ProSel-16™

Hard disk management system

by Glen E. Bredon

ProSel-16™

The first, best and most complete hard disk management system for the Apple IIgs.

NOTICE

ProSel-16, ProSel-8 and all their support and utility programs are under the

ProSel-16, ProSel-8 and their support programs are not “shareware’.

The source listings of these programs run to over 90,000 lines of code and fill binders over six inches thick. If you want to support the great effort that goes into creation of such a software package, you will respect the author’s right to a fair return on such a major project. It is fine to show the programs to others, but to give them away is illegal and immoral if not fattening.

Glen E. Bredon
521 State Road
Princeton, NJ 08540
(609) 924-5976

Contents

Part I. Introduction

Part ii. Program Selector

Part III. Disk Management

Part IV. Shell

Part V. Appointment Calendar

Part VI. Text Editor

Part VII. Number Cruncher

Part VIII. Calculator Tutorial

Appendices & Index

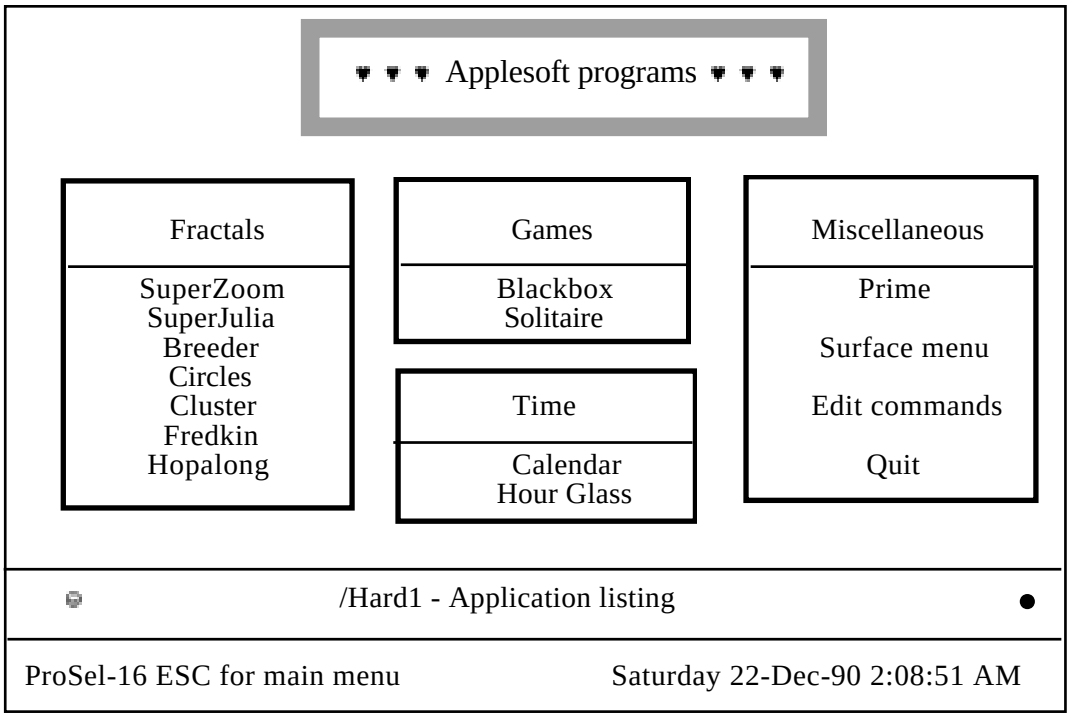


Figure 1. Rendering of a typical ProSel-16 screen.

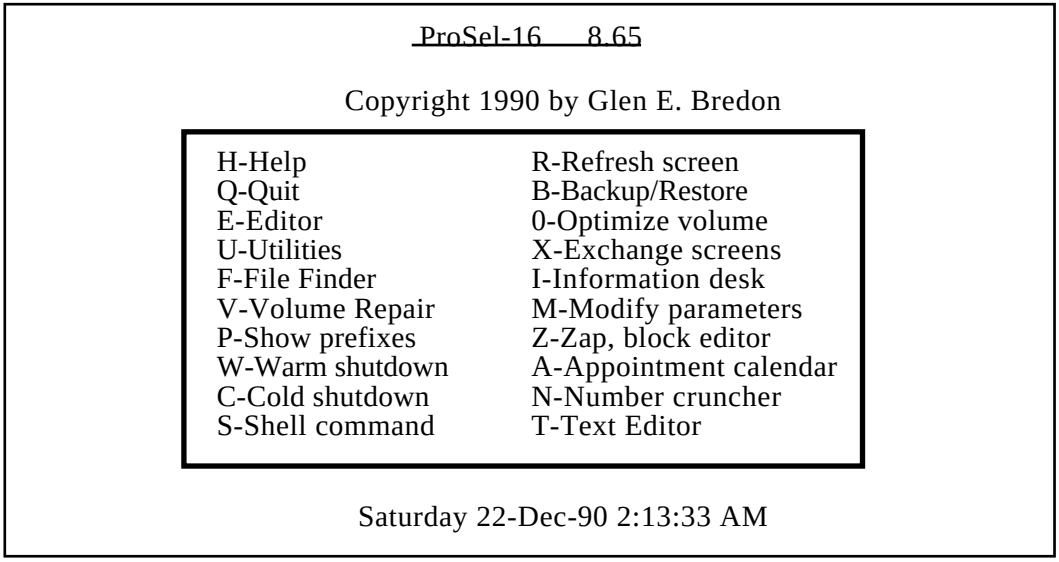


Figure 2. The Main Menu.

Part I. Introduction

☼ Overview

ProSel-16 consists of five main files:

> START, the main program and utilities, a very large file.

> PROSEL.SPECS, information about application programs that appear on the main screen of ProSeI-16, variable size, at most 17 blocks.

> PROSEL.PARMS, a file containing user modifiable parameters, 7 blocks long.

> PS. 16.TO.8, a file for use in running P8 programs, 3 blocks long.

> Appoint.CDA, appointment calendar support, 5 blocks long.

> The START file is meant to be in the SYSTEM subdirectory of the boot disk.

PROSEL.SPECS should be in the main directory of the boot disk. It, and copies of variations of it, can be on the main directory of non-boot disks, but it must be on some disk accessible at boot time.

PROSEL.PARMS should be in the same directory holding START, usually the SYSTEM subdirectory of the boot disk.

PS. 16.TO.8 should be in the SYSTEM subdirectory of the boot disk.

When you boot, the ProSel- 16 screen will appear. It holds titles of your application programs, in a very flexible, user definable format. To run a program you merely have to use the cursor keys or the mouse to highlight one of them and then press RTN or click the mouse. There are several other ways of running a program but you need not be concerned with them at first.

Pressing the ESC key will bring up the “main menu” which has several selectable items. You can go from there to the ProSel-16 utilities (a comprehensive set of file/volume manipulation features that are very powerful yet easy to use). Or you can go to the editor, which is used for changing the main screen, adding items to it, or deleting items from it. There are several other options from the main menu that will be described later.

The version number of your copy of ProSel-16 is shown on the main menu.

After installation you should consider “zipping” the START file. See the Zip help file. The START file is not supplied in zipped form in order to support the Mix ‘n Match facility.

Installation

If you have a hard disk then you should run the installation program. Just boot the ProSel- 16 disk and press G <RTN> to select the item “Greetings & installation”. When the installation is finished you should reboot because ProSel-16 uses the boot disk for some things, and, until you reboot, that disk will be the ProSel-16 disk.

For use on *3.5* inch boot disks, you may have to use the Mix ‘n Match utility; see the appendix on Mix ‘n Match.

Installation does not install everything and there may be some items you will want to copy over manually. The installation requires almost 440 free blocks on the disk.

The installation creates the subdirectory PROSEL. 16 and the subdirectories COMMANDS and HELP.FILES of PROSEL. 16. The installation copies the command files to the COMMANDS directory, but does not copy the help files or the archived calculator program file.

If you do not have a hard disk, then do the installation on a copy of your system disk, but first get rid of any items on the disk that you can do without. Check the room on the disk to see if it is adequate. (The installation also checks for sufficient room and will not attempt to do the installation if enough room is not available or if the destination disk is not a GS/OS boot disk.)

If you want the opening super-hires logo, which appears when you boot the ProSel- 16 disk, to appear when you boot your disk, just copy the file InitPic from the System/System.Setup directory to that directory of your boot disk. It should not prolong booting by more than a second on a hard disk.

> *IMPORTANT NOTE: The installation does not install GS/OS. You should already have installed GS/OS on your boot disk, using Apple’s INSTALLER.*

GS/OS almost requires a bard disk for efficient use, or at least a large ROM disk to boot from. If you don’t have one of these, you are strongly advised to consider these options.

The installation renames the present START file in the SYSTEM subdirectory to OLD.START, and installs ProSel-16 as START. Ordinarily the old START file will be the FINDER and it can be run from ProSel- 16. Should you decide you would rather boot into the FINDER, you can rename these files (from some other program) so that OLD.START becomes START. You should realize, however, that this will have undesirable effects on memory requirements and will negate the automatic virus protection in ProSel- 16.

To install a revision, simply copy the files START, PS. 16.TO.8 and Appoint.CDA after booting the revision disk. This will not change your application specification list, which is held in the PROSEL.SPECS file.

☼ Getting started

Let's assume that you have installed ProSel- 16 on your boot disk and have rebooted and are now looking at the ProSel screen. Let's also assume that you are new to ProSel. Then the first thing you are likely to want is the set of ProSel help files. Since the installer has not copied those files, let's do that now.

Press the ESC key. You now see a menu, and one of the items is Utilities. You can either move the cursor to that item using the ~ or J' keys or the mouse, or you can simply press U for Utilities. If you used one of the former methods then press RETURN or click the mouse button.

You will note some disk activity, but shortly the Utilities menu will appear. One of the items is X-Copy files. (X is used since C is taken for the Catalog function.) Either highlight that item with the cursor keys or the mouse and press RTN/button, or just press X. The copy files screen will appear with some instructions. Let's just ignore the instructions for now. The cursor indicates a request for a source pathname. Since the help files are in the /PROSEL/PROSEL. 16/HELP.FILES directory, type in that name. Then a destination is asked for. The default will probably be your boot disk name, so press TAB which will move the cursor across the volume name (or volume part of the default name). Here you can just type in the name of the directory you want to copy to (which is PROSEL. 16/HELP.FILES). But let's assume we have forgotten the name of the directory to which we want to copy the files. Press the ? key or just click the mouse button, making sure the cursor is just after the volume part of the pathname. The disk will be accessed for a while, and then you will see a 'tree diagram' of all the directories on your disk. Use the cursor keys or mouse to move to the desired directory (which is PROSEL. 16/HELP.FILES) and then press RETURN or click the mouse.

Next you will see a list of files from the source directory. They are the help files. Press the space bar. You will see that a check mark has been placed on the first file and that the cursor has moved to the next file. (Instead of the space bar, you could have clicked the mouse button to get the same result.) You could do this for all the files, but there is an easier way to select all the files: move the cursor back to the checked file and press space again, removing the check mark. Now press control-A. (This is denoted by "A, a notation we shall use from now on. Remember that the control key must be held down while the A key is pressed and released.) On pressing "A you will see that all the files are now checked. (A stands for all.) However, no files have yet been copied - this is just the selection phase of copying files. Now press RETURN. You will see all the files being copied. Each one is highlighted just before it is copied. If only some of the files had been selected then only those would be highlighted.

When the copy is done, you will be returned to the Utilities main menu, and you can exit by pressing RTN (the cursor is on Quit) or by pressing the Q key or the mouse button or ESCAPE.

You should now be back at the main ProSel screen. The next thing you are going to want to do is to place some of your applications on the ProSel screen. To do that you must use the editor. Since we just copied the help files let's see if there is some help on the editor. If there is an item called Help on the screen, select it by highlighting it and pressing RETURN or clicking the mouse. If not, then hold down the Apple key and press the H key, and then RETURN. (From now on, I will indicate this by press Apple-H <RTN> ".) This will cause some disk activity and then a list of help files will appear. To select the one called EDITOR, type ED <RTN>. (The help manager only requires a few letters of the help file name to understand which one you want, so it is not necessary to type out the complete name EDITOR.) This will display the help information on the editor. Frankly, this information will probably not make much sense to you at this time, but you may find it useful after you are more familiar with ProSel-16.

Let's go to the editor. If there is an item "ProSel-16 Editor" on the screen, select it. If not, you can still go to the editor by pressing Apple-E <RTN>, or by pressing ESC and selecting Editor from the main menu.

The editor will come up after some disk activity and will display a screen much like the ProSel-16 screen itself, except that the text "ProSel-16 editor" will be seen at the bottom of the screen.

You are going to want to add an application to the screen, so move the cursor to an empty spot on the screen and press RETURN. (For applications not already on your hard disk, see "Installing applications".)

This will bring up a screen with a great deal of text. Read it if you like. It also presents you with two choices (A)utomatic or (M)anual mode. The manual mode assumes that you already have some knowledge of pathnames and how ProSel works. Also, you have to know exactly where your desired application is on the disk. Ok, so let's try Automatic mode instead: press A.

It now asks for a directory to search and will have the volume name as a default. Giving a directory name will limit the search and also make it faster, but you can just press RETURN at this point to have it search the entire disk. Do that. The disk will run for a while and then a list of files will be shown on the screen. These are all the executable files on the volume, or, if the screen is full, it is all of them that will fit. (You could have used a subdirectory instead to get names that may not fit when you use the volume name.)

Look for a file that appears to be the program you want to add to the ProSel- 16 screen. Use the cursor keys to highlight it and press RETURN. That is it, it is done! You will now be looking at the main editor screen and the name of the application will now be there under the cursor. Do you want to change the name to something prettier? If so, press RETURN. What you see now is manual mode, but the specifications for the application will all be there. The cursor will be on the "screen title" and you can change that to whatever you want. Do not change the other three items, but ponder them if you want. Later, you will want to understand what they are all about.

Now let's save our changes. Press ESC to return to the main editor screen and another ESC to get to the editor menu. There are two save items, one is (S)ave as a screen and the other is (P) for save a new PROSEL.SPECS file. It is the P choice that you want, so press the P key.

The save will send you back to the ProSel screen from the editor. You should see the item that you added to the screen. Move the cursor to highlight it and then press RETURN or click the mouse to run that program.

When you Quit or Exit from your application, you should return, after a little disk activity, to the ProSel screen.

That is about all you have to know for the basic operation of ProSel-16. There is a lot more to ProSel than what we have described, and the rest of this documentation will go into everything. Some of what you will read may be too technical for you at this time. Don't worry about it. When you climb Mt. Whitney, you must first drive to Lone Pine, and then you have to get to Whitney Portal. Both of these are much easier than the trip to the top, and most of us will never reach the pinnacle. The higher we get the harder it is, but, however far we go, the view is magnificent.

The structure of the ProSel -16 system

We have already met several parts of the ProSel system: the main screen (which you can hardly miss), the main menu, the Editor, the Utilities (which corresponds, incidentally, to the ProSel-8 program), and the Help facility.

Other main parts of the system are File Finder, Information Desk, Backup/Restore, Zap, Volume Repair, the Optimizer, the Calculator, the Text Editor, and the Shell command processor. The latter is one of those things which present you with a blank screen and a cursor and dare you to do something intelligent. This frightens some people and, if you are one of those, let me assure you now that this is a part of ProSel-16 that you don't even have to see if you don't want to. It is there for people who want it. If you land there by mistake and you start shaking, remember the ESC key! Think of that key up in the left upper corner as your panic button - the cure of all evil. Nevertheless, we will have to talk about the Shell later on. It is one, together with the main screen and the main menu, of the three major "hinges" of the ProSel system. The other parts, such as the Utilities, as important to you as they may be, are more like separate programs that are just a little more immediately accessible than most applications. Indeed, although all these things are in the same file START, only the main screen, the main menu, and the command processor are loaded into memory upon boot. The other parts are loaded whenever you go to them. You noted the fairly sizable disk access when you went to Utilities. That was loading the Utilities (a sizable hunk) from disk into the computer. After you quit from Utilities back to the main screen, if you again went to Utilities, you would go there very quickly, without all that disk access. This is just from the fact that Utilities has already been loaded by the previous access, and now does not have to be loaded again.

Here is a diagram that may help you understand the relationships between parts of the program. It does not show everything; ProSel, being a very flexible program, usually has more than one way of accomplishing a particular goal.

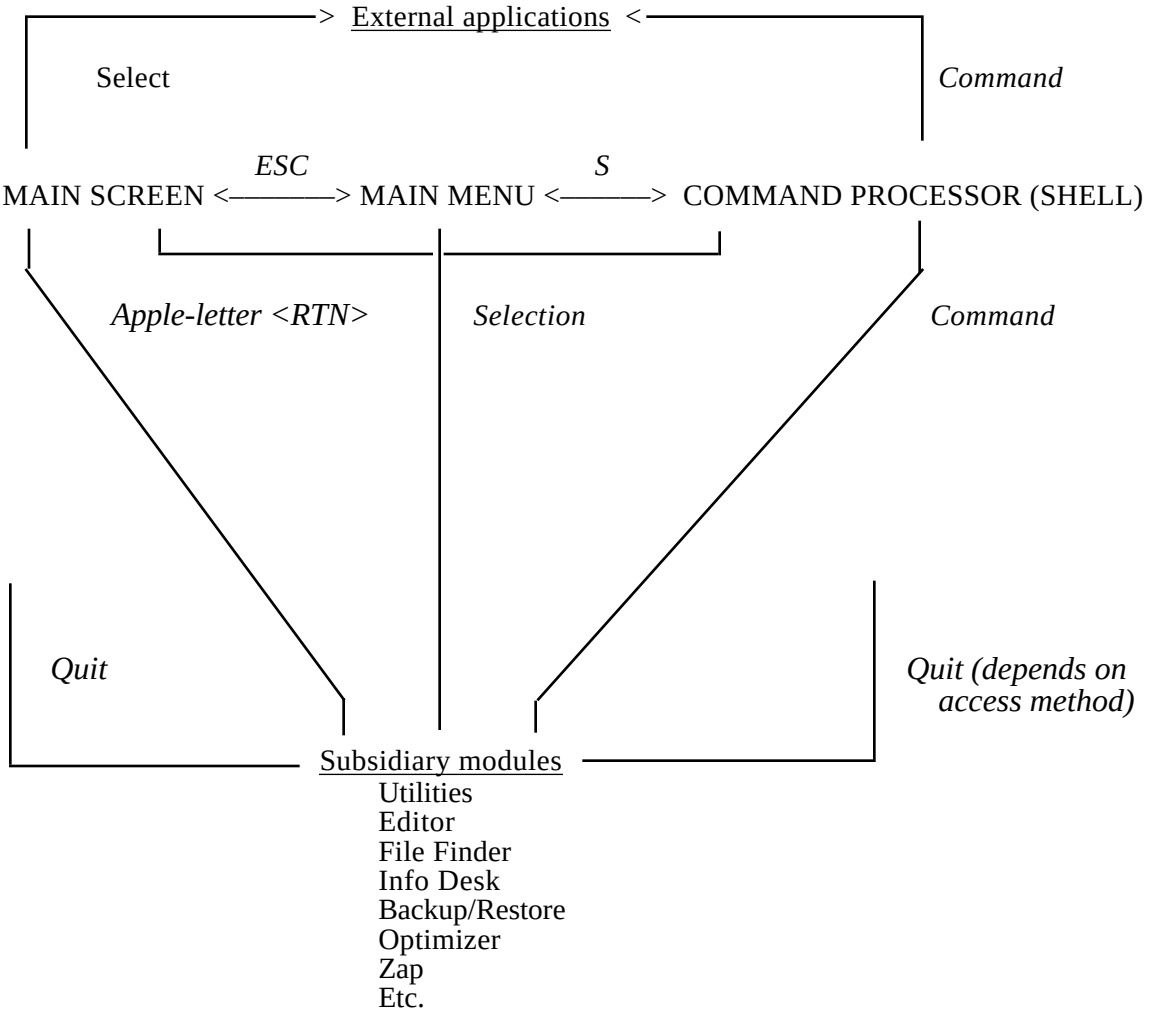


Figure 3. *ProSel-16 structural diagram.*

On the other hand, this may just look like a lot of hot air to you. If so, then never mind, perhaps it will be clearer later.

Part II. Program Selector

☼Executable files

ProDOS files can have 256 different “filetypes” (or simply “types”). The more common filetypes have names (always three characters). Some types are reserved and have no presently defined file structure, and these are indicated on catalogs by a hex byte such as $F9. Presently, programs that can be run directly can have one of three types, called:

SYS (system file)
S16 (system-16 file)
EXE (shell executable file)

The first of these, SYS, is the filetype of executable files that run under ProDOS-8. Its counterpart under GS/OS is an S16 file. EXE files are also GS/OS executable files, but often expect a “shell”, such as ProSel-16, to be in memory for various types of assistance. Unfortunately, programmers do not always follow the rules for these files, and so there are many SYS files around that are not really stand-alone executable files. Also there are many EXE files that do not behave according to the design of the EXE filetype.

These three types are the filetypes of files that can be run from ProSel. In addition, there are BAS (BASIC) files which are designed to run under BASIC.SYSTEM. In this case BASIC.SYSTEM is the SYS file that is run, and the BAS file’s name is passed by the selector to BASIC.SYSTEM when the latter is run, telling it to run, in turn, the BAS program. Similarly BIN (BINARY) files often, but not always, are programs designed to run under BASIC. SYSTEM and they are handled in the same way as HAS files by ProSel. Rarely, TXT (TEXT) files are designed to run under BASIC.SYSTEM as “exec” files. Since this is so rarely the case, and since such files are easily run from a simple BASIC program, I have decided not to allow direct access to these from ProSel-16. Instead, ProSel-16 will EXEC such a file as a set of shell commands under ProSel-16 rather than BASIC.SYSTEM. [See the section on the Command line processor, and particularly on “automatic operation”.]

☼ Virus Protection

When ProSel- 16 is asked to launch a program, it automatically checks for known viruses and will not allow such files to be launched. The viruses currently checked for are “Cyber-Aids”, “Festering Hate”, and “Burp”. The rather different viruses “Load Runner” and “Screen blanker” are checked for in memory and eliminated. The latter do not infect files and are only contracted by booting an infected 3.5 inch disk. The screen blanker virus may “go off” before it can be detected. It will seem to totally disable the machine. If this happens, press Option-Control-Reset and then press “2”. When ProSel-16 detects a virus, you will be so informed.

Graphic display

ProSel-16 has an optional super-hires graphic display option which can be defeated by one of the parameters. When looking at the main screen this mode can be toggled by pressing Apple-TAB. There are four colors: background, text and two highlight colors, and all are settable in the parameters. The first highlight color is the counterpart of inverse in text mode.

There is a parameter that causes characters under the cursor to appear in inverted color, which is recommended for readability when one chooses to have a light background and dark characters, but not necessarily otherwise.

The graphic mode is turned off just prior to launching any program.

There is a parameter that disables use of shadowing. This will provide more ram space for EXE files, but will slow down graphic scrolling by about 20% to 40%.

There are also 64 extra “mousetext” characters, corresponding to lower case characters and ascii values $20-$3F. These can be viewed with the ASCII command and with the design mode of the editor. They are block graphics characters which can be used to create some interesting looking screens.

Pleasing screens are most easily set up with the editor’s designer, see below.

All the colors (and border color within ProSel-16) are changeable at the end of the parameters (but only if Modify parameters is entered in graphic mode), and there are ten easily selectable preprogrammed setups. You can also set up separate color pallets and borders for some of the ProSel- 16 modules by pressing Option together with T, N, P. U, F, B, 0, Z, I, V or A, which will transfer the presently showing colors to the modules: Text Editor, Number Cruncher, NC-plotting mode, Utilities, Find File, Backup, Optimizer, Zap, Info Desk, Volume Repair, and Appointment Calendar respectively. Adding the control key to this key combination will read and display the pallet for that module.

Installing applications

ProSel-16 does not install application programs on your hard disk. Many programs come with their own installation. Many others cover hard disk installation in their manuals. In other cases, you usually just need to create a directory (see Utilities) for the program and copy all the files from the program disk to that directory on your hard disk. You do not have to copy the PRODOS file or any files in the SYSTEM directory of the program disk.

☼ Program selection from the main screen

It probably does not have to be said that to run a program from the main screen you use the cursor keys or mouse to highlight the desired item and press RETURN or click the mouse button. Also, as indicated on the screen, you can press ESC to go to the main menu. However, there are other actions you can take that are not so obvious.

If you press an alpha key (A-Z) at the main screen, then the highlighting will move to the next item starting with that letter. The case of the letter has no effect. If there is no item beginning with that letter then the cursor moves to the first item on the screen. Scanning is done downwards and then across.

If you press a number key on the main keyboard then the disk DEVICE corresponding to that number (0 is used to mean 10 for this purpose) will be accessed and a list of the executable files and subdirectories from that disk device will be shown on the screen. You can select an executable file to run, or you can select a subdirectory, causing that directory to be read and its files displayed, or you can press the] key (the Applesoft prompt character) and the BAS and BIN files in that directory will be displayed for selection.

DEVICE NUMBERS: These are the GS/OS or P16 equivalents of the P8 slot/drive. Device #1 is always your boot disk. The other device numbers depend on how your SYSTEM/DRIVERS directory is set up. Some devices are not disk drives at all. For example the CONSOLE (screen) may be a device. To see the correspondence between devices and their numbers, use the P command from the main menu or Apple-P <RTN> from the screen.

If you press a number key on the KEYPAD, then the directory corresponding to that PREFIX will be read, and its executable files and subdirectories will be displayed for selection as above.

*PREFIXES:GS/OS keeps a set of 33 “prefixes", which are names of some special directories. They are named */, 0/. 1/,....,31/. The */ prefix is always the boot volume name. The 0/ prefix is the “default’ prefix, meaning that it is added to any pathname that is not a "full” pathname. The 1/ prefix is the directory containing the current application file. When in ProSel-16 it is */SYSTEM if ProSel-16 is installed in that directory as recommended. Prefix 6/ is used by ProSel-16 as the “transient command” directory, prefix 7/ is used as the “help files” directory, and prefix 5/ is used for data storage by Info Desk. Backup the Text Editor and the Appointment Calendar. All these prefixes, except */ and 1/, can be set up as desired by the user by modification of the ProSel-16 parameters, as will be detailed later.*

The option key together with a number key adds 10 to the key value for both of these provisions. Similarly the Apple key adds 20 and control adds 40 (main keys only) with combinations accumulating.

In addition, if you hold the Apple key down when typing an alpha key, you will go to the main menu with the cursor on the item corresponding to the key. (If none, no action occurs.) For example, Apple-U goes to the main menu and highlights the Utilities item. Thus pressing RETURN then sends you to the Utilities module.

When you run a program from the main screen, there are also some options that can be requested by pressing one or more of the Apple key. Option key, Shift key, and Control key. (These can also be made automatically selected options via the program's "specs", to be described later.)

(1) Apple key down. This will cause ProSel- 16 to do a memory purge before running the program. This can be useful for running some very large programs that may not be able to get enough memory after the loader has already loaded the program.

(2) Option key down. This will cause ProSel-1 6 to purge memory and to force reloading of itself after the application has quit. This has the effect of cleaning up memory a little more thoroughly than just case (1). It may allow running programs for which (1) is insufficient.

(3) Apple and Option keys down. This will do the same as (2) but will also not request return when the application quits. This would be appropriate if you wanted it to return to another selector.

(4) Shift key down. This has effect only when running a SYS program, that is, a P8 application. This will cause ProSel- 16 to set up ProSel-8 before launching the program, so that it will be ProSel-8 which gains control when the application quits. Of course, this requires ProSel-8 to be available. In fact, it supposes that the PROSEL and the PROSEL.SYSTEM files (both part of ProSel-8) are in the main directory of the boot volume. [To do this automatically from a program's specs (see below) you would set the specs to run PROSEL.SYSTEM as the application and the desired SYS application as the "startup".]

NOTE: If there is a startup specification in the program's specs then the shift key will be ignored because there is a conflict between these two things. However, holding down the control key defeats the startup, so both control and shift down will always force a shift to ProSel-8.

(5) Control key down. This overrides any "startup' in ProSel-16's specification list for the file. Thus, if you have an application specification that instructs the application to automatically open a certain file, then running that program with the control key down will bypass the open instruction. You only have to hold the control key down for a moment while you press RETURN or click the mouse. You can let it up as soon as the "launching" message appears.

At various places in ProSel-16 you must type a line specifying a directory or other information. This occurs, for example, in Utilities, the Editor, and the Command processor. In all cases, tine input supports a number of line editing features:

D deletes the character under the cursor.
DELETE deletes the character to the left of the cursor.

‘E toggles insert mode.
Apple-key inserts the character whether or not in insert mode.
‘0 accepts next keypress literally where meaningful.

TAB moves Just past next / or goes to line end adding/.
Apple-TAB moves back to right of previous / or to line start.
‘N moves to line end.
‘B moves to line beginning.

‘Y clears line from cursor forward.
CLEAR clears line canceling all previous input.

ESC aborts.

RTN on first character, accepts default, if any.
RTN on other characters clears characters under and after the cursor and accepts line up to cursor as the entire input.

When you are asked for a device number (which happens in some parts of Utilities, Find File, and Info Desk), you can press the ? key. This will cause a list of up to 10 recognized device names to appear and you can use the cursor keys to select one of those. In that mode, if you press the ? key at any time, the name of the volume occupying the highlighted disk device will be displayed. For devices not on the list you can input the **device name**, starting with a period, at the device number prompt. An example of a device name is ".APPLED1SK3.5A. You can also input a two digit device number after a period at any device number prompt. You can also input the **volume name** of the volume in the device, starting with /.

Some parts of ProSel support additional features for line input For example, Utilities uses ‘X to switch between the two default pathnames. (Elsewhere ‘X is the same as CLEAR.) Similarly, in the command line processor, the ↑ and ↓ keys take you through the history of commands, while, in other places, these keys are not accepted and do nothing.

NDAs

You can access NDAs from the main screen by pressing Apple-*.

☼_The main menu

The main menu is accessed from the screen by pressing ESC. It contains a number of options:

•*Help* - brings up a list of help topics for selection.
• *Quit* - returns to a program that ran ProSel-16, if any. If none, then it just reruns ProSel- 16.
• *Editor* - enters the ProSel- 16 application list editor.
• *Utilities* - enters the file utilities.
• *File Finder* - enters the file finder module.
• *Show prefixes* - try it. It also shows some other data.
• *Warm shutdown* - use to reboot, or before shutting down system.
• *Cold shutdown* - ditto, but this one puts control panel changes into effect if you press return to reboot.
• *Shell* - enters the "shell" or command line processor.
• *Refresh screen* - rereads the program specs file, can be used to shift to a specs file on another disk.
• *Backup/Restore* - incremental hard disk backup.
• *Volume Repair* - repair facility corresponding to "Mr.Fixit".
• *Optimizer* - ProDOS volume optimizer (Beach Comber).
•*Exchange screens* - this is the "cycler".
• *info Desk* - enters the information desk module.
• *Modify parameters* - allows modification of the user definable parameters for ProSel-16 (the PROSEL.PARMS file).
•*Zap* - a block editor called Block Warden.
•*Number Cruncher* - a powerful programmable calculator.
• *Text Editor* - a useful, powerful and convenient editor.

In addition the ESC key returns you to the main screen.

You have already met some of these options. Most of them will be described in more detail in later sections.

As mentioned before, pressing Apple-▫ while at the main screen, where ▫ is the command key for one of these menu options, will take you directly to the menu with the cursor on the desired item. Although this is not less keystrokes than the regular ESC and selection of the item, it will be more intuitive to some people. There is a parameter that will make the Apple-▫ commands active immediately without the necessity of following them with RETURN.

When you choose any of the options on the main menu, and return, you are never sent back to the menu. Instead you return directly to the main screen.

Also, many of these items can be selected in other ways. They can be set up as selections on the main screen by the proper choice of "specs", as we will discuss later. Some of them can also be accessed directly from the command processor.

This section is a short tutorial on “pathnames” in ProDOS. It does not go into some of the esoteric additions that have been added to GS/OS.

