November 1979

This document contains instructions for installing the VAX-11 PASCAL compiler on the VAX/VMS operating system. It also contains information not included elsewhere in the documentation set, typically concerning software and/or documentation errors that were discovered or changes that were made late in the development cycle. This document should be read before the VAX-11 PASCAL compiler is installed or used.

VAX-11 PASCAL
Installation Guide/Release Notes

Order No. AA-J181A-TE

SUPERSESSION/UPDATE INFORMATION:	This is a new document for this release.
OPERATING SYSTEM AND VERSION:	VAX/VMS V1.6
SOFTWARE VERSION:	VAX-11 PASCAL V1.0-1

To order additional copies of this document, contact the Software Distribution Center, Digital Equipment Corporation, Maynard, Massachusetts 01754

digital equipment corporation · maynard, massachusetts

First Printing, November 1979

The information in this document is subject to change without notice and should not be construed as a commitment by Digital Equipment Corporation. Digital Equipment Corporation assumes no responsibility for any errors that may appear in this document.

The software described in this document is furnished under a license and may only be used or copied in accordance with the terms of such license.

No responsibility is assumed for the use or reliability of software on equipment that is not supplied by DIGITAL or its affiliated companies.

The postage prepaid READER'S COMMENTS form on the last page of this document requests the user's critical evaluation to assist us in preparing future documentation.

The following are trademarks of Digital Equipment Corporation:

DIGITAL	DECsystem-10	MASSBUS
DEC	DECtape	OMNIBUS
PDP	DIBOL	OS/8
DECUS	EDUSYSTEM	PHA
UNIBUS	FLIP CHIP	RSTS
COMPUTER LABS	FOCAL	RSX
COMTEX	INDAC	TYPESET-8
DDT	LAB-8	TYPESET-11
DECCOMM	DECSYSTEM-20	TMS-11
ASSIST-11	RTS-8	ITPS-10
VAX	VMS	SBI
DECnet	IAS	PDT
DATATRIEVE	TRAX	

CONTENTS

PREFACE

MANUAL OBJECTIVES

The VAX-11 PASCAL Installation Guide/Release Notes describe:

- The procedures necessary to install the VAX-11 PASCAL compiler on the VAX/VMS operating system
- The use of the VAX-11 Symbolic Debugger in debugging PASCAL programs
- Changes that were made and problems that were discovered too late in the development cycle to be discussed in the rest of the VAX-11 PASCAL manual set

INTENDED AUDIENCE

This manual is intended for use by VAX/VMS system managers and PASCAL programmers.

STRUCTURE OF THIS DOCUMENT

This manual is organized as follows:

- Section 1.0 describes the installation procedures.
- Section 2.0 briefly describes how to use the VAX-11 Symbolic Debugger to debug a PASCAL program.
- Section 3.0 lists and describes problems and restrictions in this release of VAX-11 PASCAL.

1.0 INSTALLATION PROCEDURE

This chapter describes the procedure for installing the VAX-11 PASCAL compiler on the VAX/VMS operating system. The procedure is automated and requires that the system manager or other individual performing the procedure mount floppy diskettes when prompted and respond to queries issued during the installation procedure.

This procedure can be performed by any system user who has access to the system manager's account or to an account that has the same privileges as the system manager's account.

1.1 Contents of the Distribution Kit

The VAX-11 PASCAL installation kit consists of three FILES-11 ODS 2 floppy diskettes.

Each diskette is labeled with both a serial number corresponding to the VAX-11 PASCAL compiler's product number and a unique label that differentiates that diskette from every other diskette in the distribution kit.

The diskettes in the VAX-11 PASCAL distribution kit are:

Diskette Label	Contents
VAX-11 PASCAL V1.0 RX01 (numbered 1/3)	This diskette contains the installation command procedure, the Run-Time Library object modules, and some of the compiler object modules.
VAX-11 PASCAL V1.0 RX01 (numbered 2/3)	This diskette contains compiler object modules.
VAX-11 PASCAL V1.0 RX01 (numbered 3/3)	This diskette contains the remainder of the compiler object modules.

