```
**********************************
*                                *
*        Michael's Shell         *
*          Version 1.0           *
*                                *
*          A.K.A. SHELL          *
*                                *
*     Courtesy of the author,    *
*         Michael Pender         *
*         Copyright 1991,        *
*      all rights reserved.      *
*                                *
**********************************
```

ABSTRACT

Prodos BASIC allows a user to attach external prodos commands to the BASIC interpreter. However, doing so requires an advanced knowledge of assembler language and the Prodos BASIC environment. Furthermore requiring several commands to exist memory resident deprives the apple of its already rather limited programming space.

Shell programs help to make programming easier by allowing the user to customize their environment. This shell program allows the user to link their programs to the BASIC environment and execute them as if they were external commands, but they need not be memory resident. In fact, all of the details required to make a command execute automatically are handled invisibly by the shell program itself.

Also included in this package is the multiprogramming routine known as Daemon. Daemon taps mouse card interrupts to allow the coexistence of background tasks, invisible to the BASIC program executing in the foreground. Complete source code for Daemon is also included.

NOTICE OF RIGHTS

This program is shareware, it costs $30. For your $30 you get the newest version on disk, written, bound documentation, source code, a free upgrade, and a list of other shareware products I make. Special licensing must be acquired for companies which wish to reproduce this work in any form.

I believe in the concept of Shareware, that software should be cheap, and that one should be able to try a piece of software before shelling over your money. Individuals should feel free to copy this program for their own use. If after having this program for ten days you still find yourself using it send me the $30, maybe you'll like the newest version even better. If you know me at all, you know that I tend to upgrade programs almost constantly, and I always try to incorporate improvements suggested.
If you decide you don't like it, give it to a friend. I'm far from rich, and this particular project took me over two years and some investment to complete. I write this stuff as a hobby, and I don't expect to make any real money at it, but you could always surprise me. :^)

I can also be reached to answer questions.

Michael Pender
1480 Mapleton Avenue
Suffield, CT, 06078
(203) 668-0147

Disclaimer:

I have no association with the Laser series of computers (other than owning one), and no relation to Apple computer. Products mentioned such as DOS 3.3 and Prodos are the trademarks/copyrighted products of Apple Computer Co.

Having purchased this program you are entitled to use and copy this disk and its programs as you see fit for your personal use. There are no royalties for programs developed with this product, but if you include code from this system in your programs please include a note in the copyrights of the software to indicate the source. Further, as this program is shareware you should feel encouraged to distribute copies.

If you enjoy this program and are not a registered owner, please support my efforts by becoming a registered owner by sending $30 for the disks and
documentation, as well as a free update and status reports.

TABLE OF CONTENTS

CHAPTER 1 - INTRODUCTION

I started this program in 1988 ago. It was originally named Disk Commander, when I tried to implement it under DOS 3.3. Now I just call it SHELL (I was going to call it C.SHELL but decided to put it to rest). I had a lot of ideas that didn't work out (What do you mean I can't call DOSCMD with a VERIFY request?), and found a few errors in the Prodos 8 Technical Reference Guide the hard way, but all in all it seems to work, and it makes programming under Prodos BASIC a little easier to handle. At any rate it makes one slightly less likely to tear their hair out because of a dumb mistake.

Shell actually configures itself as an external command for Prodos BASIC. It intercepts commands after Prodos is done with them, but before BASIC gets a crack at them. It intercepts the external command vector, and saves the pointer to the next external command, but for reasons discussed later you will see why it is not advisable to load other external commands before SHELL, as SHELL may interfere with their operation. The theory is simple, if the command makes it to SHELL, it is either a command for SHELL, a BASIC command, or someone goofed typing. SHELL goes to its default directory, looks for a command of the name typed, and
if a file of that name is found it is loaded and executed using the Prodos smart run (dash) command.

This program is effectively a new programming environment. The nature of the command structure is extensible, making the setup very easy to customize to one's tastes. My younger sister found it amusing to create commands to respond to swearing.