A pathname is a roadmap telling ProDOS how to find a file on the disk. ProDOS volumes are organized in a “tree” structure, sometimes called a hierarchic structure. The “root” of the tree is the volume name. Off this root are limbs, most of which are subdirectories or “folders”. Some of the limbs are ordinary files. The subdirectory limbs can have limbs growing off of them, etc. An example of such a structure is:

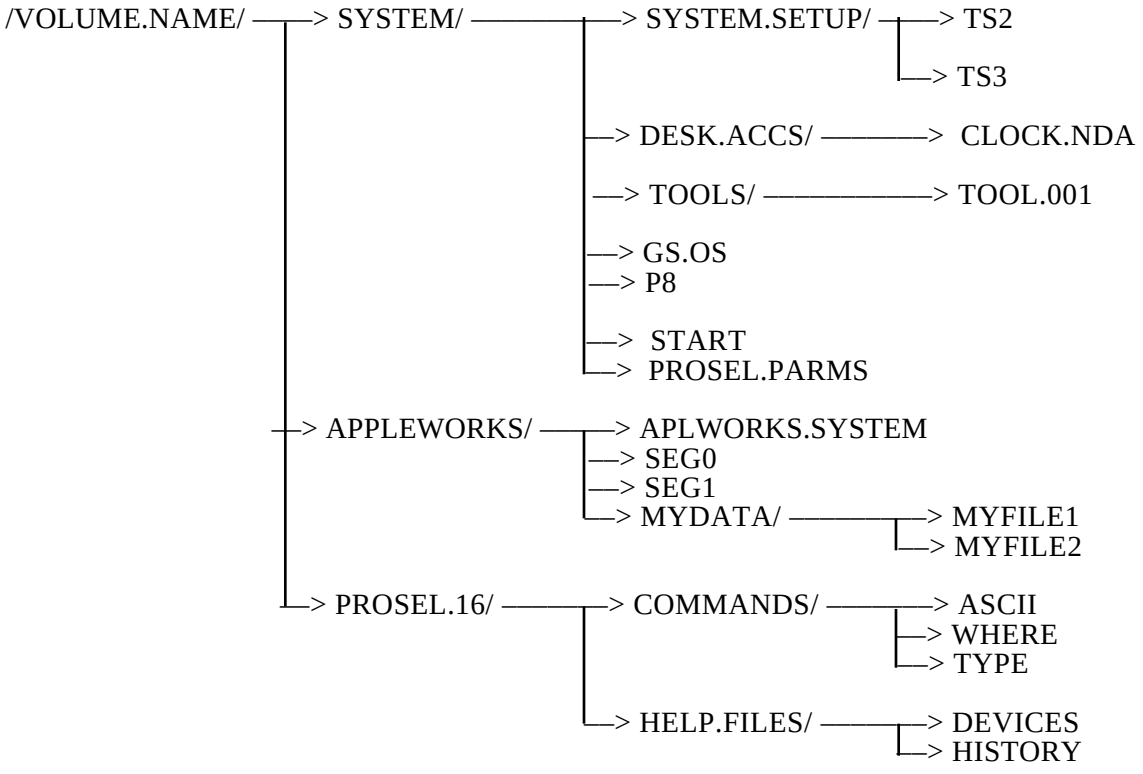


Figure *4. Typical directory tree diagram.*

Here you may ‘climb’ the tree by going either right or down (I guess this is a fallen tree) at any crook. In the diagram the titles ending in / are directories. (/VOLUME.NAME/ is the root, or main, or volume directory, and the rest are subdirectories.) Using the slash to end the name of a subdirectory is just a convenience here; it is usually optional when specifying a directory name.

The other files shown in the diagram are regular (non-directory) files. Some of them, like START, are executable programs, while others like HISTORY are data files. To construct a file's "full pathname", you just climb the tree adding each name to the last. Thus the full pathname of the file DEVICES is

/VOLUME.NAME/PROSEL.16/HELP.FILES/DEVICES

A prefix is a full pathname to a directory file. Thus the pathname above is not a prefix since it points to a non-directory, but

/VOLUME.NAME/PROSEL.16/HELP.FILES/
and
/VOLUME.NAME/PROSEL.16/
and
/VOLUME.NAME/

are all valid prefixes. One can set the system's prefix by various means and then it is called the prefix, or, in GS/OS, it is prefix #0.

A "partial pathname" is a pathname not beginning with a "/" (or some other characters see below). Giving a partial pathname for a file means that adding the given partial pathname to the prefix would result in the full pathname for the file

NOTE. When specifying a prefix, one usually does not demand that the "/" mark at the right hand end be given. It is understood. So, when forming the fill pathname, one adds the separator "/" between the prefix and the partial pathname If it is not already there.

Thus, if the prefix is set to /VOLUME.NAME/PROSEL.16/ then a valid partial pathname for the DEVICES file is HELP.FILES/DEVICES.

GS/OS supports some additional conventions that make the task of giving a file's pathname easier. As previously mentioned, there are 32 other prefixes that GS/OS remembers, named */, 1/, 2/, ..31/. The default prefix can also be called 0/. Note that these do not start with the */* character.

The prefix *1 cannot be changed, and is always the name of the boot volume. In the example, if the tree given is the tree for the boot volume then */ and /VOLUME.NAME/ mean the same thing.

The prefix 1 / is set to the directory containing the currently running program. (It can be changed by the program, but this is not advisable.) Thus, in ProSel-16 it will ordinarily be the same as */SYSTEM/. If you run APLWORKS.SYSTEM in the above example, then the prefix 1/ (or "prefix #1") will be */APPLEWORKS/ which is the same as /VOLUME.NAME/APPLEWORKS/.

☼ Application specifications & the editor

We have already discussed using the editor to add applications to the screen display by using the “automatic” mode. For most uses this is sufficient, and if “pathnames” confuse you, you can stick with the automatic mode for the time being and not worry about the more technical details in this section. Here we describe the “manual” mode of the editor, and how to access some of the special options that one has in running applications from ProSel-16.

An application specification in ProSel consists of four items:

(1) The screen title. This is what you see on the screen. It can be anything at all. It can contain control characters to do such things as turn on/off mousetext or inverse. Even the beginning character can be made invisible by following it with a backspace and another character.

If the “screen title” starts with a character whose ascii code is less than $40 (e.g., numbers, control characters, and most punctuation characters), then it will not be a selectable item. This can be used for topic headers on the screen, or similar things. The cursor will not land on such an item when the cursor keys are used. The mouse can land on them, but cannot select them, and the mouse button will cause the cursor to move to the next selectable item.

(2) The prefix. This sets prefix #0 (the “default” prefix) for the application. It is usually, but not always, the directory containing the application program. But it could be a subdirectory containing the application’s data files, or whatever you want. It could even be a directory on another volume. Note that some applications set this prefix themselves and, in that case, this item does not accomplish anything. However, it is still required and must be a full pathname, such as

*/APPLEWORKS or /VOLUME.NAME/APPLEWORKS,

of a directory.

(3) The application name. This must be a full or partial pathname of the application program itself. The program must be of file type S16, SYS, EXE or TXT. Thus and

APLWORKS.SYSTEM

and

*/APPLEWORKS/APLWORKS.SYSTEM

are valid examples, but for the former to be correct the prefix (item 2) must be such that adding this application name to it results in the correct full pathname. In the example above, APLWORKS.SYSTEM is a valid application name if the prefix is */APPLE WORKS.

(4) The startup. Usually the startup is null. It is the only part of these four-part specs that can be left empty. It can be used to communicate various things to the application program. (However, the application must be listening - i.e., It must be something that the application was programmed to look for.) In GS/OS, the startup is usually the pathname (full or partial) of a file you want the program to open automatically. It could also be some special instructions (some ProSel applications use it that way) and the program's documentation will tell you what the program looks for, if anything, in the startup.

If the application is BASIC.SYSTEM, then the startup is the BASIC program you want run by BASIC.SYSTEM. If you do not give a startup when you run BASIC.SYSTEM, then BASIC.SYSTEM looks for and, if found in the prefix #0 directory, runs the program called STARTUP. Thus, the startup, if any, in the ProSel specification is substituted for the default startup - the file called STARTUP.

Some other programs work similarly. PROSEL.SYSTEM, which is the boot program for ProSel-8, uses the startup in a similar manner, running the program given in the startup as the first thing it does. If a startup is not specified then it does not run anything but just goes to the ProSel-8 screen. In this case, in distinction to BASIC.SYSTEM, the startup must be a SYS file.

In the manual mode of the editor, you specify these four things or just three if you don't want to give a startup. The prefix does not have to end with a / character.

ProSel uses some conventions in addition to the general pathname syntax discussed above. The ? character is used by ProSel to stand for the current volume name with a slash on both ends. Thus ? usually gives the same result as ***/** but not always, since the current volume name (which means the name of the volume from which the file PROSEL.SPECS was last loaded) does **not** have to be the boot volume. Use of this character provides transportability of application specifications as well as being a space saver. I recommend its use always in any full pathname. Remember that the ? includes the / on both ends, so that ?APPLEWORKS would be a correct prefix, but ?/APPLEWORKS would not since it expands to /VOL.NAME//APPLEWORKS and the two adjacent slashes are illegal.

Similarly, the character] placed anywhere in a pathname is expanded to BASIC.SYSTEM. Thus the two characters ?] will be interpreted by ProSel as:

/VOL.NAME/BASIC. SYSTEM

The ? and] conventions are legal in both ProSel- 16 and ProSel-8.

ProSel- 16 also uses some other special notations in the prefix field as instructions to do certain actions before running the program that has been selected

(1) If the character = alone follows the prefix, then the computer will be switched to slow speed before the program is run. (However, if this item is one that inns an internal module of ProSel- 16 such as the Utilities, then the = causes text mode to be toggled. Internal modules always run at fast speed.)

(2) If the character = follows the prefix, and itself is followed by one of the digits 0 to 7, then the following actions are taken before launching the application:

=0 purge memory.
=1 purge memory and force reload of ProSel on application exit.
=2 purge memory and do not return to ProSel-16.
=3 purge memory, do not return, and force reload next time ProSel-16 is started.
=4 set TransWarp-GS or ZipGS to slow speed (if card exists).
=5,6,7 same as =4 together with = 1,2,3 respectively.
(The TWGS or ZipGS will be reset to fast speed on return to ProSel.)

NOTE: ProSel-8 uses some similar =“ notations that mean entirely different things. These things do not apply to an Apple IIGS and so there is no real conflict between the two.

ProSel-16 supports a convention that instructs it to go to the main menu and execute a particular item there. To add such an item to the screen, you must give any prefix, die command letter (e.g., U for Utilities) for the “application’ and have any startup. That is:

Screen *title:*	Utilities -16	(or anything)
Prefix:	?	
Application:	U	(main menu command char.)
Start *up:*	<none>	

An = after the prefix in such an item will toggle graphics mode.

Finally, both ProSel- 16 and ProSel-8 support some conventions that instruct ProSel to read a directory and display its contents:

Screen title:	Games directory	
Prefix:	?GAMES	(full pathname of directory)
Application:	/	(special instruction)
Startup:	<none>	

That example will read the directory given by the prefix and display all executable files (S16, SYS, and EXE) for selection.

Another example:

Screen title:	BASIC programs	
Prefix:	?BASIC	(full pathname of directory)
Application:	]	(pathname to BASIC.SYSTEM)
Startup:	]	(special instruction)

That example will read the ?BASIC directory and display all BAS and BIN type files for selection. Note that the Application path must be a valid path to BASIC.SYSTEM. It does not have to be just the] character as shown. BASIC.SYSTEM could be in a completely different directory and this would still work if the Application path is a full pathname leading to BASIC.SYSTEM. The startup here is the special instruction telling ProSel that this is a request to read a directory and not one to run a program.

After using the editor to edit the application list or add to it, press ESC to get out of manual mode and another ESC to go from the editor main screen to the editor menu. There you will have several options. Ordinarily you want to press P to save the revised PROSEL.SPECS file. There are also options to load and save screens, which will be dealt with later.

You can also set other prefixes in a program specification by putting, in the “startup’, a prefix number (two digits exactly) then a colon and then the prefix name. For example, a startup of

08:/HARD1/MISC

will set prefix #8 to /HARD1 /MISC when that program is run. You can follow this with a semicolon and some other startup or another prefix, etc. The prefix specifications are stripped from the startup before the startup is handed to the application.

Most desktop programs use prefix #8 to point to their data files. Thus, the specification:

Screen title:	Database (Sales 1990)
Prefix:	/HARD1/AWGS
Application:	Appleworks.GS
Startup:	08:0/SALES;8/SaIes.1990

will run the Appleworks.GS program located in the AWGS directory, set prefix #8 to /HARD1/AWGS/SALES and tell Appleworks.GS to open the Sales. 1990 file which is in the /HARD 1 /AWGS/SALES directory.

There is a "Design screen" option in the edit menu. If you select this, then the screen will be shown surrounded by the graphics "mousetext" characters (old and new). In this mode you enter data directly on the screen including regular text and graphic characters.

At the bottom right there is a very brief command reminder of the command keys. KEYPAD keys and control keys (ARROWS, TAB & ESC) are command keys. DELETE acts as a delete. Use Control-DELETE to put a DEL character on the screen. All other keys go on the screen or are ignored.

Keypad key 0 toggles mousetext mode, keypad keys 1-9 select the desired color, and keypad key * toggles inverse characters. The TAB key moves to the next column.

Exit design mode with the ESCAPE key.

The ^B key moves the cursor to the beginning of an item and N moves it to the end.

Use the Apple key with another key to do an insert.

Option-↓ replicates a line to the next line.

Any alphabetic character in color #1 (regular text color) in an item will cause the item to be selectable. To defeat this, press the Apple key as you move the cursor off the item. You must have at least one selectable item before you save the screen or it will not work.

The designer only modifies titles and it may change some existing titles because it will remove excess backspaces and all control characters it does not understand.

The designer cannot be used on the last 4 screen lines (normally unavailable), but you could create displays above that point and then use the regular edit mode to move them down.

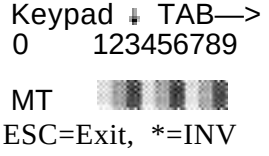


Figure 5. *Designer command indicator.*

Importing ProSel-8 application lists

Many will want to transfer their present ProSel-8 application list to a ProSel- 16 list, and a very easy procedure has been provided for doing so. Enter the ProSel- 16 editor and press ESC to get to the editor's menu. Select "Load a screen", even though it is not a "screen" that you wish to load. When asked for the pathname to load, give the full pathname of the PROSEL file (e.g., */PROSEL) Edit it if you want, and then, when you want to save it as a ProSel-16 application list, use the P command, not the S command, to save it. This will write over the current PROSEL. SPECS file thereby converting the ProSel-8 specs list to a ProSel-16 specs list. Note that the internal format of the ProSel-8 PROSEL file, and the ProSel- 16 PROSEL.SPECS file is very different, so it will not work to simply copy the PROSEL file and rename it PROSEL.SPECS. The editor automatically adjusts the format of the file when it loads and saves.

Similarly, as will be remarked again when we talk about the "screens" feature, you can import a ProSel-8 "screen" by loading it into the editor, and you can resave it back either as a ProSel- 16 "screen" (again a different format) with the S command, or as the main ProSel- 16 application list with the P command. The editor does not support conversions the other direction.

Screen blank and/or slide show and/or kinetic art

After a user selectable time the screen will blank to black, coming back at any keypress. Alternatively, you can select a "slide show" to appear after the given amount of inactivity This is selected through the "Modify parameters" function. The slide show expects to find its "slides" in the prefix #31 directory. This should also be set up in the parameters The slides must be super-hires files (unpacked of filetype $C 1 {PIC} and auxtype 0 or packed of filetype $C0 {PNT}and auxtype 1, and of length at most $8000). [Also see the PackPic transient command and help file.] At any time during a slide show, the ESC or CLEAR keys will abort the show and produce the normal screen blank mode, at which time another keypress will produce the ProSel- 16 screen. The RETURN key during a slide show will produce the ProSel-16 screen directly. The space bar will freeze the current picture until a key press. Any other key will cut short that particular slide, and bring up the next one. When the sequence of slides is used up, the show cycles to the first slide again. You can set the number of repetitions of the slide show done before passing to the screen blank, or you can specify that it is to go on forever until a key press.

The slide show is followed by a kinetic art show if that is not defeated in the parms. During the kinetic display, the ESC key goes to screen blank, RTN exits to the ProSel- 16 screen, number keys change the trailing image length, H toggles horizontal symmetry, V toggles vertical symmetry, G toggles maximum gap size large/small, S slows the display (cycling in 4 steps), R randomizes the pallet and P toggles polygonal mode.

In polygonal mode, V toggles rectangular/octagonal and H does nothing. The parameters P, V and 3-6 are selected randomly at the start.

SHOW PREFIXES

The Show prefixes selection from the main menu displays the values of the GS/OS prefixes, and the device names for the first 22 disk devices. It is advisable to get acquainted with the correspondence between device numbers and the disk drives associated with them.

Show prefixes also indicates the characters that ProSel- 16 puts on the screen to indicate the filetypes of the files listed by one of the “display directory” features (accessed either by a number key from the screen, or by one of the special specifications designed for this purpose). (See “Number key icons” in the appendix.)

Finally. Show prefixes prints thermometers showing the relative usage on up to six disk devices other than *5.25* inch drives, and memory usage. The memory usage thermometer will have a blank area without the division marks, which indicates the “maximal memory block size".

REFRESH SCREEN

The refresh screen item on the main menu simply rereads the PROSEL.SPECS file from the main directory of the disk corresponding to the current prefix #0. This can be used to get rid of a “screen’ that has been loaded over the main screen. Its primary purpose is to switch to a PROSEL. SPECS screen on another disk, and you can do that by first accessing the disk with a number key on the main keyboard and then giving the refresh command Apple-R <RTN>. ProSel- 16 will continue to use that screen until changed by another refresh command to a different disk, or until it cannot find a PROSEL.SPECS file on that drive. In the latter case it will poll all your disk drives until it locates one such file. If none can be found (or if the file is somehow damaged) then an error message will be given, and you will have the option of going to shell command mode. (One of the primary purposes of shell command mode is for just this back door, It allows you to get out of bad situations gracefully.)

In a specification for the Refresh command (R as the application), if a device name or volume name is given in the Startup then that device will be used for the refresh, i.e., the PROSEL.SPECS file on that device will be read and displayed when this item is selected. Thus the specification:

Screen title:	Switch to HARD2	
Prefix:	/HARD2	
Application:	R	(meaning the Refresh item)
Startup:	/HARD2	(= the desired volume)

will switch to the PROSEL.SPECS file on the /HARD2 volume.

Help

The built in help facility simply reads the prefix #7 directory showing all the TXT type files and lets you look at any of them. It does not require a complete filename from you, but will match the characters you input against the same number of characters from the filenames and read the first file which matches to that point. The help files are standard text files so that you can edit them or write some of your own. A carriage return is expected at the end of each line. A linefeed (^J) character will cause the output to pause and the “Press a key for more, ESCAPE to quit” message to appear.

Shutdown

There are two items on the main menu called Cold and Warm shutdown. These shut down the GS/OS operating system as recommended by Apple as preparation for turning off the computer. They should also be used, if possible, if you need to reboot for some reason. The difference between Cold and Warm is that the Warm shutdown will not resize or clear /RAM5, while Cold shutdown is the same as a power down reboot - it will clear /RAM5 and it will pick up any change in the size of /RAM5 that you have made in the control panel.

Exchange screens

This option in the main menu is equivalent to the PROSEL.CYCLER program in ProSel-8. Selecting this item will bring up a dialog with choices for changing the present application list to another one that you have previously made. These application lists are versions of the PROSEL.SPECS file and must be in the main directory with that file, and must be named PROSEL.SPECS.2. PROSEL.SPECS.3, etc. To make such a file, enter the editor, press ESC to get to the editor menu and issue the N command. Then use the TAB key to move the cursor to the end of the name and change the name PROSEL. SPECS to PROSEL.SPECS.2, etc. This will set up the P command to save to the file PROSEL.SPECS.2, etc., instead of the usual PROSEL.SPECS. You can edit such a list by using the cycler to change to the list you want to edit before going into the editor, and that will become the file loaded into the editor upon entry to the editor.

By means of a special type of application specification, you can set up a screen item that goes directly to one of these extra ProSel main screens, without going through the Exchange screens item on the main menu. This specification will cause a change to the extra main screen number 3:

Screen title:	Word Processors	
Prefix:	?	(does not matter)
Application:	X	(meaning the Exchange item)
Startup:	3	(= the desired main screen)

Modify parameters

There are a number of user modifiable parameters in ProSel-16, including printer parameters for various parts of the program, names for the “user” filetypes, the correct pathname for the BASIC. SYSTEM file that is needed by the File Finder in order to launch BASIC programs from there, selectable pathnames for the prefixes 2 through 31, etc. Selecting the Modify parameters function from the main menu will take you through all these choices, presenting the present values for approval. You can press return to accept the shown value, or you can type in the value you want to change it to. In many of the changes, you do not have to press return after entering a change. At any time, you can press ESC to move immediately to the end of the routine. At the end of the routine, you are given the choice of saving the changes (these go into the file PROSEL.PARMS which is in the prefix #1 directory - usually */SYSTEM), or just using them on a temporary basis, or of canceling all the changes you have made. Pressing TAB anywhere within Modify parameters sends you directly to the color pallet selection screen. (See “Program selection from the main screen”.)

Boot programs

One of the user-settable parameters in Modify parameters is the option of specifying the name of a program to launch automatically upon boot, bypassing the ProSel- 16 screen. In fact, Modify parameters also allows you to set its default prefix (prefix #0) and its startup (such as a file for it to open).

If this boot program is very large then there is a chance that it will force ProSel- 16 to reload when it quits. This could easily lead into a cycle where the program gets run again automatically after it quits back to ProSel- 16, which, having been purged, has forgotten that it ran this program just previously. To avoid this problem, ProSel- 16 keeps track of the day the boot program was last launched in this manner and will not honor a relaunch on the same day in this manner. (Direct selection of the program from the screen is not affected by this.) One of the user modifiable parameters can defeat this day check.

In addition, there is a provision for a keyboard action to force ProSel- 16, on boot, to ignore the boot program specification: just hold down the CONTROL key when booting (it must be down when ProSel- 16 is booting - say from about three quarters of the way along a boot until the screen clears). There is also a key that will **force** recognition of the boot program (overriding the day check): just hold the SHIFT key down in the same manner.

NOTE: The CONTROL and SHIFT keys have a similar function when running a program normally from the screen: CONTROL means to ignore the startup. SHIFT means make a certain special startup.

☼ Screens

ProSel- 16 also supports extra “screens”. There is a demo of these in the /PROSEL/ PROSEL. 16/SCREENS directory, and that demo includes some documentation on their usage. Just select that item after booting the ProSel disk.

ProSel- 16 screens are identical to the screens in ProSel-8 as far as their outward workings are concerned, but they differ in file structure and therefore have a different file type. ProSel-8 screens can be imported into ProSel- 16 by loading them into the ProSel- 16 editor and resaving them to somewhere else on the disk, with the S command.

A “screen” is just a ProSel-16 application list that is saved by the editor into a special type of EXE file that can executed by ProSel- 16. It is in the nature of a screen, however, that it leaves ProSel-16 active and just replaces the application list by the one in the screen. This provides a flexible extension of ProSel- 16 which is virtually unlimited since there is no limit on the number of screens that can be used nor on the location of these files on the disk.

There is really nothing much to know about screens. They can be loaded into the ProSel-16 editor, edited and saved back again. There is nothing different about this from editing the standard ProSel- 16 listing, except that you use the “Save screen” item in the editor rather than the “Save new PROSEL.SPECS file” function. Screens are “selected” just like any other executable program. It is recommended that screens be kept in the prefix #4 directory. That way they can easily be selected by pressing 4 on the keypad and then selecting the desired screen file.

Screens carry their own pallet and border color. To set this, just set the colors you want in Modify Parms, exit with the T option, go into the editor and load (or create) the screen and save it. To edit a screen that already has colors set up, without changing the colors, first execute the screen file and then go to the editor with Apple-E.

When in a screen with different colors than the main screen, the refresh command will return to the main screen with the main screen’s colors. You can automate the return to the original colors by having an item in the screen which does a refresh, i.e., an item with any prefix and the single character R as application.

Although there is nothing different in screens than in a regular ProSel- 16 listing, the demo screens show how one can make a screen look quite different from a standard application list. (The same things can be done with any ProSel- 16 listing, however.)

There is a command (^B anywhere in any application specification screen title) that will cause the usual bottom of the ProSel- 16 screen to be hidden from view and enabling regular titles going all the way to the bottom of the screen. To put titles below the usual bottom boundary, you use the ProSel- 16 editor, place the cursor on the lowest line it will go to, and hold the Option key down while pressing the ↓. key. This will “release the lock” and permit the cursor to go down to any line below the usual bottom.

There is another command (^C in any title) that disables the usual ESC and number key usage in ProSel. This is so you can force selection of some item on the screen and disallow exit through the “back door”.

For further ideas about how “screens” can be used, go through the screen demo. Also, to see how some things are done, use the editor to look at the actual application specifications in any of the example screens. It is not recommended that you try this feature until you are thoroughly familiar with other aspects of ProSel-16.

Display features in a screen title

You can use inverse text and Mousetext in screen titles as follows. (Also see ‘Design mode” under “Application specifications and the editor”. The designer makes most of the following automatic.)

First note that to insert a control character that otherwise would be taken as a command by the editor, you press the ^0 key (this means the control-0 key and not the two keys ^ and 0), and then the control character you want to insert.

To make an entire title show up in inverse text, just put in the @ character at the beginning of the title.

To turn on inverse for just a portion of a title, insert a ^0 at the point you want inverse to start, and “N where it should end.

To use mousetext in a screen title, turn mousetext on by inserting the character] (this is the ESC key). To turn it back off use ^X.

NOTE: The usual signal to turn mousetext on is the sequence ^[^0, and to turn it off is ^X^N, and these sequences must be used in ProSel-8, but ProSel-16 accepts the simpler commands.

Here is a brief summary of the command characters that tell ProSel (16 or 8) to take special actions when showing the main screen.

^B: If this is in any screen title, then ProSel will not display the text usually at the bottom of the screen, and regular application titles can be put there.

^A^N: This sequence in a screen title defeats inverse that is normally shown when the cursor is on that item. Another “A turns it back on for the remainder of that title.

^C: This character in any screen title defeats use of the number keys and ESC key while that screen is showing.

If you use these things to make fancy titles, just remember that for a title to be “active” it must start with a letter (ascii at least $40). To make the letter invisible, follow it with a backspace (type ^0^ H in the editor) and then whatever you want seen.

Switching between ProSel-1 6 and ProSel-8

To switch from ProSel-16 to ProSel-8, so that P8 applications will return to ProSel-8 and not ProSel-16, just execute the PROSEL.SYSTEM file. For example, this specification does that:

Screen title:	ProSel-8
Prefix:	*?*
Application:	PROSEL.SYSTEM
Start up:	<empty>

You can also go directly to a P8 application from ProSel- 16 with return to ProSel-8 by using the same specification but putting the pathname of the application in the startup part of the specs. If the application is run from a directory listing (after pressing a number key) then holding SHIFT down for a moment while RETURN is pressed, will have the same effect.

To switch from ProSel-8 to ProSel- 16, just execute the SYSTEM/START program (assuming ProSel-16 was installed the recommended way). For example, this ProSel-8 specification does that:

Screen title:	ProSel-16
Prefix:	*?*
Application:	SYSTEM/START
Startup:	<empty>

These provisions are less important today than they were when passage between GS/OS and P8 was much slower.

Part III. Disk Management

The File Finder

When you select this option from the main menu, you will be asked for a device number to search. The boot volume is always device #1 (also see “Line input’), and the others can be found by using the Show prefixes option at the main menu.

After you give the device number, you will be asked for a file name (or a string, see below). The Find File routine will then search through the entire volume for this file. When found, the name of the directory containing the file will be shown and the catalog data concerning the file will be displayed.

At this point you will be asked to press a key. The RETURN key will cause the search to continue for another file of the same name in another directory. The ESC key aborts the search. The ‘D” key produces a file dump in hex and ascii. You may select printer output of this. The “T” key “types” the file. In this case control characters other than carriage returns will be shown in inverse, or ignored if output is to a printer. If the file is a BAS type file then you can also use the “L” key to produce a formatted program listing in which all statements are shown on separate lines and loops and conditionals are indented.

When using any of these modes you can make the listing pause or step by pressing the space bar or you can abort the listing with the ESC key.

You can use ? anywhere in a file name as a wild card for any single character, and you can use as a wild card for any string of characters. More than one? or can be used.

If the file is executable then you also have the option of running the program with the “R” key.

You can also use this to run a BASIC program. For this to work you must have specified, in Modify parameters, the location of the BASIC.SYSTEM file. When a BASIC program is run this way, the prefix will be set to the directory containing the program. This directory does not have to contain BASIC.SYSTEM.

The File Finder can also find files containing a specified text string. To do this you simply type the string at the “filename or string” prompt, but precede it with a quotation mark (single or double).

The program will then ask for a directory to search. If you just press RETURN at this point, the entire disk will be searched. If you specify a directory then only that directory, and its subdirectories, will be searched.

You can then specify a filetype to search. For example, if you know that what you are looking for is in a file with filetype TXT then typing TXT at this point will limit the search to those files, substantially speeding up the search. If you want to search all files then just press RETURN at this prompt.

Then you are asked for a minimum revision date of the files to be searched. If you know that the item you want is in a file that was revised since a certain date, then giving that date will speed up the search. If you just press RETURN then dates will be ignored.

Finally, File Finder asks if it should display the filenames of the files it is currently searching. Answering N will slightly speed up the search if you are doing a search through an entire hard disk, but Y makes it more interesting to watch.

When the string is found, the program will display the catalog information about the file, and will display the portion of the file surrounding the found text, and highlight the found string. The string will be around the middle of the screen, unless it is very close to the beginning of the file, in which case it will be closer to the top of the screen. In graphic mode the siring is in highlight color 2 (usually blue) which makes it easily distinguishable from the control characters which show in highlight color I (red).

You will be told the “byte offset’ of the displayed area in the file, and asked if you want to continue the search or not, and, if so, whether you want to skip to the next file or continue with the present one. The byte offset can be used with the Type function to quickly go to the location of the found string, if you do a subsequent file search for the found file. Note that this offset is to the first byte of the displayed area, and not to the found string.

The string search is not case or “high bit” sensitive.

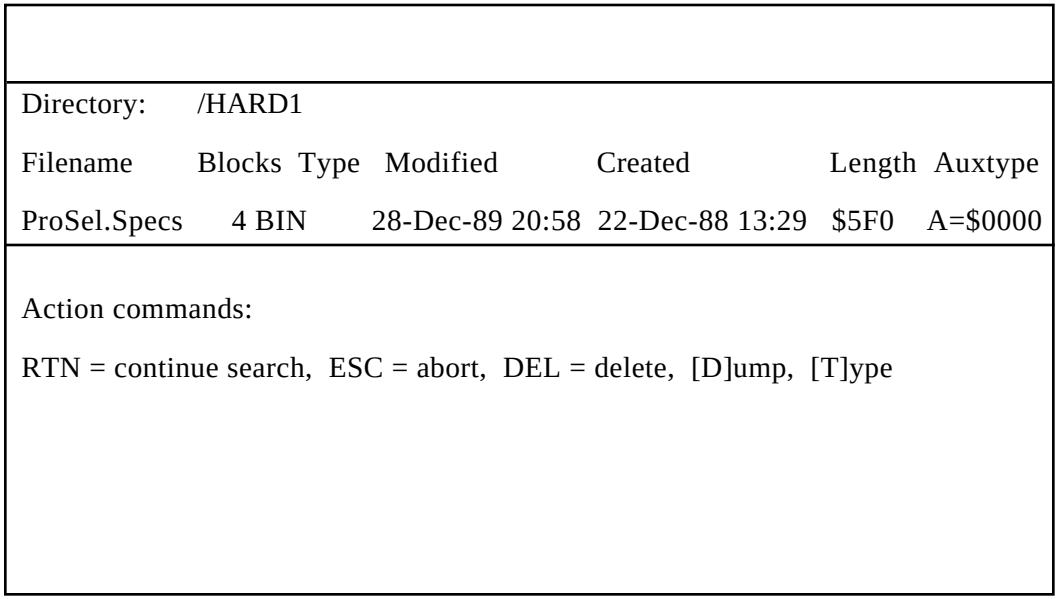


Figure 6. *Typical* File *Finder screen.*

☼_Information Desk

This module will print the entire tree structure of a selected volume or directory. The output can be sent to the 80 column screen, to a printer or to a disk file.

There are five main parts to Info Desk that give you different types of documentation of the files on your disk:

1. *Catalog.*

   This mode prints a tree structured catalog of the entire volume. Subdirectory contents are indented two spaces. All the usual catalog information is printed except the time of day and the access (locked) status. Access status is printed if the line length is set to 90 or more. The time of day in the dates is printed if the line length is set to 102 or more, in Modify parameters.

2. *Block usage by files.*

   This mode tells you just what blocks on the disk are used by each file. The printout gives the number of blocks in each file (as in mode 1). Then comes the Index field. This contains the number of the index block. This field is empty for directory files and for seedling files, which have no index block.

   Next come the actual data blocks. If two numbers in this list are separated by periods then they represent a range of blocks all belonging to that file.

   In case of a tree file (length > $20000 = 131,072) the first entry in the Index column is the master index block indicated by "< = = (Master index block)" next to the block number. After this, on subsequent lines, are the index blocks pointed to by the master index block, and their associated data blocks.

   At the end there is a "fragmentation" count given. This is a count of blocks that are not in line with the rest of the file and gives a rough indication of when an optimization might be called for. There is also a count of fragmented files.

3. *File usage by blocks.*

   This mode is the "reverse" of mode 2. It shows which files belong to the blocks on the disk. Most of the disk operations are done prior to any printing, so be patient. The printout consists of ranges of blocks in numerical order followed by the file name (the full pathname less the volume name) of the file which uses these blocks.

4. *Bit map.*

   This mode prints the volume bit map. Used blocks are shown with "x" and free ones with ".", but these are user definable in Modify parameters.

5. *Tree structure.*

   This mode prints a tree diagram of the volume.

