DRAPER PASCAL Version 2.1 ; Copyright (C) 1989 by Norm Draper ; For the Atari 400, 800, XL, or XE series computers

Table of Contents

ATR-Images#

- Draper Pascal/drpascal.atr ; Draper Pascal - Disk A
- Draper Pascal/drpascb.atr ; Draper Pascal - Disk B
- Pascal Documentation.atr ; Pascal Documentation

Manuals#

- Draper_Pascal_Manual.pdf ; Draper Pascal 2.1 manual from Norm Draper ; registered users version with complete description of all definitions ; AtariWiki says mega thank you to Norm Draper for giving this manual into PD. We really appreciate your help and contribution to the Atari community. Be sure, you are in our hearts. Thanks you so much. We further thank Kevin Savetz so much much for getting in contact with Norm Draper and sending us this lost to believe manual after 29(!) years... That is so amazing! Thank you Kevin, we all appreciate your work so much. :-)
- Draper Pascal 2.1.pdf ; Draper Pascal 2.1 disk based documentation

Disk Based Documentation#

This document contains the "Getting Started" section plus other sections from the actual Draper Pascal manual. It is designed to get you up and running and doing useful things with Draper Pascal in a very short time. It does not, however, contain a detailed description of all the Draper Pascal definitions or pretend to be a comprehensive tutorial or user's reference. Registered users will receive the actual Draper Pascal manual which does contain a complete description of all definitions plus other information. Details on registering are in the last section titled "Registration Form".

The Shareware Concept#

Draper Pascal is distributed on a Shareware basis. You may use Draper Pascal without charge for the purpose of evaluating it's suitability for your use. If you find that Draper Pascal is worth continuing to use, then you are requested to become a registered user and gain the additional benefits that go along with it. Simply complete the registration form and mail it in to us with the very reasonable payment of only $15.00. When you register your copy of Draper Pascal, you'll receive the latest version of the software if your registered version is not current, along with a comprehensive printed user manual.

If you have received this copy from a Users' Group, "Software Library" or "SIG", and have paid a small fee (usually $3 to $10), you have NOT acquired the registered rights or continued right to use Draper Pascal. This fee was for the convenience of obtaining the diskette with Draper Pascal on it. The fee does NOT apply to the registration fee.

You may freely copy Draper Pascal for distribution under the Shareware concept, without charge.

You may NOT charge any fee for the Draper Pascal program or documentation without our written approval.

You may NOT distribute Draper Pascal or it's documentation in connection with ANY commercial venture, product, publication or service unless you obtain the royalty-free license by registering.

Features#

- Draper Pascal is a complete package allowing the user to create, compile, and execute programs written in the Pascal language.
- Many features from both UCSD and ISO standard Pascals plus many extensions, such as sound and graphics, to make use of the versatile Atari hardware.
- Pascal structured programming features, including IF-THEN-ELSE, WHILE-DO, CASE-OF-ELSE, FOR-TO/DOWNTO-DO, and REPEAT-UNTIL are included.
- FUNCTIONs and PROCEDUREs using local or global variables. Integer type variables can be passed directly.
- Data types supported include: Character, String, Integer, Real, Boolean, Record, and File. One and two dimension arrays are also supported for all data types other than File and Record.
- Only one disk drive and 48K RAM are required. Multiple, and double density, disk drives are supported.
- Machine language subroutines may be loaded and called.
- Program chaining is supported.
- No limit on source program size. "Include" files are supported.
- One-pass compiler generates pseudo code directly.
- Maximum program pseudo code size is about 30K bytes.
- Textual compiler error messages.
- Execution debugging features include instruction trace and stack display.
- Special options for error display and break key disable.

- Single keystroke program execution repeat, exit to DOS, or exit to Main Menu program are featured.
- Includes Editor program to create, modify, and print Pascal source files, or other text type files.
- Includes Main Menu program (Pascal source included) for easy select of Compiler, Editor, or utility functions, such as directory or file listing. Main Menu program may be replaced with a user written program to create a turnkey operation.
- Ramdisk support. Details provided to registered users.
- Works with most popular versions of DOS.
- Easy to use. No linking required. Compile and execute immediately.
- Royalty free license available to registered users.
- Includes sample programs.
- Comprehensive user manual provided to registered users. Complete examples and BASIC equivalents given for each reserved word.

Introduction#

Draper Software welcomes you to the world of Pascal for the Atari 400/800, XL, and XE series Computer systems.

What is Pascal?#

Pascal is a high-level structured programming language developed by Niklaus Wirth in 1971. It is easy to understand and well suited for program development and maintenance.

What is Draper Pascal?#

Draper Pascal is not a "standard" Pascal. It has a number of commands which are exactly like ISO and UCSD versions, some which are similar, and many "extensions" which bring out the true power of the Atari computer in an easy to use manner. It was designed to require only one disk drive for operation, but not be limited to only one. At this time, it has been shown to work with all hardware and software configurations where enough memory is provided. This implementation also has a number of commands which are familiar to Atari BASIC users, such as POKE, PEEK, SETCOLOR, NOTE, POINT, etc..

About this manual#

This manual is intended to familiarize you with all the features of Draper Pascal. It is not intended to teach you how to program in Pascal. However, if you already know Atari BASIC, then you can understand the Pascal statements more easily by referring to their BASIC equivalents shown after the definition of each Pascal reserved word in the complete manual provided to registered users. It is recommended that you read this manual completely to be familiarized with its features and restrictions.

What is Draper Pascal made of?#

This implementation of Pascal is made up of three main components. They are the Supervisor (sometimes referred to as runtime routines), the Compiler, and the Editor. The Supervisor is a high performance machine language program which simulates a 16-bit pseudo computer. The Compiler translates Pascal source code into pseudo-code instructions to be executed by the Supervisor. The Editor is used to enter and modify Pascal source programs. It may also be used to edit data files, or BASIC programs which have been LISTed to a disk or tape. These components are explained in detail within this manual.

For a description of the various files included on the supplied diskette, refer to the 'System Information' section.

