

Quick Pascal

0.9

An OS-9 Pascal Application Programming Interface for the Tandy Color Computer 3

by

Tony Schountz
schountz@yahoo.com
©1998, 2005

What you need: NitrOS-9/OS-9 Level II
 512k RAM (minimum)
 OS-9 Pascal v.r. 02.00.00
 Hard disk (virtual or real)
 A Text Editor
 OS-9 Level I Assembler
 Printer

0. Disclaimer

This beta version of Quick Pascal and its contents may be freely distributed, so long as all source code, files and programs remain unchanged. Distribute the original archive only, not individual pieces. If you wish to modify the programs and files to suite your needs, feel free to do so. But do not give modified copies to others, as this will hamper my ability to recreate and correct runtime errors and provide support. **USE AT YOUR OWN RISK.** I am not responsible for any damage caused by this software package. If you use this product on a routine basis, for either personal or commercial purposes, I request that you make a small donation (money, time, whatever) to a charitable organization of your choice.

1. Introduction

OS-9 Pascal from Microware is a powerful programming suite. Compared to Basic09, what it lacks in ease of use and graphics capabilities is compensated by its speed. Typical OS-9 Pascal programs execute up to 10 times faster than Basic09 counterparts. The enclosed *Sieve of Erasthenes* benchmark runs 9 times faster as assembled Pascal (31 seconds) compared to the Basic09 version (over 4 minutes). On my iMac G5/2 GHz under MacMESS/NitrOS-9, it runs in 12 seconds.

However, Pascal adheres to strict programming rules, and requires diligent attention to programming detail. Those who have used C can attest to its forgiveness: Many programs will compile successfully, yet fail to do the expected job! Debugging C can be quite a chore. So there will be times when OS-9 Pascal will be a better choice for programming, particularly if you want to learn a highly portable and contemporary structured programming language. In addition, with the use of `PASCALS`, PCODE programs over 8 Mb can be run on the Color Computer 3. Quick Pascal will makes OS-9 Pascal much easier to use.

Pascal is commonly used to teach programming fundamentals throughout the world, and is used in private industry as well. OS-9 Pascal strictly adheres to ISO standard (7185.1 Level 0), and therefore will port easily to other ISO-compliant Pascal systems (with noted exceptions contained within the OS-9 Pascal Manual).

Quick Pascal is a menu driven **application programming interface** (API) which, when paired with Microware's OS-9 Pascal, forms a fluid integrated development environment (IDE). OS-9 Pascal was first released under OS-9 Level I and does not work with Level II as packaged. Radio Shack never released an updated version; however, in this distribution is a patched version of 'pascal' native compiler that will run properly under Level II and exploit the use of a hard disk drive (virtual or real). **It is hard-coded to look for /DD.**

Quick Pascal is not so much a program as it is an environment; think of it as a "toolbox" for OS-9 Pascal. Contained within this toolbox are several programs, utilities and files needed to help OS-9 Pascal function properly and efficiently under Level II. Because some of these programs are Basic09 I-code, 'RUNB' must also be in memory, along with the 'gfx2' graphics support module.

You must use shell+ 2.2a, which is included on the boot disk and virtual hard disk downloads.

2. Creating a Pascal Application

There are six steps to building an application with OS-9 Pascal:

1. Create a source code file using a text editor.
2. Compile the source code into a PCODE file using the Pascal compiler.
3. Debug any errors in the source code with a text editor.
4. Recompile until no errors are detected.
5. Translate the PCODE file into an assembly language source code using the PascalT translators.
6. Assemble the source code using the Level I assembler.

Quick Pascal combines all of these steps into a single, easy to use program, by exploiting 512k of RAM and the availability of a hard disk for all file I/O.

3. Disk Structure

Your Pascal working disk must have two directories in the root, CMDS and PASCAL.

3.1 CMDS directory. The CMDS (execution) directory must include the following programs:

```
Pascal PascalN PascalS PascalT.PRUN PascalT.MODL Pascal_Compiler
PascalErrs Support strip type asm
```

Also, if not already in memory, the utilities:

```
copy dir del list rename tmode
```

...should also be in the CMDS directory, as these are used by Quick Pascal for file management. See Appendix I for the contents of my CMDS directory.

