current length and radix mode setting.

You can use the EVALUATE command to display the current contents of a specific bit field in a value. The syntax for this command is

```
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
```

This 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
```

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

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

This 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
```

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 in terms of the current mode settings.

**FORMAT** **EXAMINE** *location [:location] [, . . . ]*

**command parameter**

### *location*

Specifies one or more locations whose contents are to be displayed. Several locations can be specified in a comma-separated list or colon-separated range. Both lists and ranges can be specified 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 will be displayed. The next sequential location depends on the current mode settings.

**command qualifiers**

### */BYTE*

Specifies that data is displayed in byte lengths. 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 word lengths. 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 longword lengths. 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 (exactly as entered) is used 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 EXAMINE to display the contents of the specified locations in terms of the current mode settings.

You can also 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. To do so, you use the backslash operator (\).

## EXAMPLES

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

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 '
```

The EXAMINE command requests a display in ASCII of all the 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 the above sequence of commands, 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.

Still 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.

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

The first EXAMINE command requests that the contents of the current location (represent 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 seion. It also ends a repetitive prompt such as NEW> , OLD> , or ECO> .

### FORMAT

EXIT

**command parameters**

*None.*

**command qualifiers**

*None.*

### DESCRIPTION

Use EXIT to terminate a repetitive prompt such as NEW>, OLD>, or ECO>, or to terminate a PATCH session and pass control back to the command interpreter. (You can also specify CTRL/Z to terminate a PATCH session. Press CTRL/Z in response to the PATCH prompt (PATCH>).)

Do not type EXIT in response to the value prompt (VAL>) for the DEFINE command. For this command, only the name prompt (NAM>) recognizes the EXIT command.

### EXAMPLES

1

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

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, shown here as a dollar sign ($), 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>
```

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 any of the other PATCH commands.

### FORMAT

**HELP** *topic [subtopic . . . ]*

#### command parameters

***topic***
Specifies the name of the command with which you need 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 or parameters or both.

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 the 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.

#### command qualifiers

*None.*

### DESCRIPTION

Use HELP to display the following information about any PATCH command: a description of the command, the format of the command, the qualifiers that may be specified with the command, and the parameters that may be specified with the command. In addition, the HELP command provides information about modes and expressions.

If you want information about a particular qualifier or parameter, specify it as a subtopic. 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.

### 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 . . .*

### command parameters

#### *location*

Specifies the address after which one or more new instructions are 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.

### command 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 (exactly as entered) is used 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 INSERT to insert VAX MACRO instructions into specific locations within an image file. To use this command, you must specify the /INSTRUCTION qualifier or you must 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 (that is, the current instruction). New instructions are inserted after the current instruction you specify.

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 last new instruction is always 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.

### Calculating the Location for the Branch 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 patch area. If the 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.

On the other hand, if the current instruction is longer than the branch instruction, the 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 patch area to make room for a branch instruction.
2. Moves the instructions calculated in step 1 to the current patch area, and inserts in their places a branch instruction. 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 into the patch area a branch instruction that reroutes 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 in terms of the current mode settings.

### FORMAT

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

### command 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 be the actual contents.

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.

### command qualifiers

#### */BYTE*
Specifies that data is replaced in byte lengths. 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 word lengths. 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 longword lengths. 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, enclose the data within matching 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, enclose the instructions within matching 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 (exactly as entered) is used 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 REPLACE to replace the contents of one or more locations with new instructions or data in terms of the current mode settings. 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 and 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. This means, for example, that if you confirm the contents of six consecutive locations, you can replace the contents of only those six locations. If the number of replacement entries is less than the number of existing entries, the remaining locations are filled with zeros.

In addition, 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.

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

## 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
```

The instruction occupying location 600 is replaced with the instruction CMPB R2,R4. This instruction occupies more bytes than the instruction TSTL R4. Thus, 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
```

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*

### command parameter

***eco-level***
Specifies a decimal integer between 1 and 128, inclusive. ECO levels outside this range of integers are illegal.

### command qualifiers

*None.*

### DESCRIPTION

Use SET ECO to define the ECO level for the patch that you are developing.

Whenever you apply a patch to an image file, the first command you type should be a SET ECO command. This command provides you with a way to identify your patches easily. More important, however, it lets you 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. Patches not represented by ECO levels are not processed.

