DataGeneral

ADDENDUM

Addendum to
SWAT™ Debugger
User's Manual

086-000045-00

This addendum updates manual 093-000258-01. See Updating Instructions on reverse.

Ordering No. 086-000045

Printed in the United States of America
Revision 00, November 1982

093-000258-01

NOTICE

Addendum to
SWAT™ Debugger
User's Manual
086-000045

Updating Instructions

This addendum contains new product information for the SWAT™ debugger.

To update your copy of 93-000258-01, please remove manual pages and insert addendum pages as follows:

Remove	Insert
Title/Notice page	Title/Notice page
iii through x	iii through x
1-9/1-10	1-9/1-10
2-1 through 2-4	2-1 through 2-4
3-1/3-2	3-1/3-2
4-1 through 4-4	4-1 through 4-4
4-13/4-14	4-13/4-14
4-23 through 4-28	4-23 through 4-28
A-3 through A-6	A-3 through A-6
C-1/C-2	C-1/C-2
C-41/C-42	C-41 through C-72
Index-1 through Index-3	Index-1 through Index-4

In the margins of replacement pages, a vertical bar or an asterisk in the margin of a page indicates substantive technical change or deletion, respectively from 093-000258-01.

The addendum number appears on all pages in this addendum.

SWAT™ Debugger
User's Manual

093-000258-01

For the latest enhancements, cautions, documentation changes, and other information on this product, please see the Release Notice (085-series) supplied with the software

Ordering No. 093-000258

Printed in the United States of America
Revision 01, May 1982

093-000258-01

Preface

The SWAT™ debugger is a high-level, interactive symbolic debugging system. It allows you to debug high-level language programs. You can use the SWAT debugger to validate a program at the source language level, rather than at the assembly or machine language level.

Who Should Read This Manual?

We have written this manual for any programmer who is developing and testing programs in one or more of the high-level languages that the SWAT debugger supports. We assume you have a good working knowledge of the operating system and your programming language. Realizing that many programmers have never used a high-level debugger before, we designed this manual to serve as a tutorial aid for the learner, and as a reference for those with more debugging experience.

We have organized the manual as follows:

Chapter 1	Introduces the SWAT debugger. It tells you what the SWAT debugger is, what it can do, and explains SWAT concepts.
Chapter 2	Shows you how to get the SWAT debugger running. It describes what you'll need to do before sitting down to debug a program.
Chapter 3	Presents a sample debugging session that introduces you to all the SWAT commands. We explain what the SWAT commands do, and demonstrate how to use them.
Chapter 4	Describes the SWAT commands in detail. This is a reference section that the experienced SWAT user can use to find answers to questions about the commands.
Appendix A	Provides a number of troubleshooting tips. We list things to check when you have trouble invoking or using the SWAT debugger. It also describes some of the differences between the AOS and AOS/VS SWAT debuggers.
Appendix B	Describes the errors that may occur during a SWAT debugging session. We give a full explanation for all error messages you may encounter.
Appendix C	Presents additional sample programs and audit files.

This manual is a revision, replacing the previous manual (093-258-00). A vertical bar in the outside margin indicates a change or addition to the technical information. An asterisk flags a substantial deletion.

Related Manuals

As we said earlier, we assume you are familiar with the operating system and the programming language you are using. To supplement your reading of this manual, you may want to refer to one or more of the following:

AOS Programmer's Manual (093-000120)
AOS Debugger and Disk File Editor User's Manual (093-000195)
AOS Link User's Manual (093-000254)

NOTICE

SWAT™ Debugger
User's Manual
093-000258

Revision History:	Effective with:
Original Release - September 1980	
First Revision - May 1982	(SWAT™ Rev. 2.0) (AOS/VS Rev. 2.0) (AOS Rev. 3.30)
Addendum 086-000045-00 - November 1982	(AOS SWAT™ Rev 2.1) (AOS/VS SWAT™ Rev 2.2)

A vertical bar or an asterisk in the margin of a page indicates substantive change or deletion, respectively from the previous revision.

AOS/VS Programmer's Manual (093-000241)
AOS/VS Link and Library File Editor User's Manual (093-000245)
AOS/VS Debugger and File Editor User's Manual (093-000246)

Command Line Interpreter User's Manual (AOS, AOS/VS) (093-000122)

COBOL Reference Manual (AOS/VS) (093-000289)

FORTRAN 77 Reference Manual (093-000162)

PL/I Reference Manual (AOS) (093-000204)
PL/I Reference Manual (AOS/VS) (093-000270)

AOS/VS PASCAL Reference Manual (093-000290)
AOS/VS C Language Reference and Runtime Manual (093-000264)

What Do You Think?

At the end of this manual you'll find a Remarks Form. This is your direct line to us in User Documentation — please take advantage of it. We want to know what you like and dislike about the manual. We welcome your suggestions, and *we really listen!* Only when the manual does its job can it help you do yours. So, help us help you.

Reader, Please Note:

We use these conventions for command formats in this manual:

COMMAND required *[optional]* ...

Where	**Means**
COMMAND	You must enter the command (or its accepted abbreviation) as shown.
required	You must enter some argument (such as a filename). Sometimes, we use: $\left\{\begin{matrix}\text{required}_1\\ \text{required}_2\end{matrix}\right\}$ which means you must enter *one* of the arguments. Don't enter the braces; they only set off the choice.
[optional]	You have the option of entering this argument. Don't enter the brackets; they only set off what's optional.
...	You may repeat the preceding entry or entries. The explanation will tell you exactly what you may repeat.

Additionally, we use certain symbols in special ways:

Symbol	**Means**
)	Press the NEW LINE or carriage return (CR) key on your terminal's keyboard.
□	Be sure to put a space here. (We use this only when we must; normally, you can see where to put spaces.)

All numbers are decimal unless we indicate otherwise; e.g., 35_8.

Finally, in examples we use

THIS TYPEFACE TO SHOW YOUR ENTRY)
THIS TYPEFACE FOR SYSTEM QUERIES AND RESPONSES.

) is the CLI prompt.

Contacting Data General

- If you have comments on this manual, please use the prepaid Remarks Form that appears after the Index. We want to know what you like and dislike about this manual.
- If you need additional manuals, please use the enclosed TIPS order form (USA only) or contact your Data General sales representative.
- If you experience software problems, please notify Data General Systems Engineering.

End of Preface

Contents

Chapter 4 - The SWAT Commands

Appendix A - Troubleshooting Tips (Or What to Do if You're Having Trouble)

Appendix B - SWAT Error Messages

Appendix C - More Examples

Illustrations

Tables

A single variable or constant can serve as an expression. To build a complex expression, however, you need to use one or more operators.

You can use a comma to separate one argument from another. Although it is normally optional, a comma may become necessary to prevent multiple arguments from being interpreted as parts of a single expression, or to separate the expression from a command or keyword.

Operators

SWAT expressions can contain one or more operators, depending on the context of the expression.

Table 1-1 lists the operators that the SWAT debugger recognizes.

Table 1-1. The SWAT Operators

	Operator		Represents
Arithmetic Operators	+	(prefix)	unary plus
	−	(prefix)	unary minus
	+	(infix)	addition
	−	(infix)	subtraction
	*	(infix)	multiplication
	/	(infix)	division
	=	(infix)	assignment
	:=	(infix)	assignment
Logical Operators	&	(infix)	logical AND
	!	(infix)	logical OR (inclusive)
Environment Operators	^	(prefix)	superior procedure block
	:	(prefix)	root environment
	:	(infix)	block identifier separator
Structure Operator	.	(infix)	level separator
Pointer Operator	->	(infix)	based reference
Indirect Operator	^	(suffix)	indirect reference

Prefix operators precede an operand and refer to that operand only. For example: +45, −2.7, and ^EXCHANGE. Except for the caret (^), you can separate the operator and operand with one or more spaces.

Infix operators define a relationship between two operands. The operator appears between the operands. For example: 36+420, GROSS − TAXES, (85.0 * FACTOR / 250.0), LOGA&LOGB, 40R8->BP.

A suffix operator appears at the end of the operand to which it refers. The operator must follow its operand with no intervening spaces. For example: DIRECT^, 500R8^, (PTR+5)^.

Data Type Conversions

In general, the SWAT debugger accepts mixed mode expressions to the same extent as the programming language you are using. Therefore, follow the conventions of your programming language when building expressions in a SWAT debugging session.

When two operands of different data types are related through an operator, the operand of the lower ranking data type is converted to the data type of the other operand. The result is therefore that of the higher ranking data type. The precedence for implicit conversions is as follows:

Highest rank:	CHARACTER
	BIT
	CHARACTER CONSTANT
	FLOAT
	DECIMAL
	DISPLAY
Lowest rank:	BINARY

In certain cases, data conversion is not possible, so an error results. Table 1-2 describes the results of various data type conversions. The data types listed at the top of the table correspond to the right operand; those at the side correspond to the left operand. These conversions apply when using the arithmetic operators +, −, *, and /.

Table 1-2. Data Type Conversions

	BINARY	DISPLAY	DECIMAL	FLOAT	CHARACTER CONSTANT	BIT	CHARACTER
BINARY	—	DISPLAY	DECIMAL	FLOAT	error	error	error
DISPLAY	DISPLAY	DISPLAY[1]	DECIMAL	FLOAT	error	error	error
DECIMAL	DECIMAL	DECIMAL	DECIMAL[1]	FLOAT	error	error	error
FLOAT	FLOAT	FLOAT	FLOAT	—	error	error	error
CHARACTER CONSTANT	error	error	error	error	—	error	error
BIT	error	error	error	error	CHARACTER CONSTANT	—	error
CHARACTER	error	error	error	error	—	error	—

1. If both operands are either DECIMAL or DISPLAY, data type conversion occurs to set the precision and scale of the result.

End of Chapter

Chapter 2
How to Run the SWAT Debugger

Before using the SWAT debugger, you need to complete a few preliminary steps. This chapter outlines everything you must do.

Required Software

The SWAT debugger operates with only certain software revisions of the operating system, the language compiler, and the Link utility. Check the SWAT Release Notice, or ask your system manager if you are not sure if the SWAT debugger will work with your software. The release notice lists the earliest revisions of related software that support the SWAT debugger. If your system operates with these or later revisions, *and* if the software revisions are compatible with each other, you can run the SWAT software.

Installing the SWAT Debugger

The system manager loads the SWAT files from the release tape into your system. Included among these files is PARSWAT.SR and SWATERMES.SR, which contain SWAT error message information. Before you can run the SWAT debugger successfully, the system manager must use the EMASM macro to build the SWAT error message file, SWATERMES.OB. Finally, the system manager updates the system error message file, ERMES, by joining the SWAT error message object file with all others. See the SWAT Release Notice for additional information.

SWAT User Privileges

As a SWAT user, you need certain privileges before you can successfully execute the SWAT debugger. Your user profile must allow you to

- create a process without blocking
- create at least three son processes

If your user profile does not grant the necessary privileges, the error message

CALLER NOT PRIVILEGED FOR THIS ACTION

or

TOO MANY SUBORDINATE PROCESSES

appears when you try to run the SWAT debugger. Check with your system operator to ensure that you have the privileges outlined above. If you do not, the system operator can change your profile.

You may need to be able to create more than three son processes. If, for example, you intend to use the SWAT CLI command, you must be able to create at least four son processes. If any son process in turn creates other processes, your profile must allow you to create that many more.

Preparing a Program For Debugging

At this point you should have both access to the necessary SWAT files and the required user privileges. Now you follow three basic steps to begin a debugging session:

1. Compile each program module, using the /DEBUG switch with the modules you want to be able to debug.
2. Link the program modules and library files using the /DEBUG switch.
3. Call the SWAT debugger.

Before using an updated version of the SWAT debugger with programs that you debugged with an earlier revision, be sure to relink them (using the /DEBUG switch) so that the program file incorporates the latest SWAT software.

Compiling Your Program

The SWAT debugger requires special information not normally built into a program's object file. To provide this information, you append the /DEBUG switch to your compiler command. Use this general format:

XEQ { CC / COBOL / F77 / PASCAL / PL1 / other } /DEBUG//.../ program//*switch* .../

This command produces a program object file called program.OB.

You can use the SWAT debugger to examine a program module only if you used the /DEBUG switch when compiling the module. So, for programs with more than one module, use the /DEBUG switch with each one that you intend to debug. The program must contain at least one module compiled in this way.

Linking Your Program

The Link utility uses your compiled module(s) and library files to produce a program file. If you intend to debug the program file using the SWAT debugger, append the /DEBUG switch to the Link command. The Link utility then builds a debuggable program file and two additional files that the SWAT software needs.

You must include the SWAT interface file, SWATI.OB (or SWATI16.OB for the 16-bit SWAT debugger under AOS/VS), in your Link command line. Be sure that you have access to this file before calling the Link utility.

The general format for the Link command line is:

XEQ LINK/DEBUG//.../ program//.../ routines SWATI lang_libraries

IMPORTANT: The SWAT interface file (SWATI or SWATI16) must appear *before* all language library files in the Link command line, otherwise your program will execute without allowing you to debug it.

You can also use the Link macro provided by your programming language. For additional information, refer to the appropriate programmer's manual. Some examples of Link macros are:

CCL.CLI	for AOS/VS C programs
CLINK.CLI	for AOS/VS COBOL programs

F77LINK.CLI	for FORTRAN 77 programs (AOS and AOS/VS)
PASLINK.CLI	for AOS/VS PASCAL programs
PL1LINK.CLI	for PL/I programs (AOS and AOS/VS)

If you use these link macros, do not specify SWATI or SWATI16 as an argument; the macro includes this file for you when you use the /DEBUG switch.

When you include the /DEBUG switch in your Link command, the Link utility produces a program file called program.PR, which you can debug using the SWAT software. Link also creates two special files: the debugger lines file (program.DL) and the debugger symbols file (program.DS).

The debugger lines file includes data taken from the debugger lines blocks of your program's object file. It determines the runtime and source file locations of all program lines that reside in the modules you selected to debug. When you refer to a line or block during a debugging session, the SWAT debugger uses this file to resolve the reference.

The debugger symbols file includes data taken from the debugger symbols blocks of your program's object file. The debugger symbols block supplies standard relocation information for this data through relocation dictionary entries. The symbols file describes all names and blocks defined within the modules you selected to debug. When you refer to a name or block during the debugging session, the SWAT debugger uses this file to resolve the reference.

