PASCAL on the DEC10

by

H. H. Nagel[*] and W. F. Burger

TR-22B February 1976

* Institut fuer Informatik der Universitaet Hamburg,
2 Hamburg 13, Schlueterstr. 70

Technical Report No. 22B
Department of Computer Sciences
The University of Texas at Austin
Austin, Texas 78712

Preface

The PASCAL compilers for the DEC-10 were developed at the University of Hamburg, Germany, based on an early version of the transportable PASCAL-P compiler [6,5]. Work continues on improving this compiler with respect to efficiency, standardization and the addition of still missing or desirable language features [7].

This report is intended to familiarize the reader with the PASCAL system on the DEC-10. It is assumed that the reader has a basic knowledge of PASCAL.

Table of Contents

1. Usage of the PASCAL Compilers

Two PASCAL Compilers are available:

PASCAL	generates absolute code
PASREL	generates relocatable code

The PASCAL compiler is recommended for programs which contain neither external procedures, nor standard procedures which are marked with R in Section 5.1 nor which contain the DEBUG-switch (Section 4).

1.1 How to Use the PASCAL Compiler

The compiler is loaded from SYS: by the command

```
     R PASCAL
```

An asterisk is typed and the compiler expects the name of the input file. The file description has the following form:

```
     DEVICE: FILNAM.EXT [PROJ.,PROGR.]
     -------       ------------------
```

The underlined parts may be omitted. They are by default:

DEVICE:	DSK:
.EXT	.PAS
[PROJ.,PROGR.]	<own project/programmer number>

The name FILNAM is used as an example throughout this section.

A listing of the program is generated on DSK: FILNAM.LST. Lines containing errors are also sent to the terminal with an appropriate error message. A successful compilation is indicated by the message

```
     NO ERROR DETECTED
```

Otherwise the message

```
     ERROR DETECTED
```

is given. This is followed by the time used for compilation. The last message from the compiler is EXIT.

The translated program is found on the files FILNAM.LOW and FILNAM.SHR after a successful compilation.

The translated program is executed by the command

```
     RUN FILNAM ##
```

where ## stands for the core requirements which must be estimated from the size of the program and the space necessary for stack and heap.

1.2 How to Use the PASREL Compiler

The compiler is loaded from SYS: by the command

```
R PASREL
```

An asterisk is typed and the compiler expects the name of the input file. The form is the same as described above. After a successful compilation the following size information is given:

```
HIGHSEG : U K
LOWSEG  : V K
```

U is the core requirement for the high segment (code) in K, V is the core requirement for the low segment (data) in K.

The compiler generates relocatable code on FILNAM.REL. The program is loaded by

```
LOAD FILNAM.REL
```

An executable program is obtained by

```
SAVE FILNAM W
```

W is the total core requirement for the program. It should be at least U + V + 4. A sharable high segment is obtained by

```
SSAVE FILNAM W
```

SSAVE may not be used if the program contains the DEBUG-switch.

The program is executed by

```
RUN FILNAM
```

Should more core be required than was specified when the program was saved then the command

```
RUN FILNAM ##
```

may be given where ## stands for the larger core requirement.

1.3 How to write a Program (Lexical Issues)

PASCAL and PASREL accept only a subset of the ASCII characters. The character set itself consists of the 64 characters with octal values from 40 to 137. This means that all input must be in upper case. The tab character is expanded to the appropriate number of blanks. Lines are ended by carriage-return/linefeed. A form feed character results in an empty line. All other characters appearing in the input text are ignored.

Next we shall describe language elements which use special characters.

Comments are enclosed in (* and *) or alternatively in % and \, e.g.:

```
        (*THIS IS A COMMENT*)
        % THIS IS A COMMENT \
```

Directives to the compiler are comments which start with a $ sign and have the form

```
        (*$T+*)
```

The directives are described in Section 1.4.