There is also a sixth option that is the same as option 2 except that it only prints the summary information at the end and is much faster, and has a suboption that lists only the fragmented files. There are also seventh and eighth options which are the same as options 2 and 4 except that they have a super-hires graphic display. Options 7 and 8 are available only if you choose output to the screen. The pixel size in options 7 and 8 depends on the volume size being mapped. A volume of 16MB or over will have the smallest pixel size. 3.5 inch disks will have a large pixel size resembling low resolution graphics. The TAB key will toggle between the graphics display and the text display.

Since there are only 64,000 pixels in super hires compared to a possible volume size of 65,535, the last 1,535 blocks will be ignored in the display whether they are used or not. The graphic mode #7 can be used to observe the level of fragmentation of a disk by how much jumping around is done by the plotting. Note, however, that the optimizer puts all directory blocks at the beginning so that there will be normal jumping between the current location and the first couple of lines of the display. Also, since tree files go at the end, some display of large numbers of blocks near the end of the display will be normal for optimized volumes. Directory blocks are plotted in white, and other blocks in 14 other colors changing with each new file. The area of white at the bottom of the screen represents block numbers beyond the volume size and thus makes possible a visual representation of the free area on the disk.

At any time during printout the space bar will stop, then step, the process, and ESC will abort it. When printing to the screen is finished, the program waits for a keypress before clearing the screen and asking if you want to do another.

If you choose the option to output the data to a disk file the file will be placed in the prefix *#5* directory. You can use Modify parameters to set this prefix to anything you want, and you can change it while in ProSel by using the shell prefix command. The filename used when the Catalog option (#1) is in effect is “CAT.” followed by the volume name. (If this is longer than 15 characters then it is truncated.) If this file already exists it will be overwritten unless it is locked. If it is locked, the File Finder will abort. Similarly, for the options 2-5 the file name is “LOC.”, “BLK.”, “BIT.”, or “TRE.” followed by the volume name.

Info Desk also accepts autokeys (see Volume Repair) and is scriptable. The following specification automatically runs all five main modes on device #1 writing to disk files (the M’s are taken as carriage returns):

Screen title: Info Desk (auto)
Prefix: ?
Application: I
Startup: 121MMYMMZMMYMM3MMYMM4MYMM5MMN

Backup/Restore

This is a fast full volume or incremental backup facility. You can backup the files on any volume or those in a subdirectory, and can choose to backup all the files, only those changed since the last backup (determined by “backup bits”) or those whose modification date is later than some specified date. It is “file oriented’ so that the volume size can be changed between backup and restore. At this time, 3.5 inch disks are the only accepted backup media. The program does a moderate amount of file compression, enough to save some disks and time but not enough to seriously degrade performance or robustness.

NOVICE INSTRUCTIONS for initial full backup (after installation):

- *Go into Modify parameters (Apple-M <RTN>) and press RTN until the backup parameters appear (on the second page of the parameters).*
- *Make the “Use error correction in backup” parameter Yes”.*
- *1f you have less than 1.5MB memory, make the “Confine backup memory use to 400K’ parameter “Yes “. Otherwise make it “No”.*
- *Make all other backup parameters “No”, then exit Modify parameters with ESC and then “P”to make these parameters permanent.*
- *Enter the Backup/Restore module by selecting it from the screen or by pressing Apple-B <RTN>.*
- *Press the B key for “Backup”.*
- *Type the name of the volume you wish to backup, with no “/”. For example, type HARD1 <RTN> ~f the name of the volume is /HARD 1/.*
- *Press the A key for “AlI files.”*
- *That is it. except for inserting the backup disks (a new set of disks is strongly recommended first time) into one or both of your 3.5 inch drives. Do not worry about missing an insertion; the program will wait for you if you forget.*

In order to maximize the backup speed, the program requires enough free space to hold the contents of a 3.5 inch disk. There is an option in the parameters that will cause it to use only 400K of memory and partition the backup disks into two parts. People with less than 1.5MB of memory should use that option. If this memory is not available then you will be told at the start. The backup will operate at about 3/4 MB per minute (substantially faster than any other backup program), but this depends on the speed of the media used. Choosing the verify option will almost double the backup time. You are strongly advised to use only good quality backup disks and to discard any that cause the slightest trouble.

There are some tiles that the program will not back up. These are read disabled files, bad block files, and “Pascal area” files. The skipped files are displayed before the reading of files for each backup disk. If backup bits are cleared then these filenames will again be displayed at the end of the backup process.

Backup will use either the first, or the first two, of the *3.5* inch drives that you have. This is fully automatic and you just have to insert disks into one or both of those drives.

One should make a full volume backup every week or two. You can use this facility for that or the ProSel-8 backup. If you use the latter, you should choose the “clear backup bits” option. The ProSel- 16 backup always clears backup bits of the files it backs up, except for the “compound” backup bits mode. This is done only at the end of a successful backup.

Using separate backup disk sets for each day, an incremental backup by “backup bits” is recommended at the end of every day. If a restore should be necessary, you can go back to any one of these sets of backup disks (including the full backup) and do a restore, followed by restores of all subsequent incremental backups in the same sequence in which the backups were made.

The “compound” mode is identical to the “backup bits” mode except for the clearing of backup bits. If you use this mode then every backup since the last full backup will backup all files changed since that last full backup. This allows you to use only two sets of backup disks - the full backup set and the incremental set.

You must not mix the three incremental modes - choose one to use and stick to it.

Backup disks are named BU.01, BU.02, etc. Best results will be had from media formatted with the default 2:1 interleave.

You do not have to reformat or erase the backed up volume before doing a restore if the volume has no present problems. However, if there is some problem on the disk (damaged files or directories), then you should erase the disk before a restore. The restore function, when done on all files of a volume directory will ask you if you wish to erase. If you have a bad block file, then you should **not** use the erase option. Instead, erase the disk by another means and use Volume Repair to rewrite the bad block file. When you do this, you should first boot the ProSel disk from a 3.5 inch drive.

> *NOTE: Erasing the disk will also remove the protection of the DOS.Master partition if you have one. It will not harm the data in the partition, but that area will be open to subsequent overwrite. To reprotect it you can use Zap to zero out the bit map (starting at block 6 and using one block for every two megabytes), and then use Volume Repair in F mode to free up all the blocks not in the partition. This requires the DOS. 3.3 file to be on the disk (see the doc for the Volume Repair module).*

When you enter the Restore routine, you will be informed of the directory that was backed up onto the present media and of the mode of backup (all files, by backup bits, or by modification dates).

Some read/write errors are not recoverable and will force an abort of the program. Must, however, will allow a retry or a “continue”. In the latter case the problem file is just ignored.

If the program wants a backup disk and there is no disk in the backup device, you will be prompted to insert the proper disk. If there is a disk in the backup device but having the wrong name, you will be asked if it can be “destroyed”. If you answer N, you will be prompted for the correct disk. If you answer Y then the disk will be written to and will inherit the proper backup disk name. You can just replace the disk with the correct one and answer Y to this.

The “selective recover” allows recovery of individual files from a single backup disk. It will ask for one of the disks, will read it, and display a list of backed up files on it. Use the ↓ and ↑ keys to scroll through the list, the space bar to mark files for recovery, and press RETURN to start recovery of the marked files. Press ESC to abort. If the last file on the disk is marked then the program will also read the next backup disk in case the file is continued there. While selecting, the ^A key toggles all marks.

Backup/Restore can be automated by puffing the desired key sequence in the startup of a specification for backup. The first character, selecting Backup or Restore can be followed by a semicolon to separate it from the following directory name. Otherwise, a semicolon will be interpreted as a RTN. Similarly, a "<" character in the input of the directory name will move the cursor to the start (useful when using a script with the Restore function). For example, you can set up this specification:

Screen title:	Backup, incremental
Prefix:	?
Application:	B
Start up:	B;HARD1 ;C

Selection of that from the screen will go into the backup routine, provide the directory name HARD 1, select the compound backup bits mode and begin the backup.

Similarly, the following specification will cause a full backup of the /HARD1 volume:

Screen title:	Backup, full
Prefix:	?
Application:	B
Startup:	B;HARDl ;A

A disk access error during backup or restore will turn off the automatic mode.

The “clear backup bits” option is intended for use by people using another backup method which does not clear backup bits (such as SiderFile) for full backups and the present system for incremental backups. In this case you should use the clear backup bits option immediately after doing the full backup. This is automatic when using the present system for doing the full backup.

The “directory” option lets you see what files are on a particular backup disk. (Selective recover does the same but not as fast.) Pressing RETURN at the end causes a return to the start of the backup program instead of exiting. ESCAPE exits. Pressing "*" when the directory is on the screen will cause it to be sent to the printer.

The “update” option restores only those files on the backup disk that either do not exist on the destination volume or have a later modification date than on the destination volume. This facility is intended for using backup disks to transfer files from one hard disk to another one. It could also be used to recover deleted files from a set of backup disks made before the deletions.

The “script” option allows you to backup via a list of files and directories. To access this option, just give the name of the script file when asked for the name of the directory to be backed up. The prefix upon entering the backup module will serve as the prefix throughout the backup. The script file must be a standard text file. (It can be written. for example, with the ProSel- 16 text editor, or with any text editor that can write standard text files.) It must contain the names of the files to be backed up. Each name must be a partial pathname (which will be added to the prefix) and must be terminated by a carriage return. The script can also contain directory names, provided they are also partial pathnames, in which case all files in the given directories and their subdirectories will be backed up. You can use any of the backup options of all files, files with backup bits set (compound or not) or files revised after a given date. The script file must not be one of the files to be backed up nor in any of the directories specified in the script.

Since the prefix on entry to the scripted option is crucial, I suggest that you do not use this option via the keyboard, but rather from a screen specification such as:

Screen title:	Backup by script
Prefix:	/HARDl/MYDIR
Application:	B
Startup:	B;HARD1/PROSEL.16/BACKUP.SCRIPT;A

The script file BACKUP.SCRIPT could contain, for example, the lines:

```
WP.FILES
DB.FILES
DATA.FILE1
DATA.FILE2
```

Then selection of this item will automatically backup all files in the directories /HARDl/MYDIR/WP.FILES and /HARD1/MYDIR/DB.FILES and also the single files DATA.FILE1 and DATA.FILE2. If the specification Startup had ended with C instead of A then only the files, among those, that have changed since the last full backup will be backed up.

Note that the script file name given in the Startup is a full pathname but without the “/” at the start. This is necessary because the directory name prompt accepts only full pathnames but supplies the first “/” automatically.

There is an option (in Modify parameters) to force formatting of the backup disks. This is convenient for a new set of disks and it is also a good idea to use this option every two months or so to prevent gradual degradation of the backup disks.

There is an option (in Modify parameters) to use error correction on the backup disks. This will correctly restore from backup disks that have a few unreadable or corrupted blocks. If the error correction is unsuccessful, an error $27 results. The error correction has a penalty of using about 10% more backup disks, and the corresponding time necessary to write to those disks. The added reliability is definitely worth it, and I advise you strongly to use this option.

Success of the error correction depends, of course, on the number of bad blocks on the backup disk. If the bad blocks are randomly distributed then the chances of a successful recovery are given by the following table:

Number of bad blocks	*Chance of successful* error *correction*
3	98%
5	94%
10	75%
15	51%
20	29%
25	14%
30	5%

Figure *7. Error correction statistics.*

These statistics were theoretically derived. Actual tests indicate somewhat better chances, probably due to the fact that non-randomness would tend to improve the chances. For example, several bad tracks in a row would always be recoverable unless one of them is the first track on the disk. (The first three blocks contain vital information and must be readable for error correction to succeed.) Similarly, you could touch the disk surface with a strong quarter inch bar magnet and have a 100% chance of recovery, unless you happened to hit one of the first three blocks. In the latter case, if you get the message “Insert BU.xx...” and the correct disk is in the drive, press the option key to force acceptance of the disk (only in error correction mode).

There are actually two varieties of error correction in the backup program, one for dealing with bad blocks and another for dealing with corrupted data. Thus it is possible to see the message “Backup disk bad - attempting fix...” twice in succession. If you are desperate and willing to live with corrupted data, you can defeat the second of these integrity tests by putting the byte $FF in the first byte of block 0 of the backup disk. The first test can be defeated only by copying the good blocks to a totally readable disk. Neither of these “back doors” should be attempted except in desperation.

Through Modify parameters, you may elect to skip certain filetypes in backing up. You might choose, for example, to skip the system files and perhaps application files (SYS and S 16 types) since those would be easily replaced. This would speed up the backup process and require fewer disks, but it would also make a full restoration of the disk more difficult. You can also elect to skip “hidden” files, a feature that lets you skip individual files in the backup.

NOTE: Backup/restore treats forked files as two separate files. Selecting the data fork in selective recover mode is sufficient to have both forks restored. Selecting the resource fork will only restore that fork and will have no effect on the data fork. The data fork is always the first one displayed in a directory or the selective recover list.

There is a parameter which will cause Backup to write a directory of all backup disks to the prefix #5 directory. The name of the file will be the name of the directory backed up preceded by “X.” where X=A,B,C,D is the type of backup. The file can be on the volume currently being backed up but, in that case, that file will not be backed up and will be marked as “read disabled”. Each item in the file is preceded by the number of the backup disk.

There is another parameter to “abbreviate” this file (mode A only) that will result in two dots replacing each subdirectory segment of the pathname, producing a file of about half the size.

In graphics mode, and when the backup directory parameter is active, a display of the date of the last full backup of any volume is printed after the input of the volume name. The date is taken from the directory file.

In graphics mode a running file count and a megabyte count are shown together with an estimate of the number of disks needed. The latter is usually mote accurate than the number given at the top of the screen (which is for a full backup only) since it uses the actual sizes of the files to be backed up. It is still an estimate since it is impossible to predict the amount of compression in a reasonable time.

The backup module in version 8.2 and later is incompatible with earlier versions of ProSel- 16. Other than that they have been upward compatible.

The ProSel- 16 Backup/Restore is, by a good margin, the fastest, most advanced and most reliable backup system in existence for the Apple II line.

☼ Zap (Block Warden)

The Zap module is a block editing facility. When “Zap” is selected from the menu, block 2 of the boot disk will be read. The module is entered in “R/W mode”. Only one memory page of a block is shown at any time and you must use the “\” key to flip between the two pages of a block in both R/W mode and EDIT mode. (In EDIT mode however, moving the cursor will automatically adjust the page displayed.) Figure 8 shows a typical Zap display.

◊ *R/W MODE*

In this mode of Block Warden the → and ← keys read the next or previous block. The ↓ and ↑ keys read the next or previous pages.

Pressing → or ← in R/W mode while holding down the Option key will act as if the arrow is pressed 10 times. The Apple key will do the same but 100 times. This is active whether in follow mode or not. Note that if following a sparse file, the next block read may be beyond 10 or 100 blocks farther in the file since the arrows skip over missing blocks in sparse files. The ↑ and ↓ keys are not affected.

The Q key will ask if you want to quit.

The R command lets you specify the next block to be read. Block input is in hex unless you backspace over the $ sign, at which point you can enter a decimal number.

The W command will allow changing the block. (just press RETURN for no change.) Then it will ask if you really want to write the block, as an extra safety to prevent accidents. Any write command will force a warm shutdown when you quit from Block Warden since otherwise GS/OS may get confused and create problems on the disk.

The C command lets you change the disk device being accessed. Note that this command will not read from the disk, so that you can transfer a block from one disk to another by use of it. Subsequent reads, however, will come from the new disk.

The P command lets you specify the prefix. This is mostly for use with the F and I commands.

The F command asks for a pathname (full or partial) to be followed. After issuing this command, the name of the file being followed will show at the top of the screen and the an-ow keys will move between blocks of the file. Pressing the ESCAPE key (and some other commands) will cancel the follow mode. This facility supports all file types including directory files and sparse files. It reads only the data blocks of a file, so it cannot be used to look at the file’s index blocks. While following a file you may go to edit mode, edit, return to R/W mode, write a block, and continue to follow using the arrows.

The I command asks for a file name and then shows the file parameters which appear in the directory entry of the file in a form that is easy to read. The bytes on the left give the offset location of the data in the block buffer. When you press a key, the program reverts to the block in effect when the command was issued. If the file is a subdirectory, however, the header of the subdirectory is shown after the first key press. You must note the block number containing this information if you are going to want to use the editor to change it.

The L command disassembles the current buffer contents. [In follow mode with a TXT file it lists in ascii instead. This can be forced in follow mode for any file type with the " command.] The ascii equivalents of bytes appear after the disassembly line. A total of 40 lines are listed on each page. You are asked for a starting byte (000- 1FF) to start the disassembly on. If you just press RTN this defaults to zero. During disassembly the → key or ↑ key or RTN produces the next page of the disassembly; another key cancels this mode. If you are following a file then the → or ↑ keys pass to the next block of the file when the buffer listing is done, and that block is listed. While in the List mode you can dump the screen to a printer by pressing Apple-D. This assumes the printer is on and in slot 1. (If a previous Dump command was issued and you selected a different slot for the dump then that slot will be used instead.) The disassembly supports the whole 65816 opcode set. The “M and X flags” attempt to follow the program logic, as in the Merlin- 16 + assembler. At the start of each page these flags can be reset by using the RTN key instead of the arrow keys and pressing, at the same time, the Apple key to set M=0 and/or the Option key to set X=0.

The D command dumps the buffer contents to a printer which is assumed to be in the slot of the number key you press when you are prompted to turn the printer on. You can abort the command by pressing the ESCAPE key.

The * key displays a catalog of the current prefix directory.

Finally, the command sends a list to the printer of all possible ‘index blocks” on the disk (starting with block 7). This is intended as an aid in a last ditch attempt to repair a blown directory. It would be better if you have an up to date list made by INFO.DESK. Also note that if block 2 is bad (so that the volume name is shown as “?“) then this command will not work - it will print all the blocks on the disk. Thus you must repair block 2 first to the extent that it has a valid volume name and the correct number of blocks (bytes $29,$2A of block 2). The list printing can be aborted by pressing the ESC key. The list will include some blocks that are not really index blocks, mostly partially full blocks at the ends of files.

◊ *EDIT MODE*

In edit mode the arrow keys move the cursor (shown in inverse). Any key other than a control character will be regarded as a change to the editing buffer. (In hex mode it is disregarded if not a valid hex digit.)

The ESC key returns to R/W mode. The ^X key cancels any changes you may have made on the current block. (This is done by rereading the block.) The TAB key toggles between hex and ascii editing modes. You can tell what the current mode is by seeing where the cursor is placed. In ascii editing mode, the high bit of a typed character will be off unless you press the Apple key at the same time, in which case the high bit will be on. Control characters can be inserted into the buffer only in hex editing mode.

The F key is a lead-in to character find mode. If the cursor is in the hex portion of the screen then the cursor will disappear and you are expected to type two hex digits. (An invalid digit causes this mode to be canceled.) The resulting byte then becomes the "find character" and the next occurrence of it in the buffer will be found and the cursor moved there. (If none exists the cursor returns to the first byte and the "Find mode" message is erased.) After this first find (that is, when "Find mode" is on the screen) subsequent T commands find further instances of the find character in the buffer. This mode can be canceled with the RTN key. If the ^F key is issued when the cursor is in the ascii portion of the display then an ascii find character will be gathered. (High bit of the find character will be off unless the Apple key is pressed when the character is typed.)

The ^S key selects global search mode. When used the first time, this brings up a request to input a search string. If the string you give begins with "$" then it will be taken as a hex string for the search. Otherwise the string is taken as an ascii string. If FOLLOW mode is in effect then only the followed file will be searched for the string, otherwise the entire disk, from the present position, will be searched. The ascii search is not sensitive to the high bit of characters, nor is it case sensitive. (Thus, for example, a search string of "Junk" will find both "JUNK" and "junk".) You can cancel a search in progress with the ESC key while the disk is reading, or during input of the search string. Strings that cross block boundaries will be found by this facility. After the first instance of the search string is found, others will be searched for upon pressing the S key in edit mode. Any block read with the R/W mode Read command will cancel the search mode. You can also cancel it by pressing RTN in edit mode. The search mode with the same string can be reinstated by typing another S command in edit mode and just pressing RTN when the default string is shown.

If the Volume name at the top of the screen is shown to be "/?" this means that the program encountered an error in trying to determine the name. This almost certainly means that there is something wrong with block 2 (the first block shown). It could also mean, however, that the disk is not a ProDOS disk.

◊ *CHANGING STARTUP NAMES*

Block Warden has a built in feature to simplify entering Startup names in SYS files. To change the default Startup file name in a SYS file, follow these steps:

1. In R/W mode, press F to follow a file, and specify the appropriate file name. (You may want to use the P command first to set the prefix.)

2. When the first block of the file is read, use the E command to go to edit mode.

3. Use → six times to move the cursor to the startup position. At this point a box will be displayed asking for the startup name. (If not, then either the file is not of SYS type, you are not in follow mode, or the file does not support startups.)

4. Type the startup name ending with RETURN. When done you will see the name you entered has been placed in the edit buffer with the appropriate length byte supplied automatically.

5. Press ESC to go to R/W mode, then the W write command to write the buffer to disk, etc

```
Block: $0002 (2)       Vol: /PROSEL          Sunday2-Dec-90 7:15:44 PM
Prefix: /PROSEl/
Byte $000400
(c)↓    00 01 02 03 04 05 06 07 08 09 0A 0B CC 00 CE 0F    R/W mode
1991
        000: 00 00 03 00 F6 50 52 4F 53 45 4C 00 00 00 00 00 ....vPROSEL.....
    b   010: 00 00 00 00 00 00 7C B5 20 17 00 00 67 B4 14 01 ....,.|5 ...g4..
B   y   020: 05 CCC3 27 0D07 00 06 00 40 06 26 50 52 4F 44 ..C'..,.@.&PROD
L       030: 4F 53 00 00 00 00 00 00 00 00 00 FF 17 00 04 00 OS..............
O   G   040: C9 05 00 CE82 lB 09 00 B0 21 00 00 CE 82 lB 09 I..N2...01...N2..
C   l   050: 02 00 06 53 59 53 54 45 40 00 00 00 00 00 00 00 ..VSYSTEM......
K   e   060: 00 00 0F 07 00 01 00 00 02 00 82 AD 00 CC 00 BE ..............-..>
    n   070: 21 00 00 7C B5 26 11 02 00 09 50 52 4F 53 45 4C !..|15&..YPROSEL
W       080: 2E 31 36 00 00 00 00 00 00 00 0F 00 00 01 00 00 02 .16.............
A   B   090: 00 25 B2 00 14 00 B6 21 00 00 65 B5 29 CE 02 00 .%2...6!..e5)...
R   r   0A0: 2C 50 52 4F 53 45 4C 2E 53 50 45 43 53 00 00 00 ,PROSEL.SPECS..
D   e   0B0: 06 08 03 04 00 6A 05 00 8A B1 09 15 78 B6 E3 00 .....j...1..x6c.
E   d   0C0: 80 12 B5 1C 08 02 00 D4 55 54 49 4C 00 00 00 00 ..5....TUTIL.....
N   o   0D0: 00 00 00 00 00 00 00 0F 14 00 01 00 00 02 00 28 ..............  (
    n   CE0: 82 28 15 00 B8 21 00 00 7C B5 03 11 02 00 03 44 2+..8!..|5.....SD
        0F0: 4F 43 00 00 00 00 00 00 00 00 00 00 00 00 0F 15 OC..............
```

Figure 8. Typical Zap (Block Warden) screen.

SUMMARY of Block Warden commands:

>> R/W mode commands:

Q . . . . . quit to the ProSel-16 screen
E . . . . . go to edit mode
Arrows . . read next or previous block or page
R . . . . . read block (input)
W . . . . . write block (input)
I . . . . . get file info (input)
",L . . . . . list/disassemble buffer contents (input)
(continue list with right/up arrow or RTN)
(RTN with Apple keys operate MX flags)
P . . . . . set prefix (input)
F . . . . . set ‘follow” file name (input)
C . . . . . change device (input device #)
0 . . . . . dump block contents to printer
^ . . . . . , send index block list to printer
* . . . . . print catalog

Edit mode commands:

ESC . . . . return to R/W mode
Arrows., move cursor
^X . . . . . cancel changes to buffer
^F . . . . . find chr next typed in current buffer
(or continue find)
^S . . . . . global disk or file search for string
(or continue search)
RTN . . . . turn off find and search modes

(Non-control keys are taken as buffer edits.)

(The Apple key after AF in ascii mode or during buffer edits sets the high bit of the character typed.)

Optimizer

CAUTION: This utility is extremely dangerous. If it misfires in any way or if you have a glitch in your system such as a slight RAM problem or if you have a power outage while it is operating it can destroy the entire volume it is working on. You absolutely must have a full backup before using it. Destruction may also occur if the disk contains bad blocks or damaged data in a directory. (Bad block files are respected by the optimizer, however, and will not cause difficulties.)

> I TAKE NO RESPONSIBILITY OF ANY KIND CONCERNING THE PROPER WORKING OF THE OPTIMIZER. YOU ARE ON YOUR OWN.
>
> > **Make a BACKUP first** !!! <

The volume optimizer will put all directories at the beginning of the volume, and all files will have contiguous data blocks. This makes for much more efficient file access.

The optimizer will ask if you want “complete” or “turbo” mode. Complete mode optimizes directories as well as files and requires no space on the disk. Complete mode should be used the first time and any time after a restoration after a formatting or erasing. Turbo mode optimizes only files and is very fast. It requires at least one megabyte of free space at the top of the volume (not counting a “Pascal area” or a DOS Master partition) and is not as dangerous as complete mode. If many files are fragmented, turbo mode may not have enough space to handle them all. Another pass will defragment more of them, but generally the complete mode should be used in those cases. There is a parameter in Modify, parameters that will limit the Turbo mode to optimize only those files past a certain cutoff of fragmentation. If 0, files with any fragmentation at all will be optimized. A value of 5 or 10 is a reasonable limit, which will speed up Turbo mode.

Next, the optimizer will ask for a disk device number. It will then read and show the volume name in that device and ask if it is all right to continue.

Complete optimization will take a number of minutes. If you absolutely must abort the program, you can use the escape key. The program will take a few moments to react to this key because it will only quit at a time when the disk is “clean”, and must also do some last moment writing to disk at this point. If you later want to continue, the program will quickly arrive at the spot where it stopped, for the simple reason that it has little to do before that spot. Do not interrupt the program with RESET.

Complete optimization makes 4 “passes”, numbered 0 to 3, through the directory. The brief pass number 0 just reads data from the disk, and an error at this phase will not affect disk contents. Pass number I relocates the directory and is not very long. From this point, the screen will show the current block being processed. Pass number 2 relocates all files except for tree files. This pass takes the great majority of the time used by the program. Finally, pass number 3 relocates tree files, if any. All tree files will be relocated following all standard files.

1 have gone to great lengths to make the program as fast as I can. One to three megabytes per minute is typical. It has been known to run at over 5 megabytes per minute in some cases. In any case, it is much faster than Beach Comber under ProDOS 8 or any other available Apple II optimizer.

The optimizer goes through directories and files in the same order as in lnfo.Desk. Thus, to speed later uses of the optimizer, you can put directories containing files that can be expected to change, late in this order, i.e., late in the main directory. Put stable directories early. Of course, this may not be practical for all people and it may not speed things up all that much anyway.

If the optimizer encounters a problem, it will shut down and display an error message. Some error messages like “Block Read” error are self-explanatory. Others like “Block Reference” error are rather technical and just generally indicate some defect in the volume (or may be caused by a hardware problem). One error that seems to occur more than the others is a “Block in use” error. This error occurs only on the initial pass before any writing is done to the disk. It indicates that there are two files, or a file and a directory or bit map block, that contain the same block. Thus one or both of the files has been corrupted. This is a serious error and this condition should be dealt with as soon as possible and as carefully as possible. Do not write to the disk until it is cured. Volume Repair should report the same error, and will indicate one of the two bad files. Info Desk can be used to determine the other file. Such files **must not** be deleted by ordinary means. Use the “problem file deletion” method described at the end of the documentation for Utilities.

NOTE: There is a very serious problem in GS/OS that can utterly destroy a volume after an optimization of the boot volume. To avoid this problem, a shutdown is forced after an optimization of the boot volume.

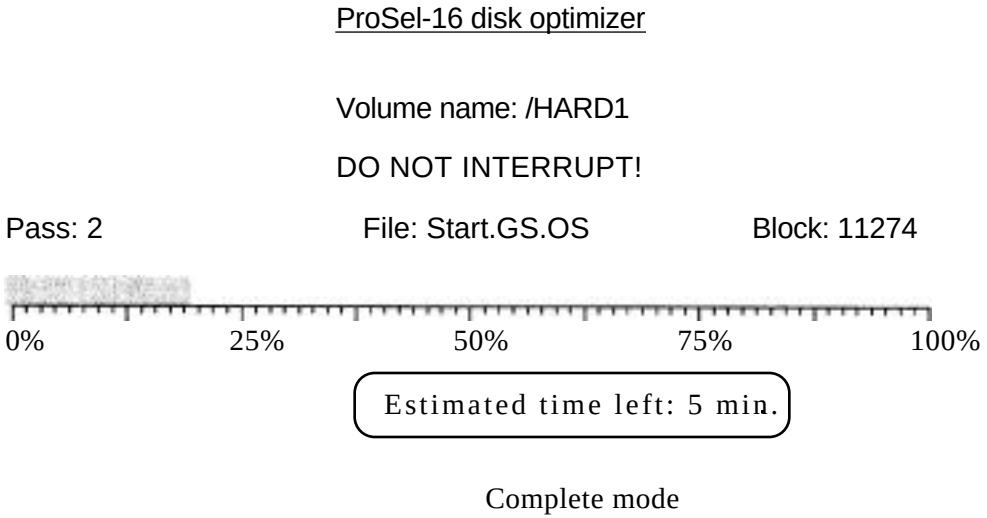


Figure 9. *Optimizer screen.*

☼ Volume Repair

This is a directory repair utility corresponding to Mr.Fixit in ProSel-8. There are seven modes: a Test mode in which nothing is altered on the disk; a Fix mode which attempts to correct defects found in the directory structure, etc.; a Main directory mode which attempts to reconstruct the main directory; a Bad blocks mode which scans for bad blocks and, if desired, places them in a bad block file; a Recover lost tiles mode; a Zero unused blocks mode; and a Statistics mode. The Fix and Main directory modes are dangerous and can change a bad situation to a worse one, so never use them without first using the test mode and never use them without an adequate backup to fall back on.

◊ *TEST AND FIX NODES*

The program tests and, if fix mode is active, fixes the following defects on any ProDOS volume:

1. Header pointers of all active files.
2. Parent pointers and parent entry numbers of all subdirectories.
3. Backwards directory links.
4. Used blocks marked free in the bit map.
5. Illegal characters in file names.
6. Entry length (the program assumes this should be $27).
7. Number of entries per block (assumes this should be 13).
8. File count in each directory.
9. Incomplete deletes (deleted files with nonzero “name length”).
10. Incorrect directory storage types.
11. File and directory block counts.
12. File and directory dates and times.

In file names (point 5) lower case characters are converted to upper case, high bits are stripped, and other illegal characters are replaced by ’7’. Illegal dates and times are zeroed. [Note that even though GS/OS now supports lower case in displayed file names, the names in directory files are still all upper case.]

The following items are checked and reported, but not acted upon:

13. Blocks used by two or more files (“block in use” error).
14. Block number out of range (past volume size).
15. Unknown storage types.

Some errors result in files or directories being skipped over. You will be told if this occurs. It happens because of information damaged in such a way that the situation cannot be handled, or the damage is such that the supposition is that the remaining data is invalid.

Some block read/write errors cannot be handled and will result in early termination of the program.

Sometimes, in the error printout, you may see a directory name printed twice. One of them refers to the directory “header”, the other to the “parent block”.

Volume Repair will also look for blocks that are marked used on the volume bitmap but are not used by any file. You will be given the option, in Fix mode, of releasing these blocks. Sometimes areas of a volume are marked off without belonging to any file and in this case you should not ask that these blocks be freed. Examples are a DOS.Master area or a Pascal area. As a general rule, if the number of blocks indicated as being marked, but unused, is large, then you should assume that they are marked for a reason and should not free them.

MAIN DIRECTORY REPAIR MODE

The most important block on a ProDOS volume is block 2. Since it is accessed much more often than any other block, it is also the most likely to be damaged. If it is, you will probably see a message that block 2 is too damaged for the program to function. There is a special provision for attempting a repair of the main directory, and it is accessed automatically when you request Fix mode and block 2 has extensive damage. You can also force this mode by selecting M at the Test/Fix prompt. You will be asked if the program should assume that the “bit map” is valid (default = Yes). Ordinarily you should select Y unless there is some reason to believe the bit map is damaged. (E.g., if you are trying to resurrect the subdirectories after a disk has been “erased” by Utilities then the bit map will not be valid and you must select N at this prompt.)

When this mode has been selected by M or automatically because block 2 has been determined to be substantially damaged, then the program will tell you that this attempt is being made, and will give a few particulars along the way. Although the routine can resurrect most subdirectory pointers, it cannot do anything for standard (non-subdirectory) files in the main directory. While this routine is operating, every block on the volume will be read. This may take some time, so be patient. When it is finished, you will get the “Another?” message. Note that this routine does not do the other repair jobs mentioned above. You can run through the disk test again to check if those things are all right. You should realize, however, that this main directory repair routine expects most of the rest of the volume to be reasonably valid and normal. (E.g., it assumes that the “bit map” starts on block 6 if the main directory header has been damaged. This is true for almost all disks, but not for some RAM volumes; this routine should not be attempted on a RAM volume or any other volume you suspect is organized in an unusual way.)