Even though you build a program file using the /DEBUG switch, you can execute the program without using the SWAT software. The debugger information generated by the /DEBUG switch does not interfere with normal program execution.

Calling the SWAT Debugger

You are now ready to execute the SWAT software and debug your program. The format of the SWAT invocation command line is as follows:

XEQ SWAT*[/global_switch ...]* program*[/switch ...] [argument[/switch ...]]*

where:

program	is the name of the program file you want to debug.
argument	is any argument required by your program.

You can append any appropriate switches to the program and its argument(s).

Table 2-1 lists the global switches that you can append to the SWAT invocation command. Some of these switches may require additional privileges, which you must grant to the SWAT debugger. By default, a son process has the same privileges as its father. Refer to the CLI User's Manual for additional information.

The Audit File

To keep a record of a debugging session dialog, you can set up an audit file. When the audit file is open, the SWAT debugger appends all debugging commands and responses to the file. During the debugging session, you can close the audit file, then later open the same or another audit file.

There are two ways to set up an audit file. You can use the /AUDIT switch when you invoke the SWAT debugger, or you can use the AUDIT command within a debugging session. In each case, the SWAT debugger uses the default audit file unless you provide the name of another file. The name of the default audit file is your program's name with the .AU extension.

Table 2-1. The SWAT Global Switches

Switch	Use
/AUDIT/=*pathname*/	Assigns an audit file and turns auditing on. The default audit file is program.AU. If the file does not exist, the SWAT debugger creates it; otherwise, the debugger appends the audit dialog to the file.
/BREAK	Creates a break file on abnormal termination of your program. (AOS/VS only — see note below.)
/CALLS=n	Sets the maximum number of concurrent system calls for the program.
/CONSOLE=consolename	Assigns a terminal for the program you are debugging. (If you omit this switch, the program uses the same terminal as the SWAT debugger.)
/CPU=seconds	Sets the maximum CPU time for the program in seconds.
/DACL	Does not pass the default ACL to the program.
/DATA=pathname	Sets the program's DATAFILE, which identifies the generic @DATA file.
/DEBUG	Starts the program in the AOS or AOS/VS user debugger, granting write access to the program. (This switch is incompatible with the /NOCONSOLE switch.)
/INPUT=pathname	Sets the program's generic @INPUT file.
/LIST=pathname	Sets the program's LISTFILE, which identifies the generic @LIST file.
/MEMORY=pages	Sets the maximum memory size for the program in pages.
/NAME=name	Assigns a simple process name to the program.
/NOCONSOLE	Prevents assignment of generic files for the program. (This switch is incompatible with the /DEBUG switch. To use the SWAT CLI command or console interrupts, you must use this switch.)
/NOCOPY	Prevents automatic copying of the program file on start-up. The SWAT debugger opens the original program file.
/OUTPUT=pathname	Sets the program's generic @OUTPUT file.
/PREEMPTIBLE	Makes the program pre-emptible.
/PRIORITY=n	Sets the priority for the program.
/RESIDENT	Makes the program resident.
/SONS=n	Sets the maximum number of sons that the program can create. The default number is one fewer than the number of sons remaining to the process that called the SWAT debugger. (This default is not the same as that of the CLI PROCESS command.)
/USERNAME=name	Assigns a username to the program.
/WSMAX=pages	Sets the maximum number of pages allowed in main memory at one time. (AOS/VS only — see note below.)
/WSMIN=pages	Sets the minimum number of pages allowed in main memory at one time. (AOS/VS only — see note below.)

The /BREAK, /WSMAX, and /WSMIN switches apply only to the 16- or 32-bit SWAT debugger running under AOS/VS. The AOS SWAT debugger ignores these switches.

Chapter 3
A Sample Debugging Session

In this chapter we take you through a sample SWAT debugging session. You will see how each SWAT command works, what options are available to you, and how to use the commands to isolate errors in your source program. If you haven't already read Chapters 1 and 2, please do. The background they provide will help you with the material in this chapter.

This sample session uses a PL/I program running on an AOS system. Appendix C presents a compilation listing and audit file of the same program running under the 32-bit AOS/VS SWAT debugger. Appendix C also illustrates the use of the SWAT debugger with programs written in FORTRAN 77, COBOL, PASCAL, and C.

The Program

Before we begin the debugging session, let's look at the program we'll be working with. The program, called EXCHANGE, is written in PL/I. It was designed for the accounting department of an American-based company having branch offices in eight other countries. The program calculates monetary exchange values based on daily exchange rates for the currencies the company deals with.

The program has two separate modules: the main program (EXCHANGE) and an external subroutine called CONVERSION. The main program module also contains an embedded subroutine labeled CHECK.

What It Does

As we explain how the program works, refer to the program listings (Figures 3-1 and 3-2) and the program flow chart (Figure 3-3). When it begins to run, the EXCHANGE program reads the date and exchange rate information stored in a disk file called RATE_FILE. The file might look like this:

```
811124,
37.52,2.2465,2.4543,1194,1.7725,1.1865,3.4190,225.90,
.02663,.4458,.4081,.000835,.5566,.8434,.2924,.004471,
```

The first data item is the date of the exchange rates in the format YYMMDD, which corresponds to PL/I's built-in DATE function. (The other entries are the 16 exchange rates.) Later we describe the RATES table and explain what each element represents. If the date in this file does not match the system date (indicating that today's rates have not been entered), the program displays the message

RATES NOT CURRENT

and terminates. If the rate file does not exist, or if it lacks the required information, the program terminates after displaying the message

RATES NOT AVAILABLE

If the dates match, the program enters a loop so that it will continually be available to perform exchange calculations. First it displays a menu of the currencies handled by the company.

Select currency code for US$ exchange:

1 Belgian francs
2 W. German marks
3 Dutch guilders
4 Italian lire
5 Swiss francs
6 Canadian dollars
7 Saudi riyals
8 Japanese yen

(Type 0 to end the program)
Enter the currency code:

The operator types the code number identifying the currency involved in the exchange. The valid currency code numbers are 1 through 8; code 0 signals the program to terminate. The subroutine CHECK verifies that the operator has given a valid code. The program stops if it encounters an invalid code.

After determining the type of currency, the program asks the operator

Do you want to convert US$ into Foreign currency?
Enter 'Y' or 'N':

If the operator responds with anything other than "*Y*" or "*y*", the program assumes the exchange will be from the foreign currency into American dollars. Based on the operator's response, the program sets an index to locate the appropriate exchange rate in the table RATES.

The program reads the RATES table directly from the disk file. It is organized as follows:

RATE	Represents	For exchanging
(1)	Belgian francs per $1 US	US$ to BF
(2)	West German marks per $1 US	US$ to DM
(3)	Dutch guilders per $1 US	US$ to gld
(4)	Italian lire per $1 US	US$ to Lit
(5)	Swiss francs per $1 US	US$ to SF
(6)	Canadian dollars per $1 US	US$ to Can$
(7)	Saudi Arabian riyals per $1 US	US$ to SRI
(8)	Japanese yen per $1 US	US$ to Y
(9)	US dollars per 1 Belgian franc	BF to US$
(10)	US dollars per 1 West German mark	DM to US$
(11)	US dollars per 1 Dutch guilder	gld to US$
(12)	US dollars per 1 Italian lira	Lit to US$
(13)	US dollars per 1 Swiss franc	SF to US$
(14)	US dollars per 1 Canadian dollar	Can$ to US$
(15)	US dollars per 1 Saudi Arabian riyal	SRI to US
(16)	US dollars per 1 Japanese yen	Y to US$

(The program requires a specific exchange rate for converting American dollars to foreign currency and for converting foreign currency to American dollars. These exchange rates are not necessarily reciprocals.)

Chapter 4
The SWAT Commands

This chapter describes all of the commands you can use in a SWAT debugging session. We have arranged them alphabetically to help you locate each command quickly.

Remember that while in a debugging session you can obtain information about SWAT commands by using the HELP command. The debugger displays a brief description about the selected command, its format and functions. Table 4-1 lists all the SWAT commands and their functions.

Table 4-1. SWAT Commands

Command	Use
AUDIT	Reports or sets the audit status.
BREAKPOINT	Displays or sets a breakpoint.
BYE	Terminates the SWAT debugger.
CLEAR	Clears a breakpoint.
CLI	Performs a CLI function.
CONTINUE	Initiates or resumes program execution.
DEBUG	Calls the AOS or AOS/VS user debugger.
DESCRIBE	Displays information about a program symbol.
DIRECTORY	Displays or sets the current directory.
ENVIRONMENT	Displays or sets the working environment.
EXECUTE	Executes a file of SWAT commands.
HELP	Displays information about SWAT topics.
LIST	Displays one or more lines from the program source file.
MESSAGE	Displays an error message for an error code.
PREFIX	Displays or sets the SWAT command prompt.
PROMPT	Displays or sets the null command response.
SEARCHLIST	Displays or sets the search list.
SET	Assigns a value to a variable.
TYPE	Displays the value of an expression.
WALKBACK	Displays the current location and calling locations.

Using Upper- and Lowercase

You can enter a SWAT command or keyword in any combination of uppercase or lowercase characters. The entries CLEAR, clear, and Clear are all equivalent.

When referring to a program symbol such as a variable, an array, a label, or a structure, however, you must follow the case-usage rules of your programming language. If the language is insensitive to character case (such as FORTRAN 77), you can use any combination of upper- and lowercase characters. But for case-sensitive languages such as PL/I, you must enter the symbol *exactly* as it appears in the source program. In this case, the SWAT debugger distinguishes between NAME, Name, and name.

The SWAT Prompt

When the debugger is ready to accept a command, it displays the prompt

```
>
```

followed by a space. You can change the prompt character by calling the PREFIX command. This command takes a string argument in which you specify the new prompt character(s).

Enter your command line, terminating it with a NEW LINE. If you make a syntax or format error in a command line, an error message appears. See Appendix B for a complete list of SWAT error messages. Following each command description is a list of errors related to the command.

Delimiting Commands and Arguments

To enter more than one command on a line, separate the commands with a semicolon. For example:

```
> ENVIRONMENT @BREAK; LIST 1 10 ↵
```

You can continue a command line over more than one typed line on the terminal. If you simply continue typing, your command line wraps around to the next line. You can also use an ampersand (&) to end an incomplete line, then continue it on the next line. In either case, the debugger acknowledges that you are continuing the command by preceding the the prompt on the next line with an ampersand. For example:

```
> AUDIT ON;BREAKPOINT 23,34,76,81,112;CONTINUE;& ↵
&>ENVIRONMENT;CLI "TIME";EXECUTE "DBUG" ↵
```

The complete command line cannot exceed 511 characters.

You can use a comma in place of a tab or space to separate arguments.

```
> CLEAR 53,76,104 ↵
```

is equivalent to

```
> CLEAR 53 76 104 ↵
```

At times, you may need to use a comma to avoid ambiguity or to separate complex arguments. Usually a comma must precede expressions that do not begin with either a digit or a name. The comma prevents the debugger from interpreting the argument(s) as an incomplete expression. For example:

```
TYPE , (A + B) * C
```

or

```
LIST , *5
```

When providing a series of expression arguments, you must use a comma to distinguish one from another.

```
TYPE A, -B * C, (D + E) * F
```

The Null Command

If you press just the NEW LINE key following a SWAT prompt, the debugger responds according to the current null command response setting. By default, the SWAT debugger performs a LIST @BREAK command when you enter a null command. To change the null command response setting, use the PROMPT command. You can specify one or more debugger operations to be performed in response to a null command.

Abbreviating SWAT Commands

You can abbreviate a SWAT command to its minimum uniqueness. For example, you can abbreviate the DESCRIBE command to DES, which distinguishes it from the DEBUG and DIRECTORY commands. If you enter only D or DE, however, the SWAT debugger cannot tell which of these commands you want to use.

You must spell out the CLI and SET commands completely to distinguish them from all other SWAT commands.

Locators

A locator identifies a program statement. The locator can be the statement's line number or label. If you specify a numeric label as a locator, precede it with an asterisk (*) to distinguish it from a line number. For example:

PROG1	label locator
250	line number locator
*250	numeric label locator

If the locator points to a line number outside the current program module, or to a label outside the working environment, you must fully qualify the locator by preceding it with an environment specifier. For example

:EXCHANGE:LOOP	label locator
:CONVERSION:24	line number locator
:.MAIN:*5	numeric label locator

Specifying Environments

Environment specifiers identify a specific procedure block. You use them with the ENVIRONMENT command to set the working environment, and with program statement locators to fully qualify them.

The general format for an environment specifier is

[:external-procedure] [[:] internal-procedure[:internal-procedure]...]

An external procedure is a block that is not contained within any other block. In other words, it is located directly below the root environment. For this reason, an external procedure identifier always begins with a colon. Do not begin an environment specifier with a colon unless the first element is an external procedure.

An internal procedure is an embedded block. Use a colon to separate an internal procedure identifier from an external procedure identifier.

An environment specifier can contain more than one internal procedure specifier, depending on the depth at which the target environment is embedded. Separate each internal procedure identifier from the previous one with a colon. Do not precede the first internal procedure identifier with a colon unless it follows another procedure identifier.

An external or internal procedure identifier is usually the block's label. The AOS SWAT debugger and SWAT16 identify an unnamed block by using the line number where the block begins.

The caret (^) symbol represents the next higher environment. You can use one or more carets alone, or as a prefix to one or more internal procedure identifiers.

There are two SWAT keywords that you can use to specify an environment. The @BREAK keyword identifies the block immediately containing the statement where execution is currently trapped. The @MAIN keyword represents the main program module.

The identifying statement of an embedded procedure block technically resides within the enclosing environment. To examine the program state when a block becomes active, set a breakpoint at an executable statement within the block. You cannot refer to symbols within a block unless execution is trapped within that block's environment.

Placing Comments in the Audit File

If you set up an audit file through the /AUDIT switch or the AUDIT command, you can keep a copy of your debugging session dialog. The debugger appends the dialog to the current audit file. The audit file is a useful tool for examining what occurred during a debugging session. To make this record more understandable, you can insert comments into the audit file as you debug the program.