Identifiers may be written using the underline character to improve readability, e.g.:

```
        NEW_NAME
```

Strings are character sequences enclosed in single quotes, e.g.:

```
        'THIS IS A STRING'
```

If a quote is to appear in the string it must be repeated, e.g.:

```
        'ISN''T PASCAL FUN?'
```

An integer is represented in octal form if it consists of octal digits followed by B. An integer is represented in hexadecimal form if it consists of a " followed by hexadecimal digits. The following representations have the same value:

```
        63      77B     "3F
```

Several PASCAL operators have an alternate representation. The alternate form is provided for compatibility with older versions of PASCAL. The form of the operator shown in the left column should be used in new programs. (Forthcoming compiler versions will discontinue the use of the alternate forms).

operator	alternate form	explanation
>=	"	greater or equal
<=	@	less or equal
AND	&	logical and, set intersection
OR	!	logical or, set union
NOT	?	
#		not equal

1.4 Compiler Directives

Compiler directives are special comments selecting compiler options. An option is turned on if it is followed by + and it is turned off if it is followed by -. The following options are accepted by both compilers:

C	generate code to perform runtime checks for indices and assignments to scalar and subrange variables.
L	list the generated code in symbolic form
T	TTY is opened at the beginning of program execution (see Section 2.6)

By default the compiler assumes these directives:

```
(*$C+,L-,T+*)
```

The PASREL compiler accepts additional options:

D	generate information for the DEBUG package (see Section 4)
M	a main program part is present (see Section 5.2)

The default directives are:

```
(*$D-,M+*)
```

2. Input/Output

Input/Output is done with the standard procedures READ, READLN, WRITE, WRITELN as described in the Revised Report on PASCAL (also referred to as PASCAL2) [2,9].

2.1 Standard Files

In addition to the standard files INPUT and OUTPUT the standard file TTY is available in DEC10 PASCAL. This file is used to communicate with the terminal. The standard files can be directly used to read or write without having to use the standard procedures RESET or REWRITE.

2.2 File Declaration

Files may only be declared in the main program. The first 6 characters of the identifier are used for the file name, the next 3 characters are used for the file extension. Blanks are used if there are not enough characters.

2.3 RESET and REWRITE

A file must be "opened" with the standard procedure RESET when it is to be used for reading. It must be "opened" with the standard procedure REWRITE when it is to be used for writing.

RESET and REWRITE may have up to 5 parameters in DEC10 PASCAL:

```
     RESET (<file identifier>,<name and extension>,
            <protection>,<project programmer number>,
            <device mnemonic>)
```

Only the first parameter is required. The other parameters are used as follows:

<name and extension>
: This parameter must be of type PACKED ARRAY [1..9] OF CHAR. The first 6 characters are used as file name, the next 3 characters as file extension. The parameter is used to overwrite the default file name which is derived from the file declaration.

<protection>
: This parameter must be of type INTEGER. It is not necessary for input. If the fourth parameter is used, the value of this parameter may be 0.

in octal representation this parameter must consist of 12 digits, where the first 3 digits are the protection code and the rest are zeroes.

<project programmer number>
This parameter must be of type INTEGER. For the PPN 1023,7777 it would have the following form (in octal representation): 001023007777B
The job's PPN is used if the value is 0, or the parameter is missing.

<device mnemonic>
This parameter must be of type PACKED ARRAY [1..6] OF CHAR. It defines the device where the file resides. If this parameter is missing then the system fills in 'DSK '.

In the following example REWRITE is used to give the file OUTPUT the actual file name TEST.LST. The file is created with protection <057>.

Example:

```
REWRITE(OUTPUT,'TEST  LST',057000000000B)
```

2.4 Formatted Output

Parameters of the standard procedure WRITE (and WRITELN) may be followed by a "format specification". A parameter with format has one of the following forms:

```
X : E1
X : E1 : E2
X : E1 : O
X : E1 : H
```

E1 is called the field width. It must be an expression of type INTEGER yielding a non-negative value. If no format is given then the default value for E1 is for type

INTEGER	12
BOOLEAN	6
CHAR	1
REAL	16

For strings the default value is the length of the string. Blanks precede the value to be printed if the field width is larger than necessary to print the value. Depending on the type involved, the following is printed if the field width is smaller than necessary to print the value:

INTEGER	asterisks
REAL	asterisks
BOOLEAN	T or F preceded by the appropriate number of blanks

For strings the leftmost characters fitting the field width are printed. No characters are printed if the field width is 0. The minimal field width for values of type REAL is 8 if they are negative, 7 otherwise.

Example:
```
WRITELN('STR':4, 'STR', 'STR':2, -12.0:10);
WRITELN(15:9, TRUE, FALSE:4, 'X':3);
```

The following character sequence will be printed (colons represent blanks):

```
:STRSTRST-1.200E+01
:::::::15::TRUE:::F::X
```

A value of type REAL can be printed as a fixed point number if the format with expression E2 is used. E2 must be of type INTEGER and yield a non-negative value. It specifies the number of digits following the decimal point. There must be enough room for the fractional part, otherwise asterisks will be printed.