Volume Repair can be used to resurrect the subdirectories in the main directory and all other files that are not in the main directory. (It is not possible to retrieve the vital pointers for other files in the main directory, although those files may still be intact on the disk somewhere.) This can be used on a disk that has been inadvertently erased or soft formatted (i.e., as long as the disk was not physically formatted). To do this, first use the M option to repair the main directory. Then use the T mode to check the types of remaining errors. (There should be a lot of block free errors, and a file Count error.) Then use the F mode to fix the remaining errors if there are no fatal errors. You must remember that this program makes decisions that sometimes may be inappropriate and it may leave some undetected problems.

◊ *BAD BLOCK MODE*

If you select the Bad block mode then you will be asked if you want only to test or to fix. If you select test then the volume will be scanned for bad blocks and the results reported. If you select fix then there will be an attempt to place the bad blocks found in a bad blocks file. This file will be created in the main volume directory (so be sure there is room for it). If a bad block is a data block and not a directory or index block then several attempts will be made to read it and relocate it. If it cannot be read then a fake block is substituted and marked with the message “DAMAGED BLOCK”.

◊ *FILE RECOVERY MODE*

The “file recovery” mode will search for files that have become “detached” (perhaps by the problem file deletion of a skipped directory). This mode should be used only after other problems on the disk have been dealt with. It will first run through the disk in test mode to gather information it needs. Then it will ask for the name of a directory on a different disk to write recovered files to. It will then make two passes through the disk looking for files that are otherwise lost. Any files found on the first pass will be completely recovered, but those on the second pass will be lacking of the information usually in the directory entry for the file, such as the exact length of the file, its name, its filetype, etc. Handwork on files of the second type may be necessary before they are usable. The filetype will almost definitely have to be changed from the NON type given to it by the recovery routine, and the length of the file may have to be cut down before the file is acceptable to whatever application program it belongs to.

During the search, the block currently being examined for file information is indicated at the top right hand corner of the screen. Those shown in the second highlight color (usually blue) are blocks pointing to recoverable files on the first pass. Those in the first highlight color (usually red) are the same during the second pass. Recovered files of the second type will be named “Recovered.000l”, etc.

If a recovered file overflows the destination disk, or any write error occurs, then that disk will be ejected (if it is ejectable) and the program will return to the prompt for a destination directory name. It will continue recovery using the new destination when you furnish that.

The best choice for the destination directory for file recovery would be a subdirectory on a hard disk volume with plenty of room. Second best would be a single subdirectory on an empty 3.5 inch disk. You should not use the root directory because the 51 file limit for root directories would likely be filled long before the disk itself runs out of room. It is advisable to have several fresh disks of this sort on hand before starting file recovery. They can all have the same name, such as “/NEWDISK/FILES”.

The lost file recovery routine cannot be used for reinstating deleted files. Use the exhume function of Utilities to do that.

ZERO UNUSED BLOCKS NODE

The “zero unused blocks” mode is for cleaning up the unused part of a disk so that old irrelevant information that may be there will not confuse the lost file recovery when it may be necessary to use that Sometime in the future. You should not use this facility on volumes that have a problem or from which you wish now to recover files. Any presently lost files will be destroyed by this function.

◊ *STATISTICS MODE*

The “statistics” mode of Volume Repair will analyze a hard disk for speed of access in a “linear mode”, reading successive blocks, a “random mode”, reading randomly ordered blocks, and an “Os overhead mode”, which times operating system and disk interface card overhead. The average time to access and read (except overhead mode) a block is reported for each of these. This mode of Volume Repair is “test Only” and never writes to disk. The tests begin as soon as the S key is pressed selecting this mode.

• • • • •

The Volume Repair can be automated by putting the desired key sequence after a semicolon in the startup position of a ProSel-16 specification for it, as in:

Screen title:	Volume Repair
Prefix:	*?*
Application:	V
Startup:	;1FS

This example will select disk device #1, Fix mode, and screen output. When done, it will exit to the main screen automatically unless there were errors, in which case the program will stop at the end so that you can see the final report of errors.

If you use my DOS.Master program, also put the pathname to the DOS.3.3 file in the startup (before the semicolon if you use one).

NOTE. A Shutdown may be forced after repair of the boot volume (only if some actual writing was done), because GS/OS will get confused and destroy the volume if this is not done.

> ***Do not use Volume Repair on Backup disks.*** *It can do no good and may cause damage. There are no files in standard file format on backup disks.*

☼ Utilities

The utilities section is a filing and directory manipulation utility. Its main function is to do batch copying, locking, unlocking, and deleting of files from specified directories. Most of the routines are limited to directories containing at most 761 entries, but more than 110 will cause truncation of file names on the display.

Throughout, pathnames must be given in their full form. The TAB key moves the cursor past the next "/", or to the end of the name. Apple-TAB moves the cursor back to the preceding "/". In some routines you can select the directory pathname by a tree search through all directories on a volume. When applicable this is indicated on the screen, and the ? key starts the process. If you type ? as the first character then disk devices will be scanned and the volumes on line will be displayed for selection for the tree search. If you type, or move the cursor over, the directory name and then press?, the device search will not be done and the tree search will be done on that directory.

Utilities maintains two default pathnames. These can be switched by pressing ^X when the cursor is on the first character of the first default name.

In addition to the general line editing features discussed before, the input of directory pathnames in Utilities has these features: Holding down the Apple key together with a number key from 0 to 9 will substitute the corresponding PREFIX for the default path given. When indicated, and when the cursor is on the first character during directory name input, pressing the Option key will bring up a request for the device number. The specified device will then be accessed through the tree search. As is true throughout ProSel-16, pressing the 1 key at the device prompt lets you select from a list of devices. (Also see "Line input".)

If you have Utilities on the ProSel-16 screen then you can set up its specs so that it will go directly into any of its parts, by putting the command key in the startup of the specs. For one of the items on the second menu of Utilities, precede the command key with "I" which will be taken as "TAB". For example, to have Utilities go directly to Volume copy, set the Startup of its specs "IV". Without the "I" this would cause it to go to Verify (on the first menu screen of Utilities) instead.

◊ *COPY FILES*

This is a batch file copier. It retains both creation dates and modification dates of files. It is compatible with all types of files including forked files, sparse files and subdirectory files. In the case of subdirectory files, however, you will be asked if you want the files inside to be copied - otherwise the new directory is created rather than copied, if it does not already exist. If "prompting" is OFF then the answer to this question will be taken automatically as "Yes".

You specify the source directory and the destination directory. Then a list of the files in the source directory is displayed. Move with the cursor keys to highlight the files you want copied and select them (or deselect them) by pressing the space bar. When ready to copy, press RETURN. If you decide not to copy any files, press ESCAPE.

You can select/deselect all files by pressing ^A. (Actually, this “toggles” all the flags so that previous selections will be deselected and vice-versa.) If “prompting” on the Utilities menu is OFF then files will be copied whether or not they exist or are locked on the destination directory. If prompting is ON then you will be prompted if a file of the same name exists on the destination directory, and you will be prompted again if it is locked. During prompting, the file name in question will blink. Files that are being copied or that have been copied are highlighted. When the copying is done you will be asked if you want to copy the same files to another disk with the same directory name. this allows you to back up files to several disks without going through the process of selection over and over. Also, if you press DELETE at this point, you will be sent to the delete routine at the point of file selection with the same files selected. If you then press RETURN, the original files will be deleted. (Note: This question is not asked if files inside subdirectories have been copied, because needed data is no longer in memory.)

When files have been selected, you can type ^C (for “changed” files), and the program will then automatically copy only those files among the selected files whose modification date/time is later than that of the same files on the destination directory, and all files that do not exist on the destination directory. The copying will start immediately without need of pressing the RTN key. Note that if you do not select any files before using ^C then none will be copied. To copy all changed files, use ^A then ^C. The ^E (exists) command works the same as C but copies only the modified files that afready exist on the destination directory. The ^B key is similar but copies only those files whose “backup bit” is set, and clears that bit. The DELETE key works in a similar manner but will mark for deletion from the source directory all selected files that exist on the “destination” directory with at least as late a revision date. (Note that although this is in the COPY routine, this use of DELETE does not copy anything.) In contrast to the ^T, ^E and^B commands, DELETE does not take immediate action and leaves you in select mode, but remember this is now selection for deletion.

◊ *TYPE FILES*

This option will type text files (or any files) to the screen or printer. You can select any number of files for display. The catalog information for the file is shown just previous to the file display, unless disabled in Modify parms. Most keys will stop the display or restart it. The ESC key aborts the display and returns to the utilities menu. The RETURN key aborts listing for the current file and proceeds to the next selection if any. At the end of each file display, the program will stop for a key press before continuing with the next file, or to the utilities menu in the case of the last file. The S key slows the display.

◊ *LOCK. UNLOCK. HIDE. UNHIDE or DELETE FILES*

These operate much as does COPY FILES, but only those files that are appropriate to the operation are displayed. (E.g., for LOCK, only unlocked tiles are displayed.) The delete function can delete entire directories, but you are warned first. You get only one warning even though you may have selected several directories for deletion. Unhide will only work if the parameter allowing hidden files to be seen in Utilities is set (see Modify parameters).

◊ *RENAME A FILE or VOLUME NAME*

After selection, file names are displayed at the bottom of the screen and can be changed from the keyboard. In order to change the VOLUME name, just select some file in the volume directory for renaming and, when presented with that file, move the cursor back and change the volume name. Press RTN when the cursor is at the end of the changed volume name. This will change the volume name and then present the file again with the new volume name. Just press ESC if you don't want to change the file name.

◊ *EXHUME FILES*

This function lets you revive deleted files, provided the files have not been overwritten. It might be wise to use Volume Repair to check for any problems if anything strange happens with this function. An exhume done on the boot volume will probably cause a forced reboot to avoid serious problems that would otherwise occur. If an error on a file occurs, the routine will just continue with the rest of the files. You can tell this only from the absence of the file from the directory after the routine has finished.

◊ *VERIFY FILES*

This will read specified files to test for bad blocks. Directory files are not themselves verified, but the files inside directories will be verified if you ask for that to be done (automatic if prompts are oft). [You could use Zap to read through a directory file if you are having trouble with one.]

◊ *DUMP FILES*

This will do a hex and ascii dump of files (mainly of use to programmers).

◊ *CREATE DIRECTORY*

This lets you create new subdirectories. It will even create multiple subdirectories. For example, if you tell it to make a directory called /HARD1/DIR1/SUB3 and DIR1 does not exist on /HARD1, then DIR1 will be created and a subdirectory SUB3 created inside it.

◊ *SORT DIRECTORY*

This powerful directory sorter asks for a directory name and then displays the names in that directory. Some instructions appear at the bottom of the screen. It accepts the following commands:

A - sort alphabetically
C - sort by creation date
M- sort by modification date
T - sort by file type
P - sort by file type and alphabetically within a type
R - reverse present file order

You can also use the Apple key in conjunction with ↑ or ↓ to move file names around by hand.

When you are done, press RETURN (or press ESCAPE to abort). When you press RTN you will still be given a chance to abort before the sorted directory is written to disk.

◊ *FORMAT or ERASE A VOLUME*

These will erase all files on the designated volume. They give you a chance to change your mind before the action is taken. ERASE is like FORMAT but is much faster and assumes the disk has been previously formatted.

◊ *AUTO FORMAT*

This will automatically format any 3.5 inch disk and repeat until the ESC key is pressed. RETURN must be pressed to activate it (a safety feature). The volume name will be /NEWDISK with the default interleave.

CAUTION: This formats **any** disk in **any** 3.5 inch drive - do not have or put a disk in such a drive that you do not want destroyed.

◊ *CHANGE FILE DATE*

This routine lets you change the modification and creation dates on any file, even the volume date (which has a creation date only). You specify the pathname to operate on and you will be shown the existing dates and allowed to modify them. Just press RTN to accept the date shown. When you are done you will be given a chance to abort the routine before the new data is written to disk. The main use for this routine is to allow you to make meaningful creation dates for files having no dates or ones on which the date was mined by a defective program. You do not have to type the dashes or colon shown in the date; any non-numeric character (such as a space) will do, but you must type the data in the correct position on the screen.

◊ *SHOW FILES*

This catalogs a directory. Use the ↓ and ↑ keys to scroll forwards and backwards. At the top of the screen is shown the number of blocks used by files in the directory exclusive of subdirectories, the number of files in the directory and the number of free blocks on the volume. If showing invisible tiles is enabled, they are marked with an open apple if unlocked and a closed apple if locked.

◊ *SHOW VOLUME NAMES*

This looks at all mounted disk devices and shows the device number and name, volume name, number of free, used, and total blocks, and the creation date of the volume.

◊ *TOGGLE BELL*

If this is OFF then the bell that is heard at some prompts will be defeated.

◊ *TOGGLE PROMPTING*

This toggles the prompting state for the FILE COPY and VERIFY, etc. The current state is shown on the menu. If prompts are OFF then deleting of locked files and copying over existing and locked files is done without asking whether you are sure. In addition, the copying and verifying of files inside selected directories is done automatically when prompting is OFF.

◊ *COMPARE DIRECTORIES*

This will compare the contents of two directories (not the contents of the files in the directories). A file listed on only one of the two columns is in that directory but not in the other. Files in both directories but of different file types are shown with their types, and those with different modification dates are shown with their dates.

◊ *COMPARE FILES (K command)*

This will ask for the full pathnames of the two files you wish compared. It then compares them and displays any differences found. A very useful feature is that it allows the adjustment of the position pointer of one of the two files independently of the other. After showing a difference (which is shown by exhibiting hex and ascii dumps of 128 bytes from each file with the differences highlighted), you can scroll through the files with the cursor keys or go to the next difference with the RTN key or abort with the ESCAPE key. Hold down the Apple key when giving the second name to compare resource forks.

◊ *MOVE FILES*

This will move files (put them on the destination and remove them from the source) on any given volume. This is very fast and can move whole directories. It does not write new files, but just moves the pointers to the files to the new directory.

◊ *LOCATE STRING*

This is for finding a text string in a group of files. (This feature is also in the File Finder module, but works somewhat differently.) You specify the directory to check and the string to look for. Found strings are displayed in context much like the same function in the File Finder.

◊ *VOLUME COPY*

This volume copy facility will use as much memory as it can find in order to copy in the most efficient manner. Its operation is fairly self-explanatory, but there are two provisions that are not announced on the screen and which are for special situations:

- If the Apple **and** the Option keys are held down when you press a number key in answer to the "Destination device" question, then all blocks will be copied. Normally the routine copies only used blocks for more speed.

- If the Apple **and** the Option keys are held down when you press a number key in answer to the "Source device" question, then the size of the disks will be assumed to be 1600 blocks *(3.5"* disks) and the program will ignore any unreadable blocks. This feature is meant for dealing with badly damaged disks.

  Using both these options also allows copying a damaged *5.25* inch disk to a 3.5 inch disk, which can then be used for a lost file search and other repairs. Ignore the read errors on the original past block 279.

◊ *USING A MOUSE WITH UTILITIES*

A mouse can be used to select options and files. It works a little differently from most mouse interfaces since it "wraps" around the screen. (Thus, for example, if the cursor is at the top of the screen then an upward mouse movement will move the cursor to the bottom, as does an up arrow.)

You can use the mouse button to select options from the utilities menu or to select or deselect files when a list of files shows on the screen.

When you are asked for a pathname, the mouse button is interpreted as a “?” (when that is accepted). The mouse can be used to move the cursor across a pathname. Thus you could, for example, use the mouse to click on “Catalog” on the utilities menu, then click again when asked for a pathname. This will bring up a list of all volumes on line. Clicking on one will produce the directory tree, and positioning the cursor and clicking on the tree will produce the desired catalog of that directory. Move the mouse up and down to scroll the catalog if it is large enough. A final click, anywhere, returns you to the utilities menu.

In the tree structured directory display, the mouse button is interpreted as a RTN to select the currently displayed directory, and mouse movement is accepted in place of cursor keys.

In a sort, the mouse button can be used in place of the Apple key to move, with the mouse, files around the screen.

◊ *GETTING RID OF PROBLEM FILES*

Sometimes some glitch in the system will ruin a file to the extent that it cannot be deleted by ordinary means. There is a “secret” (meaning not shown on the screen) provision in Utilities that will allow you to get rid of such files. It is in the SORT routine, even though it has little to do with sorting. If you highlight the last file in the list and press Apple-DELETE then that file will be deleted from the list. Nothing happens on disk until you press return and ask for the “sorted” directory to be written to disk. To delete a file which is not the last one on the list you must move that file to the end, using the arrow keys with the Apple key, and then press Apple-DELETE.

This only gets rid of the file as far as the directory is concerned and does not free the blocks used by the file and release them to the system. To do that you should then use that provision in Volume Repair.

Please note that files deleted in this fashion are not recoverable. You must use it with extreme caution. Entire directories can be deleted this way, by using it on the name of the directory.

For technical reasons this method does not work on a file that is the only file in its directory. However, the directory containing it can be deleted.

Part IV. Shell

The command line processor (Shell)

The old fashioned user interface on computers was what is called a command line processor or a ‘shell”. This is a user-hostile environment. Usually it presents you with a blank screen, a prompt, and a cursor, daring you to do something intelligent. Many old timers, mostly programmers, like this because they had to learn all the commands and spent a lot of time doing it, so why shouldn’t you?

The most familiar command line processor on an Apple II is Applesoft. If you enter Applesoft, without running a BASIC program, then you are presented a “prompt” (the] character) and a cursor (the flashing box), and Applesoft then waits for you to type a command like RUN or CATALOG, etc. Applesoft is also the most powerful such interface you are likely to see.

Another, older, example is the “monitor program’ which presents you with a prompt (the * character) and a cursor, and has another set of commands altogether, with which many fewer people are comfortable.

There are other, more recent, command line processors for the Apple, but they all share, to greater or lesser extent, the characteristic of user hostility: to use them effectively, or at all, you must learn the commands and sometimes some very curious command syntax. Some command processors try to ease the hostility by having one command that shows you all the other commands. So then the only thing you must know is that one command. Still, they do not meet present standards of friendliness.

If command line processors are so passé, why does ProSel- 16 have one? The main reason for this is that there are some situations that could happen, rarely we hope, that leaves ProSel-16 no way to display an application screen. For example, the file PROSEL.SPECS containing the screen data may be damaged. Since ProSel- 16 is the boot program, such a circumstance leaves you in a difficult position. In such instances, however, ProSel- 16 gives you the option of entering the command line processor to try to rectify the situation. For example, you can enter the Utilities module from the command line processor by typing UTIL and then you could copy the PROSEL.SPECS file from another disk, or you could rename (directly from the command line if you want) the SYSTEM/START file and reboot into another boot program.

Of course, another reason for including such a feature is that there are some people who like such an interface, and some people who don’t like it may decide it is useful when they learn more about it.

The command line processor in ProSel- 16 is basically a set of a few internal commands, and includes a provision for adding commands (“transient commands”). Some of the commands are things more powerfully done by the Utilities and other modules, but which might occasionally be more convenient to do from the command processor which is always in memory.

There is also a provision for operation by a “script”, see below.

Learning to use the command line processor is in no way indispensable for using ProSel-16. You can ignore it completely if you wish.

The command line processor, like other main menu selections can be accessed by pressing ESC to enter the menu and then S for Shell, or equivalently by pressing Apple-S <RTN> from the screen. There is also a convenience key CLEAR that sends you directly from the main screen to the command line processor.

Entering the processor yields a brief message indicating that you can exit back to the screen by pressing ESC or typing BYE <RTN>. The processor’s prompt is the colon. [Note, for example, that if you type HELP, it will print a prompt”>”. Since this is **not** the processor prompt, you are still in the Help module. Pressing RTN at this point brings the **“:”** prompt indicating that you are again in the command processor.]

The processor contains a number of internal commands, and external “transient” commands, which are just EXE type files contained in the prefix #6 directory.

> *Any executable file in the prefix #6 directory can be executed by just typing its name from the command prompt. If the file is an EXE file then ProSel does not “shut down” and it stays in control. That is the main distinction to other types of executable files. There are other differences, mainly that ProSel-16 passes certain “pointers” to EXE flies.*

Typing HELP COMMANDS <RTN> brings a list of the built in commands and some comments on each.

Typing COMMANDS <RTN> brings a list of external commands, provided that the COMMANDS program is in the prefix #6 directory.

Internal commands:

HELP <alone>	displays help topics available. (? = HELP.)
HELP <topic>	reads and displays help info on <topic>.
DELETE <filename>	deletes <filename>.
UNLOCK <filename>	unlocks <filename>.
LOCK <filename>	locks <filename>.
RENAME <fuel> <file2>	renames <fuel> to <file2>.
PREFIX <directory>	sets prefix 0 to <directory>
PREFIX #	sets prefix 0 to prefix # (1 through 31).
PREFIX #=<directory>	sets prefix # to <directory>.
PREFIX <alone>	prints a list of prefixes 0 through 31.
PFX	short for PREFIX.
REFRESH	reloads PROSEL.SPECS file from prefix #0 volume. (Scans devices if not found.)
POP	removes one level from prefix 0.
CAT or CATALOG	catalogs current prefix directory.
CAT <directory name >	catalogs named directory.
PRINTER	turns printer on.
CONSOLE	turns printer off
EDIT <pathname>	goes to text editor loading <pathname>.
UTIL	goes to ProSel-16 utilities.
BYE	quits command line mode, goes to ProSel screen.
QUIT	quits ProSel with no return.
BRUN <pathname>	runs an executable file (516/EXE/SYS type).
RUN <pathname>	runs a BAS or BIN program under BASIC.SYSTEM.
- <pathname>	runs any program of type S16, EXE, SYS, BAS, BIN or TXT (script file).
- <path> <startup>	runs <path> (S16/EXE/SYS) with startup <startup>.
CREATE <pathname>	creates DIRECTORY <pathname>.
COPY <path1> <path2>	copies file <path1> to file <path2>.
DEVICE	lists devices for choice & sets prefix.
DEVICE <1 - 9>	sets prefix 0 to volume in given device #.
EJECT	ejects device of current prefix 0.
EJECT <0 - 9>	ejects disk in device # given, 0 = all.
BOOT <0 - 7>	boots slot # (append = for slow speed).
SETBOOT <0 - 9>	sets boot slot (8,9,0 = RAM, ROM, SCAN).
MON	sets echoing of EXEC commands to screen.
NOMON	defeats echoing of EXEC commands.
BACKUP <startup>	goes to Backup passing <startup>.
REPAIR <startup>	goes to Volume Repair passing <startup>.
INFO <startup>	goes to Info Desk passing <startup>.
HIDE/UNHIDE <filename>	makes <filename> invisible/visible.
EXEC <textfile>	executes shell commands in <textfile>
HOME	clears screen.
ECHO <text>	prints <text> to screen.
WAIT :hr:min:sec	waits until given hour:minute:second.
WAIT >hr:min:sec	waits for time period given by hr:min:sec
PICSHOW	runs the picture show.
ART	runs the kinetic art show.

Separators, such as the space between <path> and <startup> can also be commas or some other characters. The < and > characters are not typed, and are shown here just for clarity.

Anything you type that cannot be recognized as an internal command is taken as an external command, meaning that the first word of the command is searched for as a filename in the prefix #6 directory, and that file is executed if found, If there is no such file then a syntax error is reported. The external commands, of course, can be supplemented or reduced by adding or deleting files in the prefix #6 directory. Those presently supplied are'

ASCII	prints ascii/mousetext table.
COMMANDS	prints list of internal commands.
PURGE.MEM	purges purgable blocks from memory showing results.
WHERE	shows all memory handles and their owners.
SHUTDOWN	does a system shutdown (warm).
GS.OS.VERS	prints version number of current GS/OS.
TYPE pathname	types the TXT file specified.
DUMP pathname	does hex dump of file specified.
PATH filename	searches for filename and sets prefix #0 to it if found.
FILETYPE pathname,TYP	changes filetype of specified file to TYP.
AUXTYPE pathname, $1234	changes auxtype of specified file to $1234 or whatever.
LOADPIC pathname	loads super hires picture and displays it.
PACKPIC pathname	packs a SHR screen image to a packed file named PATHNAME.P.
ZIP pathname	converts downloaded ProSel-16 file START to fast load format for ExpressLoad.
CASE /dir.name	converts filenames to lower case; upper case if D in Dir.name is capitalized.

The first six commands, that don't need parameters such as filenames, can also be executed from the main screen by pressing key 6 on the KEYPAD and then selecting the desired command file.

In FILETYPE, TYP can be in hex as in $12, or ascii as in BIN.

Beginners, or anyone in doubt about the consequences, should **not use** the FILETYPE and AUXTYPE commands.

Other features of the command processor

The command processor has some convenience features. One of these is the automatic expansion of commands. This is active if you use the ENTER key on the keypad to terminate the command instead of the RETURN key. [Note: the modifiable parameters include an option to interchange the action of these two keys, to defeat it altogether or to support it on both keys.] Thus, for example, if you type C SYSTEM <ENTER> the command will be expanded to CATALOG SYSTEM and the directory SYSTEM (if there is one off the present prefix) will be cataloged.

Also there is a history provision, in that the command processor remembers your last 15 commands. You can get the last command you issued by pressing the ↑ key. This prints the last command and places the cursor at the end of it. You can just press RTN or ENTER to issue the command again, or you can edit the command a little and then press RTN. Another ↑ gets the next to last command, etc., and ↓ moves through the list in the opposite order.

Automatic operation and scheduling

The command processor's EXEC command reads a named text file (called a "script") and acts on its contents, assuming it is a set of shell commands. This can be used to run a sequence of programs automatically. For example, suppose you want to run the program MYPROG in the MISC subdirectory with a startup specification of STUFF, and then run the program BASIC.PROG, a BAS file, in the BASIC subdirectory. Then make a text file, say TF, with these commands:

```
HOME
ECHO Running MyProg with Startup STUFF.
PREFIX */MISC
-MYPROG STUFF
HOME
ECHO Running Basic.Prog.
PREFIX */BASIC
RUN BASIC.PROG
```

Then thesecommands will be sequentially executed when you type

EXEC TF or just -TF

in the command interpreter, assuming TF is in the default directory.

You can also EXEC a text file from a screen specification by letting the “application” be the script file. No “startup” is needed. Assuming that TF is in the EXEC.FILES subdirectory, this might look like:

Screen title:	Exec TF
Prefix:	?EXEC. FILES
Application:	TF
Startup:	<empty>

The HOME (clears screen) and ECHO (prints following text to screen) commands are intended for use in script files for the EXEC command.

The WAIT shell command also is intended for use in script files. It simply waits until a certain time of day.

If WAIT is followed by a time (hour:minute:second) in the form :14:45:00 then the wait will terminate at 2:45:00 PM and the next item in the script file will be executed.

If WAIT is followed by a time in the format >00:30:00 then WAIT will wait that amount of time (from the present time) before executing the next line of the script.

During a WAIT, the RETURN key will terminate the wait and execute the next line of the script. Similarly, the ESC key will abort not only the wait but the entire remainder of the script.

During a WAIT, the shell displays a message like:

```
Waiting until 18:40:00   RTN to skip,  ESC to exit.

Current time: 18:33:58
```

A BYE command in a script will abort the script and exit to the main screen. Otherwise you are left in Shell mode when the script is done.

Using /RAM5 as a boot device

Some shell commands can be used to make RAM5 into a boot device by means of a script that copies the boot files to /RAM5 and then sets RAM as the boot option. This option is not recommended with less than 2MB of ram. For example, the script can look like this:

```
HOME
ECHO Copying files to the Ramdisk.
PFX *
PFX 9=/RAM5
CREATE 9/SYSTEM
COPY PRODOS,9/PRODOS
COPY SUB/PROSEL.SPECS,9/PROSEL.SPECS
PFX SYSTEM
PFX 9=/RAM5/SYSTEM
CREATE 9/SYSTEM. SETUP
CREATE 9/TOOLS
...
COPY P8,9/P8
COPY START,9/START
...
ECHO Rebooting into the Ramdisk.
SETBOOT 8
BOOT 0
```

Note that BOOT 0 is just a reboot into the default slot, which is the Ramdisk after the SETBOOT 8 command.

Of course there are many other files that must be copied in order to use RAM5 as the boot disk, such as the tools and fonts and desk accessories and system setup files. My test script for this has about 70 items to copy/create. The ramdisk must be almost 800k to hold all the needed files. Note the line for PROSEL.SPECS - the regular file on the original boot volume would not be appropriate for the ramdisk. so it is taken from a subdirectory to which an appropriate specs file has been copied from /RAM5. Similarly, a special PROSEL.PARMS file might be desirable instead of the one in the SYSTEM directory of the original boot disk.

All of this can be automated into the boot sequence by making the script be the “boot program” in the parameters on the boot disk (which is one reason to use a different PROSEL.PARMS file to copy to the ramdisk). I suggest you not defeat the day check so that a reboot into the hard disk can be done simply by a cold shutdown.

You should exit with a cold shutdown before powering down the machine because the cold shutdown will reset the boot slot to SCAN if it is currently set to RAMDISK. Otherwise the next boot will fail until you change the boot slot in the control panel.

Passwords and Hiding files

The shell command HIDE <filename> sets the “invisibility” attribute of the named file. UNHIDE does the opposite. When this attribute is set, the file will not show in a catalog in ProSel-16. It may show in other programs depending on whether the program supports this new feature of GS/OS. The command processor. Find File, Info Desk and Utilities all support this as does the number key feature of the selector screen. One can change that for the Shell, Find File, Info Desk and Utilities via Modify parameters.

If a directory file is hidden, then all of its contents are hidden as well. If you want to hide many files or directories, consider doing it via an EXEC file, and similarly for unhiding them.

If the ProSel-16 file (usually START) is hidden, then some further restrictions are imposed by ProSel-16. In that case, access to the main menu and the shell is denied and other hidden files cannot be launched. There is only one way to override this (other than using some other program to unhide START) and that is to press CLEAR from the screen. Ordinarily that will send one to the command processor, but if START is hidden it will bring up a prompt for a password. Typing the correct password will defeat this provision for the rest of the time that you are in ProSel-16 (e.g., until a reboot). Thus, at that time, you can UNHIDE START.

When you HIDE START, this provision does not take effect until you exit ProSel-16 and return, for example by a shutdown. Thus you can use cycler (X from the menu) to bring up another screen and then do a warm shutdown and reboot. After that point, the user will be locked out of the menu, etc.

The password provision is not supported (and so the HIDE START will have no effect) until you provide a password. This is done via the Modify parameters function. The password is case sensitive and supports any characters which Modify parameters will accept as input. The length of a password is restricted to 12 characters or less.

Hiding of files can also be done from Utilities, and Unhiding can be done there provided display of hidden files is enabled for Utilities (in Modify parameters).

WARNING: The password provision does not prevent access to your files by anyone sufficiently motivated to get into them. Do not rely on this for security.

Part V. Appointment Calendar

General description

The appointment calendar is a perpetual calendar (for years 0-9999), appointment reminder, event list and date oriented data depository. To make full use of it, you must enable two of its parameters in Modify Parameters. The first of these parameters enables the reminder function and the other enables appointment file access (without which, this module is simply a perpetual calendar without appointment capabilities). You must also copy the "Appoint.CDA" file from /PROSEL/SYSTEM/DESC.ACCS to your SYSTEM/DESK.ACCS directory, or appointment warnings will be disabled. You must also specify a valid directory, in Modify Parameters, for prefix *#5*, which is where the appointment calendar keeps its data files.

With reminders enabled, any appointment entered into the calendar will cause a 9 minute pre-warning when the ProSel- 16 main screen is displayed. The warning sounds a bell and prints the number of minutes till the appointment time, followed by the appointment title, on the bottom left of the screen. The title will be cycling through three colors. When the appointment time comes, a "siren" will sound each four seconds for one minute and the border will flash briefly in a multicolored display at the same intervals. This action will occur in any program (with a few exceptions of programs which disable interrupts) including ProDOS-8 programs. At any time, you can open the Appoint.CDA desk accessory to view the times and titles of all appointments on the current day.

When you enter the appointment calendar by selecting it from the ProSel-16 main menu (or Apple-A from the screen) you will see the calendar of the present month on the left of the screen and the day's appointment list (times and titles), and event list on the right. On the calendar, a red diamond on the lower right of an entry marks the present day. An inverse red date marks the cursor position (defaulting to the present day upon entry). A closed apple character indicates a day of the displayed month which has appointments set up. An open apple indicates a yearly or monthly event (see below) but the closed apple has precedence. A small rectangle in the upper left corner of a day indicates that that day is a full moon day (approximately). *(See Figure 10.)*

When the cursor is in the calendar portion of the screen, the arrow keys will move it from day to day within that month. The appointment list will follow this movement automatically. The ESC key will exit back to the main ProSel- 16 screen. The TAB key will send the cursor to the appointment list, then to the event list, and then back to the calendar. An Apple-arrow key will change the month or year (~, the month and ♯, the year), and will also cause the data file for that month (see below) to be read if the file access parameter is enabled. Again, the appointment list will be updated automatically. If you wish to view the calendar portion only, ignoring appointments, you can do that by holding down **both** Option and Apple keys while pressing an arrow key. The CLEAR key will "home" the cursor, meaning that it moves the cursor to today's date if on the current month or to day I on any other month.