To make a comment, simply type a percent sign (%) in the first space after the SWAT prompt, then enter your comment. The debugger ignores everything to the right of the percent sign. You can describe what you are doing, and document your debugging activity. Just remember to begin the comment with the percent sign, or the SWAT software will try to interpret the comment as a command.

NOT A BREAKPOINT locator

There is no breakpoint at the specified location.

UNDEFINED SYMBOL symbol

The specified locator does not exist within the working environment. Check your spelling or the environment.

Examples

```
> CLEAR, *5 ↓
Cleared at :.MAIN:10

> ENVIRONMENT :FULL; CLE 20 ↓
:FULL
Cleared at :FULL:20

> CLEAR :EXCHANGE:62 ↓
Cleared at :EXCHANGE:62

> cle 45 ↓

NOT A BREAKPOINT 45

> clear 27 ↓

NOT A BREAKPOINT 27

> Clear @all ↓
Cleared at :EXCHANGE:34
Cleared at :EXCHANGE:50
Cleared at :EXCHANGE:CHECK:87
Cleared at :CONVERSION:13
Cleared at :CONVERSION:16
Cleared at :EXCHANGE:42:43
Cleared at :EXCHANGE:71 count=4
```

CLI

Performs a CLI function.

Format

CLI *["string"]*

where:

string is a character string of one or more CLI commands.

Description

Use this command to perform a CLI function within the SWAT session.

If you type CLI with no argument, the SWAT software creates a CLI process, and passes control to it. You remain in the CLI until it terminates.

To perform one or more CLI functions in a debugging session, follow the SWAT CLI command with a command string enclosed in quotation marks or single apostrophes. Use the ASCII notation <042> or <047> respectively, for an embedded quotation mark or apostrophe. If the string contains more than one command, use a semicolon to separate one command from another.) The debugger passes the string to the CLI process for execution.

This command has no unique abbreviation; you must use its full name.

IMPORTANT: You can use this command only if you grant the SWAT debugger exclusive use of the terminal through the /CONSOLE or /NOCONSOLE switch.

If your user profile does not grant the privilege to create unlimited sons, you may also need to use the /SONS switch to ensure that SWAT can create a minimum of two son processes.

Errors

COMMAND ENDS ILLEGALLY

You did not end the string argument with a quotation mark or apostrophe.

DEVICE ALREADY IN USE

(System error): The SWAT debugger does not have exclusive use of this terminal. Use the /CONSOLE or /NOCONSOLE switch.

INVALID EXTRA ARGUMENTS ON COMMAND

You can provide only one string argument. The string can contain more than one CLI command, provided that you separate each command with a semicolon.

NONUNIQUE COMMAND command

There is no acceptable abbreviation for this command; enter CLI.

TOO MANY SUBORDINATE PROCESSES

(System error): The SWAT debugger cannot create the CLI process because it has already created the maximum number of sons allowed it.

DIRECTORY

Displays or sets the current directory.

Format

DIRECTORY *["pathname"]*

where:

pathname identifies the directory to become the working directory.

Description

Use this command to set or display the working directory for the SWAT debugging session. Initially, the working directory is the one from which you called the SWAT debugger. (This directory applies only to the current debugging session. When you terminate the SWAT process, you return to the directory in which you invoked the SWAT debugger.)

To display the current working directory, enter the command without an argument.

To set the working directory, follow the DIRECTORY command with a pathname. Enclose the pathname in quotation marks or apostrophes.

A confirmation message appears after you set the working directory.

This command does not affect your program's directory.

Errors

COMMAND ENDS ILLEGALLY

You probably omitted the final quotation mark or apostrophe to set off the string argument.

INVALID EXTRA ARGUMENTS ON COMMAND

You can specify only one directory as the working directory. Use the SEARCHLIST command if you need access to other directories.

NONUNIQUE COMMAND command

Your command abbreviation is not unique. The shortest acceptable abbreviation for the DIRECTORY command is DI.

UNEXPECTED OPERATOR operator

You probably used a pathname prefix without enclosing the pathname in quotation marks or apostrophes.

Examples

```
> DIRECTORY ⤶
:UDD:TOM

> dir "TEST_PROGRAMS" ⤶
:UDD:TOM:TEST_PROGRAMS

> DIR "^" ⤶
:UDD:TOM
```

ENVIRONMENT

Displays or sets the working environment.

Format

ENVIRONMENT [*@ALL* | *@BREAK* | *@MAIN* | *specifier*]

where:

@ALL	represents all the external environments that the SWAT debugger can examine.
@BREAK	represents the environment of the current breakpoint trap.
@MAIN	represents the main program module.
specifier	identifies a procedure block to become the working environment.

Description

If you type only ENVIRONMENT, the debugger displays the current working environment; i.e., the area of the program that you can immediately refer to.

If you use the @ALL keyword, the debugger displays the name of each program module that it can debug. This listing is helpful when only certain modules were compiled and linked with the /DEBUG switch.

You can set the working environment by supplying an argument to the command. If you use the keyword @BREAK, the SWAT debugger sets the working environment to that of the current breakpoint trap. The @BREAK keyword provides a convenient return to the breakpoint environment.

Use the @MAIN keyword to set the working environment to that of the main procedure block.

You can also use an environment specifier to identify an environment to which you want to relocate. Specify the procedure name of the block (or, for the AOS SWAT debugger and SWAT16, you can use the line number of an unnamed block). Refer to the section called "Specifying Environments" earlier in this chapter.

Whenever you use this command, the debugger displays the working environment. The working environment remains unchanged until either you explicitly reset it, or program execution traps in another environment.

Errors

ENVIRONMENT NOT FOUND specifier

The specifier does not identify an existing environment. To refer to a superior environment or an external environment, be sure to begin the specifier at the root environment, then work your way down. If the environment is embedded within the working environment, do not begin the specifier with a colon.

NONUNIQUE COMMAND command

Your command abbreviation is not unique. The shortest acceptable abbreviation for ENVIRONMENT is EN.

NOT AT A BREAKPOINT

You used the @BREAK keyword, which refers to the environment of the current breakpoint trap. Program execution, however, has not yet begun; you must first issue a CONTINUE command.

Examples

```
SWAT REVISION 02.00 ON 10/04/81 AT 14:45:16
PROGRAM -- :UDD:TOM:EXCHANGE
> ENVIRONMENT ↓
:EXCHANGE

> ENV CHECK ↓
:EXCHANGE:CHECK

> ENV :CONVERSION ↓
:CONVERSION

> env :EXCHANGE:27 ↓
:EXCHANGE:27

> ENV @MAIN ↓
:EXCHANGE

> ENVIRONMENT ↓
:.MAIN

> ENV :PAYMENT ↓
:PAYMENT

> ENV @MAIN ↓
:.MAIN
```

EXECUTE

Executes a series of SWAT commands stored in a file.

Format

EXECUTE "filename"

where:

filename identifies a file that consists of one or more debugger commands.

Description

Use this command to execute SWAT commands stored in a file. With the EXECUTE command, you can employ user-built debugger utilities.

Name the command file by providing its filename enclosed in quotation marks. You cannot use a filename template.

The SWAT debugger executes the commands in the order they appear in the file. The commands must carry all necessary arguments; you cannot pass arguments at runtime.

Errors

COMMAND ENDS ILLEGALLY

The string argument does not end with a quotation mark or apostrophe.

COMMAND REQUIRES ARGUMENT(S)

You must provide an argument that is a filename or an expression that resolves to a filename.

FILE DOES NOT EXIST

(System error): The specified command file does not exist, or does not reside in a directory on the current search list.

FILENAME TOO LONG

(System error): The filename you provided (either explicitly or implicitly) contains an illegal number of characters.

NONUNIQUE COMMAND command

Your command abbreviation is not unique. The shortest acceptable abbreviation for EXECUTE is EX.

Examples

```
> E "DBUG" ⌇
```

NONUNIQUE COMMAND E

```
> EXECUTE "DBUG" ⌇
>> command_1
response_1
>> command_2
response_2


>
```

HELP

Displays information about SWAT topics.

Format

HELP *[topic] ...*

where:

topic is an entry from the help menu.

Description

To get quick, on-line information when you're in a SWAT debugging session, simply type HELP. The SWAT debugger displays a menu of topics you can obtain specific information for.

For information about a particular topic, follow the HELP command with the name of the topic. You can abbreviate any help topic. If the abbreviation is not unique, the debugger displays the topics that match it. You can then select from these (or any other) topics.

You can append more than one topic to the HELP command.

Figure 4-1 illustrates a sample help menu. Later revisions of the SWAT debugger may list additional topics. Simply type HELP to display the current menu.

```
topics are:

      @ALL          @BIT          @BREAK        @CHARACTER
      @FLOAT        @INTEGER      @MAIN         @POINTER
      ACTION        AT            AUDIT         BREAKPOINT
      BYE           CLEAR         CLI           COMMANDS
      COMMENTS      CONTINUE      COUNT         DEBUG
      DESCRIBE      DIRECTORY     ENVIRONMENT   EXECUTE
      EXPRESSIONS   HELP          KEYWORDS      LIST
      LOCATORS      MESSAGE       OFF           ON
      OPERATORS     PREFIX        PROMPT        SEARCHLIST
      SET           SPECIFIERS    TYPE          UNIQUENESS
      WALKBACK
```

Figure 4-1. The Help Menu

Errors

UNKNOWN TOPIC topic

The specified topic is not listed on the help menu.

HELP (continued)

Examples

> HELP ↵

topics are:

@ALL	*@BIT*	*@BREAK*	*@CHARACTER*
@FLOAT	*@INTEGER*	*@MAIN*	*@POINTER*
ACTION	*AT*	*AUDIT*	*BREAKPOINT*
BYE	*CLEAR*	*CLI*	*COMMANDS*
COMMENTS	*CONTINUE*	*COUNT*	*DEBUG*
DESCRIBE	*DIRECTORY*	*ENVIRON-*	*EXECUTE*
EXPRESSIONS	*HELP*	*MENT*	*LIST*
LOCATORS	*MESSAGE*	*KEYWORDS*	*ON*
OPERATORS	*PREFIX*	*OFF*	*SEARCHLIST*
SET	*SPECIFIERS*	*PROMPT*	*UNIQUENESS*
WALKBACK		*TYPE*	

> help @all ↵

Keyword: *CLEAR @ALL*
ENVIRONMENT @ALL
LIST @ALL

CLEAR @ALL will clear all breakpoints. ENVIRONMENT @ALL will report all debugger external procedures. LIST @ALL will list all the source lines in the current environment.

> H DEB ↵

Command: DEBUG

Enter the user debugger. To return to SWAT, either type P newline (AOS), or escape R (AOS/VS).
Note: This command is usable only if the user program has a console, i.e. the /NOCONSOLE switch was not set on SWAT. Also, the user debugger will have write access only if the /DEBUG switch was set on SWAT.

> Help program ↵

UNKNOWN TOPIC program

Using the AOS/VS SWAT Debugger

The differences between the AOS/VS and AOS SWAT debuggers are few. The enhancements offered by the AOS/VS system mean that the AOS/VS SWAT debugger includes some additional features.

There are two versions of the SWAT software that you can run on an AOS/VS system: the 16-bit SWAT debugger (SWAT16) and the 32-bit SWAT debugger. You can use SWAT16 to debug 16-bit program running under AOS/VS. The differences between the AOS SWAT debugger and SWAT16 are noted in the manual.

The 32-bit AOS/VS SWAT Debugger

- supports decimal arithmetic.
- supports integer/floating-point mixed mode arithmetic.
- fully supports multitasking. Scheduling is disabled when the debugger encounters a breakpoint. When execution continues, scheduling returns to the state it was in just prior to the breakpoint trap. There is no additional overhead for single-task programs.
- includes the display format keyword @Rn where n can range from 2 through 16. This keyword specifies a radix for the display of numeric values.
- can be initiated by a process other than the Command Line Interpreter (CLI). This means that you can pass a non-CLI format initial IPC message to the SWAT debugger.
- supports the process switches /BREAK, /WSMAX, and /WSMIN. You can use these switches when invoking the SWAT debugger. (The AOS/VS 16-bit SWAT debugger also supports these switches.)

Other Considerations for the AOS/VS SWAT Debugger

- When using the AOS/VS user debugger, be aware that you cannot set a SWAT breakpoint where an assembly language breakpoint is already set. If an assembly language breakpoint and a SWAT breakpoint exist at the same instruction, you should clear the breakpoints in the opposite order that you set them.
- The debugger distinguishes between upper- and lowercase only when the programming language is case-sensitive.

Specific Language Considerations

If you are running PL/I:

In your Link command line, append the /START switch to the name of the main program module.

The keyword @MAIN refers to the highest level procedure of the first program module (.OB file) listed in the Link command line.

To build a locator to refer to a specific clause within a statement, use this format:

line-no.clause-no

The first clause on a line is clause 0, the next is clause 1, and so on. Note, however, that even if the clauses of a single statement actually appear on adjacent lines of the source code, the debugger treats them as if they were on the same line. So, use the locator that identifies the line where the statement begins, then append the clause number.

If you are running FORTRAN 77:

The identifier .MAIN represents an unnamed main program.

You can use the SET command to assign a value to a LOGICAL variable. To assign the value .FALSE., set the variable equal to 0. To assign the value .TRUE., set the variable equal to a nonzero integer value. For example:

```
SET L1 = 0      (L1 is .false.)
SET L1 = 1      (L1 is .true.)
SET L1 = -4     (L1 is .true.)
```

If you are running COBOL:

You can debug 32-bit COBOL programs with the AOS/VS SWAT debugger.

The program name is the PROGRAM-ID specified in the IDENTIFICATION DIVISION.

If you are running PASCAL:

The debugger recognizes type and constant identifiers, enumerated data types, and subrange data types; you can refer to these with SWAT commands in the appropriate context. You can also use the elementary items of packed and unpacked arrays and structures in an expression. You can refer to a packed array [1..n] of CHAR as an aggregate.

The DESCRIBE and TYPE commands let you work with set variables.

The postfix arrow operator (^) specifies a pointer or file buffer variable.

The debugger recognizes the PASCAL assignment operator (:=), as well as the use of apostrophes to identify a character string.

If you have external subprograms within a module compilation unit, the debugger treats them as internal environments within the external environment of the module.