Example:
```
WRITE(1.23:5:2, 1.23:4:1, 1.23:6:0, 1.23:4:3);
```

The following character sequence will be printed (colons represent blanks):

```
:1.23:1.2::::1.****
```

A value of type INTEGER can be printed in octal representation if the format with letter O is used. The octal representation consists of 12 digits. If the field width is smaller than 12, the rightmost digits are used to fill the field width. If the field width is larger than 12, the appropriate number of blanks precedes the digits.

Example:
```
WRITE(12345B:2:O, 12345B:6:O, 12345B:15:O);
```

The following character sequence will be printed (colons represent blanks):

```
45012345:::000000012345
```

A value of type INTEGER can also be printed in hexadecimal representation if the format with letter H is used. The hexadecimal representation consists of 9 digits. Using the format with letter H, the following character sequence will be printed for the example above (colons indicate blanks):

```
E50014E5::::::0000014E5
```

2.5 Input/Output to TTY

The first parameter of the standard I/O procedures must be TTY if they are to be applied to the standard file TTY. The standard procedure BREAK is provided in order to force the output to the terminal even if the internal buffer is not yet full. BREAK itself does not insert a carriage-return/linefeed. This allows the user to type in on the same line where the output appeared. The line will be ended by a carriage-return/linefeed only if WRITELN(TTY) was used before the call to BREAK.

2.6 Properties of the Standard File TTY

The standard file TTY is opened at the beginning of the program. An asterisk is typed on the terminal to indicate that input from the terminal is expected (to assign a value to the buffer variable TTY^). If the program does not need any input from the terminal, a carriage-return should be typed.

The opening of the file TTY at the very start of the program is suppressed when the program contains at the beginning the TTY-switch (*$T-*). Writing to the terminal then, however, only can be initiated after the call

```
      REWRITE(TTYOUTPUT)
```

when reading from the terminal is desired it must be initiated by

```
      RESET(TTY)
```

Warning: Assignment to TTY^ and the call PUT(TTY) are prohibited. (The desired effect can be achieved, however, by the assignment to TTYOUTPUT^ and the call PUT(TTYOUTPUT)). TTY may only be a parameter to the standard I/O routines. It must not be used as an actual parameter for parameters of type TEXT or FILE OF CHAR.

3. Extensions to PASCAL

3.1 INITPROCEDURE

Variables of type scalar, subrange, pointer, array or record declared in the main program may be initialized by an INITPROCEDURE. The body of an INITPROCEDURE contains only assignment statements. Indices as well as the assigned values must be constants. Assignment to components of packed structures is possible if the components occupy a full word.

The syntax of an INITPROCEDURE is as follows (the parts enclosed in [and] may appear 0 or more times):

```
     <initprocedure> ::= INITPROCEDURE ;
                         BEGIN  <assignments>  END ;

     <init part> ::= <initprocedure> [ <initprocedure> ]
```

The <init part> must follow the variable declaration part and precede the procedure declaration part of the main program.

3.2 Extended CASE Statement

The CASE statement may be extended with the case OTHERS which then appears as the last case in the CASE statement. This case will be executed if the expression of the CASE statement does not evaluate to one of the case labels.

In the following example it is assumed that the variable X is of type CHAR:

```
     CASE X OF
           'A' :  WRITELN('VALUE IS A');
           'B' :  WRITELN('VALUE IS B');
        OTHERS :  WRITELN('VALUE IS NEITHER A NOR B')
     END  (*CASE STATEMENT*)
```

3.3 LOOP Statement

The LOOP statement is an additional control statement which combines the advantages of the WHILE and the REPEAT statement.

The LOOP statement has the following syntax:

```
    <loop statement> ::= LOOP
                          <statement> [; <statement>]
                         EXIT IF <expression> ;
                          <statement> [; <statement>]
                         END
```

The expression must result in a Boolean value.

3.4 PACK and UNPACK

PACK and UNPACK have 4 parameters. The parameters are described here for UNPACK. The first parameter is a packed array which is to be unpacked. The second parameter is an index into the packed array indicating where the unpacking will begin. The third parameter is the array into which the unpacked items are to be stored. Storing starts at the beginning of the array. The fourth parameter is the number of elements which are to be unpacked.