In the appointment CDA. entries marked with a closed Apple character represent those items dated in the future. Those without an Apple have times that have already passed. In the appointment module, when ProSel-16 is in graphics mode, the passed appointment items are shown dimmed. (Actually, they are printed in color #5, which may or may not appear to be dimmed depending on your choice of colors for the appointment module.) The appointment list, in both the module and the CDA, is always in chronological order.

The appointment editor

To enter or edit an appointment in the list, press TAB to put the cursor in the appointment list, and highlight the desired one to edit or the first empty space to add a new item. Then press RETURN. This will send you to the appointment editor. Pressing ESC when the cursor is in the appointment list will exit directly back to the ProSel-16 screen just as it will when the cursor is on the calendar, and pressing DELETE will delete the item the cursor is on.

Pressing RETURN when the cursor is in the calendar portion of the screen will also send you to the editor to add a new item for the day the cursor is on, or to edit the last one if the appointment list is full.

The appointment editor has three “fields”, a title, a time (in 12 hour AM/PM format), and a text area. When you enter the editor, the cursor will be on the first character of the title field. Pressing TAB moves the cursor from one field to the next.

The title can be any text you want. It is limited to 28 characters. The title is what is displayed in the appointment list and also in the list in the appointment CDA and in the 9 minute warning display. The text field is only seen when in the editor. The time can be anything except 12:00 AM (midnight) since that time is used for internal purposes by the program.

The text field can be anything you want, or nothing. You might keep an address there, etc. The screen area allowed for the text is somewhat larger than the maximum number of characters (512) allowed for the text. If you exceed the maximum amount, and press TAB or ^S (the SAVE command) then a box will come up telling you of this and asking you to shorten the text and try again. The 512 characters is two thirds of the text area. The excess is there to add flexibility for formatting of tab fields, etc.

Pressing ESC while in the editor will abort and you will lose anything entered. To exit and keep what you have entered, you must issue the SAVE command (^S). This can be done when the cursor is in any of the fields, but it will be accepted only if the time is other than the illegal 12:00 AM. This command also sends you back to the calendar.

There is a maximum of 30 appointments allowed for any single day.

An error #$44 when trying to add an appointment probably indicates that prefix #5 has not been set up to point to an existing directory in Modify parameters. An error #$46 probably means the appointment file access parameter has not been set to Yes.

Appointment files

If “file access” is enabled, the program will automatically create and maintain data files with names such as “Appoint.Dec.91”. This example will contain the data for the month of December in 1991. These files are kept in the prefix #5 directory (the same directory used by Info Desk and Backup for their files).

When you boot up into ProSel-16, the file for the present month/year will be created if it does not already exist. When the month changes, a new file for the new month will be created automatically if it is not already there.

When you are in the appointment calendar module and you change the month or year by the Apple-arrow keys (and are not holding down both Apple *and* Option keys), a file for the new month will be created and read if one is not already there. This only happens in the current year and the next year, so that moving to the previous year or two years forward will not create such files

Holding down the Option key alone when passing to a new month will force the creation of an appointment file for that month, regardless of the year, allowing you to enter appointments farther away than the next year.

The appointment module will automatically attempt to read the data file for any month you move to with the Apple arrow keys even if the year is not the current or next one. Hold down Option as well as Apple to avoid this, and thus to move more quickly to the new month/year.

The appointment files are quite large and very sparse. They are of length $7C000 in hexadecimal, which is about half a megabyte. When created, however, they are completely “sparse” and occupy only one block on the disk. Although it is quite unlikely, it is possible for one such file to grow to occupy about 1 ,000 blocks.

You may delete these files whenever you want. The program never deletes them because some people might want to keep records of past appointments.

Hard Copy

For hard copy of appointment lists, press Apple-P.

If this is done when the cursor is on the month’s calendar then all of that month’s appointments will be printed without the text fields.

If the cursor is in the appointment list then just the day’s appointments will be printed, but with the text fields.

If the cursor is in the event list then the entire event list will be printed.

☼Events

The “event list” holds up to 127 yearly events such as holidays and birthdays. It also holds monthly events such as bill payment dates. It also holds weekly events, things that come up each week on a particular day of the week like Monday. It also holds daily events which come up every day, such as noon. The legal holidays, and a few others, are already set up. To add to or edit this list press TAB until the cursor is in the events box, then move the cursor to the end of the list or to the item to be edited and press return. You then edit or add the event in the same way you do an appointment, with the difference that there is no “text area” and that the time 12:00 AM is legal. If the time is set to the default 12:00 AM then the item is not “time related” and will not cause a warning and will not be in the CDA (except daily events). Any other time will make the event time related. A time related event will work exactly like an appointment except that it is listed in the events box instead of in the appointment box in the appointment module. It will be listed in the CDA and will cause warnings. A daily event will come up every day, and similarly with the other two types.

Events are held in a separate file called “Events” in the prefix #5 directory. This file will be created upon first entry to ProSel- 16 unless the file access parameter is off. You may delete events by pressing DELETE when the cursor is on the event to be deleted. Note that when you press TAB to move the cursor to the event box, all events will be listed and not just those for the particular day. Those not for the selected day will have a dimmed preamble (time, date, or the words “Daily”, or “Yearly”).

When you add or edit an event and press ^S to save it, you will be asked whether it is to he a daily, weekly or yearly event. In each of these cases it is the selected day (the day, month, and weekday highlighted in the calendar) that will be taken for the event. You can delete the “Events” file, in which case a new file will be recreated automatically with the default events.

Untimed daily events are put in the CDA for display purposes only. They are marked with an open Apple and the header “Daily:”. This enhances the use of daily items as a “to do” reminder list.

You can also have events tied to things like “second Monday in May” (a yearly event) or third Friday of each month (a monthly event). When you save an event in the monthly or yearly modes you are given the opportunity to make this selection. Please note that the cursor must have been on the correct day before going to the editor, as that determines the day of the week, and the “first”, “second”, etc. This option is not presented on days past day 28 since it has no meaning in such cases.

On a day corresponding to a non time related yearly or monthly event, such as Labor Day, the event will be displayed on the main ProSel-16 screen on the bottom left. If there is more than one such event, they will alternate, up to four such events. During an appointment warning, the appointment display takes precedence.

Appointment Parameters

As already stated, one of the parameters related to the appointment calendar enables "warnings", both the bottom left message on the ProSel- 16 screen, and the multicolored border flash when an appointment comes due in any program. Another parameter enables the file access, without which appointments are totally disabled and only the calendar portion functions. There are also two more minor parameters which disable the sound part of one or both of these types of warnings.

There is a parameter that governs the size of the displayed appointment list. You can have the list any length from 1 to 15. The event list size compensates automatically to a change in this parameter; both lists combined always total 17 items. This parameter only affects the display and not the file capacity.

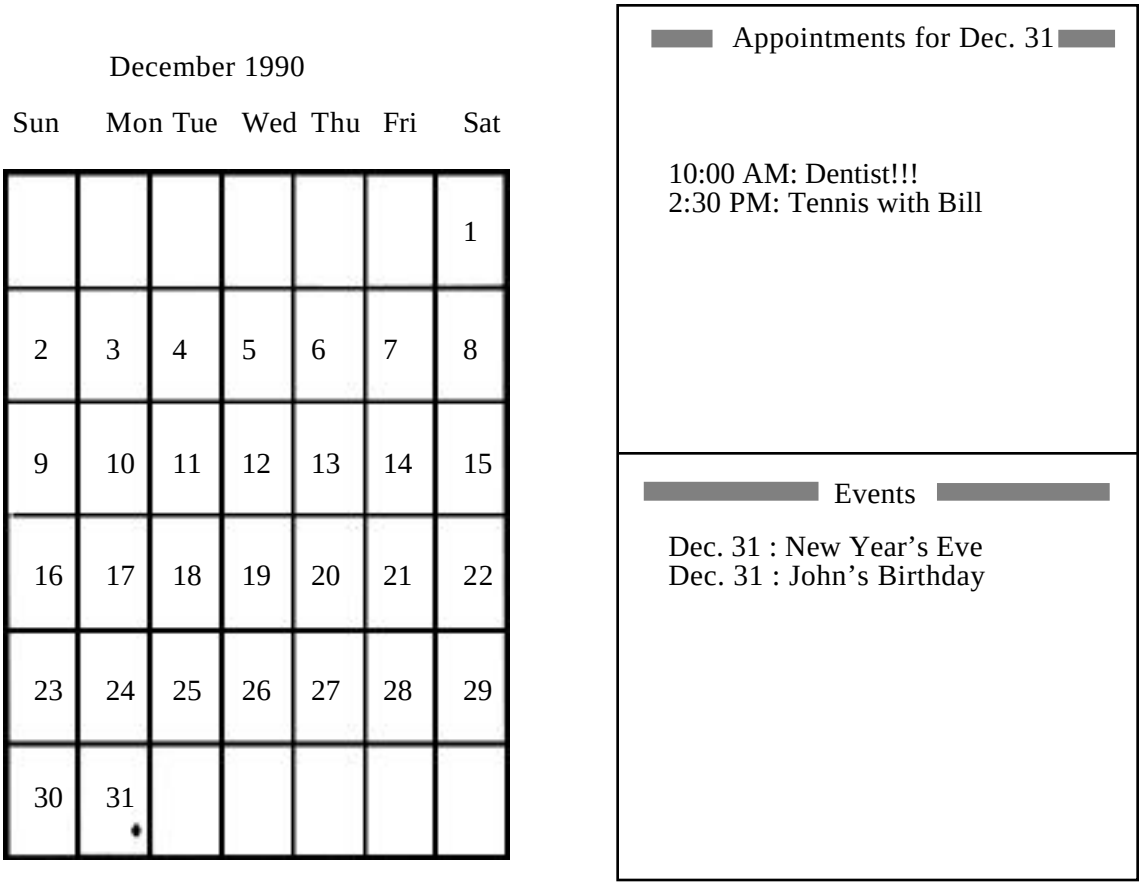
There is a parameter that will cause the appointment CDA's list to appear upon a boot, or anything that causes a reload of ProSel-16 or reinitialization of the CDA, providing a useful reminder of the day's schedule.

You can turn off the siren from within any program by holding down Apple-Shift-Control for about a second or two after the siren begins. This will turn off the siren for the current item only.

Pressing DELETE when **in** the CDA will remove, on a temporary basis, the next appointment. This will come back if any operation is done causing ProSel-16 to reload, such as the running of a very large program.

For programs that have difficulties with interrupts (used by the calendar warning mechanism), there is a provision to completely disable the calendar when running a program. This is done by putting a single dash "-" at the end (or within) the prefix field of the program's specification. You may have to do this with terminal programs and the like, and you can also do it to prevent the effect that interrupts have on the sound from certain programs that were not written to live in an interrupting environment.

There is also a less drastic provision, given by putting two dashes "--" at the end of the prefix. This will not disable appointment warnings but will disable the sound part of the warning and take other measures to prevent interference with the application program. This may be sufficient for some terminal programs.



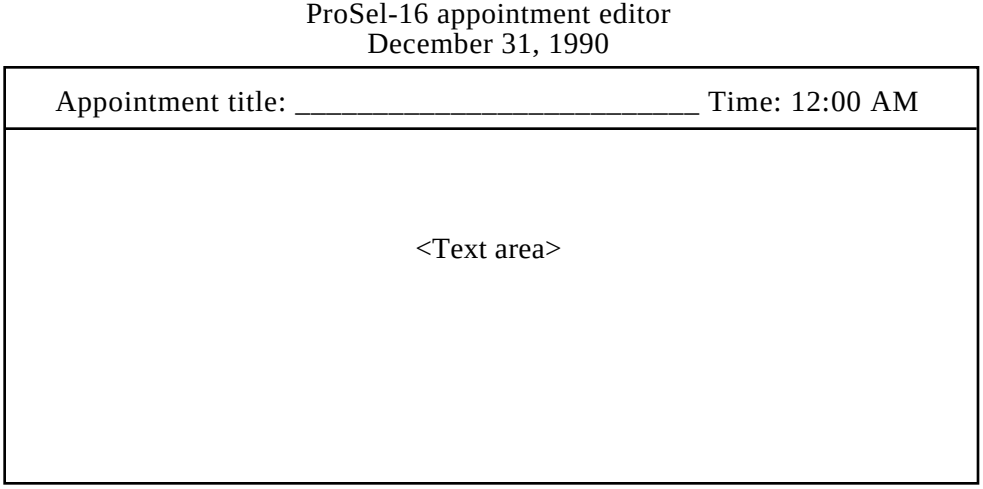


Figure 10. *Appointment ca1endar screen (top); Appointment editor (bottom).*

Part VI. Text Editor

General description

The text editor can be used for creating shell scripts, backup scripts, and anything else calling for an editor that puts carnage returns at the end of each line. It is entered by the T command from the main menu or Apple-T from the screen. If you set it up as an item on the screen you can have it automatically load a given file by placing the filename in the Startup of the specs. The editor will only save in text file format, but it will load TXT, SRC, and AWP (AppleWorks word processor) files. It will even load and repair some damaged AWP files that AppleWorks will not touch. If you leave and reenter the Text Editor, it will attempt to bring up from memory the text that was there when you left. If this is successful, it overrides the "Startup". The control key also defeats the startup, as is the case with all modules.

The editor will scroll horizontally as well as vertically.

In graphics mode, control characters will show in the first highlight color, and the cursor in the second. At the right top of the screen is the number of the line containing the cursor in the second highlight color. To its left is a vertical bar in the first highlight color. A short bar indicates the default CR insert/delete enable mode, and a long bar indicates disable.

There are six cursors depending on modes:

| = insert mode
▮ = overstrike mode
= character find & insert mode
= character find & overstrike mode
= macro definition, by pick, mode
Apple = macro definition, general, mode

The text editor has its own screen blanker which has a delay of about 10 minutes on an accelerated machine. If graphic mode is active the blanker will go into the kinetic art. At that point, the ESC key will terminate the kinetic art and just blank the screen. Any other control key, such as RETURN, or the space bar will bring the screen to life again.

The prefix used for all disk operations is prefix #0 as it was on entry or as changed by the Command Center Prefix command. It would be wise to set up an item for the text editor on the ProSel- 16 screen so that the prefix will be predictable. For example:

Screen title:	Text Editor
Prefix:	?Text. Files
Application:	T
Startup:	My.Text.File

☼ Commands

Text editor commands are arranged logically with control characters acting “locally” on lines and Open Apple characters acting globally, with some exceptions. Most of the keys are user redefinable via Modify parameters and the help provision will reflect those changes. Please note that access to the parameters for the text editor is not allowed until you go into and Out of the editor. The default command arrangement is:

Arrows	=	cursor moves
^E	=	toggle insert/overstrike
^I (TAB)	=	tab across next space
^T	=	remember line number for Apple-T command
^B	=	beginning of line
^N	=	end of line
^R	=	restore line to original
^S	=	status box; another ^S goes to history list
^F	=	find, on current line, character pressed next
^W	=	tab to next word
^D	=	delete character under cursor
DEL	=	delete previous character
^V	=	truncate line at cursor
^0	=	override commands and accept next key pressed literally
^X (CLEAR)	=	cancel global find/exchange
ESC	=	move to start of next line
Shift ←	=	local cut (cut characters out of single line)
Shift →	=	local copy
Shift ↑	=	local paste
Shift ↓	=	clear local cut buffer
Apple ↓	=	cursor down 10 lines then centered
Apple ↑	=	cursor up 10 lines then centered
Apple →	=	down 1 page
Apple ←	=	up 1 page
Apple -X	=	cut/select
Apple -C	=	copy/select
Apple -V	=	paste
Apple - F	=	find text (input)
Apple -W	=	find word (input)
Apple -E	=	exchange (find and replace) (input)
Apple -*	=	command center
Apple -T	=	return to line number remembered by the ^T command
Apple -I	=	insert line above current line
App e -ESC	=	quit
Apple -Q	=	quit
Apple -B	=	beginning of text
Apple -N	=	end of text, removing extra lines

Apple-D	=	delete current line
Apple-DEL	=	delete previous line
Apple - L	=	goto line number/label (input)
Apple - Z	=	center the line containing cursor
Apple - V	=	select to end
Apple - H	=	split screen
Apple - ?	=	help
Apple - /	=	help
Apple - S	=	save file (input)
Apple - 0	=	open file (input)
Apple - P	=	print text, all or selected (input)
Apple - !	=	swaps present text buffer with a secondary buffer
Apple - @	=	swaps present text buffer with a tertiary buffer
Apple - 0	=	catalog
Apple - U	=	undo
Apple - R	=	replace line with line delete/replace buffer (swap)
Apple - \	=	toggle CR insert/delete enable
Apple - TAB	=	tab backwards
Apple - 1-9	=	proportional position
Option +	=	macro definition (general)
Option -	=	macro definition (by pick)
Option *	=	macro definition (box)

In the list, “input” indicates that the command produces a box asking for further user instructions, such as a filename.

The character find command ^ F) will wait for another keypress and then will place the cursor on the next occurrence of that character on the present line or will move the cursor to the end of the line if none is found. It that key is repeated then the next occurrence will be found, etc.

The line delete commands place the deleted line in a special buffer used by the Apple-R command, and the latter swaps this with the current line. This can be very useful in making some types of changes involving moving lines around, even though the cut/paste commands can be used for doing similar things.

The undo will undo almost anything not involving disk access. You can undo cuts, pastes, deletions, inserts, etc. It remembers over 4000 steps, and this is extremely useful. The “history” provision will list all remembered steps of undo information. It is possible, particularly if many cuts are done, for the undo history to overflow. Except for that case, giving undo commands until the “Undo buffer empty” message occurs, will return the text to its exact initial state.

The editor automatically adds lines if you move the cursor past the end of the buffer. This should not concern you as extra empty lines at the end are automatically removed when you save the file, print the file, etc. If you want to force this removal use the Apple-N command or the Apple-9 command.

The cut and copy commands start selection if a selection is not already active. Select by ↓ and →. The ← key, and most other commands, will cancel selection. The ↑ key will narrow the selection and will cancel it if it reaches the original select line; you cannot select in an upwards direction. Find and Exchange act on the selected region if there is one, and otherwise on the whole buffer. During Exchange, an “A” key will make all changes automatic from that point. Any other printing character, such as the space bar, will accept the shown change. Any control character, such as RETURN, will reject the change, and CLEAR (^X) will cancel the entire exchange mode.

The cut and copy commands only work on whole lines and groups of lines. Thus the “local” cut and copy commands were added to enable doing such things on single lines. They work more or less like macros. The Shift → will move the cursor across text, copying it to a special buffer. Similarly Shift ←, when the cursor is on the first character of the desired string, will copy each character under the cursor to the buffer and then delete that character. Shift t then deposits the buffer at the point of the cursor just as if it were being typed at the keyboard. If the string contains control characters it would be wise to copy them with a ^O preceding each character being copied to the buffer because command characters encountered with the Shift ~ command will be taken as commands just as if they were being typed at the keyboard. Shift ~ can be issued repeatedly.

The Find and the Exchange commands are case insensitive unless you hold down the Option key when pressing RTN at the end of the find string. Exchange is limited to finding and replacing “words” (strings bounded by non-alphanumeric characters), unless you hold down the Apple key at the end of typing the replacement string, in which case it will not confine itself to “words”.

After the first Find command, subsequent Find commands will not bring up the input box but will continue to find the next occurrence of the find string. This is canceled by the CLEAR (or ^X) key. This “find” mode is indicated by one or more “+” characters next to the line number at the top right corner of the screen.

In the Go To Line # command you can input a “label” instead of a number. This will cause the text to be searched for that string as the beginning of a line of text, sending the cursor there if found.

When you save a file and later load it into the text editor, the cursor will be placed on the same line it was on when saved.

The “CR insert/delete enable” command is a toggle. In the default mode, a RTN key will insert a carriage return at the point of the cursor and carriage returns can be deleted with the DEL key. If this command is used, a RTN key will go to the next line and insert a new line at that position, and carriage returns cannot be deleted. This is useful for manipulating tables and similar things.

The mouse can be used to move the cursor. The mouse button will bring up the Command Center, but this can be changed in the pains to equal any of the command keys, and similarly for a double click which defaults to Apple →.

The Command Center

The Command Center is a pop up box that appears on a mouse click or the Apple-* keyboard command. It contains alternate entries for several of the keyboard commands as well as some commands with no keyboard equivalents.

The following commands are unique to the Command Center:

New	=	clears text buffer now in use
Un-New	=	restores text buffer if possible
Prefix	=	sets prefix for disk operations
SwapClip	=	swaps present text buffer with the clipboard
Append	=	appends a file to one presently in buffer
Segment	=	segments a file too long to load (see below)
Cement	=	cements together segmented parts of a file (see below)
Format	=	formats all or selected range to given line length

The Command Center Segment and Cement commands are for use with text files (only) too large for the editor's buffers (about 64K). The Segment command will read the file specified and cut it into pieces with the same name with ".01"," .02", etc., appended. The Cement command does the exact opposite. Ordinarily, one would edit the small segment files and issue a Cement command at the end of editing. The smaller files should not be deleted, in case further editing should be wanted. The default filename for the Cement command will be the current filename with the numeric suffix (if any) removed. The Segment command has no default filename. Ordinarily, one would have no use for the Segment command for files written with this text editor, but it may be useful for files from external sources.

The Format command asks for a line length and reformats the text to that length. If the Apple key is down when you give the length, lines will not be lengthened, which may prevent specially formatted text from being altered, If the Option key is down then filling spaces will be used to right justify.

CAUTION: The Segment. Cement, Format and Print commands and the loading of an AWPfile will write over the second Swap buffer, destroying its Contents.

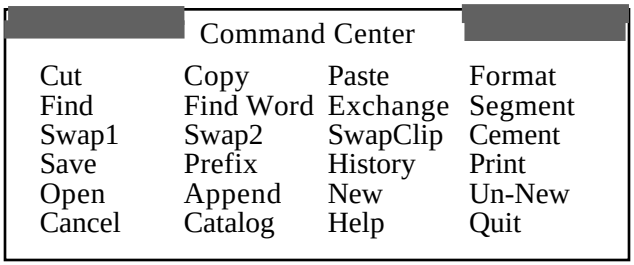


Figure 11. *The Command Center in the Text Editor.*

☼ Printing from the Text Editor

If you issue the Print command from the keyboard or the Command Center with nothing selected, then the entire text buffer will be printed, as indicated by the pop-up box that appears. If a selection is active then only that portion of the buffer will be printed.

The printer initialization string specified in the parameters for Info Desk, if any, is sent to the printer at the start of printing.

The Print command has one special character C which is not sent to the printer unless it is repeated. This causes the next character to be sent to the printer with “high bit on” This is to support printers like those from HP that use a full 256 item character set.

The following printer commands are supported. Each command begins with a period, which must be at the line start, followed by a two letter command. The command set is an extension of a subset of the old AppleWriter command Set. *(Also see Figure 13.)*

.LM# sets the left margin to **#,** defaulting to 0.
.TM# sets the top margin to **#.** defaulting to 0.
.PN# sets the starting page number, defaulting to 1.
.PL# sets the lines per page to #, defaulting to value in parms.
.LI# sets the line spacing to # (0 single space, 1 = double, etc.).
.TL#//header// sets a page header (the first “//” is required).
.UL# <char> underlines the header with line of <char> repeated # times.
.BL#//footer/f sets a page footer.
.OL# <char> overlines the footer with line of <char> repeated # times.
.RM# sets the number of printer columns to **#,** defaulting to 0 in which case columns will not be counted. If not zero, word wrap is attempted upon line overflow.
.MO# sets the printer mode to **#,** defaulting to 0.
.LE <leader> sets a leader, which must come right after the LE.
.TR <trailer> sets a trailer, which must come right after the TR.
.DE <char> sets <char> as the alias delimiter (default “I”)
.FF# does a form feed if **#=O,** or **if** # is more than the number of lines left.
.LK "filename" links <filename> during printing. Should be only at end of file.
.AL"filenam" loads <filename> alias file, overriding the default.
.WI <char> sets <char> as the alias wild card separator, defaulting to “,”
.SU suppresses header and/or footer on the next page.

The numbers after the commands are optional, defaulting to 0. For the .TL and .BL commands the numeric parameter should be 0, 1, or 2 indicating:

0 = use this header/footer on both even and odd pages.
1 = use this header/footer on odd pages only
2 = use this header/footer on even pages only.

If you omit the number in the .OL or .UL commands, they default to the length of the footer and header, respectively. That is usually what is wanted, but the repeat number lets you adjust the printing in case a proportional font is being used for the header or footer. The header or footer take up two lines on the page, the header and an empty line below it. They take three lines if an underline/overline is specified.

Nonzero values of the printer mode have the following effects:

1 = print header on page 1.
2 = print footer on page 1.
4 = defeat aliases.
8 = suppress unrecognized aliases.
16 = do not force form feed, printer will do it.
32 = print header on page 1.
64 = print footer on page 1.

For a combination of mode effects just use the sum of the numbers. Since values 4 and 8 make no sense together, that code (.M012) has a special meaning: It causes ONLY the unrecognized aliases to be printed (as they appear in text). This eases the task of finding intended aliases that have errors. (See below about aliases.)

For roman initial pages start with a .PN command with a negative page. For example, the command ".PN-3" will cause printing to start with page i, going through page iv (the same as page 0) and then continue with page 1, etc.

If a header or footer is not specified then there will be no page numbers. If the header contains the character // then the page number will be printed at that point in the header, left justified (overwriting characters to the right of the N). Similarly, the character will print the page number right justified ending at the spot of the (overwriting characters to the left of the). Also, the characters @ and % in a header or footer produce the date and time respectively. These are left justified, overwriting characters to the right; 9 characters for the date and 8 for the time). The date is in the format 15-Jan-91 and the time is in the format 11:30 AM. The overwritten characters should be spaces or something similarly expendable.

A "leader" is a printer control string that is sent just before the header and the footer. A "trailer' is a printer control string that is sent just after the header and the footer. These are intended to do things like switching fonts for the header and footer, and back again to the font used for the main text.

Holding down the Apple key when pressing RTN in the Print command box will cause only the odd numbered pages to be printed. Holding Option down at that time does the same for even pages. This lets you do two sided printing of any file easily. (Numbering here is counted from the first printed page and has nothing to do with page numbers except that they are usually the same.)

You can make printing pause by pressing the space bar. More spaces will step it one line at a time. ESC will abort it, and any other key will restart it.

The program uses form feeds to eject pages. If you get double page ejects then you need to decrease the PL parameter or the default value in the parms. This may happen with printers that do their own page ejects, not detectable by the printing routine.

This manual was written and printed, photo-ready, with this text editor.

Printer Aliases

The text editor supports printer "aliases". A printer alias is certain text in a file that instructs the Text Editor to replace it with some other text. The primary purpose is to replace printer commands that are specific to a certain printer with commands that can stay constant if you change to another printer. Only the alias file containing the conversions need be changed.

They can also be used to print control characters without actually embedding them in the text. Embedded control characters do not bother this Text Editor but do bother some other programs. For example, you could use !A! as an alias for the control-A character. Some printers have special characters assigned to otherwise unused control characters. For example, a DeskJet 500 will print a happy face when sent a ^A.

An alias should not be used, however, for a form feed (^L) or other characters that change the vertical position of the printer, since the editor does not recognize that. A form feed can either be embedded or replaced by the ".FF" printer command.

Another advantage of using aliases is simply that they are easier to remember and to understand than most printer control sequences.

An alias is any text in the file to be printed delimited by exclamation points. For example !bold! can be used as an alias instructing the editor to print, instead, the printer control sequence to switch to bold face. The I delimiter can be changed with the ".DE" printer command.

The aliases and the associated substitute text must be in a file named PS. Alias which must be in the prefix #5 directory. This file is a standard text file that can be written with the Text Editor. The first character in any line of the file determines how the line will be interpreted. Any line in the PS.Alias file starting with a control character, period, or space is ignored during printing as far as aliases are concerned, and thus can be used as a comment line in the PS.Aljas file. You can use the ".AL" command to substitute a file of any name for the PS.Alias file.

A line in the PS.Alias file beginning with any other character will signal that it is an alias designation. That character will serve as the "delimiter'. The alias is the text between the first character (delimiter) and the next occurrence of that character. The text (printer command) to replace the alias is that text between the third and fourth occurrences of the delimiter.

For example, this line in the PS.Alias file:

```
/bold//^[!/
```

asks the editor, when printing the file, to replace any occurrence of "!bold!" by ESC-!. (^1 is the ESC character.)

As in this example, you can use to signal a control character in the PS.Alias file. You could also just use embedded control characters if you prefer. The character itself is given by repeating it as ^^. The control- character is given by the two characters ^~.

Any text past the fourth delimiter is ignored while printing. Thus, you can put a comment after that point on any or all lines.

Unless a ".MO" command has been used to instruct the editor to ignore unrecognized aliases, they will just be printed as they appear in the file.

Aliases, and their replacements, are case sensitive, allowing more freedom in choosing them. Also, they are not counted as printing characters in checking the line length since they are assumed to be printer commands that will not be printed. This may not actually be the case, as with substitutes for printed control characters.

The aliases you use are completely up to you. It is wise to give them some thought, as the difficulty of the job of changing to another printer would depend strongly on such choices. There are sample alias files on the disk for the ImageWriter. Okidata 92, Epson and HP DeskJet 500 printers. The correct one must be renamed to PS .Alias for it to be used, or you must include a ".AL" printer command in your text file.

Going wild with aliases

Aliases support wild cards. This is difficult to describe in words and you should examine the examples below whether or not you understand the description.

A wild card is a special notation in an alias designation (line of the PS. Alias file) that can be replaced by characters given in the alias in the text being printed.

In the text file to be printed, substitutes for wild cards in an alias must come just before the second delimiter and are separated by a comma, or whatever has been substituted for the comma in a ".WI" printer command. For example,

```
! alias,xx,yyy!
```

stands for the alias "alias", and asks that "xx' be substituted for the first wild card and "yyy" for the second wild card.

If you want to use a different character for the comma you can use the ".WI" printer command to change the wild card separator. For example, if the text contains the command (a single line, justified left)

```
.wi\
```

then the preceding alias would be written as:

```
!a1ias\xx\yyy!
```

In the PS.Alias file, the wild cards are in the third field of the alias designation enclosed by commas. Just after the first comma must be a digit from 1 to 9 indicating which wild card it is, followed (perhaps) by a default for the wild card. For example,

```
/alias//aaa,2GG,bb,lHHH,ccc/
```

could be an alias designation, indicating that "GG" is the default for the second wild card and “HHH” the default for the *first.* This will result in the following replacements:

!alias,xx,yyy!	is printed as	“aaayyybbxxccc”
!alias,xx!	is printed as	“aaaGGbbxxccc”
!alias!	is printed as	“aaaGGbbHHHccc”
!a1ias, ,yyy!	is printed as	“aaayyybbHHHccc”

It is possible to change the separator , m an alias designation (having nothing to do with the “. WI” command except for being the same default character) by following the first character (the delimiter) of the designation with a comma, then the desired separator, and then the desired replacement for the = mark (see below). For example,

/, '=alias//aaa'2GG'bb'1HHH'ccc/

will have the same effect as the previous example. This allows the comma to be used in the designation as a non-separator.

Between the second and third delimiters you can put substitutions for the wild cards in the form /yes =A,no =B/ etc. This would result in a wild card “yes” being replaced by “A” in the text sent to the printer. An example of the syntax is the declaration:

/alias/on=YY,off=abc/^[,lxX,/

This example would result in the following replacements:

User syntax	To Printer	Explanation
!alias!	ESC xX	No wild card given, so default 'xX' used
!alias, on!	ESC YY	'on' replaced by 'YY' substituted for 'xX'
!alias, off!	ESC abc	'off' replaced by 'abc' substituted for 'xX'
!alias,Pq!	ESC Pq	'Pq', having no substitute, replaces 'xX'

A real life example for the ImageWriter printers is the declaration:

/pitch/9=n,10=N,12=E,13.4=e,15=q.17=Q/^[,IN,/

resulting in, for example:

User syntax	Sent to printer	Result
!pitch!	ESC N	pitch 10 (the default)
!pitch, 12!	ESC E	pitch 12 (12 chars/inch)
!pitch, 17!	ESC Q	pitch 17 (17 chars/inch)

This syntax is designed for convenience and simplicity of use **in** text files to be printed and not necessarily in the of construction of the PS.Alias file.

It is suggested that you try to make alias syntax as simple as possible and to use wild cards only in cases of clear utility. Aliases that are easy to remember are the best. Make the syntax describe the action.

The Open command

The Open command brings up a scroll box showing the current prefix and up to 16 filenames and their types from among DIR, TXT, SRC, and AWP type files. Keys ↑ and ↓ scroll the list. Key / brings up another box in which you can type in the pathname of the file you want loaded. The ← key closes the current directory and pops one level off the prefix and presents the list of appropriate files in that directory. A carriage return on a directory file adds that directory to the prefix and shows its files. A carriage return on another type of file loads that file into the buffer. The ESCAPE key, as usual, aborts. Pressing ← when a root directory is shown will poll disk devices, except for 5.25 inch drives, and list the volumes found for selection. Volumes can also be changed by the Command Center Prefix command, and that method must be used to change to a volume on a *5.25* inch disk.

