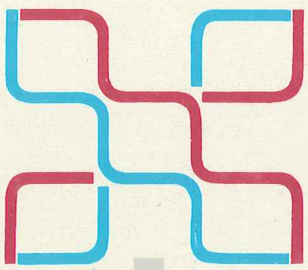
PASCAL ON POLY



polycorp

PASCAL ON POLY

VERSION 2

MAY 1983

POLYCORP

New Zealand Limited

Progeni House, 14/18 Pretoria Street
Telephone (04) 693-302, P.O. Box 30-243, Lower Hutt
Telex PO LHT NZ3740 Attn Progeni

CONTENTS

1. INTRODUCTION

Omegasoft Pascal is supplied for POLY in two forms:

(i) A single-command method of compilation which is easy to use and requires the minimum of user interaction.

(ii) For users with a more extensive knowledge of the processes involved, there is a version which allows more flexibility. However, the compilation process is much more complex than with the single-command method. (See Appendix 7.1).

There is no difference in the Pascal language supported, only in the compilation process.

Omegasoft Pascal programs are written using the text editor resident on the POLY. The programs are then SAVEd on disk and the Pascal compiler executed.

Compilation of a Pascal program consists of three phases:

1. Compiling the Pascal source code into assembler code.

2. Converting the assembler code into 6809 relocatable machine language object code.

3. Linking the machine language program with the system subroutines used in the program.

As the Pascal program is compiled into 6809 machine language, the resulting compiled program executes extremely quickly.

2. PREPARING A PASCAL PROGRAM

The POLY Text Editor is described in the POLYSYS Utilities Manual. Basically the steps to use it are:

1. Load the text editor.

   This is done by entering

   TEXT

   from either DOS or BASIC

   Ready

   is displayed in CYAN.

2. Type in the Pascal program.

   The editing commands are described fully in the POLYSYS Utilities Manual. Note that the line numbers shown on the screen are for editing purposes only and are removed when the program is SAVEd onto disk.

3. SAVE the program onto disk.

   Use the SAVE command. This saves the program as filename.TXT.

   For example:

   SAVE "MYPROG"

   saves your program as MYPROG.TXT.

The files supplied for the standard Pascal are:

PASCAL.CMD	A DOS command which sets the standard options and controls the complete compilation process.
LL.CMD	The linking loader.
PASLNK.RO	A linking file.
PC.CMD	The Pascal compiler.
RA.CMD	The relocatable assembler.
RE.CMD	The reenter command.
RL.RO	The subroutine library.

These files occupy approximately 320 sectors on disk and must be resident on the disk being used for the compilation.

To compile a Pascal program, enter DOS mode and run the Pascal compiler. The syntax of the command is:

PASCAL filename [S][D][L]

The filename is the name of the source file previously saved on disk. The default extension is .TXT. The resulting machine language program will be stored as a file with the name filename.CMD.

If S is specified then the compiler checks the program for syntax errors and no machine language program is produced.

If S is not specified and errors are found during the compile, then the machine language program is not created.

If L is specified, then a listing of the compilation on the printer is made.

If D is specified as well as L, then the listing is displayed on the screen, not the printer.

For example:

PASCAL MYPROG SL

compiles the source file MYPROG.TXT checking for syntax errors. A listing is produced on the printer.

PASCAL MYPROG.TXT LD

compiles the source file MYPROG.TXT, and if no errors are found, the object program MYPROG.CMD will be created.

Following compilation, the object program may be run by simply entering the filename.

For example:

```
MYPROG
```

After a program has been run, it may be re-run by using the re-enter command.

For example:

```
MYPROG
  .
  .
  .
RE
```

will run MYPROG twice.