The call UNPACK(Z,I,A,K) is equivalent to

```
  FOR L:=LMIN(A) TO LMIN(A)+K-1 DO A[L]:=Z[L-LMIN(A)+I]
```

where LMIN(A) is the index to the first element of A.

The parameters for PACK have the same sequence, with the array and packed array interchanged.

The call PACK(A,J,Z,K) is equivalent to

```
  FOR L:=LMIN(Z) TO LMIN(Z)+K-1 DO Z[L]:=A[L-LMIN(Z)+J]
```

where LMIN(Z) is the index to the first element of Z.

3.5 MARK and RELEASE

MARK and RELEASE can be used to organize the heap like a stack. Both have one parameter which must be of type INTEGER.

MARK(X)	assigns to X the current top of the heap. The value of X should not be altered until the corresponding RELEASE(X).
RELEASE(X)	sets the top of the heap to X. This releases all the items which were created by NEW since the corresponding MARK(X).

4. PASCAL Debug System (PASDDT)

The PASCAL Debug System is only available for programs which contain the DEBUG-switch (*$D+*) and are translated with the PASREL compiler. The system can be used to set breakpoints at specified line numbers. When a breakpoint is encountered, program execution is suspended and variables (using normal PASCAL notation) may be examined and new values assigned to them. Also additional breakpoints may be set or breakpoints may be cleared. It is helpful to have a listing of the program available as the system is line number oriented. The program should be saved with W having a value of at least U + V + 8 (see Section 1.2).

4.1 Commands

The commands described here can be given when the system enters a breakpoint. When the program is executed it will respond with an asterisk as usual. After a carriage-return is typed the initial breakpoint (set by the system) will be entered with the message

```
$ STOP AT MAIN BEGIN
$
```

Additional breakpoints are set by

```
STOP  <line>
```

where <line> is of the form line#/page# or just line# which is equivalent to line#/1, e.g. 120/3. A maximum of 20 breakpoints may be set.

The breakpoint is cleared by

```
STOP NOT  <line>
```

The breakpoints set may be listed by

```
STOP LIST
```

Variables may be examined by the command

```
<variable> =
```

<variable> may be any variable as given by the PASCAL definition (except files). In particular it may be just a component of a structured type, or the whole structure itself.

A new value may be assigned to a variable by

```
<variable> := <variable or constant>
```

The assignment follows the usual type rules of PASCAL.

The current active call sequence of procedures and functions is obtained by

```
     TRACE
```

The names of the procedures and functions together with their line numbers are printed in reverse order of their activation.

Program execution is continued by the command

```
     END
```

The program will run until another breakpoint is encountered. The breakpoint is announced by

```
     $ STOP AT  <line>
     $
```

4.2 Asynchronous Interrupt

If a program goes into an infinite loop it may be aborted by ^C^C. The monitor command DDT followed by a carriage-return will enter the PASCAL Debug System. This interrupt is announced with the message

```
     $ STOP BY DDT COMMAND
     $ STOP IN <line1>:<line2>
     $
```

5. Standard and External Procedures and Functions

5.1 Standard Procedures and Functions

The following standard procedures and functions (described in the Revised PASCAL Report) are implemented. The indicator X means that this procedure or function is implemented in both the PASCAL and PASREL compilers. The indicator R means that the procedure or function is implemented only in the PASREL compiler.

Standard Functions		Standard Procedures	
ABS	X	GET	X
SQR	X	PUT	X
ODD	X	RESET	X
SUCC	X	REWRITE	X
PRED	X	NEW	X
ORD	X	READ	X
CHR	X	READLN	X
TRUNC	X	WRITE	X
EOLN	X	WRITELN	X
EOF	X	PAGE	X
SIN	R		
COS	R		
EXP	R		
LN	R		
SQRT	R		
ARCTAN	R		

Additional mathematical functions are available with the PASREL compiler:

ARCSIN	R	SIND	R
ARCCOS	R	COSD	R
SINH	R	LOG	R
COSH	R		
TANH	R		

Additional standard functions:

RANDOM	R	result is a real number in the interval 0.0 .. 1.0
DATE	R	result is a PACKED ARRAY [1..9] OF CHAR. The date is returned in the form 'DD-Mmm-YY'.
TIME	X	current time in msec (type integer)
RUNTIME	X	elapsed CPU time in msec (type integer)

Additional standard procedures:

BREAK	X	see Section 3.7
PACK	X	see Section 3.4
UNPACK	X	
MARK	X	see Section 3.5
RELEASE	X	
GETLINENR	X	The parameter must be of type PACKED ARRAY [1..5] OF CHAR. The value returned is the current DEC-10 line number, if one is present; otherwise the value '-----' is returned. (The line number is ' ' for lines caused by form feed).

