



DECUS

PROGRAM LIBRARY

DECUS NO.	8-933 (AA)
TITLE	MOEDLING PASCAL-S
SUBMITTED BY	D. GEOFFREY CHASE
COMPANY	PORTSMOUTH ABBEY SCHOOL PORTSMOUTH, RI
DATE	V2.0, MARCH 1983
SOURCE LANGUAGE	PAL-8/PAL-III

ATTENTION

This is a USER program. Other than requiring that it conform to submittal and review standards, no quality control has been imposed upon this program by DECUS.

The DECUS Program Library is a clearing house only; it does not generate or test programs. No warranty, express or implied, is made by the contributor, Digital Equipment Computer Users Society or Digital Equipment Corporation as to the accuracy or functioning of the program or related material, and no responsibility is assumed by these parties in connection therewith.

Moedling PASCAL-S

Version: V2, March 1983

Author: Niklaus Wirth, E.T.H., Zurich, Switzerland

Submitted By: D. Geoffrey Chase, OSB, Portsmouth Abbey School,
Portsmouth, RI

Operating System: OS/8 V3-D. Tested under V3-D, including ETOS
and OMNI-8, would probably run under other OS/8 versions.

Source Language: PAL-8/PAL-III

Memory Required: 28KW

PASCAL-S is a subset of the programming language PASCAL, defined by Niklaus Wirth in Report No. 12 of the Institute for Information Science at the E.T.H. in Zurich. This implementation of PASCAL was designed for a PDP-8 with 28-K words of processor memory, running under the OS/8 operating system.

Source program text may be created or revised with any of the OS/8 File Editors, including MUTOR(1), EDIT and TECO.

Restrictions: (1). Uses BSW and MQL/MQA. See file PASCAL.8E.
(2). PASCAL-S is a subset of PASCAL. It conforms to definition of PASCAL-S given by Niklaus Wirth, the original inventor of PASCAL.

===== P A S C A L - S =====

PASCAL-S is a subset of the programming language PASCAL, defined by Niklaus Wirth in Report No. 12 of the Institute for Information Science at the E.T.H. in Zurich. This implementation of PASCAL was designed for a PDP-8/E with 28-K words of processor memory, running under the OS/8 operating system.

RUNNING PASCAL-S

Source program text may be created or revised with any of the OS/8 File Editors, including MUTOR(1), EDIT and TECO. The standard source program suffix is .PS .

To compile and run PASCAL-S programs:

```
.R PASCAL
*DEV:OUTPUT.EX<DEV:SOURCE,DEV:INPUT.EX/Option(s)
  output file   source   input file
```

[the default input and output 'file' is the user's terminal]

SEQUENCE of OPERATIONS

1. The Compiler prints a header, with version and date.
2. The source program is listed on the user's console. On the left of each line is the decimal address of the code generated by the line.
3. If syntax errors are detected, they are flagged on a line immediately below the source program line, thus:

```
20 COUNTER=COUNTER-1;
####          #51                (the = is an error)
```

At the end of the source program listings, the numeric error codes are explained in plain language. PASCAL-S then returns to the OS/8 Monitor.

4. An error-free program is executed. At its termination, the OS/8 Monitor is recalled. The user may re-run his program by typing START in response to OS/8's prompt ('.'). This option is useful when I/O is from/to the user's terminal. There is no way to respecify file I/O without recompilation; PASCAL-S has no RESET or REWRITE.

Defaults common to other OS/8 programs apply also to PASCAL. For example, an output file without a device name is written onto DSK:.

(1)MUTOR is a multi-user File Editor developed at HTL-Moedlins. Version 04 is specially adapted to writing PASCAL source code, as it permits inclusion by reference of previously-written PASCAL procedures.

===== P A S C A L - S =====

SWITCHES (Decoder Options)

/S Silent compilation, i.e., no program listing is printed.
/H Halt after compilation, return to OS/8, wait for .START.