The following Pascal program is supplied on disk:

```
program fastprimes(INPUT,OUTPUT);
(* find the first 1229 primes *)
const n=1229; n1=35; (* sqrt of n *)
var i,k,x,inc,lim,square,lin: integer;
    prim: boolean;
    p,v: array(.0..n.) of integer;
begin
    writeln;
    write(2:8,3:8); lin:=2;
    x:=1; inc:=4; lim:=1; square:=9;
    for i:=3 to n do
    begin (* find next prime *)
      repeat x:=x+inc; inc:=6-inc;
        if square<=x then
          begin lim:=lim+1;
            v(.lim.):=square; square:=p(.lim+1 .)*p(.lim+1 .)
          end;
        k:=2; prim:=true;
        while prim and (k<lim) do
        begin k:=k+1;
          if v(.k.)<x then v(.k.):=v(.k.)+2*p(.k.);
          prim:=x<>v(.k.)
        end
      until prim;
      if i<=n1 then p(.i.):=x;
      write (x:8); lin:=lin+1;
      if lin=9 then
        begin writeln; lin:=0
        end
    end;
  writeln
end.
```

This program is stored on disk as PRIMES.TXT.

To edit the program, enter

```
TEXT,PRIMES
```

When complete enter

```
LIST
```

The file will be displayed on the POLY with line numbers. Move the cursor up to line 2, change 1229 to 1000 and press <ENTER>. Repeat for line 3. Then, enter

```
SAVE "NEWPRIME"
```

and the altered file will be saved as NEWPRIME.TXT.

Enter

```
PASCAL NEWPRIME L
```

When the compilation is complete

```
DOS
```

is displayed. Enter

```
NEWPRIME
```

to run the program.

Shift <EXIT> may be pressed at any time to terminate the program and return the POLY to DOS mode.

5.1. Changes made to Omegasoft Pascal

The device AUXOUT documented in the Omegasoft Pascal Manual as the printer, has not been implemented. To print on the printer, open a text file with .PRT as the extension. When the file is closed, it is printed and deleted.

5.2. Character set

The POLY does not have square or curly brackets in the character set. Instead use (* and *) in place of curly brackets, and (. and .) in place of square brackets.

The Teletext hash (#) on POLY is the equivalent of the ASCII underline (_), while the Teletext pound sign (£) is the equivalent of the ASCII hash (#).

When writing comments curly brackets should be replaced with (* and *).

For example:

```
(* this is a comment *)
```

When using subscripts or sets the square brackets should be replaced with (. or .).

For example:

```
character:= instring(. 2 .)
digits:= (. '0'..'9' .)
```

When declaring identifiers, the Teletext hash should be used in place of the ASCII underline.

For example:

```
var input#string : string;
```

When designating a byte constant, a Teletext pound sign should be used in place of the ASCII hash.

For example:

```
const enter =£13
```

6.1. Use of Teletext colours

The screen is displayed using the Teletext conventions but Pascal programs execute in ASCII mode. In order to use the Teletext control characters, a shift must be made into Teletext mode before using the Teletext control characters, and a shift back to ASCII after using them. The example below shows how the CHR function is used to incorporate these in an expression.

The teletext control characters are:

ASCII DECIMAL VALUE		FUNCTION
0		Not used
1		Start RED characters
2		Start GREEN characters
3		Start YELLOW characters
4		Start BLUE characters
5		Start MAGENTA characters
6		Start CYAN characters
7	*	Start WHITE characters
8		Start FLASHING
9	*	End FLASHING
10		Not used
11		Not used
12	*	Normal height
13		Double height
14		Shift to ASCII
15		Shift to Teletext
16		Reverse video on
17		Start RED graphics
18		Start GREEN graphics
19		Start YELLOW graphics
20		Start BLUE graphics
21		Start MAGENTA graphics
22		Start CYAN graphics
23		Start WHITE graphics
24		CONCEAL display on rest of line
25	*	Contiguous graphics
26		Separated graphics
27		Reverse video off
28	*	No background to characters
29		Set background to current colour
30		Print graphics characters over control characters
31	*	Print space for control characters

Each of the control characters occupies ONE screen position except the reverse video on character, the reverse video off character, the ASCII to Teletext shift character, and the Teletext to ASCII shift character. These characters do not require screen positions. All control characters are reset at the beginning of each line of the screen to those with an * beside

them.