If you are running C:

The debugger recognizes user-defined data types, unsigned integers, structures and unions, and C arrays; you can refer to these with SWAT commands in the appropriate context. You must fully qualify all references to structure and union members.

You cannot refer to unsubscripted array names. The debugger recognizes the following subscript notation for multidimension arrays:

array_name [1] [2] [n] ...

The debugger recognizes the star operator (*) for C pointers.

Common User Errors

The following section may provide some assistance when you're having trouble after the SWAT debugger begins executing.

The program begins execution, but does not pass control to the debugger:

Is your program written in a language that the SWAT software supports? See the SWAT Release Notice for details.

Did you place the SWAT Interface file (SWATI or SWATI16) before all language library files in the Link command? If not, relink the program modules. Be sure that you use the latest revision of the SWATI.OB or SWATI16.OB file.

The SWAT debugger doesn't recognize a symbol or locator:

Is the symbol or locator known within the working environment? If not, did you qualify the reference by preceding it with an environment specifier?

If the programming language is case-sensitive, did you enter the symbol exactly as it appears in the source text?

If you're specifying a structure member, did you fully qualify it?

If you're specifying a numeric label locator, did you precede it with an asterisk to distinguish it from a line number?

The SWAT debugger won't set a breakpoint:

Are you located in the correct environment? You must fully qualify a locator if it is a label outside the working environment, or a label outside the current module.

If you're setting a breakpoint at a clause, does the locator identify the line where the statement begins? If you append a clause number to the locator, does it point to the right statement?

Does the locator point to a nonexecutable statement or a line number that does not exist?

The SWAT debugger won't list a statement:

Does the locator identify a line within the current module?

Did you receive a warning message that indicated that the source file was not current?

The SWAT debugger won't assign a value to a variable:

Is the result of the expression argument compatible in data type and size with the receiving variable?

Does the expression contain incompatible data types?

If the expression contains an automatic or parameter variable, is the variable known within the breakpoint environment?

If the programming language is case-sensitive, did you enter the symbol name exactly as it appears in the source text?

Did you fully qualify a structure member reference?

The program (and the SWAT session) terminated after a CONTINUE statement:

You did not place a breakpoint within the path of execution (or the breakpoint carries an unexpired proceed count).

The SWAT debugger reports that a file is not current:

The debugger lines or debugger symbols files may not be compatible with the program file.

If you are using the LIST command, the source file has been modified since you compiled it.

The SWAT debugger does not let you display a structure member with the TYPE command:

Make sure that you have fully qualified the structure member.

Use the DESCRIBE command to display a description of the entire structure. This may explain why the debugger does not let you examine the member.

Helpful Hints

Help Information from the CLI

When you are in the CLI environment, you can use the command HELP *SWAT to obtain general information about the SWAT debugger. (The file CLI.TPC.SWAT must reside in the :HELP directory.)

Calling the User Debugger

If you intend to work with the AOS or AOS/VS user debugger, we suggest that you enable a second terminal for your program. This allows you to observe the program's interaction with the terminal, uninterrupted by SWAT dialog.

For an added margin of security when using the user debugger, you can call it through the SWAT debugger and work on a copy of the program. This allows you to keep the original program file intact.

Using Screenedit Features

If you are working at a DASHER® display terminal and have screen editing capabilities, you can use a multiple command line to great advantage. For example, the SWAT command line

```
> CONTINUE; TYPE ACCOUNT.CHECKING.CURRENT_BALANCE )
```

when combined with strategic breakpoints in the program and the CTRL-A key combination, allows you to monitor the value of a specific variable through program execution. This particular example also demonstrates the amount of rekeying time you can save.

Defining Extra Variables

You may find it helpful to define more variables than the program actually needs. The extra variables, such as statement labels, can aid your debugging activity. For example, you may want to assign meaningful labels to key statements throughout a large program. Using the LIST command, you can locate a specific block of code quickly. We suggest that you remove the extra variables after debugging a program; the compiler produces more efficient code if you eliminate unnecessary elements.

Using Pointer Arithmetic

By defining based variables of various types you can examine virtually any piece of storage that is accessible to your program. You can use pointer arithmetic to display the value of an offset storage location. You can use the pointer reference operator (^) by appending it to an expression. Thus, PTR -> based_var is equivalent to PTR^.

Simulating CLI Pseudo-Macros

You can simulate the CLI pseudo-macros [!OCTAL] and [!DECIMAL] with the TYPE command. To display the octal equivalent of a decimal value, use the @POINTER keyword with the command. To display the decimal equivalent of an octal value, append the R8 suffix to the octal argument.

End of Appendix

Appendix C
More Examples

This appendix provides additional examples of PL/I, FORTRAN 77, COBOL, PASCAL, and C programs, and audit files that illustrate the use of SWAT commands with these programs.

The first example illustrates the same PL/I program used in the sample session in Chapter 3. In this case, however, the program is compiled and linked on an AOS/VS system, then run using the 32-bit AOS/VS SWAT debugger.

Figures C-1 and C-2 present the compilation listings of the EXCHANGE and CONVERSION modules, respectively. Figure C-3 presents the audit files of sample sessions similar to those described in Chapter 3.

```
Source file: EXCHANGE
Compiled on 24-Nov-81 at 14:40:38 by AOS/VS PL/I Rev 01.20.02.00
Options: PL1/DEBUG/L=EXLIST

    1  EXCHANGE:
    2          PROCEDURE;
    3
    4          /* This program calculates currency exchange values based on
    5  *       daily international exchange rates.  The operator selects
    6  *       the type of exchange then enters the incoming amount.  The
    7  *       program performs the calculation and displays the result. */
    8
    9          DECLARE (SCREEN, KEYBOARD, RATE_FILE) FILE;
   10          DECLARE CONVERSION ENTRY(FIXED BINARY,FIXED DECIMAL(10,6));
   11          DECLARE SELECTION FIXED BINARY;
   12          DECLARE 1 DAILY_RATES,
   13                  2 RATE_DATE CHARACTER(6),
   14                  2 RATES(16) FIXED DECIMAL(10,6);
   15          DECLARE RESPONSE CHARACTER(1);
   16          DECLARE INDX FIXED BINARY;
   17
   18          OPEN FILE(SCREEN) STREAM OUTPUT PRINT TITLE('@OUTPUT');
   19          OPEN FILE(KEYBOARD) STREAM INPUT TITLE('@INPUT');
   20          OPEN FILE(RATE_FILE) STREAM INPUT;
   21
   22  /* Read in today's exchange rates.  Compare date with system date. */
   23
   24          GET FILE(RATE_FILE) LIST(DAILY_RATES);
   25
   26          ON ENDFILE(RATE_FILE)
   27          BEGIN;
   28                  PUT FILE(SCREEN) SKIP LIST('NO RATES AVAILABLE.');
   29                  STOP;
   30          END; /*BEGIN Block for End of File condition */
   31
   32  /* Compare file's date with the system date. */
   33
```

Figure C-1. Compilation Listing of the Module EXCHANGE (AOS/VS) (continues)

```
34          IF RATE_DATE ^= DATE() THEN DO;
35                  PUT FILE(SCREEN) SKIP LIST("RATES NOT CURRENT");
36                  STOP;
37                  END; /*DO Block */
38
39  /* Set up ON ERROR condition for bad input */
40
41          ON ERROR
42                  BEGIN;
43                  PUT FILE(SCREEN) SKIP LIST("Invalid input.  "
44                                                  !!"Try again.");
45                  GO TO LOOP;
46                  END;
47
48  /* Display menu for exchanges */
49
50          LOOP:   DO WHILE("1"B);
51
52                  PUT FILE(SCREEN) SKIP(3) LIST("Select currency code for "!!
53                       "US$ exchange:");
54                  PUT FILE(SCREEN) SKIP(2) EDIT
55                       ("1","Belgian francs","2","W. German marks","3",
56                       "Dutch guilders","4","Italian lire","5","Swiss francs",
57                       "6","Canadian dollars","7","Saudi riyals","8",
58                       "Japanese yen","(Type 0 to end the program)",
59                       "Enter the currency code:  ")
60                       (8(X(5),A(1),X(3),A(16),SKIP),SKIP,A(27),SKIP,A(26));
61
62                  GET FILE(KEYBOARD) LIST(SELECTION);
63
64                  CALL CHECK(SELECTION);
65
66  /* Request type of exchange */
67
68          PUT FILE(SCREEN) SKIP LIST("Do you want to convert US$ into"
69                  !!" Foreign currency?");
70          PUT FILE(SCREEN) SKIP LIST("Enter 'Y' or 'N':  ");
71          GET FILE(KEYBOARD) LIST(RESPONSE);
72
73          IF RESPONSE = "Y" THEN INDX = SELECTION;
74          ELSE IF RESPONSE = "y" THEN INDX = SELECTION;
75                  ELSE INDX = SELECTION + 8;
76
77          CALL CONVERSION(INDX, RATES(INDX));
78
79  END;/* LOOP */
80
81  CHECK:
82          PROCEDURE(SEL);
83          DECLARE SEL FIXED BINARY;
84
85          IF SEL > 0 THEN
86                  IF SEL < 9 THEN RETURN;
87          PUT FILE(SCREEN) SKIP LIST("Exchange program ended.");
88          STOP;
89  END;/* CHECK SUBROUTINE */
90
91  END;/* EXCHANGE */
```

Figure C-1. Compilation Listing of the Module EXCHANGE (AOS/VS) (continued)

```
> % Use the default null command response to list the lines around
> % the current breakpoint trap.
>
59              02 FILLER       PIC X(50), VALUE SPACES.
60      01      SUMMARY-LINE4.
61              02 FILLER       PIC X(18), VALUE "MONTHLY PAYMENT = ".
62              02 SUMMARY-PAYMT, PIC $(4)9.99.
63              02 FILLER       PIC X(50), VALUE SPACES.
64
65      01      HEADLINE        PIC X(80),
66                  VALUE " NUM       INTEREST     PRIN. PAY  PRIN.
67      -                 " BAL      INTEREST PAID TO DATE".
68
69C     PROCEDURE DIVISION.
70      INIT.   OPEN OUTPUT OUTFILE.
71      OPERATOR.
72              DISPLAY "ENTER PRINCIPAL:  $" WITH NO ADVANCING.
73              ACCEPT PRINCIPAL.
74              DISPLAY "INTEREST RATE (%): " WITH NO ADVANCING.
75              ACCEPT PERCENT.
76              COMPUTE MONTHLY-INT-RATE = PERCENT / 100 / 12.
77              DISPLAY "YEARS TO PAY:      " WITH NO ADVANCING.
78              ACCEPT YEARS.
79              COMPUTE MONTHS = YEARS * 12.
> % Skip the next statement (omit opening the printer) and continue execution
> % at statement 72.
> CONTINUE AT 72

Breakpoint trap at :MORTPROG:83
> % The program asked for principal:  we entered $4000
> % The program asked for interest rate:  we entered 17%
> % The program asked for years to pay:  we entered 3
> % We then indicated that we wanted a summary, not a full schedule.
> %
> % List the line where execution trapped.
> LIST 83
83C             COMPUTE MONTHLY-PAYMT ROUNDED =
> % Display the values of program variables.
> TYPE CRT-INPUTS.PRINCIPAL
4000.
> TYPE CRT-INPUTS.PERCENT
17.
> TYPE CRT-INPUTS.YEARS
3.
> TYPE CRT-INPUTS.YEARS * 12.
 3.600000000000000E+01
> TYPE TEMPS.MONTHS
36.
> TYPE CRT-INPUTS.FUNCTION
0.
```

Figure C-9. Audit File of the MORTPROG SWAT Session (continued)

```
> % Change the principal amount to $3000.
> SET CRT-INPUTS.PRINCIPAL = 3000.
Old value: 4000.
New Value: 3000.0
> % Display computed monthly interest rate.
> TYPE TEMPS.MONTHLY-INT-RATE
 1.416666E-02
> % Change the interest rate and recompute.
> SET CRT-INPUTS.PERCENT = 15
Old value: 17.
New Value: 15.
> CONTINUE AT 76

Breakpoint trap at :MORTPROG:83
> TYPE TEMPS.MONTHLY-INT-RATE
 1.250000E-02
> % Continue execution to perform the monthly payment calculation.
> CONTINUE

Breakpoint trap at :MORTPROG:88
> TYPE TEMPS.MONTHLY-PAYMT
.08
> TYPE TEMPS.MONTHLY-INT-RATE
 1.250000E-02
> TYPE CRT-INPUTS.PRINCIPAL
3000.
> TYPE CRT-INPUTS.PERCENT
15.
> TYPE CRT-INPUTS.YEARS
4.
> TYPE CRT-INPUTS.PRINCIPAL * TEMPS.MONTHLY-INT-RATE
 3.750000E+01
> % Display the working directory.
> DIRECTORY
:UDD:TOM
5 % Display the current search list.
> SEARCHLIST
:UDD:TOM :UTIL :SWAT :LANG_RT :COBOL
> % End the debugging session.
> BYE

SWAT TERMINATED
```

Figure C-9. Audit File of the MORTPROG SWAT Session (concluded)

A Sample PASCAL Program and Audit File

This PASCAL program, called EXCHANGEP, is similar to the sample PL/I program used in Chapter 3 of this manual. It calculates currency exchange values based on daily exchange rates. The program consists of a single module, which interacts with the operator and a file called RATE_FILE.

The compilation and link commands are:

```
PASCAL/DEBUG/L=EXCHANGEP.LS EXCHANGEP

PASLINK/DEBUG EXCHANGEP
```

(The PASLINK macro includes the SWATI file automatically; you do not specify it as an argument.)