The program serves three main functions:

First, the extensible shell allows for a person to create their own commands to simplify tasks under BASIC. The shell also allows a means of support for custom drivers for different computers. For my computer (a Laser 128ex) I wrote a custom clock routine (too cheap/poor to buy one) that taps mouse interrupts. Non-ROM devices are not only possible under Prodos, but may now be made convenient, custom drivers may be

automatically loaded on startup. I wrote a routine to intercept the printer output for slot #1 and drive my electronic typewriter (and the manufacturer said its not Apple-compatible :^).

Second, this program helps to relieve some of the possibility for errors. The GO command for example makes a backup of your program in case you need to revert back. Ever accidentally delete a line in a program and can't figure out what it was? The recall command can usually repair a program accidentally NEWed out of existance. The NEW command itself was altered to allow one to back out. And even the BYE command can be recovered from using BOOTSLOT's zero function. These commands are not an absolute defense, nothing is, but they can prevent some minor catastrophes that I've encountered before.

Third, this allows one to create a customized user interface. The shell allows for nearly English command recognition. In future versions this shall be improved upon. The control-key replacements make some tasks easier. Also various editor functions have been added, giving a purpose to the TAB and DEL keys on your apple's keyboard.

CHAPTER 2 - GETTING STARTED

If you bought the disk you're all set, the program will boot up, and install itself. This is fine if all you want to do is copy the program disk and store BASIC programs on the same disk. However many people will be transferring this to another disk (/ram is a good choice). This has its benefits and drawbacks. Selecting a different disk as home for your commands has been made rather easy to do. Just set the prefix to the directory you want, using the PREFIX command, and -/SHELL/SHELL (assuming its still on the WOS disk) to invoke the shell. SHELL will read the current prefix and store this as the directory in which to ALWAYS find its commands. You can change the command directory prefix using the CPREFIX shell command. This is done because in ordinary programming you may change your current prefix several times. Some of the commands do this themselves. If the prefix is set, SHELL always knows where to look.

Remember, YOU MUST SET THE PREFIX BEFORE INVOKING SHELL!!!!

Once SHELL is invoked it loads in at $4000, and checks if it is already present. If not it relocates to the absolute address $9600. Yes, I said ABSOLUTE ADDRESS. Programmers beware!! This was necessary both to save me work, because I couldn't figure out how the bitmap worked, and because the nature of the program requires it. SHELL must be invoked before other external commands you may choose to use. For example, if you had the very popular utility DOGPAW on disk, and you loaded it before SHELL, depending on your use of directories you might end up just reloading the program every time you issued the DOGPAW command. SHELL would go to disk, looking for a file called DOGPAW. On finding the file, shell would load it. DOGPAW would see that it was already present in memory and would return to BASIC. Hence, DOGPAW wouldn't work properly.

Also the absolute address is a convenience to programmers who wish to write routines compatible with SHELL, not only because you know where it is, but when your routine is invoked SHELL made a copy of the input buffer, telling Prodos not to parse the parameters, but storing the command library prefix in a buffer at $9700, the command as it appeared in the input buffer at $9800, and the

prefix followed by the name of the routine at $9900. The library prefix is stored as length, prefix (high bit clear), the input buffer is an exact replica, and the buffer at $9900 is the string length, prefix followed by the routine name. This allows commands to have parameters that BASIC.SYSTEM would not recognize and would generate an error for.

However, the absolute address requirement of SHELL means that SHELL must be loaded before other external commands. To then load other external commands, treat them normally, as if SHELL was not present. That is, if one wanted to have SHELL, DAEMON, and DOGPAW in memory simultaneously:

```
]-shell
]-daemon
]-dogpaw
```

Every command you type after SHELL is installed that is not a normal or extended command under BASIC.SYSTEM will be looked for on disk in the volume specified. This includes basic commands like NEW or LIST. So for speed, it is advisable to make your default prefix on a ramdisk, also because you'll know exactly what commands are there: copy only the ones you use.

Any command you create and place in the COMMANDS directory then becomes part of your available disk-based command set, you need not turn the power off or reboot or any such thing. BE WARNED!!! Fooling with some of the BASIC commands can have side effects. If you make a BASIC program called LIST and then go to list a different program you just lost it when the BASIC program named LIST was loaded into memory. You could either make an EXEC file called LIST that calls the BASIC code as I did with NEW, or you could use LIST which will be interpreted as a BASIC command, not a disk command. The second method is by far more dangerous, since you only need forget once.