These diskettes must be mounted in the order listed above during the installation procedure.

1.2 Preparing to Install the VAX-11 PASCAL Compiler

To prepare for installation of the VAX-11 PASCAL compiler, perform the following steps:

1. At the system console terminal, log into the system using the system manager's account or other privileged account.

2. Be sure that the logical name SYS$DISK is assigned to the disk that contains the current version of VAX/VMS, as distributed, and contains all updates. This disk contains the command procedure that initiates the installation procedure. Note that SYS$DISK need not be (and if possible should not be) the same as SYS$SYSTEM.

3. Enter the following commands in the order shown:

   - Change the current UIC as follows:

     ```
     SET UIC [1,4]
     ```

   - Set the default directory as follows:

     ```
     SET DEFAULT [SYSUPD]
     ```

4. At the console terminal, type:

   ```
   @VMSUPDATE
   ```

The execution of the command procedure VMSUPDATE.COM initiates the installation procedure. The procedure VMSUPDATE.COM displays the following:

```
This command procedure performs VAX/VMS software updates and
unbundled software installations...

During this sequence, the standard console floppy will not be
present in the console floppy drive.

Therefore, the system is vulnerable to power failure or other
types of fatal crash.  If a system crash should occur during this
period the update sequence can be restarted at the beginning of
the first incomplete update.

Dismount the current console floppy.

Please place the first floppy of the kit in the console drive.
```

You are now ready to begin the installation procedure for the VAX-11 PASCAL compiler.

1.3 The Installation Procedure

If there is a floppy disk mounted in the console floppy drive, remove it. (Note that if there is no floppy diskette mounted, the system displays an error message. You can ignore this message and continue.)

When you remove the floppy diskette, note the direction it is facing, that is, the direction of the label on the diskette. When you insert diskettes from the distribution kit, you must insert them in the drive so that they face the same direction. (The label is on the front side of the diskette.)

Place the first diskette, labeled 1/3, in the console drive. You will receive the message:

```
ARE YOU READY TO CONTINUE?
```

If you type Y in response to this message, the installation procedure continues.

If you type N, the request to place the first diskette in the console floppy drive and the query "ARE YOU READY TO CONTINUE?" are repeated. When you have responded "Y" to the query, the command procedure on the first floppy diskette assumes control and the files needed to build the compiler are copied from the first diskette to the system disk. When this is done, you receive the next messages:

```
PLEASE PUT THE NEXT PASCAL KIT FLOPPY DISK IN THE DRIVE.

IF NO MORE FLOPPIES TO BE COPIED, PLEASE TYPE <CR>.

ARE YOU READY TO CONTINUE?
```

Replace the diskette labeled 1/3 with the diskette labeled 2/3, and type Y to indicate that the installation can continue.

When this stage is complete, you receive the messages:

```
PLEASE PUT THE NEXT PASCAL KIT FLOPPY DISK IN THE DRIVE.

IF NO MORE FLOPPIES TO BE COPIED, PLEASE TYPE <CR>.

ARE YOU READY TO CONTINUE?
```

Replace the diskette labeled 2/3 with the diskette labeled 3/3, and type Y to indicate that the installation can continue.

When this stage is complete, you receive the messages:

```
PLEASE PUT THE NEXT PASCAL KIT FLOPPY DISK IN THE DRIVE.

IF NO MORE FLOPPIES TO BE COPIED, PLEASE TYPE <CR>.

ARE YOU READY TO CONTINUE?
```

Since you have completed processing all the floppies, type a carriage return instead of a Y this time. You may now take the diskette labeled 3/3 out of the drive at any time.

During the following stage, you will get a warning message from the linker:

```
%LINK-W-MULTFR, Module "PASCAL" multiply defines transfer address
```

This a normal message and should be ignored. Then the following message will appear:

```
The default linelimit of the compiler is 2**31.

DO YOU WANT TO CHANGE IT?
```

Respond with N if you do not wish to change the linelimit. If you wish to impose a smaller default linelimit on files generated by the PASCAL I/O system, then respond with Y, in which case the next message you will see is:

```
What linelimit do you want to have?
```

Respond with the positive decimal integer value you wish to have as the default linelimit. (For example, typing 100 sets the linelimit to 100 lines.) When this stage is complete, you will receive the message:

```
DO YOU WANT TO PURGE OLDER VERSIONS OF THE PASCAL COMPILER?
```

Type Y if you wish to delete older versions; type N if you wish to retain older versions. If you have not previously installed PASCAL on the system, type N.