You can issue as many SET ECO commands as necessary during a PATCH session; however, once you specify an ECO level, that level can never be used again for that particular image file. Furthermore, whenever you begin a patch with a SET ECO command, you must also terminate the patch with the UPDATE command. If you fail to issue the UPDATE command (1) the ECO level specified with the SET ECO command is not set (hence, the patch is not applied to the image file), and (2) you cannot issue 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
```

The SET ECO command defines the ECO level for the subsequent patch as 15. The commands are then entered. When the UPDATE command is issued, 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 other PATCH commands that you enter, as well as the values that PATCH displays.

### FORMAT

**SET MODE** *mode [, . . . ]*

### command parameter

#### *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 byte lengths. The length mode qualifiers affect only ASCII and numeric data. |
| WORD | Specifies that data is interpreted in word lengths. The length mode qualifiers affect only ASCII and numeric data. |
| LONG | Specifies that data is interpreted in longword lengths. 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 | Control whether data is to be interpreted and displayed as ASCII data. When you specify ASCII, enclose the data within matching quotation marks (") or apostrophes ('). The initial default setting is NOASCII. |
| [NO]INSTRUCTION | Controls whether the data is interpreted and displayed as VAX MACRO instructions. When you specify INSTRUCTION, enclose the instruction within matching 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. |

| Mode | Description |
|---|---|
| [NO]GLOBALS | Controls whether the symbolic entry (exactly as entered) is used 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. |

## command 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
```

This 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'
```

This SET MODE command establishes INSTRUCTION mode as a current mode setting. A subsequent DEPOSIT command can then deposit an instruction without specifying the /INSTRUCTION mode qualifier.

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

In this example, the SET MODE continuously prompts for new modes until EXIT is typed.

## SET MODULE

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

### FORMAT

**SET MODULE** *module-name [, . . . ]*

### command parameter

***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.

### command qualifier

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

### DESCRIPTION

Use SET MODULE to enter local symbol information from the specified modules into PATCH's symbol table, provided you followed the rules for passing local symbol information to PATCH.

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 '
```

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
```

This 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 in favor of the patch area that you defined at assembly or compile time.

### FORMAT

**SET PATCH_AREA** *address-of-descriptor*

### command parameter

#### *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 issue 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 will locate 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.

### command qualifier

#### */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, in bytes, of the patch area. 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 will use 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

Use SET PATCH_AREA to override the use of the default patch area in favor of the patch area you defined at assembly or compile time. A user-defined patch area, like the default patch area, must have a patch area descriptor.

There are two ways to set up a user-defined patch area in the source program. First, you can initialize the patch area at assembly time. When you access this type of user-defined patch area, do not use the /INITIALIZE qualifier.

Alternatively, you can initialize the patch area at patch time. When you access the user-defined patch area, use the /INITIALIZE qualifier. PATCH will create the descriptor and locate it in the first eight bytes of the patch area.

If you plan to use the SET PATCH_AREA command without the /INITIALIZE qualifier, you must 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 will not be shareable and will incur additional overhead during image activation.

If you plan to use the SET PATCH_AREA command with the /INITIALIZE qualifier, you need not define the patch area descriptor in your program. You should, however, declare the size and starting address of the patch area as global symbols. When you issue the SET PATCH_AREA/INITIALIZE command during a patch session, PATCH will create the patch area descriptor and place 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 following sections show the two ways to set aside a patch area in a VAX MACRO program.

### 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.

In PATCH, issue 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.

This 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 issuing the SET PATCH_AREA command followed by the address of the patch area descriptor. In this example, you would access user-defined patch area like this:

```
PATCH>SET PATCH_AREA PATCH_DESC
```

In order to use the global symbol PATCH_DESC, make sure you follow the rules for passing symbols to PATCH and placing symbols in PATCH's symbol table.

## The SET PATCH_AREA Command With /INITIALIZE

When you use the /INITIALIZE qualifier PATCH creates a patch area descriptor and also 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 will place the patch area descriptor in the first eight bytes of the patch area.

In PATCH, issue the SET PATCH_AREA command with the /INITIALIZE qualifier, as shown below:

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

The size-expression indicates the size, in bytes, of the patch area. This should include eight bytes for the descriptor. The address-of-descriptor is the address where PATCH will locate the patch area descriptor. As explained earlier, PATCH builds the descriptor and places it in the first eight bytes of the user-defined patch area.

Both the size-expression and the address-of-descriptor may 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.

This 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 GIN      ;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 this example, the first free byte is the byte immediately following the descriptor.

If you specify an invalid size expression, PATCH will compute and use 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 Computing 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 Subtracting eight bytes for the patch area descriptor.

3 Subtracting 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 will substitute the default size when it builds the descriptor, as long as the default size is at least 12. PATCH will issue the following informational message:

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

If you issue 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 does, however, establish the specified user-defined patch area as the current patch area. PATCH 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
```

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 LRD-PATCHES. 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
```

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 [\ . . . ]]*

### command parameter

***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.

### command qualifiers

*None.*

### DESCRIPTION

Use SET SCOPE to establish the specified module name (and routine name, if specified) as the explicit scope to be used under the rules for translating pathnames and symbols into values.

PATCH also inserts local symbol information associated with the specified module into the symbol table when you type the SET SCOPE. 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
```

This set command establishes the scope as CARP_LPT.