3.2 PASCAL directory. The /DD/PASCAL directory must contain the following files:

```
qp.init qp.startup qp.shutdown pascal_words PascalDefs
```

`qp.init` is an initialization file of two lines that contains the (1) name of your text editor and (2) name of your hard disk (e.g., /dd, /h0). `qp.startup` is simply a procedure file that is executed by `qp` when it is first launched. At a minimum, this file should load all the relevant programs into memory. `qp.shutdown` is a procedure file that is executed upon termination of Quick Pascal. `Pascal_words` is a file used to by the program 'convert' to make all Pascal reserved words capitalized. Finally, `PascalDefs` is the definitions file that comes with OS-9 Pascal. See Appendix I for the contents of my PASCAL directory.

4. Starting Quick Pascal

Prior to launching Quick Pascal, change the working directory to that which contains the source code for your files. If starting a new project, create a directory for it, then change to it.

The main Quick Pascal (`qp`) program resides in your current execution directory. It contains several procedures: `qp`, `convert`, `smallcase`, `compile`, `tool`, and `pasc_errors`. Type `qp` at a shell prompt to launch Quick Pascal.

The first thing Quick Pascal does is open the file called `qp.init`. This file contains two lines. The first is the name of your text editor. The name cannot be greater than 15 characters in length and the editor must reside in the `CMDS` directory on your Pascal working disk or in memory. In the included `qp.init` file, the first line is `'ds'` because I use DynaStar. You can also allow parameters to be included, such as memory allocation. If you use the `'edit'` command as your editor, you might want to have the first line in your `qp.init` file to be:

```
edit #24k
```

When you edit your source code, you are effectively invoking `'edit #24k program.p'` with 24k of memory allocated to `'edit'`.

The second line contains the name of your hard disk. On my system it is `'/DD'`. This is where all your OS-9 Pascal files will be written and read.

(I've included my DynaStar initialization file [`DS.Init`] with the archive. It's set up for auto-indent, four space tab, help screen off, arrow keys enabled, and immediate edit mode. `<CTRL>-N` still gets you back to the main menu. If you use DynaStar, give it a try, if you like it, use it; if not get rid of it.)

5. Running Quick Pascal

The `qp.startup` procedure file can contain many lines. In this file you can set your printer's margins or character size (17 cpi is nice) by using `'display xx xx >/p'` command (see your printer manual for the hexadecimal control codes to set these attributes). Your creativity is the limit for configuring your system.

`Qp.startup` should also load `Pascal`, `PascalN`, `PascalS`, `PascalT.MODL`, `Support`, the Level I assembler (`asm`) as well as your editor.

After the successful execution of `qp.startup`, Quick Pascal opens the file `'pascalerrs'` in the `CMDS` directory and reads all the pascal error codes and descriptions. This is then used when compiling your source code. No more stopping to

look up error messages; they will be displayed on your screen whenever a compile error occurs.

Once `qp.startup` is complete a directory listing is provided. Next, you will be asked to enter the name of your source code. This is the name of the source file you want to edit, compile, translate, and/or assemble. If you type `'myprogram'`, Quick Pascal will convert it to `'myprogram.p'`. If you type `'myprogram.p'`, it remains `'myprogram.p'`. For now, name the file `'test.p'` and press `<ENTER>`. You'll get an alert informing you that `'test.p'` doesn't exist and asking if you want to create it. Press `'y'`.

NOTE: If this is the first time you are using Quick Pascal, take a few seconds to switch to another window with a shell running in it and do an `'mdir'` to make sure all the modules loaded. Doing so might save you some trouble later on.

After entering the filename, you'll be prompted with a menu of selections from 1 to 9.