CHAPTER 3 - INCLUDED COMMANDS

These commands are invoked by just typing their name after the SHELL program has been loaded into memory and executed.

3.1 BACKUP

Certain commands, such as the GO command make a copy of the program in memory before continuing. The program is saved on disk as a file called BKP. This is because some of the commands are written partly in BASIC, and it just wouldn't do to accidentally wipe out hours of work by typing a command you weren't familiar with. Hence the purpose of the BACKUP command. Whenever you make a major step forward in a program use the GO command to run it instead. Then should you make a major error and accidentally wipe out part of your program, merely typing BACKUP will revert to the last version saved.

See also GO, NEW, and RECALL.

3.2 BOOTSLOT

This is actually a program I wrote independently before SHELL was yet completed. BOOTSLOT is a SYS file, making it easy to run from many program selectors, including the BASIC.SYSTEM quit code.
BOOTSLOT was originally written to allow people with selector programs to chose a slot with a non-Prodos volume to boot. That is, if a person has a Prodos formatted 3.5 inch drive in slot 7, which comes up automatically when they turn on the power, and they want to boot from a device in slot 6 (say a 5 1/4 inch disk with a game in DOS 3.3 on it), all they need do is select the BOOTSLOT program, and then press the number key for the appropriate slot they wish to boot from, in this case, the '6' key.

BOOTSLOT has another important feature however. If you're working in BASIC and you use the BYE command to leave by accident, normally your program would be lost without hope. But the 0 option allows a person to exit back to the last system program that was running (sort of). Unfortunately to have this feature work the program itself had to be set up as a system program, so it must load in at $2000, possible disturbing other system programs. In general, don't expect the 0 option to allow you to return to your favorite word processor or modem program, but it will

allow you to return to BASIC quite nicely, where your program should be fine.

*** One warning, if your program is so long that it extends into the the Hi-res page one area, part of the program may be destroyed when BOOTSLOT itself loads in.
This only applies to programs longer than 6144 bytes (roughly 13 blocks on disk or bigger).

3.3 C80

This is a very simple command provided for demonstration purposes that effects a PR#3 command, followed by the CATALOG command.

3.4 CLOCK

This short routine (don't confuse short with simple, this routine was a pain in the foot to write) places a little clock in the upper left corner of the screen. Being interrupt driven, the routine is easily forced off by an SEI instruction, or if one presses RESET. To reconnect at any time, just use the DON command to reconnect DAEMON. As it is currently configured, this command shows the current time stored in the prodos standard locations. This routine was designed to remind people (myself included) when they've been spending too much time at the box and they could use a good stiff cup of tea or some sleep. This routine calls DAEMON to perform the interrupt processing, so DAEMON must be loaded prior to calling CLOCK.

See also TIMER, DON, DOFF, DREM.

3.5 CPREFIX

This command allows one to change the command library prefix after the SHELL program has already been installed. Normally SHELL selects the prefix directory active when it is first installed, but CPREFIX allows one to change it on the fly. If you copied all the commands to a ramdisk say, and wanted to change the prefix so SHELL would look there instead, or if you goofed, and hadn't set the prefix before running SHELL, this allows one to change the prefix to something new.

The new volume/directory selected need not be online at the time CPREFIX is used.

3.6 DATE

This routine prompts the user for the date and places it in Prodos-compatible form for stamping files

See also TIME.

3.7 DOFF

This routine TEMPORARILY deactivates DAEMON. It does not remove the interrupt handler, and does not free the memory DAEMON occupies.

*** Warning, disconnect DAEMON before entering another system program. Use the DREM command to permanently
remove DAEMON from memory before transferring control to another system program.

3.8 DON

This routine activates DAEMON. When DAEMON is first installed it does not activate interrupts. This was done to prevent confusing programs while a system might not be fully configured. This routine may also be called to reconnect DAEMON after reset is pressed, or it is stunned.