Double height characters extend down to the following line. If double height is used anywhere on a line, the following line is not displayed.

For example:

```
program colour(output);
var
   num : integer;
   red,si,so : char;
begin
   red:= chr(1);
   si:= chr(14);
   so:= chr(15);
   for num:=1 to 100 do
     writeln(si,red,so,num);
end.
```

This prints out the numbers 1 to 100 in red. Note the shift to and from Teletext before and after the red control code.

6.2. ASCII Control Codes

ASCII DECIMAL VALUE	FUNCTION
7	Beep
8	Move the cursor 1 space to the left
9	Move the cursor 1 space to the right
10	Move the cursor down 1 line
11	Move the cursor up 1 line
12	Clear screen and move the cursor to the home position
13	Move cursor to the start of the line (RETURN)
14	Shift to ASCII mode
15	Shift to Teletext mode
16	Reverse video on
27	Reverse video off
30	Clear to the end of the line

7. APPENDICES

7.1. Unpackaged Omegasoft Pascal

To use the unpackaged Pascal, additional manuals and utilities are required.

The extra manuals are:

- Omegasoft Pascal Utilities Manual.
- Omegasoft Relocatable Assembler and Linking Loader manual.
- Pascal Configuration Manual for FLEX (The POLY operating system is compatible with the FLEX operating system).

The extra utilities are:

CHAIN.CMD The linkage creation control program.

DB.CMD The debugger.

LB.CMD Librarian for the runtime routines.

LC.CMD The linkage-file creator.

To compile a Pascal program use the following steps:

1. Run the Pascal compiler.
2. Run the Linkage Creator.
3. Run the Linkage.

The syntax for the Pascal compiler for normal use on the POLY is:

PC <source-file [>output-file O] [>>print-file L]

Source-file is the file containing the Pascal program, the default extension is .TXT. The output-file is the compiler output file for the next phase of the compilation. The print-file is the compiler listing.

The syntax for the Linkage Creator for normal use on the POLY is:

LC

The Linkage Creator prompts the user for various parameters for the assembling and linking of the program.

The syntax for CHAIN (the last phase of the compilation) is

```
CHAIN filename
```

where filename is the output file from the linkage creator (the default extension is .CF)

Shown below is an example showing the steps used to compile PRIMES using the unpackaged method (user input are shown underlined).

```
PC <PRIMES >PRIMES O >>PRIMES L
Pascal Compiler Version 2.03
Copyright 1981 by Omegasoft

LC
Linkage Creator Version 2.00
Copyright 1982 by Omegasoft
Pascal compiler output file name : PRIMES
Pascal program name : FASTPRIMES
Auto setup ? Y
System stack size : 512
Starting load location : 100
Library drive number : <ENTER> (current drive used)
Additional files to load : <ENTER> (no extra files)
Additional library files : <ENTER> (no extra files)
Load options : <ENTER> (no load options)
Map options : <ENTER> (no map options)

CHAIN PRIMES
Chain version 2.00
Copyright 1982 by Omegasoft
RA <PRIMES.CO >PRIMES.CA O
Relocatable Assembler version 1.20
Copyright 1982 by Omegasoft
Total errors : 0 Psct size : 029D Table usage : 26
RA <PRIMES.PS >PRIMES.PA O
Relocatable Assembler version 1.20
Copyright 1982 by Omegasoft
Total errors : 0 Psct size : 002E Table usage : 5
LL
Linking Loader version 1.20
Copyright 1982 by Omegasoft
?STRP=$0100
?LOAD=PRIMES.PA PRIMES.CA
?LIB= RL
?OBJA=PRIMES.BIN
?MAPC
PSCT SIZE=098F START=0100 END=0A8E
SYMBOL TABLE USAGE: USED=45
MODULE TABLE USAGE: USED=16 OUT OF=58
?EXIT
```

```
End chain
```

The above example will result in a file PRIMES.BIN, which would be identical with PRIMES.CMD produced by the command:

```
PASCAL PRIMES L
```

7.2. Using Software Interrupts in Pascal

Pascal has no command that can directly call software interrupts. Software interrupts must be called from Assembler code which may be directly imbedded in a Pascal program.