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

In this example, there is no such module named LIP_SIGN. Therefore, the SET SCOPE command fails and the previous scope setting is retained.

## SHOW MODE

Displays the modes that are currently set.

### FORMAT

**SHOW MODE**

**command parameters**

*None.*

**command qualifiers**

*None.*

### DESCRIPTION

Use SHOW MODE to report the modes that are currently set. This command is used primarily with the SET MODE and/or CANCEL MODE commands. It indicates the condition of the current modes, enabling you to change one or more of them if necessary.

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

### EXAMPLE

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

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**

**command parameters**

*None.*

**command qualifiers**

*None.*

### DESCRIPTION

Use SHOW MODULE to request PATCH to display all the modules in the image file and indicate whether the symbols contained in the modules are available for use. It also indicates the amount of symbol table space required by each module and routine and the total amount of unused space remaining 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.
```

The first SHOW MODULE command indicates that there are two modules in the current image file, though neither one 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**

**command parameters**

*None.*

**command qualifiers**

*None.*

### DESCRIPTION

Use SHOW PATCH_AREA to report the size and starting address of the current patch area in hexadecimal representation, regardless of the current radix setting. Generally, you use this command with either the ALIGN command or the DEPOSIT/PATCH_AREA command. In both cases, you want to know 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'
```

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 the patch area.

## SHOW SCOPE

Displays the current scope setting.

### FORMAT

**SHOW SCOPE**

**command parameters**

*None.*

**command qualifiers**

*None.*

### DESCRIPTION

Use SHOW SCOPE to display the current scope setting.

### EXAMPLE

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

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**

**command parameters** *None.*

**command qualifiers** *None.*

### DESCRIPTION

Use UPDATE to apply a patch to the image file. The UPDATE command is a patch terminator. This command applies the previous patch to the image file and thus creates an output image file. If you fail to issue the UPDATE command, no output image file is created.

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

If a SET ECO command is issued, a subsequent 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
```

The UPDATE command applies the previous patch to the image file HOROSCOPE.EXE. The UPDATE display indicates that the image file has been successfully updated.

## VERIFY

Confirms that a location (or several consecutive locations) contains the specified contents–either data or instructions.

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

**command 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 be the actual contents.

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

**command qualifiers**

### */BYTE*

Specifies that data is verified in byte lengths. 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 word lengths. 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 longword lengths. 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, enclose the data within matching 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, enclose the instruction within matching 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 (exactly as entered) is used 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 VERIFY to confirm that a location or several consecutive locations contain the specified contents (instructions or data).

Generally, you should use the VERIFY command with the DEPOSIT command. The DEPOSIT command does not confirm the entries before overwriting them. Therefore, to make certain you know the contents of the locations that will be modified, issue the VERIFY command first.

The VERIFY command is also useful when you are patching an image file by means of a command procedure. Using the VERIFY command, you can check particular locations before attempting to modify them. 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'
```

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.