*** Calling DON when DAEMON is not present, or after DAEMON has been removed may have unpredicted results.

*** Warning, disconnect DAEMON before entering another system program. Use the DREM command to permanently remove DAEMON from memory before transferring control to another system program.

3.9 DREM

Daemon MUST be disconnected before entering another system program. Use the DREM command to permanently disconnect DAEMON, from memory, and from Prodos's interrupt handler.

3.10 ECHO

This command issues a PR#1 command to the system, turning on the printer for most Apples with printers, then it issues the Ctrl-I I SSC command, enabling the output of text to the screen and printer simultaneously. This command too is provided for as a demonstration.

3.11 EJECT5/EJECT7

These commands allow apple users without a convenient means of ejecting a disk in a 3.5 inch apple drive to eject the disk easily. These commands are no longer provided with the system, since some drives may be scrambled by this command. I don't want someone accidentally damaging their disk.

3.12 FLIST

This routine will take the BASIC program currently in memory, open a file named LIST, and then type the listing of the program to the file named LIST.

3.13 GO

The GO command is much like issuing the run command from BASIC, except that it also makes a copy of the current program in memory, and saves it on disk as
a file named BKP. This is useful because it is no more inconvenient than typing RUN (hell its even one character shorter), but you have a current backup of your file in case all hell were to break lose.

See also the BACKUP command.

3.14 NEW

The NEW command is activated like the others, when you type the word NEW in immediate mode. Many was the time I accidentally wiped a program out of this world with the NEW command under Applesoft. This command is very similar, it just prompts you to make sure you know what you're doing before it continues to wipe your program out of existance.

See also GO, BACKUP, and RECALL.

3.15 RECALL

This is a very short machine language

program I wrote a few years ago to bring an Applesoft program back from the dead under DOS 3.3. It works just as well under Prodos. Supposing you've neglected to avail yourself of the GO command recently, and the back-out now option of NEW, and went ahead and wiped your basic program out anyway. RECALL will rebuild the Applesoft pointers, allowing you to bring that program back from the dead. Using the GO and BACKUP commands is preferable, because RECALL will rebuild the last Applesoft program in memory, which just might be the BASIC part of the NEW command itself. But if you're in a bind, or you didn't happen to use my modified NEW command, you're probably in luck.

(Just for safety learn how this works on an unimportant file first, huh?)

See also GO, BACKUP, and NEW.

3.16 SHORT

This command redefines the output of a few of the control keys, the TAB key and the DEL key. On a Laser the selection of keys should be obvious, as for the most part they correspond to the function keys atop the keyboard.

This program reroutes the input vector from slot #1. Since most people have their printer card in this slot, it should not be a problem. To then access the driver a person need only type IN#1 to connect to it at any time, or IN#0 to disconnect it. The program itself relocates the necessary data and protects itself by lowering HIMEM. Therefore it is always available after being loaded. The loader itself checks to make sure it is not already present in memory before loading it in again.

Laser Key	Apple Key	Function
TAB	Ctrl-I	

Like pressing the forward arrow key five times. Makes editing much quicker.

DEL	ASCII 127	

Backspaces once, types space, and backspaces again.

F1	Ctrl-@	

Enters monitor by typing CALL-151 (return).

F2	Ctrl-A	

Leaves monitor by typing 3D0G (return).

F3 Ctrl-B
Types CAT (return) for a 40 column catalog.

F5 Ctrl-D Types TEXT:LIST (return)

F6 Ctrl-E Types POKE 33,33 (return)
F7 Ctrl-F Types LOAD (no return)
F8 Ctrl-G Types SAVE (no return)
F9 Ctrl-L Types PREFIX (no return)

3.18 TIMER

It is quite accurate, provided the user does minimal disk access. Interrupts are deactivated during disk accesses, and the routine is driven by the interrupts of a mouse card. If you have a mouse card in any slot (no mouse necessary), you can install the visible clock by typing the CLOCK command. This routine is much like the CLOCK command in the way it processes interrupts. Unlike the CLOCK command however, TIMER stands for Invisible CLOCK. Unlike its friend CLOCK, TIMER does not place a clock in the corner of the screen, rather it is used for time and date-stamping files for Prodos. Like CLOCK however it is easily removed, and easily re-activated by using the TIMER command. Both clocks will not run at the same time, only the first one installed will. This clock is slightly less obvious when it is disconnected, but an easy test exists. Press control-g (with SHORT deactivated), or PRINT CHR$(7). If it sounds like a warble, the routine is active. Press RESET to deactivate. TIMER reads the current time and date from Prodos, so if you've used a routine other than DATE or TIME to set the time, it will still work.