For example:

```
program swi#example(input,output);
(* example program using software
    interrupts duplicates the first
    example for SPLIT in the POLYBASIC
    Manual *)
procedure cursor(row,column:integer);

 (* position the cursor at specified text
     row and column. If the values
     supplied are outside the screen
     then the cursor is not moved *)

begin
!         PSHS   D,X,Y,U   Save registers
!         LEAU   8,U       Skip over the linkage data
!         LEAU   1,U       Point to low order byte of column
!         LDB    0,U++     Load B with the low order byte of column
!         LDA    0,U       Load A with the low order byte of row
!         SWI              Position cursor
!         FCB    9
!         PULS   D,X,Y,U   Restore the registers
end;

procedure split(splitat:integer);

(* Split the screen into two separate
    scrolling sections. The parameter
    is the row at which the split is
    to be done. If the parameter is
    0 or >24 then split is turned off *)

begin
!         PSHS   D,X,Y,U   Save the registers
!         LEAU   8,U       Skip over the linkage data
!         LDD    0,U       Load D with the 16 bit split-row value
!         EXG    A,B       Swap byte positions of D
!         SWI              Split screen
!         FCB    13
!         PULS   D,X,Y,U   Restore the registers
end;

procedure wait(milliseconds:integer);

(* wait for the specified number of
    milliseconds *)

begin
!         PSHS   D,X,Y,U   Save the registers
!         LEAU   8,U       Skip over the linkage data
!         LDD    0,U       Load the time to wait
!         SWI              Wait for x milliseconds
```

```
!         FCB    19
!         PULS   D,X,Y,U   Restore the registers
end;

procedure numberprint;

var
  loop: integer;

begin
  for loop:=1 to 21 do
    writeln(loop:2);
    wait(20);
end;

begin (* mainline *)
  page;
  split(10);
  numberprint;
  cursor(10,0);
  numberprint;
  wait(500);
  split(0);
end.
```

7.3. Current Restrictions of Omegasoft Pascal Version 2.03

Omegasoft has documented the following restrictions:

1. Long integers are currently not supported. The standard identifier longinteger and any other identifiers that are associated with long integers will not be recognised by the compiler.

2. Structures may not include devices as part of the structure. This includes arrays of devices or records containing devices. If this type of structure is desired it is possible to define the array or record as containing pointers to a device and using the addr function to set the pointer.

3. There are only two of the possible six modes available for opening a file. The last parameter of an open or create procedure is not currently used.
   The two modes for opening a file are:

   (i) RESET - open an existing file on disk

   (ii) REWRITE - open a new file on disk.

   Random access files are not currently supported by Omegasoft.

7.4. Runtime Options

Runtime options are set at compile time and affect the execution of the program. The options that can be selected are

C - Conversion checks

I - Runtime input-output checks

R - Runtime range checks

These options are enabled or disabled by placing the option in the Pascal source file. For further information see the entry for "comments" in the Omegasoft Pascal Manual.

7.5. Input/Output Example

Listed below is a simple example showing the method for disk input/output using Omegasoft Pascal.

```
program inputoutput(input,output);
var
  eofflag : boolean;
  filename,buffer : string;
  file1 : text;

begin
  eofflag:=false;
  write('Enter file name: ');
  readln(filename);
  (* open the file for output *)
  rewrite(file1,filename);
  (* put data into the file *)
  repeat
    readln(buffer);
    (* if the first character of the
      input string is a "#" then input
       is complete *)
    if buffer(.1.)='#' then eofflag:=true
      else
    writeln(file1,buffer);
  until eofflag;
  (* data input complete, close file *)
  close(file1);
  (* open the new file input *)
  reset(file1,filename);
  (* read the data back from the
    file and display it on screen *)
  while not eof(file1) do
    begin
      readln(file1,buffer);
      writeln(buffer);
    end;
end.
```