EXAMPLES:

.R PASCAL
*ROOT/S This runs DSK:ROOT.PS . No program listing is printed. The input and output default to the user's terminal.

.R PASCAL
*PLOT:<SYS:SPIRAL The program SPIRAL.PS on SYS: is compiled, listed, and run. Output goes to PLOT:, presumably a plotter. Input, if required, is taken from the user's terminal.

.R PASCAL
*DTA3:LIST.80<WAGES.DV,PTR:/H
.START DSK:WAGES.DV is compiled and listed. When the program is STARTed, output goes to LIST.80 on Dec tape #3; input comes from the paper tape reader. Handlers for PTR: generally prompt with ^ ; the user, when ready, strikes the Return key or equivalent.

The internal handler for terminal input prints ? whenever EOLN=true and keyboard input is wanted. EOLN is forced on when a program starts to run. The user then enters a line of data as required; the usual OS/8 facilities (rubout, ^U, linefeed display) are available.

Numeric data should end in one or more blanks (spaces), even if the number ends its line. Typing no blanks and Return triggers a repetition of the ? prompt.

RUN-TIME ERRORS

These take the formText.... ERROR AT xxx. The program run is aborted. It can be re-.STARTed (with, one assumes, different data). The address xxx should be compared with the compiler listings of the source program to localise the error.

SOURCE CODE

This is written, interestingly, in excellent English and includes some useful tables of core and field usage. PASCAL.SV uses fields 0-5 at compile-time, with field 6 holding the error message explanations and the run-time initialization routine. At run-time, fields 3-6 form the stack; the top two bits of the stack pointer, plus 3, determine the field.

===== P A S C A L - S =====

Version 2: EXTENSIONS and ENHANCEMENTS

1. A predefined procedure ASCII(...) with variable number of arguments, intended to output special characters. The format is

ASCII(Arg1, Arg2, ... , Argi)

where the arguments may be any legal PASCAL expression of type Integer and specify the decimal ASCII code of the desired character. All 8 bits are transmitted, except that:

0 gives unpredictable results (the contents of location CHAR). Use 256 for a pure null character.
141 (octal 215) gives CR & LF, 13 gives CR only.

2. A predefined function RANDOM of type Real without argument returns random numbers uniformly distributed in the open interval 0,1 . The cycle length is around 2^{33} . The random numbers always appear in the same sequence; to randomize, enter the sequence at an arbitrary point by "eating" some numbers.

[Note: there is room in the once-only code at 06000 ff. to include a call to a real-time or system clock, if available. Cf. examples in PATCH.ES .]

3. WRITE('string':m)...A field width expression is now allowed also for strings and performs as described in the PASCAL User Manual & Report.

4. The Real Output Conversion Routine (SUP2) of Version 1 contained a subtle bug. The routine has been completely redesigned and seems now somewhat optimal.

5. A predefined procedure HALT, like STOP in BASIC or FORTRAN.

=====

A U T H O R

Prof. Heinz Stesbauer
HTL - Moedlins
Austria

D I S T R I B U T E D b y

Wolfgang Leber
Chairman, German 12-bit SIG
Max Planck Institut fuer Hirnforschung
Deutschordenstrasse 46
D-6000 Frankfurt am Main 71
West Germany



===== P A S C A L - S =====

/ P A S C A L . H L

/NOTES ON PASCAL-S COMPILER FOR PDP-8/E

/-----
/ASSEMBLING INSTRUCTIONS:
/-----

/ .R PAL8
/ *DEV:PASCAL<DEV:PASCAL/K

/ .R ABSLDR
/ *DEV:PASCAL/9\$

/ .SA SYS PASCAL;06000

/-----
/COMPILER/INTERPRETER LIMITS:
/-----

/ 512 IDENTIFIERS
/ 63 ARRAYS -- the listing says 64 !
/ 63 BLOCKS
/ 1980 STATEMENTS OF INTERMEDIATE CODE
/ 16 LEVELS
/ 8 CHARACTERS VALID IN IDENTIFIERS
/ 80 CHAR'S/LINE MAXIMUM FOR COMPILER INPUT (NOT PROTECTED!)