1. Edit Source Code
2. Compile Source Code
3. Run Program
4. Translate Program
5. Assemble Program

6. Change Working File
7. Get Directory Listing
8. Toolbox

9. Quit

Selection:

The first five options deal directly with the OS-9 Pascal package. First you must edit your source code. By pressing `'1'` you will be put into your editor with the file that you are generating. Type following few lines of code, then save and exit your editor.

```
program test;

{ Simple example of OS-9 Pascal and Quick Pascal }
{ Comments should be enclosed in braces          }

var count, sum, square, cube, quad: integer;

begin
  sum:=0;
  square:=0;
  cube:=0;
  quad:=0;

  for count:=1 to 12 do
    begin
      sum:= sum+count;
      square:=count*count;
      cube:=square*count;
```

```

        quad:=square*square;
        a_bad_line

        writeln (count, sum, square, cube, quad);
    end; {for}
end. {test}

```

After returning to the main menu, select option '2' to compile the code. After several seconds, you'll be presented with the compilation results, which will contain an error when it comes to 'a_bad_line'. The error is flagged by an error line, followed by the error messages for the error codes. Press <ENTER> and you'll move on through the compilation results. After returning to the main menu again, select your edit function and delete the erroneous line, so that you now have:

```

program test;

{ Simple example of OS-9 Pascal and Quick Pascal  }
{ Comments should be enclosed in braces          }

var count, sum, square, cube, quad: integer;

begin
    sum:=0;
    square:=0;
    cube:=0;
    quad:=0;

    for count:=1 to 12 do
        begin
            sum:= sum+count;
            square:=count*count;
            cube:=square*count;
            quad:=square*square;

            writeln (count, sum, square, cube, quad);
        end; {for}
    end. {test}

```

Exit your editor and press '2' again. This time there should be no errors and you will have successfully compiled the source code to pcode. By pressing '7' at the main menu you can get a directory listing. There should be three files that begin with 'test', the source code file 'test.p', the pcode file 'test.pcode', and the tabulation file 'test.tab'. If no errors were encountered, press '3' at the main menu and Quick Pascal will call 'pascaln' to execute 'test.pcode'. (In the ensuing output, you'll see the variable names aren't as they appear!)

NOTE: If you have a very large program, you will have to use the swapping pcode interpreter 'pascals' instead of 'pascaln', either in another window, or use the 'shell' command in the Toolbox. See the OS-9 Pascal manual for information as to when to use pascals.

Tangent: In Pascal, convention is that reserved words are capitalized, while variables, constants and all other non-keywords are in lower case. Quick Pascal will do this for you automatically. Press '8' at the main menu. The toolbox menu will appear with nine more selections. Press '1' to get a listing of the file 'test.p' on the screen. All the characters are in lower case. Get back to the toolbox menu and press '8'. Answer 'y' and in a few seconds you'll be returned to the toolbox menu. Press '1' again, and all the Pascal reserve words in the file 'test.p' should now be capitalized, while all others are in lower case. Results follow.

```
PROGRAM test;

{ Simple example of OS-9 Pascal and Quick Pascal }
{ Comments should be enclosed in braces          }

VAR count, sum, square, cube, quad: INTEGER;

BEGIN
    sum:=0;
    square:=0;
    cube:=0;
    quad:=0;

    FOR count:=1 TO 12 DO
        BEGIN
            sum:= sum+count;
            square:=count*count;
            cube:=square*count;
            quad:=square*square;

            WRITELN (count, sum, square, cube, quad);
        END; {for}
    END. {test}
```

Notice that the 'for' enclosed in braces is not capitalized. This is because 'convert' ignores all comments within braces, as well as all strings enclosed in single quotes. This program is of greatest utility for finding when you have tried to declare a Pascal reserved word as a variable, constant or other structure. It also helps to identify typographical errors with Pascal statements.

The rest of the toolbox selections are self-explanatory. Selection '7' is the shell access. Press this and you can do one of two things. You can type in any legal shell line up to 130 characters and it will be executed, or you can press <ENTER> and you'll be in an immortal shell. To exit, type 'ex' or press <ESC>, and you'll be returned to the toolbox menu.

Getting back to the main menu, selection '4' will permit you to translate your pcode file into an assembly language file. Press '4' and on the screen will be printed three lines: the name of your pcode file, the name of the assembly file that is to be created, and the module name for the program. For our program, these will be 'test.pcode', 'test.a', and 'test', respectively. Next an overlay window will open and the Pascal translator is called. After answering in accordance to the OS-9 Pascal manual you will

have a file called 'test.a' if you enter the names as listed above the overlay window. This is an assembly language file that can be assembled by the Level I assembler 'asm'. (To my knowledge, the relocatable macro assembler [rma] supplied on the development disk does not work with Pascal translated files.) By selecting '5', Quick Pascal invokes 'asm' to work. After completion, you should have a program called 'test' in your CMDS directory on your Pascal working disk. Remember, Pascal assembled programs still need the runtime support module in memory or in the current execution directory.