Figure C-10 presents the compilation listing of the EXCHANGEP program. A sample SWAT session audit file appears in Figure C-11.

```
Source file: EXCHANGEP
Compiled on 13-Sep-82 at 10:42:27 by AOS/VS PASCAL Rev 01.00.00.00
Options: PASCAL/DEBUG/L=EXCHANGEP.LS

    1   PROGRAM
    2           exchange (INPUT, OUTPUT, rate_file);
    3
    4   {       This program calculates currency exchange values based on
    5   *       daily international exchange rates.  The operator selects
    6   *       the type of exchange, then enters the incoming amount.  The
    7   *       program performs the calculation and displays the result.    }
    8
    9   {$C+    Compiler directive allows ASCII '<nn>' notation in the program }
   10
   11   CONST
   12           TAB = '<11>';
   13           CLEAR_SCREEN = '<14>';
   14
   15   TYPE
   16           country = (belgian,
   17                      w_german,
   18                      dutch,
   19                      italian,
   20                      swiss,
   21                      canadian,
   22                      saudi,
   23                      japanese,
   24                      no_choice);
   25
   26           monetary_symbols = PACKED ARRAY [1..4] OF CHAR;
   27           ratedata = RECORD
   28                       rate_date : PACKED ARRAY [1..6] OF CHAR;
   29                       rates : RECORD
   30                                us_foreign : ARRAY[country] OF REAL;
   31                                foreign_us : ARRAY[country] OF REAL;
   32                                  END; { rates }
   33                       END; { ratedata }
   34
   35           rfile = FILE OF ratedata;
   36
   37   VAR
   38
   39           selection : country;
   40           currency : ARRAY [country] OF monetary_symbols;
   41           us_to_foreign : BOOLEAN;
   42           rate_file : rfile;
   43           daily_rates : ratedata;
   44           i : INTEGER;
   45           response : CHAR;
   46
   47   {-------------------------------------------------------------------------}
```

Figure C-10. Compilation Listing of the EXCHANGEP Program (continues)

```
48  PROCEDURE
49          builder;
50
51  {       This procedure builds the currency rate file for the exchange
52  *       program.    }
53
54  VAR
55          selection : country;
56          rate_rec  : ratedata;
57
58  BEGIN
59          REWRITE (rate_file);
60
61          WRITELN (CLEAR_SCREEN);
62          WRITELN (TAB, 'Building new currency rate file');
63          WRITELN;
64          WRITE (TAB, 'Enter the date in yymmdd format:  ');
65          READLN (rate_rec.rate_date);
66
67          selection := belgian;
68          REPEAT
69                  WRITE (TAB, 'Enter rate for US$ to ',
70                                  currency[selection], ' :  ');
71                  READLN (rate_rec.rates.us_foreign[selection]);
72
73                  WRITE (TAB, 'Enter rate for ' , currency[selection],
74                                  ' to US$ :  ');
75                  READLN (rate_rec.rates.foreign_us[selection]);
76                  selection := SUCC(selection);
77
78          UNTIL (selection = no_choice);
79
80          WRITE (rate_file, rate_rec);
81
82          RESET (rate_file);
83          READ (rate_file, daily_rates);
84  END;
85  {-------------------------------------------------------------------------}
86  PROCEDURE
87          menu;
88
89  BEGIN
90          WITH daily_rates DO
91          BEGIN
92          WRITE ('Rates for ');
93          WRITE (rate_date[3], rate_date[4],'/');
94          WRITE (rate_date[5], rate_date[6],'/');
95          WRITE (rate_date[1], rate_date[2]);
```

Figure C-10. Compilation Listing of the EXCHANGEP Program (continued)

```
 96          WRITELN;
 97          WRITELN ('Identify the type of currency:');
 98          WRITELN;
 99          WRITELN (TAB, '1   Belgian francs');
100          WRITELN (TAB, '2   W. German marks');
101          WRITELN (TAB, '3   Dutch guilders');
102          WRITELN (TAB, '4   Italian lire');
103          WRITELN (TAB, '5   Swiss francs');
104          WRITELN (TAB, '6   Canadian dollars');
105          WRITELN (TAB, '7   Saudi riyals');
106          WRITELN (TAB, '8   Japanese yen');
107          WRITELN (TAB, '(Type 0 to end the program.)');
108          WRITELN;
109          WRITE ('Enter the currency code:  ');
110          END;
111
112  END;
113  {-------------------------------------------------------------------------}
114  PROCEDURE
115          get_selection ( VAR selection : country );
116
117  {       Gets the users currency code input and sets parameter selection
118  *       to the corresponding country.                                     }
119
120  VAR
121          currency_code : INTEGER;
122
123  BEGIN
124
125          READLN (currency_code);
126
127          IF (currency_code >= 0) AND (currency_code <= 8) THEN
128
129                  BEGIN
130
131                  CASE currency_code OF
132
133                          0 : selection := no_choice;
134                          1 : selection := belgian;
135                          2 : selection := w_german;
136                          3 : selection := dutch;
137                          4 : selection := italian;
138                          5 : selection := swiss;
139                          6 : selection := canadian;
140                          7 : selection := saudi;
141                          8 : selection := japanese;
142                  END
143                  END
144
145          ELSE    selection := no_choice;
146
147   END; { get_selection }
```

Figure C-10. Compilation Listing of the EXCHANGEP Program (continued)

```
148
149  {-------------------------------------------------------------------}
150
151  PROCEDURE
152          conversion ( selection : country );
153
154  VAR
155          inval, outval, rate : REAL;
156          response            : CHAR;
157
158  BEGIN
159          WRITELN;
160          WRITELN (TAB, 'Do you want to convert US$ into Foreign currency?');
161          WRITE (TAB, 'Enter "Y" or "N":  ');
162          READLN (response);
163
164          IF ((response = 'Y') OR (response = 'y')) THEN
165                  BEGIN
166                          us_to_foreign := TRUE;
167                          rate := daily_rates.rates.us_foreign[selection];
168                  END
169
170          ELSE
171                  BEGIN
172                          us_to_foreign := FALSE;
173                          rate := daily_rates.rates.foreign_us[selection];
174                  END;
175
176          WRITELN;
177
178          IF us_to_foreign THEN WRITE (TAB, 'Enter US$:  ')
179          ELSE WRITE (TAB, 'Enter ', currency[selection], ':  ');
180
181          READLN (inval);
182          outval := inval * rate;
183
184          WRITELN;
185          IF us_to_foreign THEN WRITELN ('$', inval:10:2,
186            ' US equivalent to:  ', outval:10:2, ' ', currency[selection])
187          ELSE WRITELN (inval:10:2, ' ', currency[selection],
188            ' equivalent to:  $', outval:10:2, ' US');
189
190  END;
191  {-------------------------------------------------------------------}
192
193
194  BEGIN { main procedure }
195
196          currency[belgian] :=  'BF';
197          currency[w_german] :=  'DM';
198          currency[dutch] :=  'gld';
199          currency[italian] :=  'Lit';
```

Figure C-10. Compilation Listing of the EXCHANGEP Program (continued)

```
200         currency[swiss] :=  'SF';
201         currency[canadian] :=  'Can$';
202         currency[saudi] :=  'SRI';
203         currency[japanese] :=  'Y';
204
205         WRITELN (CLEAR_SCREEN, TAB, TAB, 'Exchange Program');
206         WRITELN;
207
208         WRITELN (TAB, 'Do you want to create a new data file?');
209         WRITE (TAB, 'Enter "Y" or "N":  ');
210         READLN (response);
211
212         IF ((response = 'Y') OR (response = 'y')) THEN builder
213         ELSE
214         BEGIN
215
216                 RESET (rate_file);
217                 READ (rate_file, daily_rates);
218
219                 WRITELN;
220                 WRITE (TAB, 'The file contains exchange rates for:  ');
221                 WRITELN (daily_rates.rate_date);
222
223                 WRITE (TAB,
224                        'Do you want to change it? Enter "Y" or "N":  ');
225                 READLN (response);
226
227                 IF ((response = 'Y') OR (response = 'y')) THEN builder;
228         END;
229
230         WRITELN(CLEAR_SCREEN);
231
232         menu;
233         get_selection (selection);
234
235         WHILE (selection <> no_choice) DO
236
237         BEGIN
238                 conversion (selection);
239
240                 WRITELN;
241                 WRITE (TAB, 'Press New Line to continue');
242                 READLN;
243                 WRITELN (CLEAR_SCREEN);
244
245                 menu;
246                 get_selection (selection);
247         END;
248
249   WRITELN (CLEAR_SCREEN, TAB, 'Exchange program ended.');
250
251   END {EXCHANGE program}.
```

Figure C-10. Compilation Listing of the EXCHANGEP Program (continued)

```
Source file: EXCHANGEP
Compiled on 13-Sep-82 at 10:42:35 by AOS/VS PASCAL Rev 01.00.00.00

                        DECLARATIONS


PROGRAM EXCHANGE ON LINE 1


VARIABLE ALLOCATION
NAME            SCOPE          LINE  SIZE   LOC   TYPE

CURRENCY        GLOBAL           40   36C   50W  ARRAY[COUNTRY] OF
                                                    MONETARY_SYMBOLS
DAILY_RATES     GLOBAL           43   39W   4W   RATEDATA
I               GLOBAL           44   2W    2W   INTEGER
RATE_FILE       GLOBAL           42   4W    44W  RFILE
RESPONSE        GLOBAL           45   1C    0W   CHAR
SELECTION       GLOBAL           39   1W    68W  COUNTRY
US_TO_FOREIGN GLOBAL             41   1W    48W  BOOLEAN


REFERENCED CONSTANTS
NAME            VALUE                                LINE

BELGIAN         ENUMERATION CONSTANT                   16
CANADIAN        ENUMERATION CONSTANT                   21
CLEAR_SCREEN
DUTCH           ENUMERATION CONSTANT                   18
ITALIAN         ENUMERATION CONSTANT                   19
JAPANESE        ENUMERATION CONSTANT                   23
NO_CHOICE       ENUMERATION CONSTANT                   24
SAUDI           ENUMERATION CONSTANT                   22
SWISS           ENUMERATION CONSTANT                   20
TAB                                                    12
W_GERMAN        ENUMERATION CONSTANT                   17


REFERENCED TYPES
NAME            FIELD OFFSET   LINE  SIZE        TYPE DESCRIPTION

COUNTRY                         16   1W         (BELGIAN,W_GERMAN,DUTCH,ITALIAN,
                                                  SWISS,CANADIAN,SAUDI,JAPANESE,
                                                  NO_CHOICE)
MONETARY_SYMBOLS
                                26   4C         PACKED ARRAY[1..4] OF CHAR
RATEDATA                        27   39W        RECORD
                      0C             6C           RATE_DATE :PACKED ARRAY[1..6]
                                                    OF CHAR
                      3W            36W          RATES :RECORD
                      0W            18W            US_FOREIGN :ARRAY[COUNTRY]
                                                    OF REAL
                      18W           18W            FOREIGN_US :ARRAY[COUNTRY]
                                                    OF REAL
RFILE                           35   4W         FILE OF RATEDATA
```

Figure C-10. Compilation Listing of the EXCHANGEP Program (continued)

```
PROCEDURE BUILDER ON LINE 48


VARIABLE ALLOCATION
NAME            SCOPE          LINE  SIZE   LOC   TYPE

RATE_REC        LOCAL           56   39W    14W  RATEDATA
SELECTION       LOCAL           55   1W     12W  COUNTRY




PROCEDURE MENU ON LINE 86
NO DECLARED NAMES




PROCEDURE GET_SELECTION ON LINE 114


VARIABLE ALLOCATION
NAME            SCOPE          LINE  SIZE   LOC   TYPE

CURRENCY_CODE LOCAL            121   2W     12W  INTEGER
SELECTION       PARAMETER      115   1W          COUNTRY




PROCEDURE CONVERSION ON LINE 151


VARIABLE ALLOCATION
NAME            SCOPE          LINE  SIZE   LOC   TYPE

INVAL           LOCAL          155   2W     12W  REAL
OUTVAL          LOCAL          155   2W     14W  REAL
RATE            LOCAL          155   2W     16W  REAL
RESPONSE        LOCAL          156   1C     18W  CHAR
SELECTION       PARAMETER      152   1W          COUNTRY
```

Figure C-10. Compilation Listing of the EXCHANGEP Program (concluded)

```
USER PROGRAM exchangep SWAT AUDIT ON 09/13/82 AT 13:14:34

AOS/VS SWAT Revision 02.20.00.00 ON 09/13/82 AT 13:14:39
PROGRAM -- :UDD:TOMF:SAMPLES:EXCHANGEP
>% This session illustrates the use of SWAT commands
>% with a program written in PASCAL.
>%
>% When invoking the SWAT debugger, we used the /AUDIT= switch
>% to open an audit file.
>%
>% Confirm that auditing is on.
>AUDIT
ON
>% Display the working environment, which should be the main procedure.
>ENVIRONMENT
:EXCHANGE
>% List all the source code.
>LIST @ALL
1       PROGRAM
2               exchange (INPUT, OUTPUT, rate_file);
3
4       {       This program calculates currency exchange values based on
5               daily international exchange rates.  The operator selects
6               the type of exchange, then enters the incoming amount.  The
7               program performs the calculation and displays the result.    }
8
9       {$C+    Compiler directive allows ASCII '<nn>' notation in the program }
10
11      CONST
12              TAB = '<11>';
13              CLEAR_SCREEN = '<14>';
14
15      TYPE
16              country = (belgian,
17                         w_german,
18                         dutch,
19                         italian,
20                         swiss,
21                         canadian,
22                         saudi,
23                         japanese,
24                         no_choice);
25
26              monetary_symbols = PACKED ARRAY [1..4] OF CHAR;
27              ratedata = RECORD
28                          rate_date : PACKED ARRAY [1..6] OF CHAR;
29                          rates : RECORD
30                                  us_foreign : ARRAY[country] OF REAL;
31                                  foreign_us : ARRAY[country] OF REAL;
32                                     END; { rates }
33                          END; { ratedata }
34
35              rfile = FILE OF ratedata;
36
```

Figure C-11. Audit File of the EXCHANGEP SWAT Session (continues)

```
37      VAR
38
39              selection : country;
40              currency : ARRAY [country] OF monetary_symbols;
41              us_to_foreign : BOOLEAN;
42              rate_file : rfile;
43              daily_rates : ratedata;
44              i : INTEGER;
45              response : CHAR;
46
47      {-------------------------------------------------------------------}
48      PROCEDURE
49              builder;
50
51      {       This procedure builds the currency rate file for the exchange
52              program.    }
53
54      VAR
55              selection : country;
56              rate_rec  : ratedata;
57
58      BEGIN
59              REWRITE (rate_file);
60
61              WRITELN (CLEAR_SCREEN);
62              WRITELN (TAB, 'Building new currency rate file');
63              WRITELN;
64              WRITE (TAB, 'Enter the date in yymmdd format:  ');
65              READLN (rate_rec.rate_date);
66
67              selection := belgian;
68              REPEAT
69                      WRITE (TAB, 'Enter rate for US$ to ',
70                                      currency[selection], ' :  ');
71                      READLN (rate_rec.rates.us_foreign[selection]);
72
73                      WRITE (TAB, 'Enter rate for ' , currency[selection],
74                                      ' to US$ :  ');
75                      READLN (rate_rec.rates.foreign_us[selection]);
76                      selection := SUCC(selection);
77
78              UNTIL (selection = no_choice);
79
80              WRITE (rate_file, rate_rec);
81
82              RESET (rate_file);
83              READ (rate_file, daily_rates);
84      END;
85      {-------------------------------------------------------------------}
```