You may now receive some of the following messages:

```
MODULE    "PAS$IO_BASIC" REPLACED

MODULE    "PAS$IO_INPUT" REPLACED

MODULE    "PAS$IO_OUTPUT" REPLACED

MODULE    "PAS$RT_UTIL"  REPLACED

MODULE    "PAS$RT_HEAP"  REPLACED

MODULE    "PAS$RT_CHK"   REPLACED

MODULE    "PAS$RT_FUNC"  REPLACED

MODULE    "PAS$LINELIM"  REPLACED

PASCAL ALREADY APPEARS IN [SYSHLP]HELP.HLP - NOT INSERTED.

%SYSTEM-F-DEVNOTMOUNT, device not mounted
```

These messages should be ignored.

At this time, an installation verification test will be run. This test compiles, links, and executes a sample PASCAL program. The only output to the terminal will be:

```
PASCAL  INSTALLATION  SUCCESSFUL
```

1.4 Completion of the Procedure

When the installation procedure completes, control returns to the system command procedure VMSUPDATE. This procedure issues the following message:

```
ARE THERE MORE KITS TO PROCESS [Y/N]
```

Respond with a Y to this message only if you have additional software kits to install. Then you will receive a request to mount the first diskette from another distribution kit. The installation of that kit will continue.

If you type N, you will receive the message:

```
Please place the system console floppy in the console drive.
```

You should immediately restore the standard console diskette (which was removed at the beginning of the update procedure) to the console drive.

After you have replaced the system console diskette, you receive the message:

```
ARE YOU READY TO CONTINUE?
```

Type Y. The console floppy diskette is automatically mounted. You receive the message:

```
Requested update sequence is complete.
```

IMPORTANT NOTE: Upon completion of the installation process, the system manager must perform the following task:

The following two system logical names must be established:

```
ASSIGN/SYSTEM SYS$OUTPUT PAS$OUTPUT
ASSIGN/SYSTEM SYS$INPUT PAS$INPUT
```

These two assignments must also be placed in [SYSMGR]SYSTARTUP.COM. They establish the default INPUT and OUTPUT files for PASCAL I/O. The VAX/VMS software release distribution kit version of SYSTARTUP will eventually include these commands, but until then, they must be treated as site-specific start-up commands.

The VAX-11 PASCAL compiler is now installed and can be invoked by the PASCAL command:

```
$PASCAL
```

2.0 INTERFACE TO THE VAX-11 SYMBOLIC DEBUGGER

The VAX-11 Symbolic Debugger does not currently contain any special features to support PASCAL. This means that many of the debugger commands that supply high-level language support are not available when debugging PASCAL object modules. However, you can still debug your PASCAL program in the same manner as you would debug a MACRO assembly language program. Before attempting to use the debugger on a PASCAL program, you should be familiar with VAX instruction architecture.

This section assumes that you are familiar with using the VAX-11 Symbolic Debugger. If you are familiar with MACRO assembly language programming, but have not used the debugger, you should read the VAX-11 Symbolic Debugger Reference Manual before attempting to understand this section.

2.1 Choosing a Language

The SET LANGUAGE command allows you to choose the debugger command formats that are most convenient. The current choices are BLISS, MACRO, and FORTRAN.

When you invoke the debugger, the language is initially set to BLISS. However, it is advised that only those individuals with a detailed knowledge of BLISS and the VAX-11 Symbolic Debugger should leave the language set in this way for debugging PASCAL programs.