*** Press reset to deactivate the TIMER routine before entering another system program.

See also CLOCK, DON, DOFF, DREM, TIME, DATE.

3.19 TYPE

By default, this command will type a text file to the screen or printer. The user is prompted for the filename to print. Typing from other file types is also supported.

3.20 VLIST

This command notes the current online volumes and displays them to the screen. The command itself is quite simple, but this command is not included merely for demonstration purposes. The current prefix is maintained, so you needn't worry about accidentally saving something on the wrong volume. However, if no prefix had been set before calling this command, the prefix after will be set to the prefix of the current drive.

3.21 VSET

This command is actually a BASIC program

I wrote independently. Its another one of the many program selectors available. This one however is accessible at the stroke of the letters VSET. It uses a smart run command (-) so it can run BASIC, Binary, Exec, or System files, assuming of course the program can normally be run using the dash command. That is, some system programs would overlay on BASIC, so they are not allowed to load via the dash.

CHAPTER 4 - PROGRAMMER'S NOTES

4.1 BASIC

To execute a SHELL-based external command from BASIC, just call it as if it were any other Prodos external command:

```
PRINT CHR$(4)"C80"
```

However one may find that commands set up as EXEC files don't respond well to activation within a program, some of these may be corrected by rewriting them in assembler, but the tools available to an assembly language programmer are also more limited in some ways. There are many Prodos commands that cannot be executed from assembly level. My advice, stick to the non-exec commands from inside your programs, or if you really neeed that function, look at its exec file to see how it was done.

The TIMER routine will prove useful from inside BASIC programs for timing sorts, or for calendar programs you may write, maintaining a record of how long a modem connection has been established, or whatever. If there is a next version of SHELL I will set up a pointer to a buffer that will contain the clock information. The timer is accurate to 1/60 second (under 17 milliseconds), which should be fine
for timing things under BASIC. It is just inconvenient with the current program. As I said, perhaps in the next release of SHELL.

4.2 ASSEMBLY LANGUAGE

As mentioned earlier SHELL loads in at an ABSOLUTE address to allow easier access to its buffers. This is not a serious drawback as it provides for subsequent versions to install smart commands. That is, if one types "TYPE filename", the TYPE command itself will begin by checking SHELL's buffer to see if a filename was already given.

A command like CPREFIX could be altered to check to see if a prefix was supplied with the command, why bother querying the user if they already typed it? But as I said, that's the next version.

4.3 HYPER C PRODOS

It wouldn't make much sense to port SHELL to Hyper C, as disk based commands are already supported by the default shell. However

DAEMON would be useful from the C environment, and I am currently attempting this port.

4.4 SCRIPTING FILES

Many routines are actually exec script files that execute BASIC or 6502 assembly language programs. As of this version, script files are normal exec files, of type txt. Wildcard and parameter replacement is not supported, but the original buffer is preserved by the SHELL program, where the individual routines may parse the parameters.

If one wishes to execute an ORIGINAL BASIC command that has been replaced by a new command of the same name, prefix the command with a colon:

eg., :PRINT

Tokenized statements that are part of a BASIC program are NOT replaced by their namesakes. eg.,

```
10    PRINT
```

CHAPTER 5 - CONCLUSIONS

SHELL is very extensible, one could even say ultimately extensible, and it works with most (well ALL the ones I've tried) Prodos external commands, and will probably work with any future command, so long as it is relocatable, or self-relocating.

Feel free to create your own commands for SHELL, that's why I made it. If you come up with something really good, let me know, maybe I'll include it in the next version (it will of course be listed under your name).