Figure C-11. Audit File of the EXCHANGEP SWAT Session (continued)

```
86      PROCEDURE
87              menu;
88
89      BEGIN
90              WITH daily_rates DO
91              BEGIN
92              WRITE ('Rates for ');
93              WRITE (rate_date[3], rate_date[4],'/');
94              WRITE (rate_date[5], rate_date[6],'/');
95              WRITE (rate_date[1], rate_date[2]);
96              WRITELN;
97              WRITELN ('Identify the type of currency:');
98              WRITELN;
99              WRITELN (TAB, '1   Belgian francs');
100             WRITELN (TAB, '2   W. German marks');
101             WRITELN (TAB, '3   Dutch guilders');
102             WRITELN (TAB, '4   Italian lire');
103             WRITELN (TAB, '5   Swiss francs');
104             WRITELN (TAB, '6   Canadian dollars');
105             WRITELN (TAB, '7   Saudi riyals');
106             WRITELN (TAB, '8   Japanese yen');
107             WRITELN (TAB, '(Type 0 to end the program.)');
108             WRITELN;
109             WRITE ('Enter the currency code:  ');
110             END;
111
112     END;
113     {-------------------------------------------------------------------}
114     PROCEDURE
115             get_selection ( VAR selection : country );
116
117     {       Gets the users currency code input and sets parameter selection
118             to the corresponding country.                                 }
119
120     VAR
121             currency_code : INTEGER;
122
123     BEGIN
124
125             READLN (currency_code);
126
127             IF (currency_code >= 0) AND (currency_code <= 8) THEN
128
129                     BEGIN
130
131                     CASE currency_code OF
132
133                             0 : selection := no_choice;
134                             1 : selection := belgian;
135                             2 : selection := w_german;
136                             3 : selection := dutch;
137                             4 : selection := italian;
138                             5 : selection := swiss;
139                             6 : selection := canadian;
140                             7 : selection := saudi;
141                             8 : selection := japanese;
```

Figure C-11. Audit File of the EXCHANGEP SWAT Session (continued)

```
142                     END
143                     END
144
145             ELSE    selection := no_choice;
146
147     END; { get_selection }

148
149     {-------------------------------------------------------------------}
150
151     PROCEDURE
152             conversion ( selection : country );
153
154     VAR
155             inval, outval, rate : REAL;
156             response            : CHAR;
157
158     BEGIN
159             WRITELN;
160             WRITELN (TAB, 'Do you want to convert US$ into Foreign currency?');
161             WRITE (TAB, 'Enter "Y" or "N":  ');
162             READLN (response);
163
164             IF ((response = 'Y') OR (response = 'y')) THEN
165                     BEGIN
166                             us_to_foreign := TRUE;
167                             rate := daily_rates.rates.us_foreign[selection];
168                     END
169
170             ELSE
171                     BEGIN
172                             us_to_foreign := FALSE;
173                             rate := daily_rates.rates.foreign_us[selection];
174                     END;
175
176             WRITELN;
177
178             IF us_to_foreign THEN WRITE (TAB, 'Enter US$:  ')
179             ELSE WRITE (TAB, 'Enter ', currency[selection], ':  ');
180
181             READLN (inval);
182             outval := inval * rate;
183
184             WRITELN;
185             IF us_to_foreign THEN WRITELN ('$', inval:10:2,
186               ' US equivalent to:  ', outval:10:2, ' ', currency[selection])
187             ELSE WRITELN (inval:10:2, ' ', currency[selection],
188               ' equivalent to:  $', outval:10:2, ' US');
189
190     END;
191     {-------------------------------------------------------------------}
```

Figure C-11. Audit File of the EXCHANGEP SWAT Session (continued)

```
192
193
194     BEGIN { main procedure }
195
196             currency[belgian] := 'BF';
197             currency[w_german] := 'DM';
198             currency[dutch] := 'gld';
199             currency[italian] := 'Lit';
200             currency[swiss] := 'SF';
201             currency[canadian] := 'Can$';
202             currency[saudi] := 'SRI';
203             currency[japanese] := 'Y';
204
205             WRITELN (CLEAR_SCREEN, TAB, TAB, 'Exchange Program');
206             WRITELN;
207
208             WRITELN (TAB, 'Do you want to create a new data file?');
209             WRITE (TAB, 'Enter "Y" or "N":  ');
210             READLN (response);
211
212             IF ((response = 'Y') OR (response = 'y')) THEN builder
213             ELSE
214             BEGIN
215
216                     RESET (rate_file);
217                     READ (rate_file, daily_rates);
218
219                     WRITELN;
220                     WRITE (TAB, 'The file contains exchange rates for:  ');
221                     WRITELN (daily_rates.rate_date);
222
223                     WRITE (TAB,
224                             'Do you want to change it? Enter "Y" or "N":  ');
225                     READLN (response);
226
227                     IF ((response = 'Y') OR (response = 'y')) THEN builder;
228             END;
229
230             WRITELN(CLEAR_SCREEN);
231
232             menu:
233             get_selection (selection);
234
235             WHILE (selection <> no_choice) DO
236
237             BEGIN
238                     conversion (selection);
239
240                     WRITELN;
241                     WRITE (TAB, 'Press New Line to continue');
242                     READLN;
243                     WRITELN (CLEAR_SCREEN);
```

Figure C-11. Audit File of the EXCHANGEP SWAT Session (continued)

```
244
245                     menu:
246                     get_selection (selection);
247             END;
248
249     WRITELN (CLEAR_SCREEN, TAB, 'Exchange program ended.');
250
251     END {EXCHANGE program}.



>% Display the working directory.
>DIRECTORY
:UDD:TOMF:SAMPLES
>% Display the current search list.
>SEARCHLIST
:UDD:TOMF :UTIL :SWAT :LANG_RT :PASCAL
>% Obtain information about global symbols.
>DESCRIBE CLEAR_SCREEN
CLEAR_SCREEN (1 byte) CONST CHAR
>DESCRIBE country, swiss, monetary_symbols
COUNTRY (1 word) TYPE (BELGIAN, W_GERMAN, DUTCH, ITALIAN, SWISS, CANADIAN,
    SAUDI, JAPANESE, NO_CHOICE)
SWISS (1 word) CONST COUNTRY
MONETARY_SYMBOLS (4 bytes) TYPE PACKED ARRAY [1..4] OF CHAR
>DESCRIBE rfile  selection  currency
RFILE (4 words) TYPE FILE OF RATEDATA
SELECTION (1 word at 16000001046R8 words) VAR COUNTRY
CURRENCY (36 bytes at 34000002050R8 bytes) VAR ARRAY [COUNTRY] OF
    MONETARY_SYMBOLS
>DESCRIBE i, response
I (2 words at 16000000744R8 words) VAR INTEGER
RESPONSE (1 byte at 34000001704R8 bytes) VAR CHAR
>DESCRIBE get_selection
GET_SELECTION (at 16001744532R8 words) SUBROUTINE line 115 to 147
>% Set the environment to that of the procedure called builder.
>ENVIRONMENT builder
:EXCHANGE:BUILDER
>% Display information about this procedure's symbols.
>DESCRIBE selection, rate_rec
SELECTION (1 word at +14R8 words) VAR COUNTRY
RATE_REC (39 words at +16R8 words) VAR RATEDATA
>% Set the environment to that of the procedure called get_selection.
>ENVIRONMENT ^get_selection
:EXCHANGE:GET_SELECTION
>% Display information about this procedure's symbols.
>DESCRIBE currency_code
CURRENCY_CODE (2 words at +14R8 words) VAR INTEGER
>% Set the environment to that of the procedure called conversion.
>ENVIRONMENT ^conversion
:EXCHANGE:CONVERSION
>% Display information about this procedure's symbols.
>DESCRIBE inval response
INVAL (2 words at +14R8 words) VAR REAL
RESPONSE (1 byte at +22R8 words) VAR CHAR
>% Return to the environment of the main procedure
>ENVIRONMENT @MAIN
:EXCHANGE
```

Figure C-11. Audit File of the EXCHANGEP SWAT Session (continued)

```
>% Set breakpoints at key statements
>BREAKPOINT 76 ACTION := 'LIST 76;TYPE rate_rec.rates.us_foreign[selection];
  TYPE rate_rec.rates.foreign_us[selection]'
Set at :EXCHANGE:BUILDER:76 action="LIST 76;
  TYPE rate_rec.rates.us_foreign[selection];
  TYPE rate_rec.rates.foreign_us[selection]"
>BREAKPOINT 147 ACTION := 'TYPE currency_code;TYPE selection'
Set at :EXCHANGE:GET_SELECTION:147 action="TYPE currency_code;
  TYPE selection"
>BREAKPOINT 167 ACTION := 'LIST 167; WALKBACK'
Set at :EXCHANGE:CONVERSION:167 action="LIST 167; WALKBACK"
>BREAKPOINT 173 ACTION := 'LIST 173; WALKBACK'
Set at :EXCHANGE:CONVERSION:173 action="LIST 173; WALKBACK"
>BREAKPOINT 182 ACTION := 'TYPE inval, rate'
Set at :EXCHANGE:CONVERSION:182 action="TYPE inval, rate"
>BREAKPOINT 212
Set at :EXCHANGE:212
>% Display the breakpoints currently in effect.
>BREAKPOINT
Set at :EXCHANGE:BUILDER:76 action="LIST 76;
  TYPE rate_rec.rates.us_foreign[selection];
  TYPE rate_rec.rates.foreign_us[selection]"
Set at :EXCHANGE:GET_SELECTION:147 action="TYPE currency_code;
  TYPE selection"
Set at :EXCHANGE:CONVERSION:167 action="LIST 167; WALKBACK"
Set at :EXCHANGE:CONVERSION:173 action="LIST 173; WALKBACK"
Set at :EXCHANGE:CONVERSION:182 action="TYPE inval, rate"
Set at :EXCHANGE:212
>% Begin execution of the program.
>CONTINUE

Breakpoint trap at :EXCHANGE:212
>% The program asked if we wanted to create a new data file; we said N.
>% We can, however, change our response.
>TYPE response
"N"
>SET response := 'Y'
Old value: "N"
New Value: "Y"
>CLEAR 212
Cleared at :EXCHANGE:212
>CONTINUE

Breakpoint trap at :EXCHANGE:BUILDER:76 action="LIST 76;
  TYPE rate_rec.rates.us_foreign[selection];
  TYPE rate_rec.rates.foreign_us[selection]"
76C                    selection := SUCC(selection);
 4.752000E+01
 2.100000E-02
>% We entered the date and the first exchange rate pair (Belgian francs)
>CONTINUE
```

Figure C-11. Audit File of the EXCHANGEP SWAT Session (continued)

```
Breakpoint trap at :EXCHANGE:BUILDER:76 action="LIST 76;
  TYPE rate_rec.rates.us_foreign[selection];
  TYPE rate_rec.rates.foreign_us[selection]"
76C                    selection := SUCC(selection);
 2.492500E+00
 4.015000E-01


>% We entered the next pair of exchange rates (West German marks)
>% We'll clear this breakpoint, then fill in the rest of the table.
>TYPE selection
W_GERMAN
>CLEAR 76
Cleared at :EXCHANGE:BUILDER:76 action="LIST 76;
  TYPE rate_rec.rates.us_foreign[selection];
  TYPE rate_rec.rates.foreign_us[selection]"
>CONTINUE

Breakpoint trap at :EXCHANGE:GET_SELECTION:147
  action="TYPE currency_code;TYPE selection"
2
W_GERMAN
>% The program displayed the menu.  We entered code 2 (West German marks)
>% We can change the variable and redirect execution.  This time we'll
>% convert Swiss francs.
>SET selection := swiss
Old value: W_GERMAN
New Value: SWISS
>CONTINUE

Breakpoint trap at :EXCHANGE:CONVERSION:167 action="LIST 167; WALKBACK"
167C                    rate := daily_rates.rates.us_foreign[selection];
Current location is :EXCHANGE:CONVERSION:167
Called from :EXCHANGE:238
>% The program asked if we are exchanging US dollars to foreign currency.
>% We responded Y.
>TYPE daily_rates.rates.us_foreign[selection]
 2.116500E+00
>CONTINUE

Breakpoint trap at :EXCHANGE:CONVERSION:182 action="TYPE inval, rate"
 2.500000E+02
 2.116500E+00
>TYPE inval * rate
 5.291250E+02
>% The result of the exchange should be 529.12 Swiss francs.
>CONTINUE

Breakpoint trap at :EXCHANGE:GET_SELECTION:147
  action="TYPE currency_code;TYPE selection"
6
CANADIAN
```

Figure C-11. Audit File of the EXCHANGEP SWAT Session (continued)

```
>% The program displayed the expected result.  It then presented the menu
>% and we entered 6 (for Canadian dollars).
>CONTINUE

Breakpoint trap at :EXCHANGE:CONVERSION:173 action="LIST 173; WALKBACK"
173C                          rate := daily_rates.rates.foreign_us[selection];
Current location is :EXCHANGE:CONVERSION:173
Called from :EXCHANGE:238
>% The program asked the exchange direction.  We entered N.
>% We will set a breakpoint at 190 so that we do not exit that procedure.
>BREAKPOINT 190
Set at :EXCHANGE:CONVERSION:190
>CONTINUE

Breakpoint trap at :EXCHANGE:CONVERSION:182 action="TYPE inval, rate"
 2.500000E+02
 8.014000E-01
>CONTINUE

Breakpoint trap at :EXCHANGE:CONVERSION:190
>SET inval := 250.
Old value:  2.500000E+02
New Value:  2.500000E+02
>SET us_to_foreign := TRUE
Old value: FALSE
New Value: TRUE
>SET rate := daily_rates.rates.us_foreign[selection]
Old value:  8.014000E-01
New Value:  1.247800E+00
>CONTINUE AT 184

Breakpoint trap at :EXCHANGE:CONVERSION:190
>CONTINUE

Breakpoint trap at :EXCHANGE:GET_SELECTION:147
  action="TYPE currency_code:TYPE selection"
0
NO_CHOICE
>% We entered 0 to end the program.  If we continue, both the program
>% and SWAT will terminate.  We can also end this session by typing
>BYE

SWAT TERMINATED
```

Figure C-11. Audit File of the EXCHANGEP SWAT Session (concluded)

A Sample C Program and Audit File

This C program, called EXCHANGEC, is similar to the sample PL/I program used in Chapter 3 of this manual. It calculates currency exchange values based on daily exchange rates. The program consists of a single module, which interacts with the operator and a file called RATE_FILE.