If you hold down the Apple key when pressing RTN to open a file, carriage returns will be inserted to make line lengths at most 79 characters. Otherwise they can be as long as 255 characters. This is a convenience for dealing with files from outside sources that do not use carnage returns to terminate each line. The Format command can also be used to do this and is more versatile.

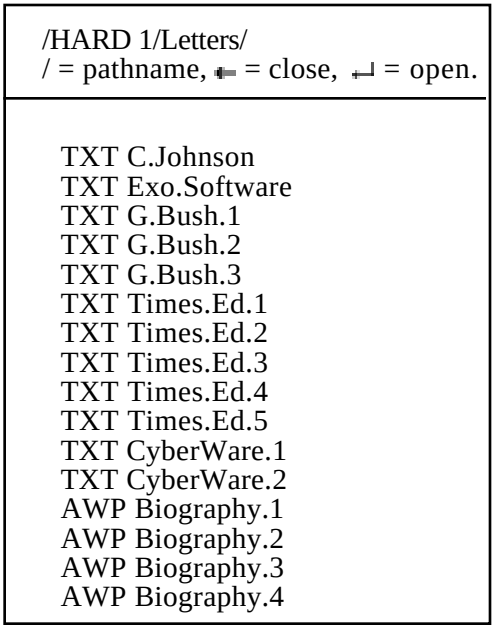


Figure 12. *The Open box.*

Keyboard macros

The Text Editor has a built-in provision for keyboard macros. The macro definitions are kept in a file named PS. Macros which is placed in the prefix #5 directory. If the file is not there, it will be created when the macros are first saved, provided prefix *#5* has been set up in Modify parms to point to a valid directory.

Note that prefix #5 is also used by the Appointment Calendar, Info Desk, and Backup.

A macro is invoked by holding down the Option key while pressing another key. If a macro for that key does not exist then the key is passed through as if Option were not being held down.

There are three ways to teach the Text Editor new macros, or to replace old ones by new ones. These are also accessed by the Option key together with three other keys, all of which are changeable in Modify parms:

Option *

> This is the easiest way of defining a macro. It produces a pop-up box asking first for the command key, and then for the definition. RTN terminates the definition and ESC aborts it.

Option —

> This lets you define a macro from text showing on the screen. When Option — is pressed the cursor changes to a ▬ and will not be flashing. The non-flashing indicates that the program is waiting for the key attached to the macro, and that will be the first key you press. Then the cursor, still a ▬, will flash normally. At this point you are expected to press the -. key a number of times to copy over the text on the screen. When you have copied as much as you wish, cancel the macro definition mode with the CLEAR key (or ^X). The cursor will return to its normal condition.

Option +

> This is the most general way to define a macro, and the only way that will accept the RTN key and Apple commands as part of the definition. When you give this command, the cursor will change to a non-flashing open apple. Again, this indicates that it is expecting the key to attach to the macro. When the desired macro key is pressed the cursor will start flashing indicating it wants the definition. At that point you type any sequence of commands. When the definition is done, press CLEAR or ^X, the only way to finish the definition. At that point the regular cursor will return.

In all of these cases a macro can be “locked” if the Option key is down when you press the key the macro will be assigned to, or at RTN in case of the macro box. If a macro is locked then that key cannot be redefined unless it is unlocked and deleted via the macro box (Option *) first.

The PS.Macros file is loaded when you first enter the Text Editor. It will not be reloaded unless the Text Editor itself has to be reloaded. If you change or create any macros while you are in the Text Editor, a pop-up box will appear when you exit asking if you want the changes in the macros saved.

You can see the macro definitions at the end of the Help display (Apple ?). Long macros are truncated at 40 characters in that display.

The limit on the number of macros is 254. The average length a macro can have (if you use that many) is 32 characters, but there is no limit on the length of an individual macro except for the total length of the entire macro table.

Since the macro command keys *, +, and — (or whatever you have changed them to) are not accepted for attaching to macros, you can press any one of them if you accidentally go into macro definition mode, and no macro will be created.

Here are some examples of creating and using keyboard macros:

Example 1.

(a) Press Option + which should produce the non-flashing apple cursor.
(b) Press the → key The apple cursor should start flashing.
(c) Press the → key 10 times. The cursor should remain an apple.
(d) Press CLEAR. The cursor should return to normal.

Try it Out by pressing Option →. The cursor should move 10 places right. You can try the same with the other arrow keys. For the ↑ and ↓ keys, 5 repetitions might be better than 10.

Example 2.

(a) Press Option * which should bring up a pop-up box asking for the “macro key”.
(b) Press the first letter in your name. The box should display it and ask for the definition.
(c) Type your name ending with RTN.

Try it out by pressing Option together with the first letter of your name.

Example 3.

(a) Position the cursor on some text on the screen.
(b) Press Option — which should produce a non-flashing → cursor.
(c) Press any key you would like to be the macro command for the text. The cursor should begin to flash, but is still the → character.
(d) Copy over some of the text with the → key.
(e) Press CLEAR to terminate the definition.

Try it Out by holding down Option while pressing the key in step (c). This method of definition is most useful for quick and dirty macros you don’t intend to keep.

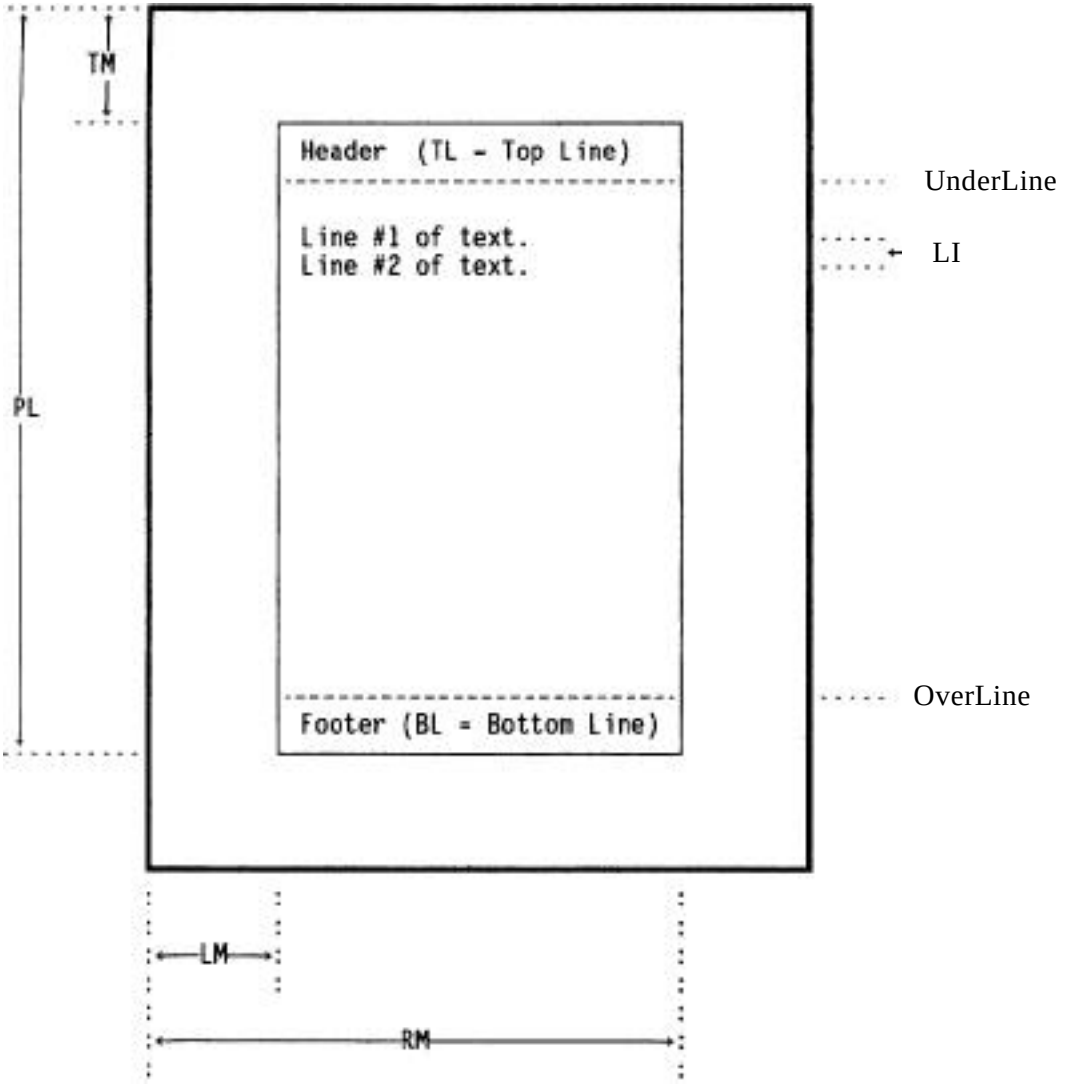


Figure 13. *Text Editor page layout.*

Part VII. Number Cruncher

☼ Introduction

This module provides the powerful number handling abilities of the most capable calculators, as well as their convenience of use. For quick numerical calculations, and for handling many types of numerical algorithms, it is unmatched. It is also a splendid tutorial device because of its exclusive advanced capabilities.

The calculator is a much improved 16-bit version of the 8-bit Number Cruncher I wrote in 1982 and revised for ProDOS-8 in 1986.

The main features of the calculator are:

- It has a complex number mode as well as the usual real mode.
- does numerical integrations including complex line integrals.
- It is programmable, and its programs can be written and edited with a built-in editor. Programs can be stored to disk.
- It directly supports such functions as sin, cos, tan, arcsin, arccos. arctan, and the hyperbolic analogues of these. It also supports exp, log, abs, powers, l/x, real part, imaginary part, angle, square root, π, and int. **All** functions are supported by the complex mode, so you can calculate things like SQR(-1), LOG(i) and EXP(1 +2i).
- It operates in either degree or radian measure.
- It has 9 user definable functions, which can also be written and edited using the built-in editor, and 9 user “formulas”.
- Besides the usual decimal mode there is a hexadecimal mode allowing operation in both 32 bit hex integer form and the SANE internal hex extended floating point format. This is a great convenience as well as an educational tool.
- It is capable of reading a text file for data. This allows running several programs on the same data. There is no limit on the size of the data file.
- It has a printer interface with output formatting abilities.
- There are 144 memory registers (in 16 banks of 9 each) and 4 stack registers and a number of memory manipulation commands. All registers can be used transparently in complex mode.
- It has a “step” mode for debugging programs.

- It has resident help information accessible at any time, and a resident program editor.
- All keys are redefinable.
- It can plot functions on the super high resolution screen, including polar and parametric equations and real or complex surfaces.
- Over *75* programs are included to do such calculations as computing averages and standard deviations, doing linear regression and chi square calculations, finding the residue of a complex function, computing the area of a polygon, plotting functions and surfaces, etc.
- There is a provision for algebraic functions, meaning functions defined the way you write them.

NOTE.' Because of space considerations, the calculator programs are provided in an archived form and require unpacking by the UNPACK.ARC program in the ProSel-16 directory. Copy the files UNPACK.ARC and CALC.ARC to your disk, go into BASIC, set the prefix to the place the files were copied to, and then type RUN UNPACK.ARC. This will create 78 program files and I test datafile.

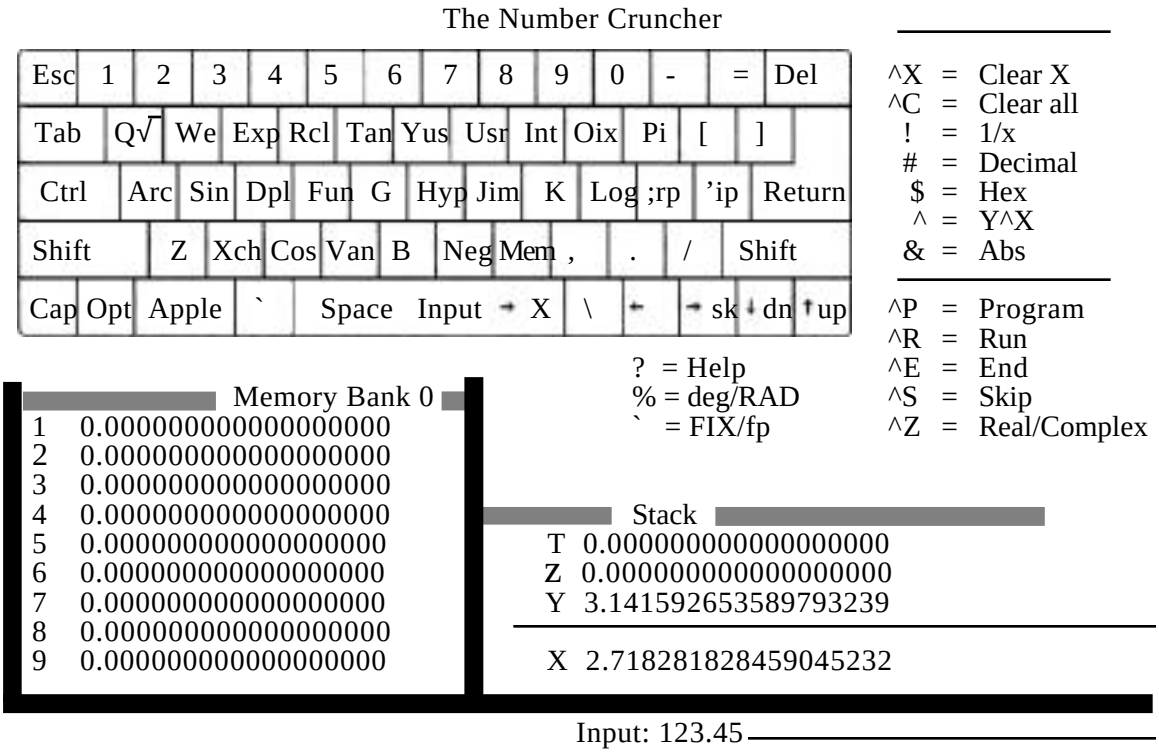


Figure 14. Calculator screen in real mode.

Getting familiar with the calculator

The calculator works with the “Reverse Polish Notation” (RPN) used by the most advanced calculators. This is a very efficient and logical method of calculation that uses no parentheses, keeping, instead, intermediate results in a four register “stack” This notation may require a little getting used to, but after a few minutes using it, its definite advantages will become clear and its operation will become quite natural. Since all registers are shown on the screen, the operation of the calculator is easily and quickly learned, much more so than with calculators showing only the main register.

Reverse Polish Notation is based on the work of the famed Polish mathematician Jan Lukasiewicz (1878-1956).

The Reverse Polish method can be quickly summarized by saying that the operations (sin, exp, log, etc.) that operate on only one number, are performed as soon as the key is pressed and are performed on the number held in the “X” (main) register. Those operations (*, +, etc.) which operate on two numbers, also are performed as soon as the key is pressed and are performed on the X and Y registers, leaving the result in the X register. In the latter case, the previous X and Y registers are lost, and the “stack” moves down. (That is, the contents of Z are put in Y, those of T are put in Z and T retains its former value.)

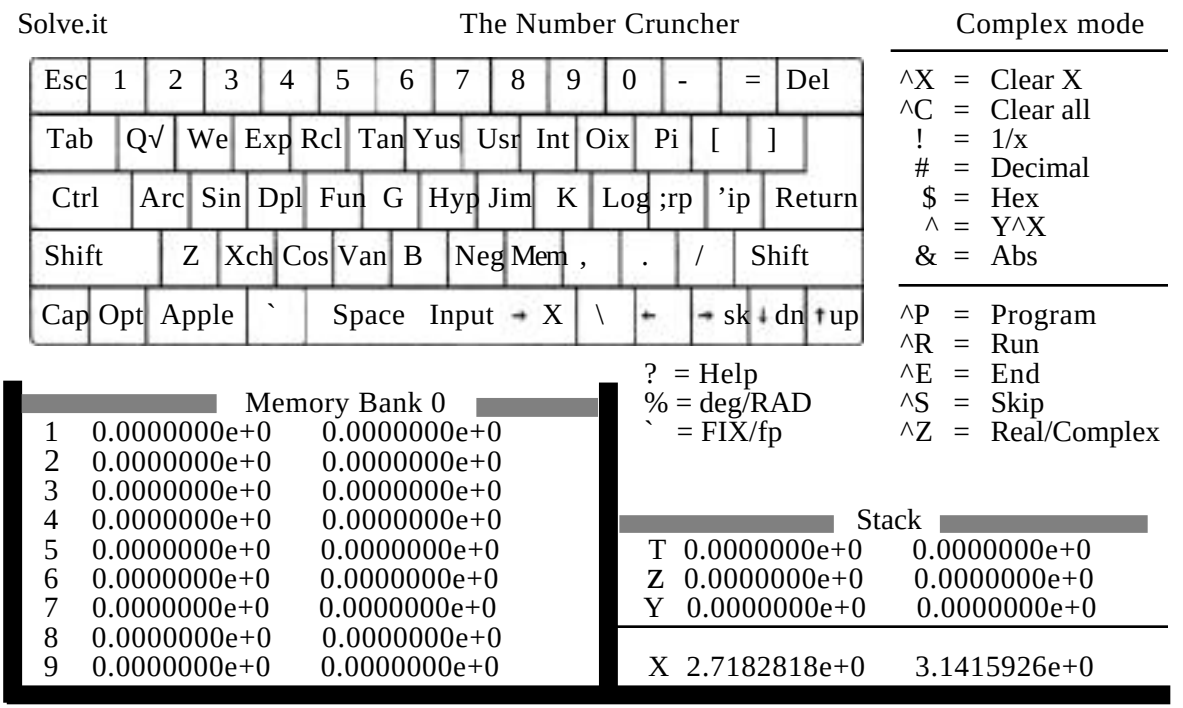


Figure 15. Calculator screen in real mode.

☼ Short RPN Tutorial

Let us go though a few simple operations to gain familiarity with the operation of the calculator. These examples illustrate the regular RPN mode of the calculator. You can also perform the operations using the algebraic entry provision: just press F, which will bring up the Function prompt, and then type the formula you want evaluated, ending with RTN. You are urged to learn the RPN method, however, since it is much more efficient.

> ***Example 1:*** Multiply 73 by 231:

(a) Enter the number 73 by pressing the *7* and then the 3 key.
(b) Copy this to the Y register by pressing RETURN or ENTER.
(c) Enter the number 231 by pressing 2 then 3 then 1.
(d) Do the multiplication by pressing the * key.

> ***Example 2.*** Compute 87 · 87 · 87:

(a) Enter the number 87.
(b) Copy it to the Y and Z registers by pressing RETURN twice.
(c) Multiply all three by pressing * twice.

> ***Example 3.*** Compute successive integral powers of 2:

(a) Press the key 2.
(b) Copy it to Y, Z and T by pressing RETURN three times.
(c) Press * several times. Each time produces the next power of 2.

> ***Example 4:*** Compute (3 · 4) + 7:

(a) Press 3.
(b) Press RETURN to copy 3 to Y.
(c) Press the 4 key.
(d) Press the * key
(e) Press the 7 key. (Note that the previous result is pushed up to the V register when you do this.)
(f) Press the + key.

> ***Example 5:*** Compute (3 · 4)+(7 · 8):

(a) Press 3.
(b) Press RETURN.
(c) Press 4
(d) Press *
(e) Press 7. (Note that V now contains 3 · 4 and X contains 7.)
(f) Press RETURN to move the contents of V to Z and to copy X to V.
(g) Press 8 (Now X has 8, V has 7, Z has 3 · 4.)
(h) Press * (Now X has 7 · 8 and V has 3 · 4.)
(i) Press + to get the final answer.

> ***Example 6:*** Compute 2 · EXP(2):

(a) Press 2.
(b) Press RETURN to copy it to V.
(c) Press the E key to get EXP(2).
(d) Press * for the final result.

> ***Example 7***: Compute 2 · L0G(3 · SQR(4 · 4+5))

(a) Press 5. (One computes from the inside out.)
(b) Press RETURN.
(c) Press 4.
(d) Press RETURN to copy 4 to V and to push 5 to Z.
(e) Press * to compute 4 · 4 in X and pull 5 down to V.
(f) Press + to add 4 · 4 to 5.
(g) Press Q to compute the square root.
(h) Press 3 which will automatically push the previous result to Y.
(I) Press *.
(j) Press L to compute the logarithm (natural).
(k) Press 2.
(1) Press * for the final result.

> ***Example 8:*** Compute ARCSIN(.5)/π in radians:

(a) If "RAD" is not highlighted then press %.
(b) Press the . key then 5.
(c) Press A then S to get the ARCSIN.
(Note that after pressing A only S, C, T, or H will be accepted. If you pressed A then H then S then ARGSINH(.5) would result.)
(d) Press P to get π and to push the previous result up.
(e) Press / to divide for the final answer.

> ***Example 9***: Compute the square of SIN(30˚):

(a) Press the % key if needed to highlight DEG on the screen.
(b) Press 3 then 0.
(c) Press S to compute the SIN.
(d) Press RETURN to copy the result to Y.
(e) Press * to multiply it by itself.

> ***Example 10:*** Compute l0-π:

(a) Press 1 and then 0 to put 10 in X-π:
(b) Press the P key to get π in X and push the 10 to V.
(c) Press N to negate the X register which then has -π.
(d) Press ^ for the final answer.

General description of operation

As indicated by the examples, there are certain times when an automatic “enter” is performed when you start a numerical entry (or π or a memory recall). The question arises as to when this happens and when it does not. The answer is that the auto-enter is done when the previous operation is such that there would be no reason for it having been made if its result is not desired. More precisely, an auto-enter is done when one starts numerical input or π or a memory recall after any function (sin, log, etc.) or arithmetic operation (*, +, etc.) or any user defined function (Y followed by 1-9) or a memory recall (R followed by 1-9). An auto enter is not done following a memory store (M followed by 1-9), an ENTER (RETURN key), a stack operation (↑, ↓, X), or a register clear (^X). The mode change keys *(%, $, #,* `, J, ^Z, D), the increase or decrease memory keys (<, >), and the keys used for programming instructions do not affect the status of the auto enter. When in doubt, of course, one should just press the RETURN key to do the “enter” manually, or, if an enter is not desired, just use ^X (or CLEAR) which clears the X register and turns off any auto enter.

The effect on the stack of any function of one argument (sin, exp, 1/x, etc.) is simply that the operation is performed on the X register and the other registers are not affected. This is also the case for a user defined function, even if it makes use of other registers. Operations which require two arguments (*, +, ^, etc.) place the result in the X register, destroy the Y register, move Z to Y, T to Z and retain the T register. This is also true with an integration, even if the function being integrated requires use of other registers.

Key M followed by 1-9 copies the X register to the specified memory register. Key R followed by 1-9 recalls the specified memory register to the X register (after performing an “auto enter” if the previous operation calls for one). Key ^A followed by 1-9 adds the X register to the specified memory register.

Radian or degree mode can be toggled by the % key. In complex mode only the radian measure is supported and a switch *(*^Z) to complex mode automatically brings up radian measure mode. Radian measure is automatically selected upon running or stepping a program. The program may switch it back to degree mode.

Hex mode, produced by key $, causes the registers to be displayed in their exact 10 byte floating point hex format, just as they are stored in memory, and also the integer part of each register is displayed in the first highlight color if it is in range of 32 bits. This is a great facility for studying how SANE handles floating point numbers.

Numeric entry in hex mode is taken as a 32 bit integer if there are at most eight digits entered, and otherwise it is taken as a floating point hex number. Thus this mode can be used to do hex arithmetic and conversions between hex and decimal in either integer or floating point. Key # switches back to decimal mode. The sign (high bit) and exponent of the floating point display is separated from the 8-byte mantissa by a space. Also note that the bytes of the display are high byte first, the opposite of how they are kept in memory, because they are better understood that way.

Since the E key is the EXP function and the - key is the subtract function, other keys must be used to produce the - and E characters for numeric input. These are the keys — and W. Note that the negate key “N” is convenient for inputting a negative number, however. Holding the Apple key down when pressing - or E will also allow those keys to be interpreted as numeric input.

Exit from the Calculator

To exit the calculator, press the? key to get to the HELP menu and then press ESC. You will be asked if you mean it.

Loading and Saving programs

The L and S commands on the help menu are for loading and saving calculator programs. Pressing them will show the appropriate files and then ask for the program name.

Numerical input

When the word DATA is displayed at the lower left of the screen, the calculator is expecting numeric input. This ordinarily happens when a program is running that wants input from the keyboard. It can be brought up in immediate mode, however, simply by pressing the ESCAPE key. When DATA is showing, the memory and stack operations are not available and, similarly, the two-register functions cannot be used. You may type a number, use a single argument function, or combinations of these. (E.g., to input π, just press P.) The input is terminated by the ESCAPE key or the RETURN key, which, in this mode, is no longer an ENTER! In this mode the - key and the E key just give those characters. (It is possible to perform an EXP function in this mode by using the @ key.) When input is terminated by ESC or RTN the word DATA will disappear from the screen.

Example: Input -1.34E-12

Method 1: type_1.34W_12

Method 2: type ESC (if DATA is not showing) then -1.34E-12 RETURN

Method 3: type 1.34W_12N

Method 4: type ESC (if DATA is not showing) then 1.34E-12N RETURN

Methods: type Apple-"-" then 1.34 then Apple-"E-" then 12

Degree/Radian modes

Key % switches between these modes. The only things affected by the status of this mode are the trigonometric functions sin, cos, tan, arcsin, arccos and arctan. A switch to complex mode automatically selects radian mode and a switch to degree mode cannot be done when in complex mode. Running a program automatically selects radian measure, so programs intended for degree mode should start with "%".

Help provisions

At any time, even when a program is requesting input, you can read the help information or you can even list the program that is in memory without affecting the status of the program or the registers. To do this, simply press the help command key "?" This leads to the help menu from which several disk commands can be issued as well as the help listing and entry to the program editor. The help listing describes the action of all active keys.

Note that only the most common commands are indicated on the main display screen (keyboard replica) and a couple of those are cryptic.

Complex mode

The Z key toggles complex mode. When in this mode, the input of imaginary parts can be toggled with the J key. Real parts are shown as normal text, while the imaginary parts are in the first highlight color (usually red).

For technical reasons, degree mode is not available when in complex mode.

All functions are supported by complex mode. (E.g., try the square root or log of-1 or the ARCSIN of 2.)

Running a program automatically sets real mode. Thus a program that is to use complex mode must start with the ^Z command. The program will also default to real part data entry and should keep track of the status of this mode if unexpected results are to be avoided.

To input a number in complex mode just type the real or imaginary part, whichever is indicated, then J then the other part. Usually the entry of a number will automatically zero the opposite part, so it may not be necessary to do that manually. You can also input a complex number by pressing the function key F and then typing the number (e.g., 3+8i or l0-ipi or l0-i*pi).

Although hexadecimal mode is compatible with complex mode, only the real parts of the registers are shown because of display limitations.

Since complex mode entails significantly more internal work than does real mode, you should expect slower operation and somewhat less accurate computations.

In complex mode, integrations (see the section on definite integrals) are done along the line joining the contents of the X and Y registers. By piecing together such segments, more complicated line integrals can be done. (See the program RESIDUE.) Because of well known properties of complex analytic functions, these line integrals are completely general.

Programming

A program can be entered into memory either from the calculator or from the editor. The latter is to be preferred for complicated programs, but in the former case you can see what is happening to the registers.

The command to begin entry of a program is ^P. (From the editor, however, you just type A < RTN> for add.) You must end entry of a program with a ^E. In between the ^P and ^E just type any sequence of operations you want performed by the program.

If you want the program to pause for user input then press ESC. If you are writing the program from the calculator mode rather than from the editor then you can type any number at this point (while the word DATA is displayed) and this will not go into the program. To terminate the data entry and continue writing the program, press escape again or the return key. If, when programming with the editor, you press ESC to put an INPUT line in the program, the editor will allow you to follow this with a short prompt string to be displayed after the word DATA. End the prompt string by pressing RETURN, which will not go into the program.

To write a looping segment into a program, type a ‘W (for “while”), followed by -, #, or + (indicating while negative, while nonzero, and while positive or zero). The loop segment must be terminated by a L followed by specification of a memory register 1-9. When the program is running, the loop will continue while the condition holds for the specified memory register, or until a “skip” command **(→)** is pressed by the user in place of a numerical input. Note that the decrement memory command can be used to do a loop with an explicit number of steps. Loops may be nested.

A \key followed by -,#, or + puts an IF condition in the program. This causes a skip to the next ELSE or ENDIF if the X register is non-negative, zero, or negative, respectively. An ELSE is provided by the ^V key and reverses the IF condition. An ENDIF is given by ^Q**.**

For example, to write a program which will get input from the user and then compute x sin(x), where x is the input, just type the sequence of eight keys:

^P ESC 1 ESC RETURNS * ^E.

(From the editor, you would type “Dl,” to first delete a program in memory, then “A <RTN>” to go to the add mode, then ESC RETURN RETURN S *^Ê.)

For another example, to write a program which would select complex mode, get input, and compute log(sin(x)) of the input, just type the seven keys:

^P ^Z ESC ESC S L ^E.

Running a program

Pressing ^R will run the program in memory. In order to increase speed (by a factor of 3 to 300) the register display is not updated while a program is running.

However, sometimes it is interesting to watch the registers, so I have provided a means to force registers to be displayed while a program is running: Hold the Apple key down when pressing ^R to run the program. If you have a free half hour you might try this on one of the Fourier programs.

If the program stops and displays the word DATA at the lower left then it is expecting numerical input. The input should be done as described under “Numerical input” and terminated with either an ESC or a RETURN.

Some programs will loop forever if they expect an indeterminate amount of data input. (For example, a program to average a list of numbers input by the user.) Such programs should be terminated with the → key which causes the current loop to be skipped. (Some programs may have more than one of these.)

You can also “step” a program by issuing ^S instead of ^R. In this case the registers are displayed at each step. Each step waits for a keypress to continue. In addition, the operation to be processed next is indicated at the bottom. In this indication, control characters are shown in the first highlight color. The inverse hyperbolic functions are indicated by a lower case “a”. Otherwise the command is shown by the key or key sequence used to issue the command.

At any time, a running or stepping program can be aborted with the ^E (end) command.

During program execution the ^R and ^S keys can be used to switch between the RUN and STEP modes.

To load a program from disk, press the ? key to get to the help menu and then issue the L command. That will list all program tiles in the current prefix #0 directory and ask for the name of one to load. (If necessary, first use the P command to change the prefix.) After the file is loaded you will be returned to the main calculator screen and can run the program with the ^R command.

Regardless of the mode when a program is run with the ^R key, the program starts in real, radian, floating point, and decimal modes. This is so that the program starts in a standard, determined manner. The program can switch into any of the other modes.

Using Datafiles

For some types of programs, particularly statistical ones, you may want to run the program on a list of data without having to type the data in every time. For this, there is a provision for the access by a program of a text file for numerical input rather than input from the keyboard. The text file, called here a datafile, must be opened by going to the help menu and typing the 0 command and then giving the name of the datafile. At this point you will be returned to the calculator and the name of the datafile will be displayed at the top of the screen after the name of the last loaded program. Then you should press ^R to run the program. When the program first pauses for input, just press ^D instead and the program will proceed to read input from the open file. At the end of the program, the file will be closed and will have to be reopened to be used again.

If the program expects termination by the "skip" command, then the datafile should end with that character (^U). If the file reaches its end before the program ends and does not have the proper skip, you will see "Press a key" at the bottom of the screen. When a key is pressed, you will be sent to the help menu. At that point you can return to the calculator and press E to abort the program, or you can open another datafile. In the latter case, the program will continue operation with the new datafile, automatically. Thus, datafiles can be chained in this manner.

Datafiles can be written by the Text Editor. Commas may be used, in place of carriage returns, to separate data in a datafile. The calculator interprets commas and carriage returns equivalently when read from a datafile.

Using the Editor

Pressing E from the help menu, sends you to the editor, where you can do various things. For example the program in memory can be listed by the L command. "Ll0,20" will list lines 10 to 20, "LI0," will list line 10 to the end, etc. The listing can be paused by pressing the space bar, and then stepped by repeated pressing of the space bar. The listing is aborted by ^C or ESC. Any other key restarts the listing. A listing can be sent to a printer by typing a P < slot #> command, any required initialization string and then the list command. (Each of these must be terminated by a carnage return.) "P0" will turn the printer interface off again as will a return to the help menu by the Q command.

You may delete any range of lines and may insert or add new ones. Note that typing "Dl, <RTN>" will delete the entire program. Any insertion or addition must be terminated by a 'E (for end). This character should be remembered at all times. It is used to end a program, end programming commands, end insertion of lines, etc.

From the editor you can also access key redefinition. See the section describing that option.

☼ Numerical integration

The calculator has a built in facility for doing numerical definite integrals. The algorithm used is Simpson's rule with 128 subdivisions. In most cases the accuracy is quite good. To do a numerical integration from the keyboard, you must write a short program segment that starts with **"(",** which is the signal for start of integration, then contains the function to be integrated, then ends with ")", meaning end of integral. The function inside the parentheses should be thought of as acting on the contents of the X register and terminating with the value of the function in the X register. (For example the "function" RETURN RETURN * * is the cube function, and the integration program written by the key sequence

ˆP (RETURN RETURN * *) ˆE

computes the integral of X X X

To perform the integration, once the program segment has been entered, you just place the left end point **in** the X register, the right end point in the Y register and run the program. Thus, to integrate X X X from 3 to *5,* you would enter the program segment shown above, then type 5 RETURN 3 ˆR.