About the DOS#

Draper Pascal can be used with most popular Disk Operating Systems. It has been tested with Atari DOS 2.5, SpartaDOS 3.2d, and MYDOS. You should format a diskette with DOS on it to contain the Draper Pascal system. Since the Draper Pascal Supervisor is named AUTORUN.SYS, it will execute immediately after the disk is booted. For XL and XE computers, you do not need to hold down the Option key while booting unless you are using SpartaDOS. If using SpartaDOS, you may want to rename AUTORUN.SYS to PASCAL.COM and create a STARTUP.BAT file containing the following two lines:

```
BASIC OFF
PASCAL
```

This documentation file is not required to be on that same diskette.

Ramdisk support#

Draper Pascal supports the use of the "Ramdisk" capability provided by using a DOS that supports a ramdisk like Atari DOS 2.5 or SpartaDOS 3.x with an Atari computer system having sufficient memory to support the ramdisk. While using this feature, the Editor takes less than two seconds to load and the Compiler takes less than three seconds. Details on setup and use of this feature are provided in the complete manual provided to registered users.

Getting Started#

This section is intended to show by example how to use the Draper Pascal system. You will edit, compile, and run a sample program. Information displayed by the computer is shown in normal type while responses to be entered by you are shown underlined with dashes (---). To begin with, make sure you have 48K RAM installed and no cartridge in place. Boot the disk now by placing it in disk drive 1 and turning on the power to the Atari computer. After the Supervisor has finished loading, you will see a screen that looks like this:

```
DRAPER PASCAL


1 - Run Program

2 - Disk Directory

3 - Compile Program

4 - Edit a Program

5 - Exit to DOS

6 - List a file

7 - Trace on

                DRAPER SOFTWARE
                    EDITOR
```

```
        A - Add line(s) at end
        C - Change line(s)
        D - Delete line(s)
        E - Edit a line
        F - Filer menu
        I - Insert before line
        L - List line(s)
        M - Menu
        P - Print line(s)
        Q - Quit
        S - Scan line(s)
        X - Exit to Compiler
A,C,D,E,F,I,L,M,P,Q,S,X,?->F          Select Filer menu
-


        A - Append file
        D - Directory list
        L - Load file
        S - Save file

        L                             Load a file
        -

Enter filename  -> SAMPLE1            Enter the name of the file
                   -------            to be loaded. The name of
                                      the last file edited,
                                      compiled, or run will be
                                      filled in by the Editor.
                                      You may have to overtype
                                      it with the name shown.

A,C,D,E,F,I,L,M,P,Q,S,X,?->L          List the file on the
                           -          screen

Line from ->                          Just press RETURN for
            -                         'Line from' and 'Line to'.
Line to   ->                          This will give a list of
            -                         the entire program in
                                      memory.


  1:PROGRAM KALEIDOSCOPE;
  2:VAR I,J,K,W,X:INTEGER;
  3:BEGIN
  4: MAXGRAPH(19);
  5: GRAPHICS(19);
  6: X:=0;
  7: REPEAT
  8: FOR W:=3 TO 50 DO
  9:  BEGIN
 10:    FOR I:=1 TO 10 DO
 11:    BEGIN
 12:     FOR J:=0 TO 10 DO
 13:      BEGIN
 14:       K:=I+J;
 15:       COLOR(J*3/(I+3)+I*W/12);
 16:       PLOT(I+8,K);
 17:       PLOT(K+8,I);
```

```
 18:       PLOT(32-I,24-K);
 19:       PLOT(32-K,24-I);
 20:       PLOT(K+8,24-I);
 21:       PLOT(32-I,K);
 22:       PLOT(I+8,24-K);
 23:       PLOT(32-K,I)
 24:      END
 25:    END
 26:  END
 27: UNTIL X=99 (* UNENDING LOOP *)
 28:END.
A,C,D,E,F,I,L,M,P,Q,S,X,?->I
                           -
Line -> 15
        --

 15:       (* MY FIRST EDIT *)
           ------------------
 16:
     -
```

Let's insert a comment before line 15.

Enter the data to be inserted when prompted for line 15. Just press RETURN when prompted for line 16. This will terminate insert mode.

```
A,C,D,E,F,I,L,M,P,Q,S,X,?->L
                           -

Line from ->
            -
Line to   ->
            -

  1:PROGRAM KALEIDOSCOPE;
  2:VAR I,J,K,W,X:INTEGER;
  3:BEGIN
  4: MAXGRAPH(19);
  5: GRAPHICS(19);
  6: X:=0;
  7: REPEAT
  8: FOR W:=3 TO 50 DO
  9:  BEGIN
 10:    FOR I:=1 TO 10 DO
 11:    BEGIN
 12:     FOR J:=0 TO 10 DO
 13:      BEGIN
 14:       K:=I+J;
 15:       (* MY FIRST EDIT *)
 16:       COLOR(J*3/(I+3)+I*W/12);
 17:       PLOT(I+8,K);
 18:       PLOT(K+8,I);
 19:       PLOT(32-I,24-K);
 20:       PLOT(32-K,24-I);
 21:       PLOT(K+8,24-I);
 22:       PLOT(32-I,K);
 23:       PLOT(I+8,24-K);
 24:       PLOT(32-K,I)
 25:      END
 26:    END
 27:  END
```

```
 28: UNTIL X=99 (* UNENDING LOOP *)
 29:END.
A,C,D,E,F,I,L,M,P,Q,S,X,?->F
                           -
```

Let's save the program back to disk drive 1 under the same name.

```
        A - Append file
        D - Directory list
        L - Load file
        S - Save file


        S
        -
Enter filename  -> SAMPLE1
                   -------
A,C,D,E,F,I,L,M,P,Q,S,X,?->X
                           -
```

Now let's exit directly to the Compiler.

```
         Draper Software

         Pascal Compiler

           Version 2.1


         Copyright 1989
         by Norm Draper



         Enter Filename:
         SAMPLE1
         -------
```