The compilation and link commands are as follows:

CC/DEBUG/LS EXCHANGEC

CCL/DEBUG EXCHANGEC

(The CCL macro automatically includes the SWATI file; do not specify it as an argument.)

Figure C-12 presents the compilation listing of the EXCHANGEC program. A sample SWAT session audit file appears in Figure C-13.

```
Source file: exchangec
Compiled on 15-Sep-82 at 10:41:39 by AOS/VS C Rev 01.00.08.126
Options:  cc/l=exchangec.ls/debug

   1 0  /*      This program calculates currency exchange values based on
   2*0          daily international exchange rates.  The operator selects
   3*0          the type of exchange, then enters the incoming amount.  The
   4*0          program performs the calculation and displays the result.    */
   5 0
   6 0  #nolist
 633 0  #list
 634 0
 635 0  #define TRUE 1
 636 0  #define FALSE 0
 637 0  typedef enum    {belgian,
 638 1                   w_german,
 639 1                   dutch,
 640 1                   italian,
 641 1                   swiss,
 642 1                   canadian,
 643 1                   saudi,
 644 1                   japanese,
 645 1                   no_choice} country ;
 646 0
 647 0  typedef char monetary_symbols[5];
 648 0  typedef struct {
 649 1                    char rate_date[10];
 650 1                    struct {
 651 2                         float us_foreign[no_choice];
 652 2                         float foreign_us[no_choice];
 653 2                    } rates;
 654 1          } ratedata;
 655 0
 656 0  FILE *rate_file;
 657 0
 658 0  country selection;
 659 0  static monetary_symbols currency[no_choice] =
 660 0                {"BF","DM","gld","Lit","SF","Can$","SRI","Y",""};
```

Figure C-12. Compilation Listing of the EXCHANGEC Program (continues)

```
661 0  int us_to_foreign;
662 0  ratedata daily_rates;
663 0  int i;
664 0  char response;
665 0  /*---------------------------------------------------------------*/


666 0   builder()
667 0
668 0  /*      This procedure builds the currency rate file for the
669*0          exchange program.                                        */
670 0  {
671 1  country selection;
672 1  ratedata rate_rec;
673 1
674 1          rate_file = fopen ("rate_file", "u");
675 1
676 1          printf ("\f");
677 1          printf ("\tBuilding new currency rate file\n\n");
678 1
679 1          printf ("\tEnter the date in yymmdd format: ");
680 1          scanf ("%s", rate_rec.rate_date);
681 1
682 1          for (selection = belgian; selection != no_choice; selection++)
683 1          {
684 2          printf ("\tEnter rate for US$ to %s : ", currency[selection]);
685 2          scanf ("%f", &rate_rec.rates.us_foreign[selection]);
686 2
687 2          printf ("\tEnter rate for %s to US$ : ", currency[selection]);
688 2          scanf ("%f", &rate_rec.rates.foreign_us[selection]);
689 2          }
690 1
691 1          fwrite (&rate_rec, sizeof(rate_rec), 1, rate_file);
692 1
693 1          fseek (rate_file, 0L, 0);
694 1          fread (&daily_rates, sizeof(daily_rates), 1, rate_file);
695 1  }
696 0  /*---------------------------------------------------------------*/
697 0  menu ()
698 0
699 0  /* Displays the selection menu.                                  */
700 0
701 0  {
702 1  printf ("Rates for ");
703 1  printf ("%c%c/", daily_rates.rate_date[2], daily_rates.rate_date[3]);
704 1  printf ("%c%c/", daily_rates.rate_date[4], daily_rates.rate_date[5]);
705 1  printf ("%c%c\n", daily_rates.rate_date[0], daily_rates.rate_date[1]);
706 1
707 1          printf ("Identify the type of currency:\n\n");
708 1          printf ("\t1   Belgian francs\n");
709 1          printf ("\t2   W. German marks\n");
710 1          printf ("\t3   Dutch guilders\n");
711 1          printf ("\t4   Italian lire\n");
712 1          printf ("\t5   Swiss francs\n");
713 1          printf ("\t6   Canadian dollars\n");
714 1          printf ("\t7   Saudi riyals\n");
```

Figure C-12. Compilation Listing of the EXCHANGEC Program (continued)

```
>% The program asked if we are exchanging US dollars to foreign currency.
>% We responded Y.
>TYPE daily_rates.rates.us_foreign[selection]
 2.116500E+00
>CONTINUE

Breakpoint trap at :conversion:776 action="TYPE inval, rate"
 2.500000E+02
 2.116500E+00
>TYPE inval * rate
 5.291250E+02
>% The result of the exchange should be 529.12 Swiss francs.
>CONTINUE

Breakpoint trap at :get_selection:744 action="TYPE currency_code:
  TYPE selection"
6
canadian
>% The program displayed the expected result.  It then presented the menu
>% and we entered 6 (for Canadian dollars).
>CONTINUE

Breakpoint trap at :conversion:767 action="LIST 767"
767C                    rate = daily_rates.rates.foreign_us[selection];
>% The program asked the exchange direction.  We entered N.
>% We will set a breakpoint at 785 so that we do not exit that procedure.
>BREAKPOINT 785
Set at :conversion:785
>CONTINUE

Breakpoint trap at :conversion:776 action="TYPE inval, rate"
 2.500000E+02
 8.014000E-01
>CONTINUE

Breakpoint trap at :conversion:785
>SET inval = 250.
Old value:  2.500000E+02
New Value:  2.500000E+02
>SET us_to_foreign = 1
Old value: 0
New Value: 1
>SET rate = daily_rates.rates.us_foreign[selection]
Old value:  8.014000E-01
New Value:  1.247800E+00
>CONTINUE AT 779

Breakpoint trap at :conversion:785
>CONTINUE

Breakpoint trap at :get_selection:744 action="TYPE currency_code:
  TYPE selection"
0
no_choice


>% We entered 0 to end the program.  If we continue, both the program
>% and SWAT will terminate.  We can also end this session by typing
>BYE

SWAT TERMINATED
```

Figure C-13. Audit File of the EXCHANGEC SWAT Session (concluded)

End of Appendix

```
>% Display the breakpoints currently in effect.
>BREAKPOINT
Set at :builder:684 action="TYPE rate_rec.rates.us_foreign[selection];
  TYPE rate_rec.rates.foreign_us[selection]"
Set at :get_selection:744 action="TYPE currency_code;TYPE selection"
Set at :conversion:762 action="LIST 762"
Set at :conversion:767 action="LIST 767"
Set at :conversion:776 action="TYPE inval, rate"
Set at :main:799
>% Begin execution of the program.
>CONTINUE

Breakpoint trap at :main:799
>% The program asked if we wanted to create a new data file; we said N.
>% We can, however, change our response.
>TYPE response
"N"
>SET response = "Y"
Old value: "N"
New Value: "Y"
>CLEAR 799
Cleared at :main:799
>CONTINUE

Breakpoint trap at :builder:684
  action="TYPE rate_rec.rates.us_foreign[selection];
  TYPE rate_rec.rates.foreign_us[selection]"
 1.222220E-76
-9.414063E-01
>% We entered the date and the first exchange rate pair (Belgian francs)
>CONTINUE

Breakpoint trap at :builder:684
  action="TYPE rate_rec.rates.us_foreign[selection];
  TYPE rate_rec.rates.foreign_us[selection]"
 7.183055E-50
 0.000000E+00
>% We entered the next pair of exchange rates (West German marks)
>% We'll clear this breakpoint, then fill in the rest of the table.
>TYPE selection
w_german
>CLEAR 684
Cleared at :builder:684
  action="TYPE rate_rec.rates.us_foreign[selection];
  TYPE rate_rec.rates.foreign_us[selection]"
>CONTINUE

Breakpoint trap at :get_selection:744 action="TYPE currency_code;
  TYPE selection"
2
w_german
>% The program displayed the menu.  We entered code 2 (West German marks)
>% We can change the variable and redirect execution.  This time we'll
>% enter 5 for Swiss francs.
>SET selection = swiss
Old value: w_german
New Value: swiss
>CONTINUE

Breakpoint trap at :conversion:762 action="LIST 762"
762C                    rate = daily_rates.rates.us_foreign[selection];
```

Figure C-13. Audit File of the EXCHANGEC SWAT Session (continued)

```
>% List this procedure.
>DESCRIBE :conversion
conversion (at 160017326518 words) extern line 751 to 785 int ()
>LIST 751 785
751     {
752     float inval, outval, rate;
753     char response;
754
755             printf ("\nDo you want to convert US$ into Foreign currency?");
756             printf ("\tEnter Y or N: ");
757             scanf ("%s", &response);
758
759             if ((response == 'Y') || (response == 'y'))
760             {
761                     us_to_foreign = TRUE;
762                     rate = daily_rates.rates.us_foreign[selection];
763             }
764             else
765             {
766                     us_to_foreign = FALSE;
767                     rate = daily_rates.rates.foreign_us[selection];
768             }
769
770             printf("\n");
771
772             if (us_to_foreign == TRUE)  printf ("\n \tEnter US$:  ");
773             else printf ("\n \tEnter %s :", currency[selection]);
774
775             scanf ("%f", &inval);
776             outval = inval * rate;
777
778             printf ("\n");
779             if (us_to_foreign == TRUE)
780                     printf ("\n\t$ %5.2f  US equivalent to: %5.2f  %s",
781                                     inval, outval, currency[selection]);
782
783             else    printf ("\n\t %5.2f %s equivalent to: $ %5.2f US",
784                                     inval, currency[selection], outval);
785     }
>% Display information about this procedure's symbols.
>DESCRIBE inval  response
inval (2 words at +14R8 words) auto float
response (1 byte at +12R8 words) auto char
>% Return to the environment of the main procedure
>ENVIRONMENT @MAIN
:builder
>% Set breakpoints at key statements
>BREAKPOINT 684 ACTION="TYPE rate_rec.rates.us_foreign[selection];
  TYPE rate_rec.rates.foreign_us[selection]"
Set at :builder:684 action="TYPE rate_rec.rates.us_foreign[selection];
  TYPE rate_rec.rates.foreign_us[selection]"
>BREAKPOINT 744 ACTION = "TYPE currency_code;TYPE selection"
Set at :get_selection:744 action="TYPE currency_code;TYPE selection"
>BREAKPOINT 762 ACTION = "LIST 762"
Set at :conversion:762 action="LIST 762"
>BREAKPOINT 767 ACTION = "LIST 767"
Set at :conversion:767 action="LIST 767"
>BREAKPOINT 776 ACTION = "TYPE inval, rate"
Set at :conversion:776 action="TYPE inval, rate"
>BREAKPOINT 799
Set at :main:799
```

Figure C-13. Audit File of the EXCHANGEC SWAT Session (continued)

```
826             {
827                     conversion ();
828
829                     printf ("\n\tEnter Y to continue : ");
830                     scanf ("%s", &response);
831                     printf ("\f");
832
833                     menu ();
834                     get_selection ();
835             }
836
837     printf ("\f\tExchange program ended.");
838     }
>% Display information about this procedure's symbols.
>DESCRIBE selection, ratedata
selection (1 word at 16000011703R8 words) country
ratedata (37 words) typedef
    rate_date (10 bytes at +0R8 bytes) char [10]
    rates (32 words at +5R8 words) struct
      us_foreign (16 words at +0R8 words) float [8]
      foreign_us (16 words at +20R8 words) float [8]
>% Set the environment to that of the procedure called get_selection.
>ENVIRONMENT ^get_selection
:get_selection
>% List this procedure.
>DESCRIBE :get_selection
get_selection (at 16001732543R8 words) extern line 725 to 744 int ()
>LIST 725 744
725     {
726
727     int currency_code;
728
729             scanf ("%d", &currency_code);
730
731             switch  (currency_code)
732             {
733                     case 1 : selection = belgian; break;
734                     case 2 : selection = w_german; break;
735                     case 3 : selection = dutch; break;
736                     case 4 : selection = italian; break;
737                     case 5 : selection = swiss; break;
738                     case 6 : selection = canadian; break;
739                     case 7 : selection = saudi; break;
740                     case 8 : selection = japanese; break;
741
742                     default : selection = no_choice; break;
743             }
744     }
>% Display information about this procedure's symbols.
>DESCRIBE currency_code
currency_code (2 words at +12R8 words) auto int
>% Set the environment to that of the procedure called conversion.
>ENVIRONMENT ^conversion
:conversion
```