Integrations may also be performed in complex mode. There is no difference except that the "program" must begin with ˆZ to turn on complex mode. Also, of course, more time will be required for the computations to be performed in this mode. By putting together such several integrations in a program, one can integrate over any polygonal path in the complex plane. The program RESIDUE, for example, integrates along a diamond shaped path about the origin.

In the real mode (only) there is a provision for precision integration. This is an "adaptive iterative routine" which continues to subdivide the interval until the numerical approximation becomes stable within a specified tolerance.

The command for precision mode is simply two integral **"("** commands in a row. (It is also suggested that the end of integral command **")"** be doubled so that the editor listing will have the correct tabbing.) On entry to a precision integration, the T register should contain the desired tolerance. If this is zero, or if it is less than the calculator deems appropriate, then the calculator will choose a tolerance automatically, usually 1E-6. On completion of the integration, this T register will have dropped into the Z register and the T register will Contain the estimated maximum error m the result. Note that the actual precision of the result will usually be better than shown by the tolerance in the T register, often by two decimal places or more. This is because the tolerance represents a worst case estimate.

The precision mode is recommended when the function is very wiggly such as those involved in computing the higher Fourier coefficients of a function. The tolerance should not be taken too small, since this causes longer runs. Usually a tolerance of 1E-4 works well. An OUT OF MEMORY error will indicate that the integration "stack" has overflowed and the integral cannot be computed within the specified tolerance.

☼ Algebraic functions

Key F allows definition and/or evaluation of an algebraic function such as sin(2x)cos(4x). Its action depends on the circumstances when it is called. If it is called in immediate mode then it puts a prompt at the bottom of the screen asking for you to type in the function you want. It then evaluates the function placing the result in X. If you press F while entering a program with the editor, it will wait for you to type in the function, showing it at the top of the screen. The input of the function is terminated by RETURN. This enters the function into the program you are typing. When a program is running and a FORMULA line (key F) is executed then one of three actions occur:

(1) If the line has no formula attached (you just pressed RTN at the function input prompt in the editor), and if the program is not in a plot, solve, or integrate segment, then the calculator will prompt you to enter a function, and the function will then be evaluated with the result going to the X register.

(2) If the line has no formula attached and the program is in a plot, solve, or integrate segment, then the calculator will evaluate the existing function (the last one entered or the last one defined previously in the program) and place the result in the X register.

(3) if the line has a formula attached then the calculator just parses and compiles that formula for later use by a FORMULA line without a formula or by a SAVE FUNCTION 1-9 command (ˆ@ 1-9), see below.

Formulas can use any of the standard functions:

SIN	ARCSIN	SINH	ARGSINH
COS	ARCCOS	COSH	ARGCOSH
TAN	ARCTAN	TANH	ARGTANH
CSC	ARCCSC	INT	INTR
SEC	ARCSEC	SQR	
COT	ARCCOT	ABS	
EXP	LOG	LOG10	LOG2

Of course, the common operators +, -, *, /, ˆ can be used in addition to \; x\y yields the remainder of x divided by y.

For parentheses you can use 0, [1, or {}to make reading the function easier.

The “ARG” in the inverse hyperbolic functions can be replaced with the more common, if less meaningful, “ARC”. “LN” can be used in place of “LOG”. “ASN”, “ACN” and ATN” can be used in place of “ARCSIN”, “ARCCOS” and “ARCTAN”.

“INTR” means “round to integer”.

Any of the stack registers X, Y, Z and T and the memory registers R1-R9 can also be used in the formula as can numeric constants and the constant “PT”

Moreover, you can use the standard mathematical notation of juxtaposition without the explicit "*" whenever it is unambiguous. For example, “4xsin(2pix)” is a perfectly acceptable substitute for "4*x*sin(2*pi*x)" and is much more readable. You can also use spaces for even more readability, as in “4x sin(2pi x)”.

In complex mode you can use “i” or “j” for SQR(-1). Thus, l+2i is perfectly legal and, if the X and Y registers contain real numbers then x + iy yields exactly what you would expect. The characters “i’ and ‘j” will give syntax errors in real mode. Complex mode also allows the functions RE and TM (real and imaginary parts). These, as all functions, must have an argument enclosed in parentheses.

Algebraic functions have some advantages over RPN formulas: they are easier to read and understand, and they can be entered on one line. They have the disadvantage that they are not as versatile: discontinuous functions or functions that cannot be expressed by a single formula are usually not available, but often can be defined easily in RPN form. Which one is faster varies with the individual function.

If you make an error when entering an algebraic function, which is detected by the parser (such as the wrong right parenthesis], then the words “Syntax error” will appear on the screen for a couple of seconds and the formula will reappear with the cursor in the position where the parser had a problem.

Example 1: To use the algebraic input mode to calculate SIN(π/12): Press the F key, at which the “Function:” prompt will appear at the bottom left of the screen. Then type in “sin(pi/ 12)” and press RTN. The answer will be in the X register.

Example 2: To compute SIN(x · π) for many values of x: Enter the first number x and then press the F key. At the “Function:” prompt type “sin(x pi)”. This will display the answer in the X register. Enter the next value of x, and press F again. This time at the “Function:” prompt, just press RETURN, because the default formula is still the proper one and will be retained if RETURN is pressed on the first character of the displayed formula. As before, the answer for the new datum will be returned in the X register. Repeat as required.

Example 3: To compute i^i: If not in complex mode, press ^Z to go into it. Press F to get the “Function:” prompt, and then type “i^i”. The answer is returned in the **X** register.

☼ User definable functions

There are 9 user definable RPN functions given by key Y followed by 1-9. These are defined by writing short program segments and using the editor command “>” followed by I to 9. This places the current program in the user function area. The user functions can also be downloaded to the program area for listing by using the editor command “<” 1-9. The user function will not be permanent, however, unless you give the command to save the current user functions (U) from the help menu. The user functions and key redefinitions (see that section) are contained in a disk file named “UserFunc” which is loaded upon initial entry to the calculator. If this file is not found, then all user functions will be disregarded and all keys will be interpreted in the standard manner.

The UserFunc file is looked for in (in this order):

(I) the present prefix #0 directory,
(2) the PROSEL.16 directory on the boot disk,
(3) the PROSEL.16/PROGS directory on the boot disk.

User functions can use all the registers, but when performed, will not change any registers except the X register. (The registers are saved and restored.) There is an auto-enter after a user function. The calculator comes with the user functions defined as:

Y 1: Multiplication by i (complex mode only).

Y2: Negate both real & imaginary parts.

Y3-Y7: Get primitive N-th root of unity, N =3-7 (complex mode only).

Y8: 10^X.

Y9: $LOG_{10}(X)$ (common log).

There is also a provision for calling up to 9 separate user defined algebraic formulas. These are accessed by key U followed by 1-9. Any one of these that has not been defined will produce a random number between 0 and 1. Some supplied programs assume that formula #9 is RANDOM. (In complex mode, RANDOM produces both random real part and imaginary part.) One defines these functions through the F command, either at the keyboard or in a program, and then by issuing the ^@ 1-9 (SAVE FORMULA) command. Algebraic formulas are saved to disk and loaded from disk along with the RPN user functions and the key redefinition table in the UserFunc file.

Printer interface

The printer interface refers to a provision for sending the number in the X register to a printer and has nothing to do with printing of program listings, which has already been discussed.

For a program to print the X register, the ^N command should be put in the program at the appropriate place. To instruct a carriage return to be sent to the printer, | should be used.

Note that “N never does a carnage return, so the | must be used if a carriage return is desired.

In addition, output can be tabbed, just before printing, by the ^T command followed by 1-9. The tab will be the specified number times 8.

Output can be formatted by using the ^F command. This must be followed by a format string containing dashes “-“ representing digit or space positions, possibly a decimal point, and possibly commas. There should be no more than 20 dashes; more will defeat the format command as will a format command not followed by a legal format string.

The format string will be used for all output by the program until another format statement occurs.

Numbers that are too large for the allotted space will be printed in their usual format.

Key redefinition

All keys used for calculator commands can be redefined by the user. This does not apply to help menu or edit commands. You should be careful about this provision. The redefinition will have no affect on the text shown on the screen.

Key redefinition is accessed through the editor and is self prompting. Redefined keys are shown and may be reset by defining each key to mean itself.

Redefinitions are not made permanent unless you use the help menu command U to save user defined functions, since the key definitions are kept in the same file with those.

Presently all lower case characters are defined to be their upper case equivalents.

Memory manipulations

There are eight memory manipulation commands:

M	1-9	Stores X in memory register indicated
R	1-9	Recalls memory register indicated to X (may do auto enter)
<	1-9	Decreases indicated register by one
>	1-9	Increases indicated register by one
^A	1-9	Adds X register to indicated memory register
^0		Stores X in register N (1-8) if register 9 contains N
^I		Recalls register N (1-8) to X if register 9 contains N
^B	-#+	Rotates memory banks

No commands except B affect the memory banks not shown on the screen.

The ^B # option just aborts the B command.

You should not use a ^B command in a user function because the registers are saved and restored when doing a user function and this would just result in one bank being copied to another.

For examples of the use of the indexed store command, see the Fourier coefficient computation programs.

Subroutines

There is a provision for up to 9 subroutines. The routines must be placed at the start of the program, or, at least, before they are used. (They are not executed unless called by a GOSUB.)

To define a subroutine while programming, use the K key followed by specification (1-9) of the subroutine number. These need not be in numeric order. Terminate the subroutine definition with key B for “return” (or “back”).

To call the routine insert a GOSUB in the program with key G, followed by the number 1-9 of the subroutine to be called.

Subroutines may be nested. Calling a nonexistent subroutine yields an UNDEFINED STATEMENT error.

☼ Plotting functions

To plot a function on the super high resolution screen, you must write a short program segment that contains the function to be plotted inside square brackets. For example, the key sequence ^P [RTN *]^E sets up a program segment to plot the squaring function. To do the plot, you must then set X and Y to the left and right boundaries, and Z and T to the bottom and top boundaries desired and then press ^R to run the program. Of course, the stack registers may be set up at the start of the program instead of from the keyboard, if desired.

In complex mode this works somewhat differently. Instead of plotting the value of the function against the argument, the complex mode plots the complex value (a point in the plane) of the function for values of the argument running along the straight line between the points which are contained in the X and ***Y*** registers at the start of the plot command. Also, the Z register contains the scale factor (usually 1) and the T register contains the point (complex) to be put at the center of the screen (usually 0). Only the real part of the Z register is used. Scale 1 will give a distance of 1 from the center of the screen to the top. Scale .5 gives distance 2, etc.

Complex mode can be used to plot polar equations and more general parametric functions. See the demonstration programs for several examples of this.

Key ^Y is the lead in to various “screen functions”. Control Y followed by 1-9 gives:

1. Clear screen at end of next plot. *
2. Don’t clear screen at end of next plot.
3. Plot axes. *
4. Don’t plot axes.
5. Use “dotted” plotting. *
6. Use “solid” plotting.
7. Sound bell and wait for keypress at end of plot. *
8. Do not wait for keypress and do not return to text mode.
9. Set all defaults.

Conditions marked with * are the defaults and all programs start with these. Function 8 provides a way to “continue” a plot without user intervention.

Screen function 3, if issued when axes are already enabled, will also cause grid points to be plotted. These will occur at all tick marks or every 5 tick marks depending on the scale.

In all cases, 640 points are plotted by the plot command, but multiples of this can be obtained using screen functions 2, 4, and 8 followed by another plot.

Plotting of polar equations is simplified by the function Exp(i*realpart) given by key “O”. Note that this just gives the complex number cos(t) +i*sin(t) where t is the real part of X.

A plot saves and replaces all registers, so that the registers will be the same at the end of the plot as they were at its start.

Ticking of axes is done according to the coordinate dimensions given in the program and not by meaningless screen coordinates. The tick interval is always some integral power often (e.g., 1/100, 1/10, 1, 10, 100 etc.). The ticks on the horizontal axis can be somewhat denser than those on the vertical axis since the horizontal resolution is three times the vertical resolution. Vertical ticks are as close as they can be to avoid spacing errors that are obvious to the eye.

Key ^G is the lead in to setting the color of the axes (if used) and the color of the plot. Control G followed by 1-9 gives (using Yellow, Red and Blue to denote the primary text color, the first highlight color and the second highlight color respectively):

	Axes color	Plotting color
1.	Blue	Yellow
2.	Blue	Red
3.	Blue	Blue
4.	Yellow	Yellow
5.	Yellow	Red
6.	Yellow	Blue
7.	Red	Yellow
8.	Red	Red
9.	Red	Blue

When the bell sounds at the end of a plot, you can press Apple-S which will bring up a dialog asking for a filename to save the graphics screen to. The resulting file will be of screen image type (PlC filetype, $8000 in length).

Plotting surfaces

If the plot command “[” is repeated (“[[”) then it will be a surface plot, plotting the value of a real function of two variables x and y. Upon entry to this plot mode. Z and T should hold the vertical limits, and X and Y should hold the limits on the x variable. The limits for the y variable are taken the same as for x. Thus the plot is always of a square area and most often you want that to be centered at the origin. If you want to plot another region, or to stretch one of the variables, you can do that by adding a constant to X or Y or by multiplying X or Y by a constant at the start of the function calculation (after the ([). Between the [[and the end of plot command] (or I] to keep tabbing nice) should be the function itself acting on the X, Y registers and ending with the value of the function in the X register. The plot is a “hidden line” plot.

You can set several parameters for the surface plot by the SETPARMS command "=". This command transfers the contents of the displayed registers to the plotting routine, after which you can change the registers. The meanings of the parameters are:

(T)The T register gives the tilt of the plot. A positive tilt will place the eye above the x-y plane and a negative tilt, below. Tilt 0 gives a direct on view. The default (no “=” command) is a tilt of 1, which is a slope of 1 / 10 (angle about 5.7˚) if the vertical and horizontal scales are equal.

(Z)The Z register gives the color scheme for plotting, consisting of three colors, one for the upper part of the surface, one for the underneath part and one to show delimitation of ridges and edges. This scheme is given by a number between 0 and 87 in Z, default 0. The help facility shows the correspondence between numbers and color which depends on how your color pallets are set up in Modify parms. Numbers 30-57 give the same color scheme as do 0-27, but signal that the plot should be truncated above at level 0. (You can truncate anywhere by adding a constant to the result of the function just before the end of plot command “]”.) Similarly, 60-87 are the same as 0-27 but cause truncation below at level 0. The truncation given by colors 30 and up will work well with many surfaces but can be incompatible with the hidden line algorithm for some surfaces. Only trying it will tell.

(Y)The Y register gives a rotation that is to be performed on the x-y plane (i.e., provides a view of the surface from a viewpoint other than along the y axis). The default is 0. Nonzero values will slow the plotting slightly.

(X)The X register gives the “separation” and must be one of the numbers 1, 2, 4, 8, or 16. This indicates skipping of points for plotting. Each of these is about 4 times faster than the previous one. Separation 16 (and often 8) is usually too sparse to make a good picture but it does give a good indication of what the denser modes will give, and it is useful, because of its speed, for “tuning” a plot before changing to the final desired separation. Separations 1 and 2 are usually too dense for a good picture except for some very crinkly surfaces. Separation 4 usually gives quite good results and it is the default. Any illegal value is changed to the default of 4.

(1) Memory register 1 gives the limit of the x-coordinate. That is, points with x-coordinate larger than register 1 in absolute value will not be plotted. (Only if register 1 is nonzero.)

(2) Memory register 2 gives the limit of the y-coordinate.

(3) Memory register 3 gives the limit of the radius ($\sqrt{(x^2 + y^2)}$). Use of this when the rotation is zero can appreciably speed up the plotting.

The limits given by registers 1-3 differ from those in the X, Y registers at the start of the plot command "[["in that the latter determine the screen dimensions, whereas the former limit the plotting within the established dimensions. With no limits in registers 1-3 the whole horizontal range of the screen is plotted, and registers 1-3 limit that range. Even if one uses the color parameter to limit the size of the plot, using registers 1-3 can speed up the plotting even without altering the plotted range because these limits are checked before any calculation for the present coordinates is carried Out, and the calculation is skipped if the point is outside the range specified by registers 1-3.

There are many examples of surface plots in the provided example programs. An FPE card is strongly recommended, and will speed up the plotting by a factor of 3 to 40+.

Plots of any sort can be labeled with the LABEL command (key "). This is only accepted in a program. Just press “ in the editor and you will be prompted for the text of the label. Previous to that line in the program you must also set the registers X, Y, and Z: Z must contain a number from 0 to 17. Values 0-8 are one less than the color of the text as given in the Appendix “Summary of colors...”. Values 9-17 are the same as 0-8 but print the text in inverse. Register X must contain 0-78 and is the horizontal tab, as if on the text screen, and register Y must contain 0-23 and is the vertical tab.

A LABEL command in a program without text attached (just press RETURN after the") will print the current default formula. Thus this use of LABEL should ordinarily come right after an F formula command (either with an explicit formula or asking for the input of a formula).

The LABEL command must come before the plot segment that is to be labeled. The label will be printed automatically at the end of the plot. Putting a LABEL command inside the plot brackets will work but will slow down the plotting.

Surface plotting can also be done in complex mode. The input parameters are the same as in real mode (in particular, only the real parts of the registers are used for parameters). The only difference is that the function computed is a function of the complex X register instead of the real X and ***Y*** registers. The value that is plotted (the height) is the real part of the X register at the end of computing the function (at the ENDPLOT “]” command). To plot the imaginary part of the function, you just have to put the IMAGPART "," command at the end of the function (or, in algebraic form, enclose the rest of the function in "im( )"). To plot the absolute value of the function, just put an ABS command at the end, and so on.

☼ Solving equations

There is a built in facility for solving equations, or, more precisely, finding zeros of a function. This is done from a program by key Z. This modifies a PLOT segment (function in square brackets) into a segment containing a function to be solved for a zero.

This is done by an iterative procedure and the initial guess should be in the X register when entering the solve mode.

For example, to find a value of x for which cos(x) =x, (converted to finding a zero of the function x-cos(x)), enter the program:

ˆP Z [RETURN C -]ˆE.

In the editor this reads:

```
SOLVE
PLOT
    ENTER
    cos
    Y-X
ENDPLOT
```

Then enter a guess into the X register and run the program with ˆR. The solution will be returned in the X register.

This facility also operates in complex mode. Note that the iterative procedure may not converge and may yield errors (INF or NAN). Convergence may depend on a reasonable guess for the first try. This is particularly true in complex mode.

SOLVE uses a modified form of Newton's method.

Two SOLVE instructions **in** a row (Z key twice) will result in the Calculator searching for all zeros in the range between the X and Y registers. Upon completion, the number of zeros found is in the Z register and the zeros are in the memory registers (perhaps including the other banks). This works only in real mode. Note that all the memory registers are set to zero upon the second SOLVE command, but they may be set up prior to the "[" start of the function and may be used in the function itself. There is no guarantee that all zeros will be found. Double zeros are particularly difficult for the range solve routine to locate.

About SANE

SANE stands for “Standard Apple Numerics Environment” and is what is used for all low level calculations. SANE has a quirk in that it has both +0 and -0. It also has + ∞ and -∞ denoted by INF and -INF. For example, calculating 1/0 will yield INF and not an error. Likewise, 1/(-0) yields -INF.

SANE also has a NAN which is a type of error. It means “not a number” and will occur when any illegal operation is done not yielding INF. For example SQR(-1) will give a NAN (real mode only). Also things like 0/0 and INF-INF will yield NAN. The NAN has an error code attached to it such as NAN(004). This indicates what kind of error caused the NAN, but it is mostly irrelevant for this calculator.

Math coprocessors

The FPE (floating point engine) card from:

Innovative Systems
P.O. Box 444
Severn, MD 21144-0444

is a math coprocessor which is highly recommended. It will speed up time consuming computations, such as those used in integration or plotting, by a factor of 3 to 45+.

The calculator uses the FPE card by direct access for the greatest speed. This can be defeated by one of the parameters (possibly necessary for early versions of the FPE). If the FPE init file is active, the card would still be used through SANE but with less speedup.

☼ Supplied programs . Description of operation

["FPE" flags those which the user may deem too slow for practical use without an FPE math coprocessor card.]

1. QUADRATIC

   Purpose: Solves $ax^2 + bx + \mathbf{c} = 0$.
   Input: a,b,c.
   On exit: roots in X, Y.

2. AVERAGE

   Purpose: Computes average of list of numbers.
   Input: data, end with —>
   On exit: average in X and in 3, sum in 1, #entries in 2.

3. FACTORIAL

   Purpose: Computes n! (or r(n+1) if n is not integral).
   Input: n.
   On exit: n! in X, n in 1.

4. BINOMIAL

   Purpose: Computes binomial coefficient (n:k).
   Input: n,k.
   On exit: n in Z, k in Y, (n:k) in X.

5. CASH.REG

   Purpose: Simulation of cash register.
   Input: data (dollars and cents but with no decimal point).
   While running: Total is in 1, #items in 2, last item in 9
   (Item sent to printer)
   End input with —>
   On exit: Z: total
   Y: 7% tax
   X: grand total (Sent to printer.)

6. POLAR

   Purpose: Converts cartesian to polar coordinates.
   Input: x,y
   On exit: T: r
   Z: *0*
   Y: y
   X: x

7. VARIANCE

Purpose: Computes average, variance, standard deviation of data.
Input: data x..., end with
While running: 1: $\sum x$, 2: $\sum x^2$, 3: #entries, Y: last entry
On exit:
1: average
2: variance
3: standard deviation
X: # of entries

8. CORRELATION

Purpose: Computes correlation coefficient of input data.
Input: x,y pairs, —i to end.
On exit:
1: $\sum xy$
2: $\sum x^2$
3: $\sum y^2$
X: correlation coefficient

9. LINEAR.REGRESSION

Purpose: Computes line of regression.
Input: *x,y* ..... pairs, —> to end.
On exit:
Z: coefficient of regression
Y: average of y' s
X: average of x' S
1: # of pairs x,y
2: $\sum x$
3: $\sum y$
4: $\sum x^2$
5: $\sum y^2$
6: $\sum xy$
7: covariance
8: variance of x's
9: standard deviation of x's

Regression line is (y-Y)=C(x-X) where C = coefficient of regression.

10. CHI.SQUARE

Purpose: Compute chi square of data.
Input: O,E...end with
On exit:
X: $x^2 = \sum\{90\text{-}E)^2/E$
9: #entries

11. DETERMINANT

Purpose: Compute determinant of 3x3 matrix.
Input: matrix entries in reading order.
On exit:
1-9: matrix entries
X: determinant

12.STIRLING

Purpose: Computes Stirling's approximation to $1og_{10}(n!)$.
Input: n.
On exit: Approximation to $1og_{10}(n!)$ in X, n! in Y (or INF).

13. ERROR

Purpose: Computes the error integral from 0 to X:
Input: x.
On exit: 1: x
X: error integral $\mathrm{erf}(x) = 2\sqrt{\pi}\int_0^x e^{-\tau 2}dt$

14. ELLIPTIC

Purpose: Computes the complete elliptic integrals:

$$K = \int_0^{\pi/2} (1 - K^2\sin^2 x)^{1/2} dx$$

$$E = \int_0^{\pi/2} (1 - K^2\sin^2 x)^{1/2} dx$$

Input: **k.**
On exit: **1: k**
V. E
X: K

15. RESIDUE

Purpose: Computes residue at 0 of input function (e.g., 1/X).
On exit: X: residue (NAN indicates an essential singularity).

16. CUBIC

Purpose: Computes roots of cubic polynomial: ax^3+bx^2+cx+d
Input: coefficients a, b, c, d.
On exit: 1: a
2: b
3: c
4: d
X, Y, Z: roots (complex)

17. AREA

Purpose: Computes area of polygon.
Input: x,y,... (coordinates of vertices of polygon).
End input with ->
On exit: Area in X.

18. FOURIER.ODD

Purpose: Computes odd Fourier coefficients of f(X) = X+|X|.

On exit: Register N (1-8) has N-th coefficient: $1/\pi\int_{-\pi}^{\pi} f(x)\sin(Nx)dx$

The answers are rounded to four decimal places.

19. FOURIER.EVEN

Purpose: Computes even Fourier coefficients of f(X) = X+|X|.

On exit: Register N (1-8) has N-th coefficient: $1/\pi\int_{-\pi}^{\pi} f(x)\cos(Nx)dx$

Register X has 0-t coefficient: $1/21/\pi\int_{-\pi}^{\pi} f(x)dx$

To use these with an arbitrary function, replace lines 2-4 with the desired function.

20 FOURIER.PLOT

Purpose: Demonstrate Fourier approximation.

Plots 8-term Fourier approximation of the discontinuous function which is x for x>0 and π for x<0. After keypress, plots in red the actual function.

21 FOURIER.PRINT

Purpose: Print (to printer) Fourier coefficients of f(X) = X+|X|.
End with ^E.

22 PLOT.DEMO.1

Purpose: Demonstrate plotting of real functions.

Plots $2(X^4 - x^2+X/3)$ and then, without erasing, sin(2πX).
After keypress, plots √|X|, then X^2, then |X|.

23. PLOT.DEMO.2

Purpose: Demonstrate polar plotting (using complex mode).

Plots the cardinid r=1-cos(*0*) by plotting the complex function (1-cos(X))exp(iX) for 0≤X≤2π. After keypress, plots an Archimedian spiral and then a logarithmic spiral.

24. PLOT.DEMO.3

Purpose: Demonstrate error handling for plots.
Plots the function 1/X.
After keypress, plots $1/(X^2-4)$.
After keypress, plots log(X).

25. PLOT.INTEGRAL (FPE)

Purpose: Demonstrate use of integral with a plot.

Plots values of E-1 where E is the complete elliptic integral (see above) against the parameter k running from 0 to 1. The vertical axis extends from 0 to .6. This program takes about an hour to run (with a TransWarp.GS) since it computes 640 definite integrals of a complicated function. With an FPE card it takes about 3 minutes.

26. DERIVATIVE.PLOT

Purpose: Plots the function sin(3x)-cos(5x) and its numerically computed derivative.

27. LISSAJOUS

Purpose: Demonstrate parametric plotting with Lissajous curves.

Plots these Lissajous curves:

x = sin(3t), y = sin(4t)
x = sin(3t), y = cos(5t)
x = sin(7t), y = cos(11t)
x = sin(13t), y = cos(17t) (in two parts)

28. LISSAJOUS.RAND (FPE)

Purpose: Plot random Lissajous curves, multicolored.

29. CYCLOID (FPE)

Purpose: Plot random cycloids with cycling colors. Screen clears at random intervals.

30. LIN.REG.PLOT

Purpose: Demonstrate use of datafile.

Open datafile from help menu before running. Program will prompt for the AD command to start reading the datafile. It then does a scatter plot of the data and, when the file is done, plots the linear regression line computed from the data.

31. COMPOUND

Purpose: Demonstrate the Compound function.
Input: Principal, then interest rate.
On exit: Value after N time periods in memory register N, after 10 periods in X.

32. ANNUITY

Purpose: Demonstrate the Annuity function.
Input: Principal, then interest rate.
On exit: Present value after N time periods in memory register N, after 10 periods in X. Future value after 10 periods in Y.

33. COMPOUND.PLOT

Purpose: Plots compound interest (for principal 1) for the interest rates 6%, 8% and 10% in a single graph.

34. ANNUITY.PLOT

Purpose: Plot present and future annuity values, with automatic scaling for any number of periods input by user.

35. PLOT.ANY

Purpose: Plot any real function input by user.

36. ANY.DERIVATIVE

Purpose: Plot a function input by user and then its numerically computed derivative.

37. PLOT.ANY.SURF

Purpose: Plot the graph (a surface) of any real function of X,Y input by user.
Input: Function, Radius, Tilt, Rotation, Separation, plot limits.

38. SOLVE.ANY

Purpose: Demonstrate the SOLVE feature.
Input: Function, limits of range to solve within.
On exit: Number of solutions in Z, solutions in memory registers.

39. SOLVE.ANY.CMPLX

Purpose: Demonstrate SOLVE for complex functions.
Input: Function and initial guess.
On exit: Solution in X (if none found end with ^E).

40. MOVING.LINE

Purpose: Demonstrate flexibility of complex plotting.

This is one of those moving line graphics demos. There are better ones, but this one is far from bad and shows how complex plotting can be used to do things far from the purpose of the plotting facility. It would be hard to find as good a program using only 190 bytes and 121 program lines. (Typing this in is like typing two lines of text.)

41. STARS

Purpose: Just a nice colorful graphics program.

Plots random starlike (sometimes) patterns in cycling colors. The screen is cleared at random intervals.

42-46. CONTOUR.1-5 (FPE)

Purpose: Demonstrate production of contour maps.

Again this shows how complex plotting can be used for things for which it was not designed. These programs are self-documenting. Because of the massive amounts of calculation necessary to do a contour map, these programs take from 15 to 21 minutes with an FPE card, but 45 minutes to over 5 hours without.

41-68. SURFACE.1-22

Purpose: Demonstrate surface plotting.

These are self-documenting, and, anyway, some of them are surprises which I don't want to ruin. They take from 10 seconds to 7 minutes with an FPE card, 3 to 40 times that, without. The first 4 of these correspond to the first 4 contour maps.

69-78. CX.SURFACE.1-10 (FPE on 1-4)

Purpose: Demonstrate surface plotting in complex mode.

Actually, Surface.4 is also in complex mode (because it is faster in that case - usually it is not). These are functions that are simple in complex mode, but would be difficult to do in real mode. (Several more of the real surfaces could have been done in complex mode, but are faster done in real mode.)