If you choose FORTRAN, the debugger will understand expressions and assignments containing simple variables of the INTEGER, REAL, SINGLE and DOUBLE types. However, in the FORTRAN language mode, the debugger

does not allow you to do address arithmetic. As a result, it is difficult to examine structured variables and object code with the debugger in FORTRAN mode. FORTRAN mode is useful only if:

1. Your program contains many simple variables (INTEGER, REAL, SINGLE and DOUBLE).

2. You do not need to examine structured variables (arrays and records).

3. You need to set break points only at the entries to your functions and procedures.

MACRO mode allows you to examine structured variables in the same way that you would examine structured data in an assembly program. You can also easily examine the object code instructions. Therefore, you should probably choose MACRO as the language mode to use with the debugger.

For information about using the debugger with MACRO assembly programs, refer to the VAX-11 Symbolic Debugger Reference Manual. Using the debugger with FORTRAN programs is described in the VAX-11 FORTRAN IV-PLUS User's Guide. Information on using the debugger with BLISS programs is included in the VAX-11 BLISS-32 User's Guide.

2.2 Describing Object Program Locations

PASCAL produces its machine code listing before it performs branch optimization. Therefore, the hexadecimal locations in the machine code listing do not correspond to the locations in the final object code. As a result, you cannot safely set a break point based on the locations specified in the listing. However, you can easily set a breakpoint at the entry of a procedure or function. By using the EXAMINE/INSTRUCTION command, you can look at the actual object code and use this information to set break points inside routines.

PASCAL produces a line number table for use by the traceback program and the debugger. When you examine the stack of routine calls (with the SHOW CALLS command), you can see the line numbers where the procedure calls have occurred. However, the debugger currently only allows line number tables produced by FORTRAN to be used as input. If your PASCAL program does not contain any procedures, the object code is similar enough to that produced by FORTRAN so that you can use %LINE parameters with debugger commands in FORTRAN mode. But for most PASCAL programs, the %LINE parameter will not be available for use with the current debugger. Furthermore, you cannot use the %LABEL parameter because PASCAL labels do not look like FORTRAN statement numbers.

2.3 Examining and Modifying Variables

PASCAL generates complete symbol table entries only for the INTEGER, REAL, SINGLE, and DOUBLE types. The EXAMINE, EVALUATE, and DEPOSIT commands work easily with variables of these types. When using variables of other types, you must be familiar with the internal format that PASCAL uses to store these types. PASCAL generates debugger symbol table entries for all variables, but the symbol table entry for a structured variable currently gives only the variable's location. As a result, the debugger always prints storage allocated to a variable as an unsigned numeric value using the debugger's prevailing number base.

Because PASCAL is a block structured language, you may find it necessary to provide extra qualification to your program's symbol names. For example, if variable NAME is declared in procedure PROC in program PROG, then variable NAME should appear as PROG\PROC\NAME on input and output to the debugger. This qualification allows the debugger to differentiate this use of the symbol NAME from other uses of this symbol elsewhere in your program.

PASCAL does not nest its procedure symbol tables. Thus, even if you declare a procedure INNER local to a procedure OUTER, the produced symbol table describes both INNER and OUTER as unnested procedures declared in main program block. As a result, you never need to qualify a variable name with more than two context names (the program or module name and the procedure name).

Variables declared in the outermost block look like they were declared in a procedure with the same name as the program. Therefore, a variable with the name NAME1 declared in program PROG appears as PROG\PROG\NAME1 when used with the debugger. The scoping rules of the debugger usually make this qualification unnecessary. Refer to the VAX-11 Symbolic Debugger Reference Manual for a description of how these scoping rules work. However, since PASCAL does not nest its debug symbol tables, the debugger sometimes confuses two internal procedures with identical names.

The current version of the debugger does not reconstruct the proper context when accessing a variable. You must issue a SHOW CALLS command to determine which PASCAL routine is uppermost in the stack of called procedures. You can then examine variables only in this uppermost procedure and in the main program block. If your program references variables declared in outer blocks (other than at program level), the debugger accesses the wrong location.

PASCAL statically allocates the variables in the outer level of the main program so that you can examine these variables at any time.

2.4 Debugger Commands

Most debugger commands work as described in the VAX-11 Symbolic Debugger Reference Manual. The CALL and STEP commands are the two exceptions.