Figure C-13. Audit File of the EXCHANGEC SWAT Session (continued)

```
>SEARCHLIST
 :UDD:TOMF :UTIL :SWAT :LANG_RT :C
>% Obtain information about global symbols.
>DESCRIBE country, swiss, monetary_symbols
.country (1 word) typedef { belgian, w_german, dutch, italian, swiss, canadian,
    saudi, japanese, no_choice }
swiss (1 word) enum
monetary_symbols (5 bytes) typedef char [5]
>DESCRIBE ratedata  selection  currency
ratedata (37 words) typedef
    rate_date (10 bytes at +0R8 bytes) char [10]
    rates (32 words at +5R8 words) struct
      us_foreign (16 words at +0R8 words) float [8]
      foreign_us (16 words at +20R8 words) float [8]
selection (1 word at +12R8 words) country
currency (45 bytes at 34000023664R8 bytes) static typedef char [9][5]
>DESCRIBE i, response
i (2 words at 16000005014R8 words) extern int
response (1 byte at 34000023604R8 bytes) extern char
>% Set the environment to that of the procedure called main.
>ENVIRONMENT :main
 :main
>% List this procedure.
>DESCRIBE :main
main (at 16001733173R8 words) extern line 792 to 838 int ()
>LIST 792 838
792     {
793             printf ("\f\t\tExchange Program\n\n");
794
795             printf ("\tDo you want to create a new data file?\n");
796             printf ("\tEnter Y or N:  ");
797             scanf ("%s", &response);
798
799             if ((response == 'Y') || (response == 'y')) builder ();
800             else
801             {
802
803                     rate_file = fopen ("rate_file", "r");
804                     fread (&daily_rates, sizeof(daily_rates), 1, rate_file);
805
806                     printf ("\n\tThe file contains exchange rates for:  ");
807                     printf ("%s\n", daily_rates.rate_date);
808
809                     printf ("\tDo you want to change it? Enter Y or N:  ");
810                     scanf ("%s", &response);
811
812                     if ((response == 'Y') || (response == 'y'))
813                     {
814                             fclose (rate_file);
815                             builder ();
816                     }
817             }
818
819             printf ("\f");
820
821             menu ();
822             get_selection ();
823
824             while (selection != no_choice)
825
```

Figure C-13. Audit File of the EXCHANGEC SWAT Session (continued)

```
USER PROGRAM exchangec SWAT AUDIT ON 09/15/82 AT 12:27:38

AOS/VS SWAT Revision 02.20.00.00 ON 09/15/82 AT 12:27:42
PROGRAM -- :UDD:TOMF:SAMPLES:EXCHANGEC
>% This session illustrates the use of SWAT commands
>% with a program written in the C language.
>%
>% When invoking the SWAT debugger, we used the /AUDIT= switch
>% to open an audit file.
>%
>% Confirm that auditing is on.
>AUDIT
ON
>% Display the environments in this module.
>ENVIRONMENT @ALL
:builder
:menu
:get_selection
:conversion
:main
CURRENT ENVIRONMENT IS :builder
>% Display this procedure.
>DESCRIBE :builder
builder (at 16001732066R8 words) extern line 670 to 695 int ()
>LIST 670 695
670     {
671     country selection;
672     ratedata rate_rec;
673
674             rate_file = fopen ("rate_file", "u");
675
676             printf ("\f");
677             printf ("\tBuilding new currency rate file\n\n");
678
679             printf ("\tEnter the date in yymmdd format: ");
680             scanf ("%s", rate_rec.rate_date);
681
682             for (selection = belgian; selection != no_choice; selection++)
683             {
684             printf ("\tEnter rate for US$ to %s : ", currency[selection]);
685             scanf ("%f", &rate_rec.rates.us_foreign[selection]);
686
687             printf ("\tEnter rate for %s to US$ : ", currency[selection]);
688             scanf ("%f", &rate_rec.rates.foreign_us[selection]);
689             }
690
691             fwrite (&rate_rec, sizeof(rate_rec), 1, rate_file);
692
693             fseek (rate_file, 0L, 0);
694             fread (&daily_rates, sizeof(daily_rates), 1, rate_file);
695     }
>% Display the working directory.
>DIRECTORY
:UDD:TOMF:SAMPLES
>% Display the current search list.
```

Figure C-13. Audit File of the EXCHANGEC SWAT Session (continues)

```
Function conversion on line 751
-------------------------------

Identifier     Storage   Size  Loc.Off.  Line   Type

inval          auto        2W    12W      752   float
outval         auto        2W    14W      752   float
rate           auto        2W    16W      752   float
response       auto        1C    10+ 0C   753   char


Function main on line 792
-------------------------

       No declared entries.

# -- This identifier was not referenced.

Character following size field:
    C      Size is number of bytes ( 8 bits)
    W      Size is number of words (16 bits)

(Location field is word address + bit/byte offset)
```

Figure C-12. Compilation Listing of the EXCHANGEC Program (concluded)

```
strcpy          #extern      0W  ext.       456   char *()
strncpy         #extern      0W  ext.       461   char *()
strcat          #extern      0W  ext.       470   char *()
strncat         #extern      0W  ext.       475   char *()
strlen          #extern      0W  ext.       484   int ()
strsave         #extern      0W  ext.       488   char *()
strnsave        #extern      0W  ext.       493   char *()
strchr          #extern      0W  ext.       501   char *()
strrchr         #extern      0W  ext.       507   char *()
wdleng          #extern      0W  ext.       517   int ()
$classify       #extern   -----  ext.       570   short int []
rate_file        extern      2W  ext.       656   FILE *
selection        extern      1W  ext.       658   country
us_to_foreign   extern       2W  ext.       661   int
daily_rates      extern     37W  ext.       662   ratedata
i               #extern      2W  ext.       663   int
response         extern      1C  ext.       664   char
free            #builtin     0W      0W     415   void ()
sqrt            #builtin     0W      0W     515   double ()


Function builder on line 670
----------------------------

Identifier      Storage   Size  Loc.Off.  Line   Type

selection       auto         1W    10W     671   country
rate_rec        auto        37W    12W     672   ratedata


Function menu on line 701
-------------------------

        No declared entries.


Function get_selection on line 725
----------------------------------

Identifier      Storage   Size  Loc.Off.  Line   Type

currency_code   auto         2W    10W     727   int
```

Figure C-12. Compilation Listing of the EXCHANGEC Program (continued)

```
swiss           const.   ----- -----       641   Constant 4
canadian        const.   ----- -----       642   Constant 5
saudi           const.   ----- -----       643   Constant 6
japanese        const.   ----- -----       644   Constant 7
no_choice       const.   ----- -----       645   Constant 8

FILE            typedef    32W -----        67   struct _iobuf
country         typedef     1W -----       645   enum
monetary_symbols
                typedef     5C -----       647   char [5]
ratedata        typedef    37W -----       654   struct

currency        static     45C     0+ 0C  659   monetary_symbols [9]
$iob           #extern   2048W  ext.        75   FILE [64]
fclose          extern      0W  ext.        83   int ()
fopen           extern      0W  ext.        87   FILE *()
freopen        #extern      0W  ext.       101   FILE *()
fgetc          #extern      0W  ext.       109   int ()
fputc          #extern      0W  ext.       114   int ()
$getc          #extern      0W  ext.       120   int ()
$putc          #extern      0W  ext.       124   int ()
ungetc         #extern      0W  ext.       128   void ()
fputs          #extern      0W  ext.       136   int ()
fgets          #extern      0W  ext.       143   char *()
puts           #extern      0W  ext.       152   char *()
gets           #extern      0W  ext.       158   char *()
fread           extern      0W  ext.       165   int ()
fwrite          extern      0W  ext.       178   int ()
getw           #extern      0W  ext.       192   int ()
putw           #extern      0W  ext.       196   int ()
fseek           extern      0W  ext.       200   int ()
ftell          #extern      0W  ext.       214   long int ()
fflush         #extern      0W  ext.       220   int ()
printf          extern      0W  ext.       284   int ()
fprintf        #extern      0W  ext.       291   int ()
sprintf        #extern      0W  ext.       299   int ()
scanf           extern      0W  ext.       308   int ()
fscanf         #extern      0W  ext.       317   int ()
sscanf         #extern      0W  ext.       327   int ()
atoi           #extern      0W  ext.       342   int ()
atof           #extern      0W  ext.       347   double ()
atol           #extern      0W  ext.       352   long int ()
atou           #extern      0W  ext.       358   unsigned ()
ftoa           #extern      0W  ext.       364   char *()
itoa           #extern      0W  ext.       377   char *()
utoa           #extern      0W  ext.       383   char *()
calloc         #extern      0W  ext.       395   char *()
malloc         #extern      0W  ext.       401   char *()
alloc          #extern      0W  ext.       407   char *()
intss          #extern      0W  ext.       429   int ()
intso          #extern      0W  ext.       432   int ()
strcmp         #extern      0W  ext.       442   int ()
strncmp        #extern      0W  ext.       449   int ()
```

Figure C-12. Compilation Listing of the EXCHANGEC Program (continued)

```
 826 1          {
 827 2                  conversion ();
 828 2
 829 2                  printf ("\n\tEnter Y to continue : ");
 830 2                  scanf ("%s", &response);
 831 2                  printf ("\f");
 832 2
 833 2                  menu ();
 834 2                  get_selection ();
 835 2          }
 836 1
 837 1  printf ("\f\tExchange program ended.");
 838 1  }

-------
1234567       Listing format:

Columns 1-4:  Line Number.

Column  5:    One of the character(s):
                *  Line started out inside comment.

Column  6:    Number of nested {'s:
                0 ... 9 --  0-9  nested {'s.
Source file: exchangec
Compiled on 15-Sep-82 at 10:41:52 by AOS/VS C Rev 01.00.08.126

                              Allocation Map

External Block
--------------

Identifier     Storage   Size  Loc.Off.  Line   Type

_iobuf        #struct     32W -----        53   struct
_ptr          #member      2W     0W       54   unsigned char *
_bstart       #member      2W     2W       55   unsigned char *
_xmt          #member      2W     4W       56   int
_icount       #member      1W     6W       57   short int
_ocount       #member      1W     7W       58   short int
_bflag        #member      1W     8W       59   short int
_bufsize      #member      1W     9W       60   short int
_fd           #member      1W    10W       61   short int
_tsize        #member      1W    11W       62   short int
_look_ahead   #member      1C    12+ 0C    63   unsigned char
_temp_buf     #member      1C    12+ 1C    64   unsigned char
_pad          #member      1W    13W       65   short int
_reserved     #member     18W    14W       66   int [9]

(no name)      struct     37W -----       648   struct
rate_date      member     10C     0+ 0C   649   char [10]
rates          member     32W     5W      653   struct

(no name)      struct     32W -----       650   struct
us_foreign     member     16W     0W      651   float [8]
foreign_us     member     16W    16W      652   float [8]

(no name)     #enum        1W -----       637   enum
belgian        const.   ----- -----       637   Constant 0
w_german       const.   ----- -----       638   Constant 1
dutch          const.   ----- -----       639   Constant 2
italian        const.   ----- -----       640   Constant 3
```

Figure C-12. Compilation Listing of the EXCHANGEC Program (continued)

```
768 2          }
769 1
770 1          printf("\n");
771 1
772 1          if (us_to_foreign == TRUE)  printf ("\n \tEnter US$:  ");
773 1          else printf ("\n \tEnter %s :", currency[selection]);
774 1
775 1          scanf ("%f", &inval);
776 1          outval = inval * rate;

777 1
778 1          printf ("\n");
779 1          if (us_to_foreign == TRUE)
780 1                 printf ("\n\t$ %5.2f  US equivalent to: %5.2f  %s",
781 1                                inval, outval, currency[selection]);
782 1
783 1          else    printf ("\n\t %5.2f %s equivalent to: $ %5.2f US",
784 1                                inval, currency[selection], outval);
785 1    }
786 0    /*---------------------------------------------------------------*/
787 0    main ()
788 0
789 0    /* The main procedure drives the interactive session through the
790*0          rate file processing and conversion selection phases.   */
791 0
792 0    {
793 1          printf ("\f\t\tExchange Program\n\n");
794 1
795 1          printf ("\tDo you want to create a new data file?\n");
796 1          printf ("\tEnter Y or N:  ");
797 1          scanf ("%s", &response);
798 1
799 1          if ((response == 'Y') || (response == 'y')) builder ();
800 1          else
801 1          {
802 2
803 2                 rate_file = fopen ("rate_file", "r");
804 2                 fread (&daily_rates, sizeof(daily_rates), 1, rate_file);
805 2
806 2                 printf ("\n\tThe file contains exchange rates for:  ");
807 2                 printf ("%s\n", daily_rates.rate_date);
808 2
809 2                 printf ("\tDo you want to change it? Enter Y or N:  ");
810 2                 scanf ("%s", &response);
811 2
812 2                 if ((response == 'Y') || (response == 'y'))
813 2                 {
814 3                        fclose (rate_file);
815 3                        builder ();
816 3                 }
817 2          }
818 1
819 1          printf ("\f");
820 1
821 1          menu ();
822 1          get_selection ();
823 1
824 1          while (selection != no_choice)
825 1
```

Figure C-12. Compilation Listing of the EXCHANGEC Program (continued)

```
715 1          printf ("\t8   Japanese yen\n");
716 1          printf ("\t(Type 0 to end the program.)\n\n");
717 1
718 1          printf ("Enter the currency code: ");
719 1    }
720 0    /*-------------------------------------------------------------*/

721 0    get_selection ()
722 0
723 0    /* Gets the users currency code input and sets parameter selection
724*0          to the corresponding country.                          */
725 0    {
726 1
727 1    int currency_code;
728 1
729 1          scanf ("%d", &currency_code);
730 1
731 1          switch  (currency_code)
732 1          {
733 2                  case 1 : selection = belgian; break;
734 2                  case 2 : selection = w_german; break;
735 2                  case 3 : selection = dutch; break;
736 2                  case 4 : selection = italian; break;
737 2                  case 5 : selection = swiss; break;
738 2                  case 6 : selection = canadian; break;
739 2                  case 7 : selection = saudi; break;
740 2                  case 8 : selection = japanese; break;
741 2
742 2                  default : selection = no_choice; break;
743 2          }
744 1    }
745 0    /*-------------------------------------------------------------*/
746 0    conversion ()
747 0
748 0    /* Performs the actual conversion based on the selection and
749*0          outputs the result.                                    */
750 0
751 0    {
752 1    float inval, outval, rate;
753 1    char response;
754 1
755 1          printf ("\nDo you want to convert US$ into Foreign currency?");
756 1          printf ("\tEnter Y or N: ");
757 1          scanf ("%s", &response);
758 1
759 1          if ((response == 'Y') || (response == 'y'))
760 1          {
761 2                  us_to_foreign = TRUE;
762 2                  rate = daily_rates.rates.us_foreign[selection];
763 2          }
764 1          else
765 1          {
766 2                  us_to_foreign = FALSE;
767 2                  rate = daily_rates.rates.foreign_us[selection];
```

Figure C-12. Compilation Listing of the EXCHANGEC Program (continued)

Index

Within this index, the letter "f" (or "ff") following a page entry means "and the following page" (or "pages").