Enter name of program to be compiled. The name of the last program edited, compiled, or run will be filled in by the Compiler.

```
Enter List Output Filespec
Default is E:
             -
```

Just press RETURN at this point to have the compile list directed to the screen.

```
0000 PROGRAM KALEIDOSCOPE;
0000 VAR I,J,K,W,X:INTEGER;
0003 BEGIN
0003  MAXGRAPH(19);
0017  GRAPHICS(19);
001B  X:=0;
001E  REPEAT
0022  FOR W:=3 TO 50 DO
002A   BEGIN
0035     FOR I:=1 TO 10 DO
003D     BEGIN
0048      FOR J:=0 TO 10 DO
004F       BEGIN
```

```
005A         K:=I+J;
0062         COLOR(J*3/(I+3)+I*W/12);
008A         PLOT(I+8,K);
0098         PLOT(K+8,I);
00A6         PLOT(32-I,24-K);
00B8         PLOT(32-K,24-I);
00CA         PLOT(K+8,24-I);
00DC         PLOT(32-I,K);
00EA         PLOT(I+8,24-K);
00FC         PLOT(32-K,I)
010A        END
010A      END
010C    END
011C   UNTIL X=99 (* UNENDING LOOP *)
0142 END.
0147
ADDR   NAME
---- --------
0003 I
0004 J
0005 K
0006 W
0007 X
5 Compiler table entries used
*** Program Execution Completed ***
Highest Stack Address Used = $AFF8
<START>Repeat,<SELECT>Menu,<ESC>Exit
```

Press the SELECT key at this point to take us to the main menu.

```
           DRAPER PASCAL

            VERSION 2.1

         1 - Run Program

         2 - Disk Directory

         3 - Compile Program

         4 - Edit a Program

         5 - Exit to DOS

         6 - List a file

         7 - Trace on



           Copyright 1989
           by Norm Draper


       1
       -
```

Select '1' to run the program that was just compiled.

```
  Enter name of program to be run

       SAMPLE1
```

The name of the last

```
-------                          program edited, compiled,
                                 or run will be filled in
                                 by the main menu program.
                                 Overtype the name if you
                                 want to run a different
                                 program.
```

At this point you should have a nice kaleidoscope pattern being displayed on your television screen. To stop it, press the BREAK key. To repeat execution, press the START key. To return to the main menu, press the SELECT key. To exit to DOS, press the ESC key.

Another program, SAMPLE2, is also provided for you to practice with. It will display Roman numerals for powers of two between 1 and 4096. Compile it, turn on the trace via the main menu, and run it. After it is finished, press CTRL-T to display the trace table, and CTRL-S to display the stack contents. When prompted for 'Where? Filespec', enter 'E:'. For a description of the stack display line, refer to the 'DUMPSTK' command in the 'Pascal Definitions' section of the manual provided to registered users.

Main Menu#

The Main Menu is the initial program to be run by the Supervisor. It is written in Pascal. The source code is provided for it and you may customize it as you see fit. The disk filename for the source is 'INIT.PAS'. The pseudo code program that is initially executed is 'INIT.PCD'. It would be wise to copy 'INIT.PCD' to another name to be used in case your compile of the menu program is not successful. Or, you could rename INIT.PAS to something else, like NEWINIT.PAS, and compile it to produce NEWINIT.PCD. Then you can use the 'run' option (mentioned below) to test your modified program.

The Main Menu appears as follows:

```
                DRAPER PASCAL

                 VERSION 2.1

            1 - Run Program

            2 - Disk Directory

            3 - Compile Program

            4 - Edit a Program

            5 - Exit to DOS

            6 - List a file

            7 - Trace on


               Copyright 1989
               by Norm Draper
```

Each of the menu options will now be explained:

Run Program#

Use this option to execute a program that has previously been successfully compiled. You will see the following prompt:

```
Enter name of program to be run
```

The Main Menu program will fill in the name of the last program edited, compiled, or run. If this is the one you want, all you have to do is press RETURN. If it is not the one you want, just overtype the name shown with the one you want.

Disk Directory#

This option will provide you with a list of all, or selected, files on one of your disk drives. You will receive the prompt 'Filespec?'. If you just press RETURN at this point, you will see a list of all files on the default drive. If you enter 'D2:', you will see all files on drive 2. To show only selected files, use wildcards in the normal manner. For example, enter 'D1:INIT.*' to show only files named INIT with any suffix from drive one. At the end of the list, you will be prompted to press any key to continue. After pressing any key, the Main Menu will be re-displayed.

Compile Program#

This option sends you directly to the Pascal compiler. You will be prompted for the name of the program to be compiled, after the Compiler is loaded. If you have already edited, compiled, or run a program, the name will be shown and may be used by just pressing the RETURN key. For more information, refer to the section of this manual on 'The Compiler'.

Edit a Program#

Control is transfered to the Draper Pascal Editor when this option is chosen. For more information, refer to the section of this manual on 'The Editor'.

Exit to DOS#

Pascal execution is terminated by this option. Control is passed to the Disk Operating System.

List a file#

This convenience entry is provided to allow you to view, on the screen, any text file on disk or tape. You are prompted to enter the name of the file to be listed. The file is assumed to reside on the default drive if a colon (:) is not found within the name you specify. At the end of the list, you will be prompted to press any key to continue. After pressing a key, the Main Menu will appear again.

Trace on#

The wraparound internal trace may be turned on (or off) with this option. The trace is used only for debugging purposes and may be viewed at program termination time by pressing CTRL-T. Program execution speed is slightly degraded while the trace is active. You will be prompted to enter the number of trace entries to be maintained by the system. Each trace entry requires 10 bytes of storage at the high end of memory. The trace may not be used during graphics displays because screen memory is also at the high end of memory. To turn the trace off and remove the memory allocation of the trace table, enter zero when prompted for the number of entries to maintain. The trace format is described in the manual provided to registered users.