/-----
/DIFFERENCES from "WIRTH'S" PASCAL-S:
/-----

/ MAXINT = $2^{35} - 1 = 34359738367$
/ REALS BETWEEN $2.78E-309$ AND $8.98E+307$, PRECISION ABOUT $5.0E-11$

/ MAX. ARRAY-BOUNDS
/ MAX. CASE-ITEMS : $-2048 < N < 2048$

/ EOF AND EOLN W I T H O U T (INPUT)

/ ADDITIONAL PREDEFINED PROCEDURE HALT

/ N O LINE-SPACING CONTROL CHARACTERS PROVIDED!
/ (USE SPECIAL PREDEFINED PROCEDURE ASCII(N))

/ OUTPUT LINE LENGTH NOT LIMITED (USER'S RESPONSIBILITY!)



===== P A S C A L - S =====

CHANGES MADE to the MOEDLING CODE

1. The compile-time error messages were rendered from German into English. The run-time messages were in English from the beginning.
2. The liberty has been taken of mapping characters read from input files, whether source (.PS) or runtime data input. The changes are shown in the table below. Briefly, Horizontal Tab and lower-case letters are now legal in source programs and in data input files. Note that comments in PASCAL-S programs should be enclosed between (* and *), since { and } become [and] respectively.

Octal -----	Character -----	Becomes -----
11 (or 211)	H. Tab	One blank (space)
140 (or 340)	` (Accent Grave)	One blank (space)
141	a	A
142	b	B
172	z	Z
173	{	[
174		\
175	}]
176	~	^
177	Rubout (Delete)	-

PASCAL-S compiles the following correctly:

```

Program Above96 (Input, Output);          (* show conversion *)
Var   lowercase:   integer;
Begin
      for lowercase := 97 to 127 do begin
        writeln;
        write (lowercase, ' ', Chr(lowercase-32))
      end
End.

```

Program YourName(Input, Output); syntax is standard for all PASCAL-S programs(1). Read(ln) and Write(ln) are directed to this standard Input and Output, respectively.

'Lowercase' is significant to 8 characters ('lowercas'). At least one variable needs to be declared, even if no variables are actually used.

(1) See Eisenbach & Sadler: PASCAL for Programmers, Springer-Verlag 1981, Ch. 8, page 122, "large main-frame implementations of PASCAL...".



===== P A S C A L - S =====

CHANGES MADE, continued.

3. According to John Easton, the University of Minnesota experimented with PASCAL-S before deciding to write their own fuller implementation of PASCAL. There were two grammatical oddities in PASCAL-S that disturbed them:

(A.) The use of double-quote (") to enclose text when other PASCALS use single-quote (apostrophe: '); and,

(B.) The use of pound-sign (#) as a not-equals sign, like <>.

Since the double-quote requires the programmer to do something not standard in PASCAL, it has been removed. The quoting character is now the apostrophe (single-quote). To quote an apostrophe, type it twice, e.g. Writeln ('PASCAL'S syntax'); .

The pound-sign doesn't impose itself on the programmer. It might be considered an 'enhancement', a bit like ASCII(N) and RANDOM. But the user should not expect enhanced code to be portable to another version of PASCAL.

4. The source changes made in PASCAL.PA are shown in PASDIF.LS . Optional patches to RANDOM and to the '#' syntax are shown in PATCH.ES . PASCAL-S has no overlays; the .SV file is easy to patch with ODT or with FUTIL.

NOTE: N. Wirth's definition of PASCAL-S and a description of its compiler have been reprinted in D. W. Barron: PASCAL, The Language and its Implementation, Wiley 1981/1982, Chapter 8.