5.2 External Procedures and Functions

A procedure or function heading may be followed by the word EXTERN. This indicates to the compiler that the routine will be supplied at load time. In addition it may be specified that the routine is a PASCAL, FORTRAN, ALGOL or COBOL routine. PASCAL is assumed if no language is specified. The language symbol determines how the parameters are passed to the external routine. The relocatable file also contains information to direct the loader to search the corresponding library on SYS:.

Example: `PROCEDURE TAKE(VAR X,Y: INTEGER); EXTERN FORTRAN;`

A group of PASCAL routines without a main program can be compiled separately if the compiler directive (*$M-*) is used at the beginning. The END of the last routine must be followed by a period (instead of a semicolon). Only the first 6 characters of names of the outermost routines are significant. The resulting .REL file must be loaded with any PASCAL program which contains EXTERN declarations for these routines.

Assume the files TEST.REL, AUX1.REL and AUX2.REL are to be loaded, along with a routine from the library NEW: ALGLIB.REL. Loading is accomplished by:

```
LOAD TEST,AUX1,AUX2,NEW: ALGLIB/LIB
```

Should the loader not know about the library SYS: PASLIB.REL then it must be added to the load request as SYS: PASLIB/LIB.

6. Miscellaneous

6.1 Implementation Restrictions

a) No LABEL section is yet implemented. It is not possible to leave a procedure by an exit label. Execution of the program may, however, be terminated by a call to the procedure QUIT. This procedure must be declared external:

```
        PROCEDURE QUIT; EXTERN;
```

b) Printer control characters are not available. A new page is started by a call to the standard procedure PAGE.

c) The PROGRAM declaration is not implemented.

d) Procedures and functions cannot be formal parameters.

6.2 Known Bugs

a) The comment delimiter *) is not recognized if it is preceded by an even number of asterisks, e.g. ***)

b) The number of characters used for a positive real number is field width + 1.

c) Comparison of variables of type PACKED RECORD or PACKED ARRAY may cause errors if these variables appear in a variant part or were assigned from a variant part.

d) Files may be created with date 5-Jan-75.

e) Linking of ALGOL or COBOL REL-files may not proceed as expected.

6.3 Utility Programs

CROSS is a program which formats a PASCAL program, produces a crossreference list of identifiers, indicates the nesting of statements, and reports the static nesting of procedures. It also lists for a given procedure all the procedures which it calls as well as the procedures by which it is called. The program is executed by the command

```
      R CROSS
```

The program asks for the file to be processed. Assuming the file is FILNAM.PAS it will create the files FILNAM.NEW and FILNAM.CRS. The file FILNAM.NEW contains a formatted PASCAL program which is accepted by the compilers. The file FILNAM.CRS contains a formatted listing with nesting indicators and crossreference information.

REFERENCES

[1] N. Wirth
The Programming Language PASCAL (Revised Report)
Bericht Nr. 5
Berichte der Fachgruppe Computer-Wissenschaften
ETH Zurich, November 1972

[2] K. Jensen, N. Wirth
PASCAL - User Manual and Report
Lecture Notes in Computer Science, vol 18
Springer Verlag Berlin, Heidelberg, New York, 1974

[3] C.A.R. Hoare and N. Wirth
An Axiomatic Definition of the Programming Language PASCAL,
Bericht Nr. 6
Berichte der Fachgruppe Computer-Wissenschaften
ETH Zurich, November 1972

[4] K.V. Nori, U. Ammann, K. Jensen, H.H. Naegeli
The PASCAL-P Compiler: Implementation Notes
Bericht Nr. 10
Berichte der Fachgruppe Computer-Wissenschaften
ETH Zurich, December 1974

[5] C.-O. Grosse-Lindemann, H.-H. Nagel
Postlude to a PASCAL-compiler bootstrap on a DEC System-10,
Bericht Nr. 11
Institut fuer Informatik der Universitaet Hamburg
October 1974
and Software - Practice and Experience 6, 29-42 (1976)