The Editor#

The Editor is used to create, modify, and save Pascal source files. It may also be used to process other text type files, like BASIC programs which have been LISTed to disk or tape. It is a line oriented editor. Combined with some type of formatting program, it may be used for word processing applications. The entire source to be edited must be in memory at one time. If your Pascal program will not fit within the limits of the Editor, then you can use the INCLUDE feature of the Compiler to allow segments of a program to be edited separately. Refer to the section on "THE COMPILER" for more information on the INCLUDE feature. Source code for the Editor is listed in the manual provided to registered users. Some key points to be noted about this editor are as follows:

1. Each line is referred to by line number, however, no line numbers are stored either internally or on the disk or tape.
2. Each line may contain up to 80 characters. This may be changed by altering the constant called MAXLENGTH and re-compiling the Editor. A source listing of the Editor is provided to registered users.
3. A maximum of 250 lines of text may be edited at one time. This may be changed by altering the constant called MAXLINES and re-compiling the Editor. An increase in MAXLINES should correspond with a decrease in MAXLENGTH, and vice versa. A source listing of the Editor is provided to registered users.
4. When entering or editing a line, the line must be terminated by pressing the RETURN key.
5. As lines are inserted into, or deleted from, the source file, the remaining lines are automatically renumbered.
6. A line of source may extend onto more than one screen line.
7. Due to operation of the Atari operating system, a blank line may not be directly entered. To enter a blank line, you must first enter a non-blank character (like a period), then use the Editor Change command to change the character to a space.
8. Input operations (Append and Insert) are terminated by entering a null line (just pressing the RETURN key).
9. The BREAK key is disabled by the Editor to prevent loss of data. It is enabled again at termination of the Editor.
10. If you enter or change data then try to Quit or exit to the Compiler without first saving the data onto disk, you will receive an option to either save the data or ignore it and continue.
11. Cassette tape files may be loaded, edited, and saved by the Editor. The Compiler does not support tape input, though. You would first have to load the file from tape, with the Editor, then save it to disk.

Wiki-Editing pending

```
                                EDITOR COMMANDS



                                General Prompts


      The following prompts are general in nature and are common among
      many of the editor commands to be described below.

      Line ->

      You are prompted to enter one line number, as opposed to a range
      of line numbers. It is used by the INSERT Editor command and
      refers to the line before which the inserted line(s) will be
      placed.
```

Line from ->

This is the first prompt for a range of line numbers. Enter the low number of the range. If you just press RETURN, line number 1 is assumed.

Line to   ->

Enter the high line number in the range desired. If only one line is to be acted upon, that number must be entered in both this prompt and the one mentioned above. If you just press RETURN, the highest line number in the buffer will be assumed. If the number you enter is less than the 'Line from' value, the 'Line from' value will be used here.

Enter filename ->

This prompt is shown when loading, appending, and saving files. The last filename used is filled in after the arrow. If this is the file you wish to use now, then all you have to do is press RETURN. A full filespec may be entered, but is not required. If a colon (:) is not found within the filename specified, then the default drive is assumed. If the filename given does not contain a period (.), then a suffix of .PAS is assumed.

The Commands

A - Add line(s) at end

This command is used to add lines after the last line currently in the buffer. If the buffer is currently empty, then line 1 will be assumed as the starting point. In this manner, you can create a new file if one has not been loaded. You can append as many lines as you like. When you are finished entering lines, just press RETURN without entering any data on the line (null line).
Prompts used: None

C - Change line(s)

The Change command allows you to change one specified string pattern to another for the first occurance in each line within the range of lines specified. After being prompted for the line number range, you are asked for the data to 'Change from ->' and 'Change to ->'. Enter any string of characters at each prompt. Imbedded blanks are allowed. If you just press RETURN for the 'Change to' prompt, the first occurance of the 'Change from' data within each line will be deleted.
Prompts used: 'Line from', 'Line to', 'Change from', 'Change to'

D - Delete line(s)

This command allows you to delete a line or a range of lines from the file in memory. The whole file in memory will be deleted if you just press RETURN when prompted for both 'Line from' and 'Line to'. Be aware that all lines following the range deleted will be renumbered, to fill the gap just made. If you desire to delete a number of line ranges, delete those with the highest numbers first and proceed toward the beginning of the file. That way, you won't have to do a LIST after each range delete to find out what the new line numbers for the following lines are.
Prompts used: 'Line from', 'Line to'

E - Edit line(s)

The Edit command is used to edit (or make individual changes to) a line or range of lines that already exist in memory. If a range is specified, the lines are presented to you one at a time. As each line is presented, you may use any of the normal Atari editing keys (like right and left cursor, insert, delete), to alter the data. Just press RETURN when you are finished with each change. If you don't want to make a change to a line shown, just press RETURN.
Prompts used: 'Line from', 'Line to'

F - Filer menu

The Filer is a subsystem which handles communication with an external device (disk or tape). The features provided are as follows:

A - Append file

A file is read from disk or tape and added to the end of the file currently in memory. The data in memory prior to the append remains unchanged.
Prompts used: 'Enter filename'

D - Directory list

This command is used to provide a directory list of the different files on a diskette. You are prompted for 'Filespec?'. Enter the disk drive number and selection criteria for the directory list. If you just press RETURN you will see a directory list of all files on the default drive. To see all files on drive two, enter 'D2:' or 'D2:*.*'. To see only files with a suffix of PAS on drive one, enter 'D1:*.PAS'.
Prompts used: 'Filespec?'