☼ Summary of available functions and commands

FUNCTION KEYS:

+	Plus
-	Minus
*	Times
/	Divide
S	Sin
C	Cos
T	Tan
HS	Sinh
HC	Cosh
HT	Tanh
AS	Arcs in
AC	Arccos
AT	Arctan
AHS	Argsinh
AHC	Argcosh
AHT	Argtanh
Q	Square root
E	Exp
L	Log (natural)
&	Absolute value Negate X (In complex mode only the active "part".)
^	YˆX
!	Reciprocal
I	Integer part
P	π
@	Annuity (X = number of periods, Y = rate)
~	Compound (X = number of periods, Y = rate)
;	Real part
'	Imaginary part
V	Angle (*0* in complex mode)
0	Exp(i·realpart) (cos(t) + i·sin(t))
Y 1-9	User function (RPN)
U 1-9	User formula (algebraic)

MEMORY AND STACK MANIPULATION KEYS:

RETURN	Enter
^X	Clear X register (same as Clear key)
^C	Clear all registers
↓	Rotate stack down
↑	Rotate stack up
X	Exchange X and V
R 1-9	Recall memory to X
M 1-9	Store X in memory
< 1-9	Decrement memory
> 1-9	Increment memory
^A 1-9	Add X to memory
^I	Recall of register 1-8 pointed to by register 9
^O	Store in register 1-8 pointed to by register 9
^B -#+	Rotate memory banks

MODE KEYS:

#	Decimal mode
$	Hex mode
%	Degree/Radian toggle
'	Fixed/Floating toggle
^Z	Complex mode toggle
J	Real/Imag entry toggle
D 0-9	Decimal places/significant digits in fix/fp, 0 = max

PRINTER INTERFACE KEYS:

^T 1-9	Tab
^N	Print X
|	Carriage return
^F	Format (followed by format string)

PROGRAMMING AND SPECIAL PURPOSE *KEYS:*

^P	Start programing
^E	End programming or abort running program
^W -#+	While
^L 1-9	Loop
\ -#+	If condition
^V	Else
^Q	End if
^R	Run program in memory
^S	Step program in memory
K 1-9	Subroutine definition start
G 1-9	Gosub
B	Return from subroutine
→	Skip past current loop
^D	Get input from datafile (if one is open)
ESC	Input request
(	Start of integral (Two in a row give a PRECISION INTEGRAL, precision in T.)
)	End of integral
[	Start of function to be plotted (Two in a row give a hidden line surface plot.)
]	End of function to be plotted
=	Transfers stack register parameters to surface plot routine (T=tilt, Z=color scheme {O-87}, Y=rotation, X=separation)
^Y 1-9	Screen functions (see PLOTTING)
^G 1-9	Plotting colors (see PLOTTING)
Z	Modifies following PLOT segment to a function to SOLVE (Two in a row give a SOLVE IN RANGE.)
@	Exp during input request
W	The character E in immediate numerical input
_	The character - in immediate numerical input
?	Go to help menu
F	Define and/or evaluate algebraic formula
^@ 1-9	Move defined algebraic formula to User Formula 1-9
"	Define text for labeling a plot (X,Y,Z = HTAB, VTAB, COLOR)

☼ Advanced functions quick reference

Integration (f(X) → **X**) **Precision integration** ((f(X) → **X**))

T: tolerance in precision mode		T: error estimate (prec. mode)
Z:		Z: tolerance (prec. mode)
Y: right end		Y
X: left end	→	X: integral = $\int_{\mathbf{x}}^{\mathbf{y}} f(x)\,dx$

Plotting functions [f(X) → **X**]

real	complex
I: top	T: center
Z: bottom	Z: scale
Y: right end	Y: finish
X: left end	X: start

Plotting surfaces

real [[f(X,Y) → X]] complex [[f(X) → X→ Re(X)]]

T: top
Z: bottom
Y: right end
X: left end

Setparms for surface plot **("="** command)

T: tilt	1: x limit
Z: color (0-87)	2: y limit
Y: rotation	3: radial limit
X: separation (1,2,4,8,16)	

Solve Z [f(X) → **X**]

X: initial estimate → **X:** solution

Range solve ZZ [f(X) → **X**]

Y: right end	→	Z: number of solutions
X: left end		(solutions in memory registers)

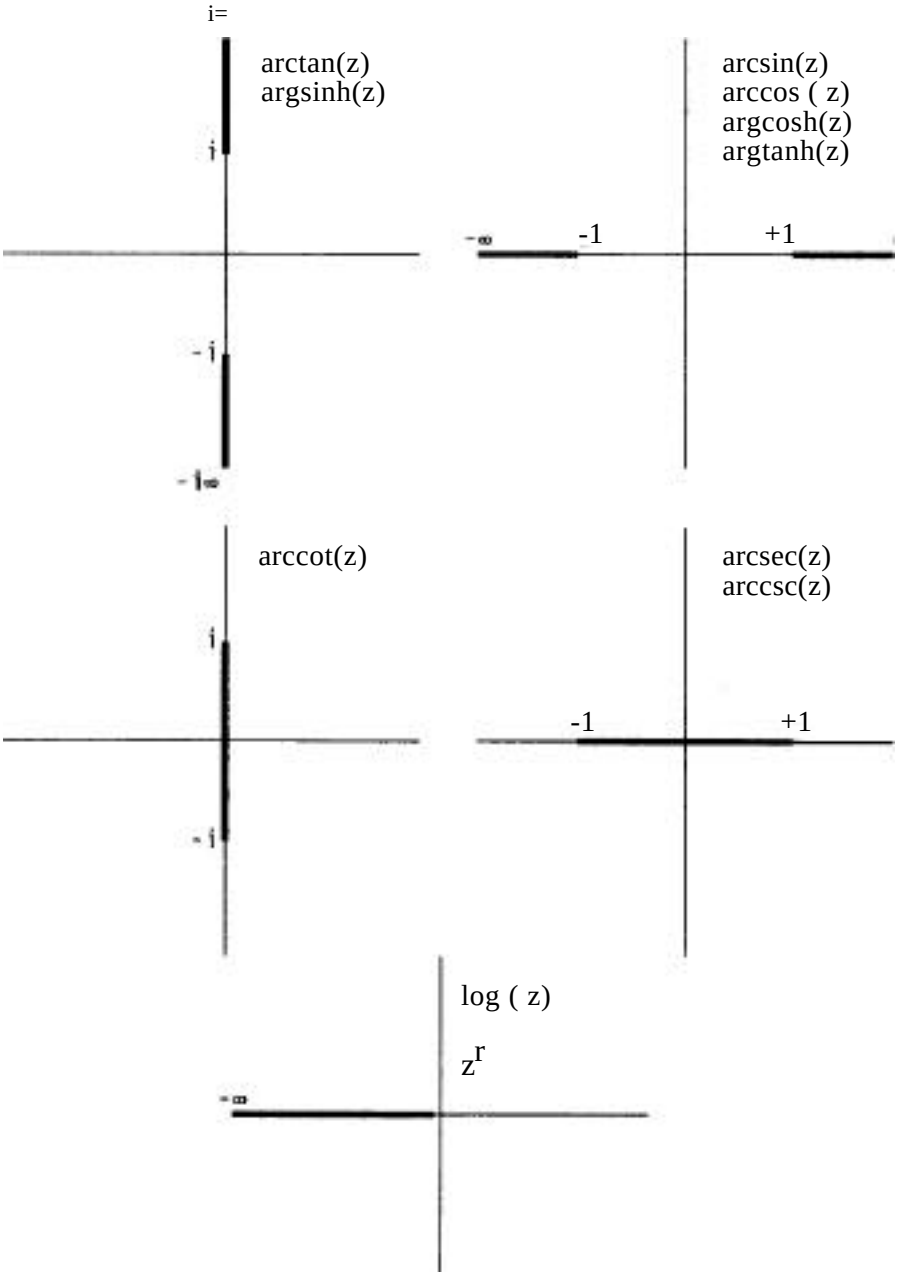


Figure 16. *Cut loci (discontinuity sets) of elementary complex functions.*

Part VIII. Calculator Tutorial

This tutorial explains in detail how to program the Calculator. It presumes you are already familiar with the RPN notation and how to use the calculator in direct mode. All programming examples are described as being written from the editor rather than from the calculator proper.

EXAMPLE 1. Write a program to calculate the distance a projectile will travel over flat ground given its initial velocity and the angle to the horizontal at the start. Ignore air resistance.

Procedure: Use the formula range = v2 SIN(2 angle)/32 where v is the initial velocity. Input the value of v from the user and then the angle. Calculate the range (in feet) and end.

Solution: Type "?", to go to the help menu and then “E” to go to the editor. Press “D <RTN>” to delete the program in memory, if any. Type “A <RTN>” to go into Add mode. Now type the program as follows:

(You type only the keys shown in the “Keys” column. The “ˆ”indicates a control key; do not type the character. The “<ESC>” and “<RTN>” stand for the ESC and RETURN keys.)

Keys	Editor displays	Comment
%	1 DEG<->RAD	Turn on degree mode
ˆX	2 CLEARX	Clear the X register
<ESC>	3 INPUT	Get velocity
<RTN>		Or optional prompt message
<RTN>	4 ENTER	Copy to Y
*	5 X*Y	Square it
RTN>	6 ENTER	Copy to Y, to prepare for:
ˆX	7 CLEARX	Clear X (this is just cosmetic)
<ESC>	8 INPUT	Input angle in degrees
RTN>		Or optional prompt message
<RTN>	9 ENTER	Copy it to V
+	10 X+Y	Twice the angle
S	11 SIN	SIN of 2.angle
*	12 X*Y	Times v^2 now in V
*		(Numbers show at top of screen
/	13 32	until non-numeric key is hit)
32	14 Y/X	Divide by 32 and end
		End programming

Now type Q < RTN> to go back to the help menu, and Q to return to the calculator proper. Press ˆR to run the program. At the data request, type the value for the velocity and, at the next one, give the angle. Terminate both inputs by either <RTN> or <ESC>. The program will end with the calculated range in the X register.

EXAMPLE 2. Write a program to calculate the surface areas and volumes of cylindrical cans, and to total these areas and volumes for several different can sizes.

Procedure: The program will request the input of the radius r and the height of each can from the user. It will calculate the volume and area and place these in memory registers 1 and 2. The accumulated total volumes and areas will be placed in memory registers 3 and 4.
Solution: Type "?" to go to the help menu and then "E" to go to the editor. Press "D <RTN>" to delete the program in memory, if any. Type "A <RTN>" to go into Add mode. Now type the program as follows:

	Editor displays	Comment
^C	1 CLEARALL	Clear all registers
^W+	2 WHILE +	Set up a while loop
^X	3 CLEARX	Clear X register for input
<ESC>	4 INPUT	Input r
<RTN>		Or optional prompt message
<RTN>	5 ENTER	Duplicate to V
<RTN>	6 ENTER	Duplicate to Y and Z
*	7 X*Y	Now r in Y, r^2 in X
<RTN>	8 ENTER	Stack reads r^2, r^2, r
<RTN>	9 ENTER	Now r^2, r^2, r^2, r
P	10 PI	Now πr^2, r^2, r
*	11 X*Y	Area πr^2 of base
M9	12 MEMORY 9	Save for later use
^X	13 CLEARX	Prepare to input h
<ESC>	14 INPUT	Request input of h
<RTN>		Or optional prompt message
<RTN>	15 ENTER	Duplicate h to V
^J	16 ROTDOWN	Rotate stack down
*	17 X*Y	Calculate hr^2
P	18 P1	Get n and move stack up
*	19 X*Y	Calculate πhr^2
Ml	20 MEMORY 1	Store volume in register 1
^A3	21 ADDTO 3	Accumulate volumes in reg 3
^J	22 ROTDOWN	Rotate r,h to X,Y
*	23 X*Y	Calculate rh
P	24 P1	Get π and move stack up
*	25 X*Y	Calculate πrh
<RTN>	26 ENTER	Duplicate to V
+	27 X+Y	$2\pi rh$ (area of sides)
R9	28 RECALL 9	Get area of base
<RTN>	29 ENTER	Duplicate to V
+	30 X+Y	Area of top + bottom
+	31 X+Y	+ area of sides
M2	32 MEMORY 2	Store total area
^ A4	33 ADDTO 4	Add to accumulated total area
^ L8	34 LOOP 8	Loop always (until a — key)
^E		End programming

Now type Q < RTN> to go back to the help menu, and Q to return to the calculator proper. Press R to run the program. At the data request, type the value for r and then <RTN> or <ESCAPE>. At the second data request type the value for h and then <RTN>. Repeat this for as many cans as you want and press ➡ when finished.

After the calculation for each can, memory register 1 shows the volume of the last can and register 2 shows the surface area. Register 3 shows the total volume and register 4 shows the total surface area.

Note that this program of 34 lines appears moderately long when listed, but, in fact, takes only 47 keystrokes to enter including the A <RTN> at the start and the E at the end. This program occupies only 45 bytes Out of the total available program memory space of 2048 bytes.

Let us now modify this program. Suppose we wish the user to input the number of cans to be processed by the program. We would do the following:

Press ? to go to the help menu and E to go to the editor. Type 134 <RTN> to insert above line 34. Then type as follows:

Keys	Editor displays	Comment
<8	34 LOWER 8	Decrease register 8 by 1
^E		Terminate the insertion

Now type 12 <RTN> to insert above line 2 and enter this:

<ESC>	2 INPUT	Get # of cans
<RTN>		Or optional prompt message
I	3 INT	Make sure it is an integer
&	4 ABS	and positive or zero
M8	S MEMORY 8	Put count in register 8
<8	6 LOWER 8	Decrement it
^E		Terminate the insertion

Now if you return to the calculator and run this program, it will go through the loop the number of times given by the initial data input. Note that, instead of the LOWER 8 at line 6, you could replace the WHILE+(while positive or zero) with a WHILE # (while nonzero).

For the remainder of this tutorial we shall assume that you are now familiar with the means of passing between the calculator proper and the editor, and will not repeat those instructions.

EXAMPLE 3. Write a program to plot the exponential function exp(x) and its tangent line y=x+l at(0,1).

Procedure: Set up the plot for the range -2 to 1 on the x-axls and 0 to 2 on the y-axis. Plot the EXP function and then (without erasing the screen) the function y x + 1

Solution: Enter this program from the editor:

Keys	Editor displays	Comment
2	1 2	Upper limit (shows at top till:)
<RTN>	2 ENTER	Move to Y register
^X	3 CLEARX	Lower limit 0
<RTN>	4 ENTER	Push stack up
1	5 1	Right limit (shows at top till:)
<RTN>	6 ENTER	Push stack up
_2	7 -2	Left limit (shows at top till:)
^Y2	8 SCRNFNC 2	Don't erase screen at plot end
^Y8	9 SCRNFNC 8	Don't stop for key at plot end
[	10PLOT	Start function to be plotted
E	11 EXP	Take EXP(X)
]	12ENDPLOT	End of function to be plotted
^E		Terminate programming for now

We now wish to put in the instruction to pause for a key at the end of the next plot. Suppose we have forgotten the keys to do this. This is why we ended the programming. Now go to the help menu by typing "Q <RTN>" and get the help information by typing "H". Press the space till we come to the page describing the "screen functions". There we see that the proper command is ^Y7. Press <ESC> to abort the help listings and then "E" to return to the editor. Press "A" to add to the program and then:

^Y7	13 SCRNFNC 7	Wait for key at plot end
^G2	14 COLOR 2	Plot derivative in red
[	15 PLOT	Start of function to be plotted
<RTN>	16 ENTER	Copy X to Y
1	17 1	Put 1 in X
+	18 X+Y	Calculate X+1 (value of function)
]	19 ENDPLOT	End of function X → X+1
^E		Terminate programming

Now return to the calculator proper and try it Out by pressing the ^R key. You might experiment with the operation of the other 'screen functions

REMARK. One is often tempted to hit RTN at the end of a number when entering it into a program. This puts an undesired (perhaps) ENTER into the program. You could just continue entering the program and delete these extra lines when finished.

EXAMPLE 4. Write a program to look for solutions of $3x^4-3x^3+x=0$.

Procedure. Write this polynomial in its "Homer form" (((3x±0)x-3)x-v 1)x+0, that is, ((3xx-3)x+ 1)x. Put it into a SOLVE segment.

Solution:

Keys	Editor displays	Comment
z	1 SOLVE	Set up a solve segment
]	2 PLOT	Start of function to be solved
<RTN>	3 ENTER	Copy x to all stack registers
<RTN>	4 ENTER	
<RTN>	5 ENTER	
*	6 X*Y	Square x
3	7 3	Coefficient 3
*	8 X*Y	$3x^2$
_3	9 —3	Coefficient -3
+	10 X+Y	$3x^2-3$
*	12 X*Y	$(3x^2-3)x$
1	12 1	Coefficient 1
+	13 X+Y	$(3x^2-3)x+1$
*	14 X*Y	End of function to be solved
]	15 ENDPLOT	
^E		Terminate programming

Now reenter the calculator proper, place an initial guess in the X-register and press ^R to execute the solve routine. Try various initial guesses.

Now let us return to the editor and modify this program to attempt to find all solutions between -2 and 1. Type I] <RTN> (in the editor) to insert at the start of the program and then:

1	1 1	Right hand of range
<RTN>	2 ENTER	Copy to Y
-2	3 -2	Left hand of range
Z	4 SOLVE	2nd solve to make a range solve
^E		Terminate insertion

Return to the calculator and just run this with ^R. When done, the calculator will have 4 in the Z-register indicating that 4 zeros were found. The zeros are in memory registers 1,2,3 and 4. If more than 9 zeros were found then they would be in the next memory bank and you could view them by typing ^B +.

EXAMPLE 5. Write a program to calculate the Bessel function

$$J_0(t) = 1/\pi \int_0^{\pi} \cos(t \sin x)\, dx$$

Procedure: Have the user input t and do the integration with a precision integral with tolerance .0001.

Solution:

Keys	Editor displays	Comment
^C	1 CLEARALL	Clear up garbage
<ESC>	2 INPUT	Get t from user
<RTN>		Or optional prompt message
Ml	3 MEMORY 1	Save in register 1
.0001	4 .0001	Set tolerance
<RTN>	5 ENTER	Push up stack
^X	6 CLEARX	Clear X
<RTN>	7 ENTER	Push stack up
<RTN>	8 ENTER	Now tolerance is in T
P	9 P1	Get π
X	10 X<->Y	π to Y, 0 to X
)	11 INTEGRAL	Start of fnc to be integrated
)	12 INTEGRAL	Precision integral
S	13 SIN	sin(x)
R1	14 RECALL 1	Get t
*	15 X*Y	t sin x
C	16 COS	cos(t sin x)
)	17 ENDINTL	End of fnc to be integrated
)	18 ENDINTL	just for nice listing
^K	19 ROTUP	Get error estimate
^K	20 ROTUP	Error to Y
P	21 PI	Get π
/	22 Y/X	Error/π
M9	23 MEMORY 9	Save it in register 9
P	24 PI	Get π, integral is in Y
/	25 Y/X	Divide integral by π
^E		Terminate programming

Now go to the calculator and press ^R to run this. Enter the desired value for t when the program asks for data. As always, terminate this input with <RTN> or <ESC>. $J_0(t)$ will be in the X register when the program finishes and the maximum error estimate will be in register 9. Register 1 holds t.

Exercise: Modify this program so that it loops forever, requesting new values of t until the user presses ➡.

EXAMPLE 6. Write a program to calculate the "sine integral"

$$Si(t) = \int_0^t \sin(x)/x \, dx$$

Procedure: Have the user input t and do the integration with the efficient 128 subdivision approximation. Since the expression sin(0)/0 is meaningless but sin(x)/x approaches 1 as x goes to 0, we will use a conditional to substitute I at that point. (The calculator would do the integration without this but the error would be much greater.)

Solution.

Keys	Editor displays	Comment
^C	1 CLEARALL	Clear up garbage
<ESC>	2 INPUT	Get t from user
<RTN>		Or optional prompt message
X	3 X<->Y	t to Y, 0 to X
(	4 INTEGRAL	Start of fnc to be integrated
\#	5 IF #	Do following if x not 0
<RTN>	6 ENTER	Copy X to Y
S	7 SIN	Calculate sin(x)
X	8 X<->Y	sin(x) to Y, x to X
/	9 Y/X	sin (x) /x
^V	10 ELSE	Do following if x was 0
1	11 1	Get number 1
^Q	12 ENDIF	End of conditional
)	13 ENDINTL	End of function to be integrated
^E		Terminate programming

Return to the calculator, press ^R to run the program, input desired value of t, Si(t) will be in the X-register on completion. This will be a very good approximation as long as t is not too large.

Exercise. Rewrite this with a precision integral. Use both versions for various values of t and compare the results.

EXAMPLE 7. Rewrite USER FUNCTIONS 4 and 5 to give conversions from the usual cartesian complex number x + iy form to polar form (r ± ie) and back again.

Procedure. Write a program to compute r = ABS(x + iy) and e ANGLE(x + iy) and combine into the complex number r+ je. Transfer this to USER function 4. Similarly compute x=r cos(e) and y=r sin(e), combine into x+iy and save this to USER *5.*

Solution.

Keys	Editor displays	Comment
^Z	1 REAL->CPX	The purpose of these two switches
^Z	2 REAL<->CPX	is to ensure real parts are "up"
<RTN>	3 ENTER ABS	Copy to Y
&	4 ABS	Compute r
X	5 X<->Y	r to V, original X back
V	6 ANGLE	Compute *0*
J	7 REAL<->IMAG	Switch to imag parts
1	8 1	Get i
*	9 x*Y	i*0*
J	10 REAL<->IMAG	Back to real parts
+	11 X+Y	r+i*0* (done)
^E		Terminate programming

Now type >4 to copy this program to user function 4. Next type "D," to delete this program and enter the next with "A <RTN>"

^Z	1 REAL<->CPX	Make sure real parts are up
^Z	2 REAL<->CPX	
<RTN>	3 ENTER	Copy to Y
;	4 REALPART	Get r
X	5 X<->Y	r to Y, orig r+i*0* to X
,	6 IMAGPART	Get *0*
0	7 EXP(I*REALPART)	Compute cos(*0*)+is in(*0*)
*	8 X*Y	Multiply by r
^E		Terminate programming

Now type >*5* to copy this program to user function 5. Return to the calculator to test them out: Switch to complex mode with "^Z". Type lJ1 to give the complex number 1+i. Type Y4 to call USER 4 which should give $\sqrt{2}+i\pi/4$. Type Y5 to convert back. There is a very small rounding error.

If you want to make these user functions permanent, you must use the Help Menu "U" command.

Exercise: Write USER functions to do degree ⇔ radian conversions.

EXAMPLE 8. Write a program to calculate successive Fibonacci numbers from seed values in Y (1st seed) and X (2nd seed) at start.

Procedure: Convert Y,X to X,X+Y. After each calculation have the user do a dummy input in order to see the result of each calculation.

Solution:

Keys	Editor displays	Comment
<RTN>	1 ENTER	Save X,Y on stack
^X	2 CLEARX	Set to clear 9
M9	3 MEMORY 9	Set loop register to 0
^J	4 ROTDOWN	Get X,Y again
^W+	5 WHILE +	Set up loop
<RTN>	6 ENTER	Copy X to Y, bump V to Z
^J	7 ROTDOWN	Stack now X,Y,?,X
+	8 X+Y	Now X+Y,?,X,X
^K	9 ROTUP	X,X+Y,?,X
X	10 X<->Y	X+Y,X,?,X
<RTN>	11 ENTER	Protect from input
<ESC>	12 INPUT	Just to pause program
<RTN>		Or optional prompt message
^L9	13 ROTDOWN	Retrieve X+Y,X
^E	14 LOOP 9	Loop forever (until —+ or ^E)
		Terminate programming

Now go to the calculator, set up desired seeds in X, Y and run the program with ^R. Just press return at each DATA request. The X register holds the successive Fibonacci numbers.

Exercise. Rewrite the routine to compute generalized Fibonacci numbers, where seeds Y,X are converted to X,aX+bY, where a and b are constants held (On entry) in memory registers 1 and 2.

EXAMPLE 9. Write a program to plot a function and its derivative. Use a numerical algorithm to approximate the derivative.

Procedure. Since the function will be calculated several times, it will be placed in a subroutine. Use the second order central difference formula

$$(f(x-2h)-8f(x-h) \pm 8f(x + h)-f(x+2h))/12h$$

as the derivative approximation. Write the routine using the sin function, easily replaceable by any other function.

Solution: First write the subroutine to calculate f(x):

Keys	Editor displays	Comment
K1	1 SUBROUTINE 1	Calculate f(x)
S	2 SIN	
B	3 RETURN	End of subroutine

Next we write a subroutine to calculate the term f(x-2h)-8f(x-h) in the difference formula. It is entered with h in Y and x in X.

K2	4	SUBROUTINE 2	f(x-2h)-8f(x-h)
Ml	5	MEMORY 1	Save x
X	6		
M2	7	MEMORY 2	Save h
<RTN>	8	ENTER	Duplicate h
+	9	X+Y	2h
-	10	Y-X	x-2h
G1	11	GOSUB 1	f(x-2h)
M3	12	MEMORY 3	Save it in 3
R1	13	RECALL 1	Get x
R2	14	RECALL 2	Get h
-	15	Y-X	x-h
G1	16	GOSUB 1	f(x-h)
<RTN>	17	ENTER	
8	18	8	
*	19	Y*x	8f(x-h)
R3	20	RECALL 3	Get f(x-2h)
-	21	Y-X	
N	22	NEGATE	f(x-2h)-8f(x-h)
B	23	RETURN	

Next, the subroutine to calculate the full difference formula, entered with h in Y and x in X:

Keys	Editor displays		Comment
K3	24	SUBROUTINE 3	Est. deriv. x in X, h in Y
G2	25	GOSUB 2	f(x-2h) -8f(x-h)
M4	26	MEMORY 4	Save in 4
R2	27	RECALL 2	Get h
N	28	NEGATE	-h
R1	29	RECALL 1	Get x
G2	30	GOSUB 2	f(x+2h)-8f(x+h)
R4	31	RECALL 4	
-	32	Y-X	f(x+2h)-8f(x+h)+8f(x-h) -f(x-2h)
R2	33	RECALL 2	Get -h
/	34	Y/X	Divide difference by -h
12	35	12	
/	36	Y/X	Divide by 12
B	37	RETURN	

Finally the main program, entered with X,Y,Z,T holding the left end a, right end b, bottom, top as in a plot. Save a,b and use h = (b-a)/640 for the difference formula (this seems as good as any choice for h). Use a solid plot for f(x) and a dotted plot for the numerical derivative:

^Y6	38	SCRNFNC 6	Solid plot
^Y8	39	SCRNFNC 8	No key wait
M5	40	MEMORY 5	Save left end a
X	41	X<->Y	
M6	42	MEMORY 6	Save right end b
-	43	Y-X	
640	44	640	
/	45	Y/X	
N	46	NEGATE	h=(b-a)/640
M7	47	MEMORY 7	Save h in 7
R6	48	RECALL 6	Retrieve a
R5	49	RECALL 5	Stack now as at start
[	50	PLOT	
G1	51	GOSUB 1	Plot f(x)
]	52	ENDPLOT	Returns stack to original state
^Y9	53	SCRNFNC 9	Set defaults
^ Y2	54	SCRNFNC 2	Don't clear screen
[	55	PLOT	
X	56	X<->Y	Save x
R7	57	RECALL 7	Get h
X	58	X<->Y	h to Y, x to X
G3	59	GOSUB 3	Calculate f'(x) estimate
]	60	ENDPLOT	
			Terminate programming

Now go to the calculator proper, place -2π, 2π-, -1, 1 in the stack by typing 1 <RTN> N P 2 * <RTN> N. Then run the program with ^R. Try other ranges and other functions.

Appendices and Index

☼ Updating ProSel-16

Updates of ProSel-16 will be $10 if ordered from me, $25 if you want printed documentation. You must return your original disk with your update order. Updates will also be available (free) on the information services: CompuServe, GEnie, and America Online. In the latter case, the files will he encrypted and the update procedure will require your original disk (not a copy). Thus, keep your original in a safe place. Prices are subject to change without notice.

☼ Summary of colors and embedded control character toggles

This information is not needed if you use the design mode of the editor exclusively.

Color #0 = background (rusty brown default)
Color #1 = standard text color (yellow default)
Color #2 = highlight #1 (red default and anything in inverse on the normal text screen)
Color #3 = highlight #2 (cyan default)
Color #4 = dither of colors 1 & 2
Color #5 = dither of colors 1 & 3
Color #6 = dither of colors 2 & 3 (= cursor color)
Color #7 = dither of color 1 & background
Color #8 = dither of color 2 & background
Color #9 = dither of color 3 & background

ˆS toggles Color 2
ˆD toggles Color 3
ˆP toggles Color 4
ˆQ toggles Color 5
ˆR toggles Color 6
ˆPˆE toggles Color 7
ˆQˆE toggles Color 8
ˆRˆE toggles Color 9
ˆT toggles inversion with background color

Any of the toggles EXCEPT ˆT, when toggling off defeat all of the toggles including ˆT. (Thus use ˆT only after another toggle to do an inversion.) Dithered characters often look better inverted. Also, as with standard output controls, ˆO sets inverse (an inverted Color #2) and ˆN turns it off.

☼ Mix ‘n Match

This little program in the UTIL directory of the ProSel- 16 disk lets you make a custom version of ProSel- 16 retaining only some of its modules. This is intended for making Cut down versions for use on *3.5* inch disks and similar crowded environments. It generally has no use on hard disks since it will not speed up anything. To use it:

1. Set up a /*RAM5*/ volume with sufficient room. 800 free blocks will be more than sufficient.

2. Copy an UNZIPPED copy of the main ProSel-16 file START to /RAM5/.

3. Execute the Mix.N.Match program from anywhere; it does not have to be on /RAM5/.

4. You will be given the opportunity to keep or discard any selection from the fifteen modules:

Cycle	(X function)
Utilities	
Optimizer	
FindFile	
NDA	(The NDA screen, * function)
Backup	
He1p	
Zap	
Editor	(The ProSel editor, not the Text Editor)
Calculator	
MrFixit	(Volume Repair)
Appoint	(The appointment calendar)
InfoDesk	
TextEd	(The Text Editor)
Config	(Modify parms)

You just answer Y (keep) or N (discard) to each of these as they appear on the screen.

The cut down version will be written to /RAM5/ as “Custom.Start”. This can be as small as 60 blocks or as large as the original file. All versions will have full launching and shell facilities including the screen blanker with slide show and/or kinetic art.

It might seem that Config and Editor would be required, but this is not the case - you could set up the parms with the full version and transfer them to the desired disk and have no further need for changing the parms. Similarly for the Editor.

In all, this little program allows creation of 32,768 different versions of ProSel- 16. It will work with all (unzipped) versions of ProSel-16, past, present and future. You can Zip the final file if you wish, and copy **it** to the desired volume.

Quick reference

When selecting from the screen:

Main keyboard number key X:	accesses device X.
Keypad number key X:	accesses prefix X.
Alpha key:	moves cursor to item starting with key.
Apple-alpha key:	goes to that item on main menu.
CLEAR:	goes to command line processor.
ESC:	goes to main menu.

When executing a program:

Control down:	defeats “startup specification.
Shift down:	shift to ProSel-8 if SYS & no startup.
Shift & Control down:	shift to ProSel-8 if SYS file.
Apple key down:	purge memory.
Option key down:	purge mem & force reload of ProSel-16.
Apple key & Option down:	purge, force reload, don’t return.

When booting:

Control down:	defeat “boot program” in parameters.
Shift down:	force running boot program (defeat day check).

Line edit keys:

^D	deletes character under the cursor.
DELETE	deletes the character to the left of the cursor.
^E	toggles insert mode.
Apple-key	inserts character whether or not in insert mode.
^O	accepts next keypress literally where meaningful.
TAB	moves just past next / , or goes to line end adding /.
Apple-TAB	moves back to right of previous / or to line start.
^N	moves to line end.
^B	moves to line beginning.
^Y	clears line from cursor forward.
CLEAR	clears line cancelling all previous input.
ESC	aborts.

☼ ProDOS error codes

MLI (Machine Language Interface) ERRORS, • = *common error.*

	$01	Invalid call number
	$04	Invalid parameter count
	$05	Call pointer out of bounds
	$07	ProDOS is busy
	$10	Device not found
	$11	Invalid device request
	$20	Invalid request
	$21	Invalid control/status code
	$22	Bad call parameter
	$23	Character device not open
	$24	Character device already open
	$25	Interrupt table full
	$26	Resources not available
•	$27	I/O error (usually bad block on disk)
•	$28	No device connected
	$29	Driver is busy
•	$2B	Write protected
	$2C	Invalid byte count
	$2D	Invalid block address
•	$2E	Disk switched
•	$2F	Device not online (usually no disk in drive)
	$30-$3F	Device specific errors
•	$40	Pathname/device name syntax error
	$41	Memory manager error
	$42	Out of file buffers
	$43	Invalid file reference number
•	$44	Path not found
•	$45	Volume not found
•	$46	File not found
	$47	Duplicate pathname
•	$48	Volume full
•	$49	Volume directory full (already has 51 files)
	$4A	Incompatible file format
	$4B	Unsupported storage type (usually damaged file)
	$4C	End of file encountered
	$4D	Position out of range
•	$4E	Access not allowed (usually file locked)
	$4F	Buffer too small
	$50	File open
	$51	Directory structure damaged
	$52	Unsupported volume type (usually damaged volume directory)
	$53	Parameter out of range
	$54	Out of memory
	$55	Volume control block full
	$56	Buffer area in use
	$57	Duplicate volume name
•	$58	Not a block device (invalid device number)

$59	Invalid level
$5A	Block number out of range (usually damaged file)
$5B	Illegal pathname change (invalid RENAME command)
$5C	Not an executable file
$5D	File system not available
$5E	Cannot remove /RAM
$5F	Return stack overflow
$60	Data unavailable
$61	End of directory reached
$62	Invalid FST call class
$63	Resource not found
$64	Invalid FST operation
$67	Duplicate device name
$70	Cannot expand file, resource fork exists
$71	Cannot add resource fork

FATAL SYSTEM ERRORS

$01	Unclaimed interrupt
$0A	Volume Control Block unusable
$0B	File Control Block unusable
$0C	Block zero allocated illegally
$0D	Interrupted while I/O shadowing is off
$11	Wrong operating system version
$15	Segment loader error $24
$17-$24	Can’t load package
$25	Out of memory
$26	Segment loader error
$27	File map destroyed
$28	Stack overflow

SYSTEM LOADER ERRORS

$1101	Entry not found
$1102	Incompatible OMF version
$1103	Pathname error
$1104	File is not a load file
$1105	Loader is busy
$1107	File version error
$1108	User ID error
$1109	SegNum out of sequence
$110A	Illegal load record found (usually damaged file)
$110B	Load segment is foreign

SELECTED TOOL ERRORS

$0201	Unable to allocate memory block
$0305	Damaged heartbeat queue detected (fatal)
$0308	Damaged heartbeat queue detected (fatal)
$0681	Event queue damaged (fatal)
$0682	Queue handle damaged (fatal)
$08FF	Unclaimed sound interrupt (fatal)
$0911	Desktop Bus Tool cannot synchronize with system (fatal)

Number key icons

List of icons, indicating filetype, displayed in a list of tiles obtained upon pressing a number key at the main ProSel- 16 screen:

 = DIR (subdirectory)

 = S16 (16 bit program)

 = SYS (8 bit program)

 = EXE (shell executable file)

] = BAS (BASIC program)

 = BIN (Binary file, perhaps a program under BASIC)

Printer commands

.LM# sets the left margin to #, defaulting to 0.
.TM# sets the top margin to #, defaulting to 0.
.PN# sets the starting page number, defaulting to 1.
.PL# sets the lines per page to #, defaulting to value in parms.
.LI# sets the line spacing to # (0 = single space, 1 = double, etc.).
.TL#//header// sets a page header (the first "//" is required).
.UL#<char> underlines the header with line of (char> repeated # times.
.BL#//footer// sets a page footer.
.0L#<char> overlines the footer with line of <char> repeated # times.
.RM# sets the number of printer columns to #, defaulting to 0 in which case columns will not be counted. If not zero, word wrap is attempted upon line overflow.
.MO# sets the printer mode to #, defaulting to 0.
.LE<leader> sets a leader, which must come right after the LE.
.TR<trailer> sets a trailer, which must come right after the TR.
.DE<char> sets <char> as the alias delimiter (default "!").
.FF# does a form feed if #=0, or if # is more than the number of lines left.
.LK "filename" links <filename> during printing. Should be only at end of file.
.AL "filename" loads <filename> alias file, overriding the default.
.WI<char> sets <char> as the alias wild card separator, default ",".
.SU suppresses header and/or footer on next page.

INDEX