[6] G. Friesland, C.-O. Grosse-Lindemann, F.H. Lorenz,
H.-H. Nagel, P.-J. Stirl
A PASCAL compiler bootstrapped on a DECSystem-10
in 3. GI-Fachtagung ueber Programmiersprachen
Lecture Notes in Computer Science, vol 7
Springer Verlag Berlin, Heidelberg, New York, 1974

[7] H.-H. Nagel
PASCAL for the DECSystem-10, Experiences and further Plans
Mitteilung Nr. 21
Institut fuer Informatik der Universitaet Hamburg
November 1975

[8] W.F. Burger
PASCAL Manual
TR-22, Department of Computer Sciences
University of Texas at Austin, July 1973

[9] W.F. Burger
An Introduction to PASCAL2
TR-22A, Department of Computer Sciences
University of Texas at Austin, September 1975

Appendix A

A routine to read a standard DEC-10 file description is available as relocatable file FILNAM.REL in area [1025,2]. The file FILNAM.PAS [1025,2] is the source for this routine. It may be copied into a PASCAL program if it is to be translated by the PASCAL compiler.

The name of the routine is GETFIL. It reads a file description from TTY and opens the file if it is available, otherwise it requests a new file description. A letter sequence separated by a slash may follow the file description.

The following program shows a general application of GETFIL. Actual files are assigned to the standard files INPUT and OUTPUT.

```
(*$T-*)
TYPE CHARSET= SET OF 'A' .. 'Z';
     FNAME  = PACKED ARRAY [1..9] OF CHAR;

VAR FILENAME: FNAME;
    FLAGS: CHARSET;
    CH: CHAR;

PROCEDURE GETFIL(VAR F: TEXT; VAR FILENAME: FNAME;
                 VAR FLAGS: CHARSET);  EXTERN;

BEGIN
  REWRITE(TTYOUTPUT);
  WRITE(TTY,'FILE: '); BREAK;
  RESET(TTY);
  GETFIL(INPUT,FILENAME,FLAGS);
  WRITELN(TTY,'INPUT FILE IS  ',FILENAME);
  IF FLAGS#[] THEN BEGIN
     WRITE(TTY,'FLAGS ARE ');
     FOR CH:='A' TO 'Z' DO
     IF CH IN FLAGS THEN WRITE(TTY,CH,' ');
     WRITELN(TTY);
  END;
  FILENAME[7]:='L'; FILENAME[8]:='S'; FILENAME[9]:='T';
  REWRITE(OUTPUT,FILENAME);
  WRITELN(TTY,'OUTPUT FILE IS ',FILENAME);
END.
```

A TTY session might look as follows:

```
FILE: DSK: TEST1.PAS [1025,2] /F
FILE TEST1 .PAS NOT FOUND, REENTER
* TEST.PAS/ABC
INPUT FILE IS  TEST  PAS
FLAGS ARE A B C
OUTPUT FILE IS TEST  LST
```

Appendix B

PASLC is an experimental compiler in area [1025,2]. This compiler accepts programs written with a 95 character set. It generates relocatable code.

The character set consists of the characters with octal values from 40 to 176. These include lower case letters and the five special characters { } ~ | ` in addition to the standard 64 character set. Lower case letters are converted internally to upper case, except in strings or character constants. This means e.g. that the identifier NewName is the same as NEWNAME. However, in the listing of the program all characters will be printed as they appear, without conversion. Also, braces are now legal comment delimiters as in the Revised PASCAL Report, e.g.

```
        {This is a Comment}
```

PLCLIB.REL (also in area [1025,2]) is the library which is used by programs which were translated with the PASLC compiler. The read routines accept the 95 character set. Care must be taken with values of type SET OF CHAR (or subranges thereof). Only values of the original 64 character set may be assigned to sets, also tests of membership may only involve values of the original 64 character set.

A conversion from the 95 character set to the 64 character set can be accomplished by

```
      IF ORD(CH) > 95 THEN CH := CHR(ORD(CH)-32)
```

(A different mapping for special characters may be desirable).

If the routine GETFIL (see Appendix A) is loaded together with a PASLC program then filenames may be entered in upper or lower case.

(These modifications were implemented by Robert E. Wells).