6. Pascal Graphics Procedures

The procedures contained within `pgfx.p` file inside the PASCAL directory utilize the OS-9 Level II drawing primitives found in the section of the Level II manual entitled "Windows". Each procedure expects integer parameters. I have not fully tested all the procedures, but the ones I did test worked fine. Although OS-9 Pascal runs many times faster than Basic09 counterparts, I don't know if this also applies to these graphics procedures. I'll explore this in future releases.

NOTE: Please exercise caution when writing graphics code. If you try to draw on a hardware screen you'll run into trouble real fast, possibly crashing your system.

I have most of the graphics capabilities of the operating system defined; however some are not included. This is simply because I'm not too sure of what I'm doing when it comes to buffers and groups (I'm a much better microbiologist than I am a programmer :-). I hope someone will come up with these procedures and donate them to the public domain. The ones not included are:

```
getblk  gpload  putblk  arc3p
```

I also hope someone will write routines for accessing the mouse port (both low-res and high-res) through Pascal. I've also included a source code for a graphics example. When you run it, you'll see it's not too sophisticated. You can also get additional information for these routines from the Basic09 high resolution graphics functions found in the Level II manual.

7. Considerations

I've tried to include as much flexibility as possible with the program, but I'm sure there are better things that can be done. I welcome any comments or suggestions for improvements. I hope this program is as valuable for others as it is for me.

- It's good practice to make frequent saves of your files to your floppy or hard disks.
- Pascal is a great language to learn. Visit Amazon.Com, your local bookstore, or your library for suitable self-teach books. Look for books that use ISO or USCD compliant code. Avoid books on Turbo Pascal.

- For updates, visit the OS-9 Pascal Resource Page at <http://www.unco.edu/schountz/pascal/>

8. Epilogue

Any questions, comments, bug reports or suggestions, please send email to me (schountz@yahoo.com). Also, I visit bit.listserv.coco on a regular basis, and that forum will probably help others who might be having similar problems. Please post there if your questions or comments are of benefit to all users. Have fun!

Tony Schountz
August 12, 2005
Greeley, Colorado

Appendix I - Contents of my CMDS and PASCAL directories. Here's what my current working disk looks like:

```
Directory of /dd/CMDS 21:17:46
PASCALMODS      alias      asm        attr       cmp
date            deldir    dirm       dmode      ds
dsort          dump      edit       findprimes format
free           ident     makdir     modpatch   pascal
pascale        pascalN  pascals   pascalt.modl pascalt.prun
pascal_compiler pascalerrs pwd        pxd        reboot
rename         save     sievep    sled       strip
support        type     verify    wcreate    xmode
```

```
Directory of /d1/PASCAL 21:03:16
ds.init          PascalDefs  qp.startup  qp.shutdown  qp.init
pascal_words
```

Appendix II - qp.init, qp.startup, qp.shutdown file contents. Comments follow <-

qp.init

```
ds <- DynaStar, my screen editor
/r0 <- The name of my RAM disk descriptor
```

qp.startup

```
-x <- Keep going, even if error detected
t <- Echo to the screen
copy cmds/pascal_compiler /r0/PasCompiler
copy cmds/pascalt.prun /r0/pascalt.prun
copy pascal/pascaldefs /r0/PascalDefs
copy pascal/ds.init /r0/DS.Init <- My DynaStar init file
load pasmods <- Merged file containing pascal and pascaln
load support <- Pascal runtime module
load swap <- Merged file containing pascals and pascalt.modl
load ds <- DynaStar, my screen editor
load asm <- Level I assembler
```

qp.shutdown

```
-x
t
unlink pascal
del /r0/pascalt.prun
del /r0/pascompiler
del /r0/pascalDefs
del /r0/ds.init
del /r0/p_errs
unlink ds
unlink pascals
unlink support
unlink asm
```