L - Load file

This is the way to load a file into memory from disk or tape. If any data was currently in memory, it is deleted and replaced by the file read in.
Prompts used: 'Enter filename'

S - Save file

Data is copied from memory to disk or tape with this command. The data currently in memory remains unchanged. You are prompted for filename and may use whatever name you wish. It is not necessary to save a file under the same name as was used to load the file. You should save data to disk frequently

if you are making extensive changes. That way you won't have to re-do as much if something goes wrong.
Prompts used: 'Enter filename'

I - Insert before line

This command allows you to insert one or more lines at any point within the file in memory. The inserted data is placed before the line number you specify. To terminate insert mode, just press RETURN without entering any data on the same line (null line). Note that all lines after the point of insertion will automatically be renumbered.
Prompts used: 'Line ->'

L - List line(s)

One or more lines of data from memory are listed on the screen with this command. During the list, you may stop the scrolling by pressing either the space bar or RETURN. To resume scrolling, press any other key other than ESC. The ESC key may be pressed to prematurely terminate the listing.
Prompts used: 'Line from', 'Line to'

M - Menu

The main Editor menu is presented in response to this command. A question mark (?) may also be used to display the main menu.
Prompts used: None

P - Print line(s)

This command is used to create a list of data in memory on a printer attached to the Atari parallel port (P:). Internal line numbers are also directed to the printer although they do not actually exist within the file on disk or tape.
Prompts used: 'Line from', 'Line to'

Q - Quit

This command is used to exit from the Editor when you are finished editing your data. Control is given to the Main Menu program. If you have changed the data in memory and have not saved it prior to quitting, you will be given the option of saving the data or ignoring the changes and exiting. If you are going to compile a Pascal program immediately after quitting the Editor, you may use the 'X' command described below.
Prompts used: None

S - Scan line(s)'

This command allows you to display all lines within a specified range which contain a specified character string. The character string may contain any characters, including imbedded blanks. To temporarily stop the listing, press either the space bar or RETURN. To abort the listing, press ESC. Press any other key to continue as normal.
Prompts used: 'Line from', 'Line to', 'Scan for'

X - Exit to Compiler

This command terminates the Editor and transfers control directly to the Compiler. If the file in memory has been changed but not saved prior to the Exit command, you will be prompted to either save the file or ignore the changes and proceed to the Compiler.
Prompts used: None

The Compiler

The Compiler is used to translate words that we humans understand into "words" that the computer can understand. The computer words are referred to as pseudo-code, or p-code for short. These pseudo-code instructions are understood and executed by the Supervisor.

This is a single pass goal oriented compiler. It expects the proper syntax for a statement. If correct syntax is not found, the compilation stops, and an error number with associated text description is displayed. At this point, you are given the option of quitting or returning to the Editor to correct the problem and do the compile again.

The Compiler itself is written in Draper Pascal and occupies about 28K of RAM memory space.

The first prompt from the Compiler is 'Enter filename:'. The name of the last program edited, run, or compiled is filled in for your convenience. If this is the one you want, just press RETURN. If it is not the one you want, just overtype it with the name you desire. The name you provide will become the new default name for the Editor, Compiler, and Main Menu 'Run' option. No suffix is allowed when specifying filename. The Compiler will add the standard '.PAS' to it for you. If the source does not reside on the default disk drive, then you must prefix the filename with 'Dn:' where 'n' is the disk drive number where the source resides. The default disk drive is normally disk drive number one, but is changed to the Ramdisk drive number if you are taking advantage of the Ramdisk feature of a Disk Operating System that supports it. Ramdisk initialization is explained in the manual provided to registered users.

The next prompt is 'Enter List Output Filespec'. The default (if you just press RETURN) is the screen (E:). The list output may go to any normal output device, such as printer (P:) or disk (D:LISTNAME.PRN).

A number of additional points are mentioned below:

1. Comments are delimited by '(*' on the left end and '*)' on the right end. Any characters may appear within comments. Comments may appear anywhere within the program.

2. 'Include' files are supported. You may have procedures, functions, or any part of a program included in a compile, even though it is not actually part of the file being compiled. It is a variation of a comment which allows you to do this. The format is as follows:

   (*$I XXXXXXXX *) or (*$I D1:XXXXXXXX *)

The dollar sign and 'I' must be right next to '(*' and must be followed by one space. Then you may mention the 'D' for disk and drive number (if other than the default drive is to be used). Follow it with a colon (:) and the filename. A suffix of '.PAS' will be automatically added to the file name. Then have at least one space and '*)'.

3. Pascal source files must reside on disk.

4. The output pseudo-code from the compile will be directed to the same disk drive that the Pascal source resides on. It will be created with a filename suffix of '.PCD'. If you have multiple disk drives and the source and pcode will not both fit on one disk, have a small file on the output disk with an 'include' for the source which resides on the other disk.

5. The hexadecimal offset of the pseudo instructions generated is given at the left side of the output listing. This offset may be useful for debugging purposes. It may be referred to when looking at a program trace (see TRACEON in the Pascal Definitions section of the manual provided to registered users). It also may be referred to in case of an error message or termination caused by pressing the BREAK key. The offset shown may not always be accurate. If not exact, the values are very close.

6. The name and stack offset of each variable defined is shown at the end of the compile listing. The offset value is shown in hexadecimal. Each stack entry is two bytes wide. The first three stack entries are reserved for system use. Therefore, the offset of the first variable will be 0003, which is actually six bytes into the stack. If a variable is defined within a procedure or function, the offset shown is relative the beginning of that procedure or function.

7. The program is ready to run immediately after the compile is finished. No linking is required. (Some Pascal systems require linking of output code after the compile and before execution).

8. Nested procedures are supported. You may define one procedure within another.

9. Recursive procedures are supported. A procedure may call itself. If variables are defined within the procedure, they

are cleared with each entry into the procedure and refreshed upon exit from the recursive procedure call.

10. No forward references are allowed. A procedure may not be referenced before it is defined. In most cases, nesting the procedures will take care of this problem.

11. Double density disk drives are supported for both source and pcode files. The pcode will be written to the same drive that the initial source is taken from.

12. Only integer type parameters may be passed to procedures and functions. Other types of data may be passed by using global type variables setup at the beginning of the program (not within a procedure or function).

13. A function may only return an integer type value. Procedures do not return values.

14. Hexadecimal constants and literals are prefixed by dollar signs ($).

15. To write out an integer in hexadecimal format, precede the variable name with a percent sign (%).

16. A total of 170 compiler table entries may be used. One table entry is used for each variable definition, procedure name, function name, and parameter name used with procedures and functions. Table entries for variables defined within procedures are re-used following the 'END' for that procedure. The number of table entries used within a compile is displayed at the end of the output list from the Compiler.