- CALL -- Although FORTRAN always passes parameters by-reference, and the debugger considers MACRO to use call by-value, PASCAL allows a mixture of call by-reference and call by-value (as described in Chapter 6 of the VAX-11 PASCAL User's Guide). In addition, PASCAL uses its own conventions when passing procedures and functions as parameters. Therefore, you must carefully consider the effect of the debugger language mode when using the CALL command to call PASCAL routines. Use the CALL command only if you are familiar with the run-time representation of parameters for PASCAL functions and procedures.

- STEP -- The STEP command works in INSTRUCTION mode, but not in LINE mode. If you select the FORTRAN language mode, you must also issue the SET STEP INSTRUCTION command after the SET LANGUAGE command.

3.0 RELEASE NOTES

The following is a summary of all the known restrictions that affect the operation of the VAX-11 PASCAL compiler.

3.1 LIST Option

The /LIST command line qualifier does not work as expected in connection with the (*L+*) source code qualifier. This is due to an error in the way the VAX/VMS command line processing works.

For now, you must explicitly specify /LIST in the command line if you want a source code listing, as follows:

```
$PASCAL/LIST=file-spec file-spec

     or

$PASCAL file-spec/LIST=file-spec
```

If you specify neither /LIST nor /NOLIST, the compiler behaves as though you specified /NOLIST. Therefore, attempts to set the listing option in the following format within a PASCAL source program will have no effect:

```
(*$L+*)
```

A future release of the VAX/VMS operating system will correct this problem.

3.2 Rewrite of a File Using DECnet

There is a restriction concerning the rewriting of files that exist on a remote (not local) node of a DECnet network.

A new file can be created and written on a remote node through VAX-11 PASCAL. An existing file can be read but not rewritten unless it is empty.

The following code sequence will fail if MASTER.DAT is not empty:

```
OPEN (FILEX, 'BANGOR"PLUGH XYZZY"::MASTER.DAT', OLD);
REWRITE (FILEX);
WRITE (FILEX, VAR1, VAR2);
```

When you call REWRITE in VAX-11 PASCAL, the file buffer variable is set to the beginning of the file and all existing records are overwritten and therefore lost. You could also open the file as "NEW", in which case the previous version of the file is not destroyed, but a new version is created. The result is the same, except that the file's version number has been incremented.

This stems from a restriction in DECnet-VAX that will be fixed in a future release.

(For a discussion of "NEW" and "OLD" files, see Section 7.5 of the VAX-11 PASCAL Language Reference Manual concerning the OPEN statement. An example of writing data to a remote node is shown in Section 5.6 of the VAX-11 PASCAL User's Guide.)

3.3 Reinstallation of VAX-11 PASCAL

The VAX-11 PASCAL compiler must be reinstalled after a new VAX/VMS operating system is generated. A new system generation will produce a new copy of the library STARLET.OLB which will not contain the VAX-11 PASCAL modules. The PASCAL specific modules can be written into the library only by the reinstallation of VAX-11 PASCAL.

3.4 Reading REAL or DOUBLE Precision Numbers

When you read REAL or DOUBLE values using the procedure READ or READLN, the exponent designator must be a capital letter, "E" or "D."

The following are illegal values to be input:

```
1e1
3.1415e0
10.1111d-3
```

This problem will be fixed in a future release of the VAX/VMS operating system.

3.5 Using Descriptors in VAX-11 PASCAL

VAX-11 PASCAL uses the uniform descriptor mechanism in transmitting %DESCR arguments to routines written in languages other than PASCAL. (For a complete discussion of argument descriptors, see Appendix C of the VAX-11 Architecture Handbook.)

However, there is a restriction on the use of descriptors in VAX-11 PASCAL. The length and type fields of descriptors of some arrays may not be filled in. This is due to a deficiency in the size of these fields. It is not recommended that these fields be referenced directly.

3.6 Number of Significant Digits

The number of digits printed for a single-precision real number in VAX-11 PASCAL is 9. Typically, however, only 7 of these digits are significant due to the functioning of the VAX/VMS hardware.