===== P A S C A L - S =====

```

Program ASCCHR (input, output);
  (* Compare Ascii(N) with Write( Chr(N) ) *)
Var
  counter, ascicode: integer;
begin
  for counter := 64 to 84 do
    begin
      ascicode := counter;

      repeat
        Write (ascicode, ' ');
        Ascii (ascicode);
        Write ( ' ', Chr(ascicode) );
        ascicode := ascicode+21
      until ascicode > counter+42;
      (* don't omit that semi-colon ^ ! *)

      writeln
    end
  end.

```

64	@	@	85	U	U	106	j	*
65	A	A	86	V	V	107	k	+
66	B	B	87	W	W	108	l	,
67	C	C	88	X	X	109	m	-
68	D	D	89	Y	Y	110	n	.
69	E	E	90	Z	Z	111	o	/
70	F	F	91	[[112	p	0
71	G	G	92	\	\	113	q	1
72	H	H	93]]	114	r	2
73	I	I	94	^	^	115	s	3
74	J	J	95	_	_	116	t	4
75	K	K	96	`	`	117	u	5
76	L	L	97	a	!	118	v	6
77	M	M	98	b	"	119	w	7
78	N	N	99	c	#	120	x	8
79	O	O	100	d	\$	121	y	9
80	P	P	101	e	%	122	z	:
81	Q	Q	102	f	&	123	{	;
82	R	R	103	g	'	124		<
83	S	S	104	h	(125	}	=
84	T	T	105	i)	126	~	>

G. Chase, O.S.B.
 Portsmouth Abbey School
 December 1982--February 1983



===== P A S C A L - S =====

Some April Foolishness: footnotes to PASCAL(1)

1. PASCAL is forgiving of type change from Integer to Real, thus 'If Trunc(X) = X then ...' is accepted even though Trunc() returns an integer. But otherwise the language is very strongly typed. Integers, Reals, Booleans, Chars: don't mix them!

2. PASCAL wants its semi-colons. Two favorite traps are after Program Moo (Input, Output); and Procedure Foo (var Wu: integer); . Likewise after Var Fee, Fie, Fo, Fum: integer; .

In general, every complete statement is ended by semi-colon. Note, however, If J>K Then J:=K Else J:=1; -- don't embed a semi-colon between If and Else.

The semi-colon may, really should, be left out if the next word is End or Until (but an End or an Until clause must itself be ended with ';' -- unless, of course, it be followed by another End).

3. Programs MUST begin with PROGRAM ... and end with END. . This is the only END that may and must be followed by full stop.

Errors

4. Wirth's write-up on PASCAL-S mentions care taken to avoid superfluous error messages: those resulting from an earlier program syntax error. Still, PASCAL's structure makes this difficult. You will often get error messages which seem to point at text which is, in fact, perfectly sound -- and would be recognized as sound IF there hadn't been that little mistake two lines before.

The compiler flags an error when it's sure it's an error; this may not be where the programmer made his mistake. Missing semi-colons, for example, are usually missing before the text flagged with #14 .

) PASCAL experts may not read this page. It's too embarrassing.



===== P A S C A L - S =====

Printins Formats

5. Write ('text':N) has been mentioned on Page iii.

Write (realnumber:8:2) resembles FORTRAN F8.2 or PRINT USING "####.##" . But format :8:0 prints in exponential format, not in F8.0 . A rule of thumb is to make formats wider than seems, at first sight, necessary.

Write (intvalue: 4) is the I4 or "####" integer format.

Booleans

'If STUFF' is equivalent to 'If STUFF = True' .
'If Not STUFF' is equivalent to 'If STUFF = False' .

It is often necessary to parenthesize Boolean expressions. The operator precedence can cause trouble.(1) Warning: an 'And' will test both conditions even if the first be found false. If you don't want this try, e.g., IF A<B Then IF C<D Then ...

PAS April, 1983

(1) See PASCAL for BASIC Programmers, Addison-Wesley 1983, pages 54-55, 'Operator Precedence'.