17. The time needed to compile a program can be reduced by turning off the ANTIC chip within the computer. This turns off the display to the screen yet gives a fairly significant increase to the Atari's internal speed. In a normal Pascal program, you can have POKE(559,0) to turn it off and POKE(559,34) to turn it back on. But a special compile time option is provided to make use of this feature to speed up compiles. It is as follows. Have a statement (*$S+*) to turn the ANTIC off (increase speed), and use (*$S-*) to turn the ANTIC on (resume normal speed). These options may appear anywhere within a program. The ANTIC is automatically turned back on at compile termination and at time of error (if any).

The Supervisor

The Supervisor is a high performance machine language program which simulates a pseudo 16-bit stack oriented computer. It executes the pseudo code that is generated by the Compiler.

It is loaded into memory by disk operating system at the hex location $1D7C, which is just above DOS in memory. It should work with any DOS that allows a program to load at that address, such as Atari DOS 2.1S, Atari DOS 2.5, or SpartaDOS version 2.x or higher. A message will be displayed if the Supervisor cannot be loaded at the proper location.

The disk filename for the Supervisor's object code is 'AUTORUN.SYS'. It may be renamed to anything you desire, such as 'PASCAL.COM', but will not be automatically loaded when the disk is booted if the name is other than 'AUTORUN.SYS'. To start the Pascal system from the DOS menu, use the 'L', binary load, option to load 'AUTORUN.SYS' into memory. Execution will begin automatically.

The Supervisor begins execution by loading and executing the Pascal program 'INIT.PCD' from the default drive, which is always disk drive 1 immediately after loading the Supervisor. 'INIT.PCD' is the name of the main menu program. You may substitute any compiled Pascal program of your own by naming it 'INIT.PCD'. In this manner, you can have a true turnkey system where your program begins execution after booting the disk.

After termination of each Pascal program, the Supervisor gives you a choice of what to do next. You are prompted with the following line:

<START>Repeat,<SELECT>Menu,<ESC>Exit

If you press the START key, your Pascal program will execute again from the beginning. If you press the SELECT key, control will be transfered to the main menu program (INIT.PCD). If you press the ESC key, you will exit to the DOS utility menu. You also have two other options at this point. They are both used for debugging purposes. If you press CTRL-S (the 'S' key while holding down the CTRL key), the stack values, at termination time, will be displayed. If you press CTRL-T, the internal trace table, if active, will be displayed. With either of these two debugging options, you will be asked where the display should be sent by the prompt 'WHERE? (FILESPEC)'. To see it on the screen, enter 'E:'. It also may be sent to printer or disk by following normal filespec naming conventions. If the display is sent to the screen, you may stop the scrolling by use of the space bar. Press the ESC key if you have seen enough and wish to return to the Supervisor termination prompt. Any other key causes scrolling to continue as normal.

Pascal Definitions

Syntax is shown below for each of the Draper Pascal reserved words. The complete manual provided to registered users, however, contains COMPLETE definitions, COMPLETE sample programs demonstrating the use of each reserved word, as well as BASIC equivalents.