## EXAMPLES

This example 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. You will see how to patch the image file to correct the error.

```
0000            .TITLE  NEGATION - Calculates negation of decimal integer
0000            .IDENT /01/
0000            .SUBTITLE - Pure Data
0000            .PSECT RO_DATA,NOWRT,NOEXE,LONG
0000    PROMPT: .ASCII  /Enter decimal integer: /
000C
0017    PROMPT_LEN = .-PROMPT
0017    RESULT_FAO:
0017            .ASCID /The negation is !SL./
0025
0031
0033            .SUBTITLE - Impure Data
0000            .PSECT RW_DATA,NOEXE,LONG
0000    BUFFER_SIZE     =       132             ;Buffer size
0000    BUFFER_DESC:
0000            .LONG   BUFFER_SIZE,BUFFER      ;Descriptor for buffer
0008    BUFFER:
0008            .BLKB   BUFFER_SIZE             ;Buffer to get line
008C    INTEGER:
008C            .BLKL   1                       ;Input integer
0090    RESULT:
0090            .BLKL   1                       ;Result of operation
0094
0094            .SUBTITLE - Define RAB and FAB for terminal I/O
0094    TRMFAB: $FAB    FNM=TT:,-       ;FAB for terminal
0094                    FAC=<PUT,GET>,- ;Use for input and output
0094                    RAT=CR
00E4
00E4    TRMRAB: $RAB    FAB=TRMFAB,-    ;RAB for terminal
00E4                    UBF=BUFFER,-
00E4                    USZ=BUFFER_SIZE,-
00E4                    ROP=PMT-
00E4                    PBF=PROMPT,PSZ=PROMPT_LEN
0128
0128            .SUBTITLE - Program Entry Point
0000            .PSECT RO_CODE,EXE,NOWRT
0000
0000            .ENTRY  ENTRY_POINT, ^M<>
0002            $OPEN   FAB=TRMFAB              ;Open terminal file
000F            $CONNECT        RAB=TRMRAB      ;Connect record stream
001C            $GET    RAB=TRMRAB              ;Get input from terminal
0029            PUSHAL  INTEGER                 ;Push integer address on
002F            PUSHL   RAB$L_RBF+TRMRAB        ;Push starting address of
0035                                            ;input on stack
0035            MOVZWL  RAB$W_RSZ+TRMRAB,-(SP)  ;push length of input on stack
003C            CALLS   #3,G^LIB$CVT_DTB                ;Convert to binary
0043            MOVL    INTEGER,RESULT          ;Calculate negation
004E
004E    ;determine length and contents of result
004E            $FAO_S  CTRSTR=RESULT_FAO,-             ;Use system service to
004E                    OUTLEN=TRMRAB+RAB$W_RSZ,-       ;convert binary result
004E                    OUTBUF=BUFFER_DESC,-            ;return ASCII character
004E                    P1=RESULT                       ;in output string
006D
```

```
006D    52 ;output result
006D    53          MOVAB    BUFFER,TRMRAB+RAB$L_RBF
0078    54          $PUT     RAB=TRMRAB
0085    55          MOVZBL   #SS$_NORMAL,RO
0089    56          RET
008A    57          .END     ENTRY_POINT
```

The program will assemble and link, but when executed will produce erroneous results, as follows:

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

Locate the incorrect code, by using the VAX/VMS Symbolic Debugger or simply by examining the listing. You can invoke PATCH to make a permanent change in the image file as shown below. The numbers to the right of the example are keyed to the explanations that follow the example.

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

❶ Invoke PATCH—Invoke PATCH for an interactive terminal session by typing the DCL command line PATCH NEGATION.EXE;1. PATCH, in return, displays an introductory message on your terminal. This message indicates the version of PATCH that you are using and its release date.

❷ Define an ECO level—The SET ECO command defines the ECO level for the ensuing patch. Note that the ECO level is not set until you issue the UPDATE command.

❸ Alter the symbol status—The SET MODULE/ALL command inserts all local symbol information into the symbol table, provided that you followed the rules for passing local symbols at compile or assembly and link time.

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

❺ Alter entry and display mode status—The SET MODE command sets the INSTRUCTION mode, causing PATCH to display the contents of the specified locations as VAX MACRO instructions.

❻ Change the code—A number of PATCH commands can be used to modify code. Each command alters the code in a unique fashion. In this example, the REPLACE command was used to overwrite the existing instruction with a new, correct instruction.

❼ Set the ECO level and update the image file—After you correct the code, issue the UPDATE command. This sets the specified ECO level and applies the patch to a new version of the image file.

❽ Exit from PATCH—Type the EXIT command to end the PATCH session and return control to the VAX/VMS command interpreter.

Now run the program NEGATION.EXE;2 and it will execute properly.

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

# Index

VAX/VMS
Patch Utility
Reference Manual
AA-Z426A-TE

## READER'S COMMENTS

**Note:** This form is for document comments only. DIGITAL will use comments submitted on this form at the company's discretion. If you require a written reply and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Did you find this manual understandable, usable, and well organized? Please make suggestions for improvement.

Did you find errors in this manual? If so, specify the error and the page number.

Please indicate the type of user/reader that you most nearly represent:

- ☐ Assembly language programmer
- ☐ Higher-level language programmer
- ☐ Occasional programmer (experienced)
- ☐ User with little programming experience
- ☐ Student programmer
- ☐ Other (please specify) ____________

Name ____________ Date ____________

Organization ____________

Street ____________

City ____________ State ____________ Zip Code or Country ____________

Do Not Tear - Fold Here and Tape

digital

No Postage Necessary if Mailed in the United States

**BUSINESS REPLY MAIL**
FIRST CLASS PERMIT NO.33 MAYNARD MASS.

POSTAGE WILL BE PAID BY ADDRESSEE

SSG PUBLICATIONS ZK1-3/J35
DIGITAL EQUIPMENT CORPORATION
110 SPIT BROOK ROAD
NASHUA, NEW HAMPSHIRE 03062-2698

Do Not Tear - Fold Here

Cut Along Dotted Line