Consider the following program:

```
PROGRAM TESTFP(OUTPUT);
VAR  X : REAL;
BEGIN
   X := 3.14;
   WRITELN ('The value of X is ',X);
   WRITELN ('nine digits ',X:16:9);
   WRITELN ('eight digits ',X:16:8);
   WRITELN ('seven digits ',X:16:7);
   WRITELN ('six digits ',X:16:6);
   WRITELN ('five digits ',X:16:5)
END.
```

The output from this program would be:

```
The value of X is 3.140000105E+00
nine digits     3.140000105
eight digits     3.14000010
seven digits      3.1400001
six digits         3.140000
five digits         3.14000
```

When doing precise arithmetic calculations, you should keep this degree of error in mind.

3.7 Assignment of PAS$INPUT and PAS$OUTPUT

The assignment of PAS$INPUT and PAS$OUTPUT to SYS$INPUT and SYS$OUTPUT is specified in the installation section of this manual. However, on versions of VAX/VMS later than 1.6, these assignments may already be present in the system startup command procedure file.

The reassignment of PAS$INPUT and PAS$OUTPUT at installation time will then be unnecessary. If performed, reassignment will have no effect and the compiler will continue to function correctly.

3.8 Run-Time Errors Not Signaled

Errors at run time from the VAX-11 PASCAL compiler cannot be fielded by condition handlers (see Chapter 7 of the VAX-11 PASCAL User's Guide).

A future release of the VAX-11 PASCAL compiler will signal these errors so they may be fielded by condition handlers.

3.9 Output Characteristics Differ

The predeclared files INPUT and OUTPUT (SYS$INPUT and SYS$OUTPUT) are opened with default attributes. However, OUTPUT is not opened in exactly the same manner as is done in other VAX/VMS languages. For more information on the predeclared file OUTPUT, see Chapter 7 of the VAX-11 PASCAL Language Reference Manual.

3.10 No Checking for Dereferencing a NIL Pointer

When the CHECK qualifier is set at compile time, VAX-11 PASCAL does not generate code to check for dereferencing a NIL pointer. Currently, an access violation occurs when a program dereferences a NIL pointer.

To avoid the problem at this time, you can check for dereferencing a NIL pointer in the source program by writing a routine to do so.

3.11 Overflow of REALs and INTEGERs

There is a restriction in the diagnosis of overflow conditions for REAL and INTEGER numbers.

For example, consider the following program:

```
PROGRAM TOOLARGE (INPUT, OUTPUT);
VAR BIG : REAL;
BEGIN
   READ (BIG);
   WRITE (BIG)
END.
```

When this program is run and the following number is entered:

```
1.8E39
```

The following result is printed:

```
0.000000000E+00
```

This problem will be fixed in a future release of the VAX-11 PASCAL compiler.

3.12 TRUE and FALSE Comparisons

VAX-11 PASCAL uses more than just the low-order bit when comparing Booleans. This will cause incompatibilities with Boolean values from FORTRAN IV-PLUS, system services, and other VAX/VMS languages that do not use 0 for FALSE and 1 for TRUE.

3.13 RUN-TIME ERROR TERMINATION MESSAGE

One of the following messages may appear when your program encounters an error at run time:

```
%PAS-F-NOMSG, Subsystem 33 -- Message number 33700

%NONAME-F-NOMSG, Subsystem 33 -- Message number 33668
```

The message appears along with other run-time error messages that provide more information about the error. These two messages will change in a future release.

VAX-11 PASCAL
Installation
Guide/Release Notes
AA-J181A-TE

READER'S COMMENTS

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

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

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

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

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

Please indicate the type of reader that you most nearly represent.

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

Name_____ Date_____

Organization_____

Street_____

City_____ State_____ Zip Code_____
or
Country

Please cut along this line.

— — Do Not Tear - Fold Here and Tape —

digital

No Postage
Necessary
if Mailed in the
United States

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

POSTAGE WILL BE PAID BY ADDRESSEE

RT/C SOFTWARE PUBLICATIONS TW/A14
DIGITAL EQUIPMENT CORPORATION
1925 ANDOVER STREET
TEWKSBURY, MASSACHUSETTS 01876

— Do Not Tear - Fold Here —

Cut Along Dotted Line