ABS	FUNCTION ABS(Number):INTEGER;
ADDR	FUNCTION ADDR(Var):INTEGER;
AND	
ARCTAN	FUNCTION ARCTAN(Var):REAL;
ARRAY	ARRAY[Number1] OF Type ARRAY[Number1,Number2] OF Type
ASC	FUNCTION ASC(Cvar):INTEGER;
BEGIN	
BLOAD	PROCEDURE BLOAD(Program);

```
BOOLEAN

CALL                PROCEDURE CALL(Address);

CASE                CASE expr1 OF const1 : stmt1;
                               const2 : stmt2;
                               ...
                               constn : stmtn
                    END;

                    CASE expr1 OF const1 : stmt1;
                               const2 : stmt2;
                               ...
                               constn : stmtn
                    ELSE stmtx
                    END;


CHAR

CHR                 FUNCTION CHR(expr1):CHAR;

CLOSE               PROCEDURE CLOSE(File);

COLOR               PROCEDURE COLOR(Number);
```



```
CONCAT              PROCEDURE CONCAT(Parm1,Parm2,...):STRING;

CONST               CONST name1=value1; name2=value2; ...

COPY                FUNCTION COPY(Source,Index,Length) : STRING;

COS                 FUNCTION COS(Var):REAL;

CVTREAL             FUNCTION CVTREAL(Ivar):REAL

DEG                 PROCEDURE DEG;

DELETE              PROCEDURE DELETE(Source,Index,Size);

DIV

DOS                 PROCEDURE DOS;
```

```
DRAWTO              PROCEDURE DRAWTO(X,Y);

DUMPSTK             PROCEDURE DUMPSTK;

DVSTAT              PROCEDURE DVSTAT(A,B,C,D);

END

EOF                 EOF(File);

EOLN                EOLN(File);

EXIT                PROCEDURE EXIT;

EXP                 FUNCTION EXP(Var):REAL;

EXP10               FUNCTION EXP10(Var):REAL;

FALSE

FILE

FOR                 FOR var := expr1 TO expr2 DO statement;
                    FOR var := expr1 DOWNTO expr2 DO statement;

FUNCTION

GOTOXY              PROCEDURE GOTOXY(X,Y);

GRAPHICS            PROCEDURE GRAPHICS(Number);

HIMEM               PROCEDURE HIMEM(Value);
```



```
IF                  IF expr1 THEN stmt1;
                    IF expr1 THEN stmt1 ELSE stmt2;

INSERT              PROCEDURE INSERT(Source,Destination,Index);

INTEGER

IORESULT            FUNCTION IORESULT:INTEGER;

KEYPRESS            FUNCTION KEYPRESS:INTEGER;
```

```
LENGTH            FUNCTION LENGTH(svar):INTEGER;

LN                FUNCTION LN(Var):REAL;

LOCATE            FUNCTION LOCATE(X,Y):INTEGER;

LOCK              PROCEDURE LOCK(Filename);

LOG               FUNCTION LOG(Var):REAL;

LPENH, LPENV      FUNCTION LPENH:INTEGER;
                  FUNCTION LPENV:INTEGER;

MAXGRAPH          PROCEDURE MAXGRAPH(Mode);

MOD

NOT

NOTE              PROCEDURE NOTE(Iocbno,Sector,Byte);

ODD               FUNCTION ODD(iexp);

OPEN              PROCEDURE OPEN(Fileno,Aux1,Aux2,Filename);;

OPTIONKEY

OPTIONS           OPTIONS(Opt1,Opt2,...,Optn);

OR

ORD               FUNCTION ORD(Realvar):INTEGER;

PADDLE            FUNCTION PADDLE(Number):INTEGER;

PEEK              FUNCTION PEEK(Address):INTEGER;

PLOT              PROCEDURE PLOT(X,Y);

POINT             PROCEDURE POINT(Iocbno,Sector,Byte);
```



```
POKE              PROCEDURE POKE(Address,Value);

POS               FUNCTION POS(Pattern,Source):INTEGER;
```

```
PROCEDURE          PROCEDURE Name;
                   PROCEDURE Name(Parm1,Parm2,...,Parmn);

PROGRAM            PROGRAM Name;

PTRIG              FUNCTION PTRIG(Number):INTEGER;

PURGE              PROCEDURE PURGE(Filespec);

RAD

READ, READLN       PROCEDURE READ(File,Var1,Var2,...Varn);

REAL

RECORD

REPEAT             REPEAT Stmt1; ... ;Stmtn UNTIL Condition;

RESET              PROCEDURE RESET(File,Filespec);

REWRITE            PROCEDURE REWRITE(File,Filespec);

RND                FUNCTION RND(Iexp):INTEGER;

SELECTKEY

SETCOLOR           PROCEDURE SETCOLOR(Register,Hue,Luminance);

SHL                Expr1 SHL Expr2

SHR                Expr1 SHR Expr2

SIN                FUNCTION SIN(Var):REAL;

SOUND              PROCEDURE SOUND(Voice,Pitch,Distortion,Volume);

SQR                FUNCTION SQR(Var):REAL;

SQRT               FUNCTION SQRT(Var):REAL;

STARTKEY

STATUS             PROCEDURE STATUS(Iocbno,Ivar);

STICK              FUNCTION STICK(Number):INTEGER;
```

```
STR                 FUNCTION STR(Var):STRING;

STRIG               FUNCTION STRIG(Number):INTEGER;

STRING

TRACEOFF            PROCEDURE TRACEOFF;

TRACEON             PROCEDURE TRACEON;
                    PROCEDURE TRACEON('Number);

TRUE

UNLOCK              PROCEDURE UNLOCK(Filespec);

VAL                 FUNCTION VAL(Svar):INTEGER or REAL;

VAR                 VAR Name1,Name2,...,Namen : Type;
                    VAR Name1,Name2,...,Namen : ARRAY[Number]
                        OF Type;

WAIT                PROCEDURE WAIT(Number);

WHILE               WHILE Condition DO Statement;

WRITE               WRITE(File,Expr1,Expr2,...);
                    WRITE(File,Expr1:Fldwdth...);
                    WRITE(File,Expr1:Fldwdth:Numdec...);

WRITELN             WRITELN(File,Expr1,Expr2,...);
                    WRITELN(File,Expr1:Fldwdth...);
                    WRITELN(File,Expr1:Fldwdth:Numdec...);

XCTL                PROCEDURE XCTL(Filespec);

XIO                 PROCEDURE XIO(Number,File,Aux1,Aux2,Filespec);
```

System Information

The Supervisor uses zero page locations $A0 - $BF. Locations $80 - $9F are available for your use if desired. Various locations between $D4 and $FD are used by the floating point routines. Page six ($600 - $6FF) is available for your use and not used by the Pascal system.

The Supervisor is loaded into memory by DOS at the address $1D7C. If this memory location is not available, then an error message is given, along with an explanation of the probable cause of the problem. The pseudo code program to be executed is loaded in memory immediately after the end of the Supervisor. The pseudo machine stack extends from the end of the pseudo code program to the MEMTOP position, just before screen memory.

Filename Descriptions

The files named below are included in this ARC file:

AUTORUN.SYS	Supervisor object code
COMPILER.PCD	Compiler pcode
EDITOR.PCD	Editor pcode
INIT.PCD	Main Menu pcode
INIT.PAS	Main Menu Pascal source
EXPLNERR.PCD	Error code explainor (used by Compiler)
RSVDWRDS.TXT	Reserved word list (used by Compiler)
ERRORS.TXT	Text for compile errors (used by EXPLNERR.PCD)
RAMDISK1.DAT	Ramdisk setup (See complete manual)
RAMDISK2.DAT	Ramdisk setup (See complete manual)
RAMDISK3.DAT	Ramdisk setup (See complete manual)
NOTITLE.OBJ	Used to suppress title (See complete manual)
SAMPLE1.PAS	Kaleidoscope sample program source
SAMPLE2.PAS	Roman numeral sample program source
PASCAL.DOC	This introduction manual

Reserved Word List

ABS	DEG	FUNCTION	ODD	READLN	STRING
ADDR	DELETE	GOTOXY	OF	REAL	THEN
AND	DIV	GRAPHICS	OPEN	RECORD	TO
ARCTAN	DO	HIMEM	OPTIONKEY	REPEAT	TRACEOFF
ARRAY	DOS	IF	OPTIONS	RESET	TRACEON
ASC	DOWNTO	INSERT	OR	REWRITE	TRUE
BEGIN	DRAWTO	INTEGER	ORD	RND	UNLOCK
BLOAD	DUMPSTK	IORESULT	PADDLE	SELECTKEY	UNTIL
BOOLEAN	DVSTAT	KEYPRESS	PEEK	SETCOLOR	VAL
CALL	ELSE	LENGTH	PLOT	SHL	VAR
CASE	END	LN	POINT	SHR	WAIT
CHAR	EOF	LOCATE	POKE	SIN	WHILE
CHR	EOLN	LOCK	POS	SOUND	WRITE
CLOSE	EXIT	LOG	PROC	SQR	WRITELN
COLOR	EXP	LPENH	PROCEDURE	SQRT	XCTL
CONCAT	EXP10	LPENV	PROGRAM	STARTKEY	XIO
CONST	FALSE	MAXGRAPH	PTRIG	STATUS	
COPY	FILE	MOD	PURGE	STICK	
COS	FOR	NOT	RAD	STR	
CVTREAL	FUNC	NOTE	READ	STRIG	

Operators

Operator	Operation
:=	assignment
arithmetic:	
+	addition
-	subtraction
*	multiplication
/ or DIV	division
MOD	modulo (remainder after division)
Relational:	
=	equality
<>	inequality
<	less than
>	greater than
<=	less than or equal to
>=	greater than or equal to
Logical:	
OR	
AND	
NOT	

Error Messages

COMPILE TIME ERROR MESSAGES

01: Compiler table overflow (max 170)
02: Number expected
03: '=' expected
04: Identifier expected
05: Constant type identifier, number, or string constant expected
06: 'BEGIN' expected
07: Too many nesting levels
08: ':' expected
09: '.' expected
10: ';' expected
11: Undeclared identifier
12: Invalid type of identifier
13: ':=' expected
14: 'END' expected
15: ';', 'ELSE', or 'END' expected
16: 'THEN' expected
17: '#' expected
18: 'DO' expected
19: '#' or FILE type identifier expected
20: '[' expected
21: ']' expected
22: ')' expected
23: Illegal factor or identifier type
24: INCLUDE file nesting too deep

25:
26: 'OF' expected
27: Mismatched data types
28: 'TO' or 'DOWNTO' expected
29: 'UNTIL' expected
30: Range error
31: '(' expected
32: ',' expected
33: Literal too long or missing end quote (')
34: 'END' but no RECORD started
35: Incorrect number of parameters
36: INTEGER type identifier expected
37: STRING type identifier expected
38: REAL type identifier expected
39: CHAR type identifier expected
40: FILE type identifier expected
41: HEX type identifier expected
42: STRING constant expected

EXECUTION TIME ERROR MESSAGES

INDEX TOO HIGH

This message occurs if an attempt is made to store a string array element into an occurance that is higher than defined for the variable. For example, if you tried to store the twentieth entry of an array that was only defined to hold ten occurances, you would get the message. This message only applies to string arrays since other array types are not checked for valid occurance numbers.

UNABLE TO OPEN DEBUG IOCB (7)

This message is issued if the list output device you specify in response to the 'WHERE? (FILESPEC)' prompt cannot be opened. The

prompt is issued only for the debug features trace and stack display.

CIO ERROR xxx FOR IOCB # y

Some kind of Input-Output operation was performed which resulted in an abnormal return code from the Atari operating system. Refer to your BASIC or DOS manual for the meaning of the error number 'xxx'. 'y' is the IOCB number which the error occured on. Note that this message will not be printed if OPTIONS(0) is in effect. In this case it is your responsibility to check the return code by interrogating IORESULT after each I/O type instruction.

AT OFFSET

This message accompanies some other error message and refers to the offset within the pseudo code of the instruction that had the error. Refer to the offset shown on your compile listing to determine the Pascal instruction that experienced the error.

STOPPED BY <BREAK> KEY

This message indicates that execution of the program was stopped because the BREAK key was pressed. The offset of the instruction executing is shown in the 'AT OFFSET' message. Note that this message will not occur (and the program will not stop after BREAK is pressed) if OPTIONS(4) is in effect.

INSUFFICIENT MEMORY

This message indicates that an attempt was

made to increase the value of the stack pointer to a value which would overlay screen memory or the trace buffer, if the trace was active. It may also be caused by manipulation of a record without sufficient room between the top of the stack and the top of available memory (MEMTOP) to temporarily hold it.

INVALID OPCODE

This message should not occur. It indicates that a pseudo instruction was encountered which is invalid. If you get this message, it means that your '.PCD' file has been corrupted somehow or an XCTL was made to a file that was not a pseudo code file. To correct, re-compile the program in question. It may also occur if you attempt to run a Draper Pascal program which was compiled under a previous release of this software.

Registration Form

When you register for a fee of only $15.00, you will receive a copy of the latest version of the Draper Pascal manual. Also, if a later version of the software is more current than the version you are registering, you will receive a diskette containing it. In addition to all relative information in this manual, the

complete manual provided to registered users contains the following:

* Complete Draper Pascal reserved word definitions

* Complete Draper Pascal example programs demonstrating the use of each reserved word

* BASIC equivalents for each reserved word, where applicable

* Information on setup and use of a Ramdisk for Draper Pascal. After proper initialization, the Editor will take less than two seconds to load and the Compiler will take less than three seconds. Also, the default drive will be automatically set to the Ramdisk.

* A royalty-free, non-exclusive license to allow you to distribute software developed using Draper Pascal

* Complete Draper Pascal source listings for the Editor, Main Menu program, Ramdisk initialization program, and sample programs

* Editor command summary

* Information on how to suppress the title screen from being displayed

* Trace format descriptions

* Internal data format descriptions

* Notes on printer usage with Draper Pascal

* Technical support, by mail, at no charge

If you purchased Draper Pascal directly from Draper Software in your own name, then your copy is already registered and you will recieve all the benefits of registration. You do not need to send in a registration form.

If you received Draper Pascal some other way, you may register your copy by filling out the following form and mailing it to the listed address along with your check or money order.

==========

Draper Pascal 2.1 Registration Form

Mail payment to: Draper Software
307 Forest Grove Drive
Richardson, TX 75080-1939

Texas residents: Please add 8% tax.

Note: Please allow three to five weeks for delivery.

NAME _____

COMPANY (if any) _____

ADDRESS _____

CITY/STATE _____

ZIP _____

Where did you obtain this copy of Draper Pascal?

_____

_____

_____

==========