MEMORY SPACE FOR ASSEMBLER

When assembling a very large program, it is possible for the Assembler to run out of memory space. There are two remedies to try when this happens:

Use the command-level swapping option to get 11ØØ words of additional memory space.

If the program uses "include" files, use the Filer's M(ake command to create a 4-block file named SYSTEM.SWAPDISK on the same diskette that contains the Assembler. This allows a segment of the Assembler to be swapped out onto the diskette before the operating system segment that opens files is swapped in.

FILE SPACE FOR ASSEMBLER

When the code file is automatically sent to the workfile the default size for the file is [*]. In all other cases, the default size is [Ø], which means that the code file will be allocated all of the largest space available on the diskette that it is sent to. If there is only one available space on the diskette, the code file takes all of it.

This can cause problems if the code file is on the boot diskette or on the same diskette used for the listing file. The Assembler requires some space on the boot diskette for temporary files, and will fail if the code file takes all of the space on the boot diskette. Likewise, the Assembler will fail if it tries to create a listing file and the code file has taken all the available space on the specified diskette.

If you run into these problems, specify a different diskette for the code file or the listing file, or specify a definite length for the code file that will leave enough room for other required files.

Addendum to the

Apple Pascal

Operating System Reference Manual

TABLE OF CONTENTS

APPLE Part #Ø31-Ø1ØØ-ØØ

INTRODUCTION

The document you are reading is an addendum to the Apple II Pascal Operating System Reference Manual. Most of the items described are features that have been added to the operating system since the printing of this manual. Corrections to the manual also have been included.

COMMAND LEVEL

EXEC FILES

An exec file is composed of a series of commands that have been stored in a text file. When an exec file is executed, each command included in the file is executed, just as if you were typing the commands from the keyboard. Exec files are often used to store sequences of commands that must be entered into the system over and over again.

The following sections include an explanation of exec files and a detailed example demonstrating the creation, execution, and editing of an exec file.

Using Exec Files

To create an exec file, Type M for M(ake from the main Command level. You will be prompted

```
    NEW EXEC NAME:
```

Next type the name you want to give to your exec file, following the same rules that govern the naming of other Pascal files. If no file size is specified, the file will be opened with eight blocks. Now you will see the prompt

```
    TERMINATOR=% , CHANGE IT?
```

The terminator is a character that is used to signify the beginning and end of an exec file. The terminator character that is used to begin the file is automatically supplied by the system. The two terminator characters that end the file must be typed by the user. If you answer the above prompt by typing N for No, the system will use a percent sign as a terminator. If you type Y, meaning that you want to change the terminator character, you will be asked

```
    NEW TERMINATOR:
```

Whatever character you enter becomes the terminator character for that exec file. The system will accept any character as the terminator character except the control characters CTRL-C, CTRL-S, CTRL-F, CTRL-@, CTRL-A, CTRL-Z, CTRL-E, and CTRL-W.

The terminator character that signals the beginning of an exec file is supplied by the system. Do not begin an exec file by typing the terminator character. If you do, the system-supplied terminator immediately followed by your typed terminator will be interpreted as the end-of-file signal and the system will close the exec file.

Once the system knows what your terminator character is, you can begin entering the series of commands that will make up your exec file. Commands will be executed as you enter them. End the exec file by typing the terminator character twice.

When you are ready to execute your exec file, type X for eXecute from the main Command level. When you are prompted

```
    EXECUTE WHAT FILE ?
```

you should respond by typing

```
    EXEC/<filename>
```

The system will instantaneously perform the series of commands listed in your exec file, flashing the prompts and your previously-entered responses as it goes.

When using an exec file, you must make sure that the system will be able to go through EXACTLY the same sequence of events that it went through when you created the exec file. For example, suppose you create an exec file that enters the Filer, transfers the file MYFILE.TEXT from diskette OLDSTUF: to diskette NEWSTUF: and then returns to the main Command level. If you later run your exec file without removing the original diskette NEWSTUF: from its disk drive, the system would find MYFILE.TEXT already present on that diskette. Consequently, the system will ask

```
    REMOVE OLD NEWSTUF:MYFILE.TEXT BEFORE TRANSFER ?
```

This is a question that was not asked when the exec file was created. The system will use as its response the next character in the exec file which, in this case, happens to be Q for Quit. In order for the system to remove a file under these conditions, it must receive an N for No as the response to the above question. Thus, the old version of MYFILE will not be removed and the new version of MYFILE will not be transferred. Because the Q was used to respond to this question, the exec file never uses the Q to Quit the Filer and the exec file closes with the system still at the Filer level. The lesson to be learned from this is that, when creating an exec file, you must make sure that the steps the system goes through will not change from one execution to another.

There are some conditions under which you may want to create your exec file using the Editor rather than the Make command. Suppose, for example, that you want to create an exec file that will transfer several files to the printer. If you were to use the Make command to create the file, it would be necessary for you to wait at your terminal while each of the files is actually sent to the printer. A more sensible approach would be to create the exec file in the Editor. Remember, however, that the exec file you create must include every character required by each command. Forgetting to include a single key stroke, such as a carriage return, will prevent the proper execution of your exec file. If you do use the Editor to create an exec file, make sure that you begin the file with a single terminator character and end the file with two terminator characters.

The Editor will not put certain special characters, such as CTRL-C, into its text. Thus you cannot use the Editor to create an exec file that creates a text file.

If you decide to edit an existing exec file, you will notice that certain characters, such as <backspace>, are included in the exec file but do not appear on the screen. For this reason exec files are often difficult to edit.

There is no way to stop the execution of an exec file part way through except by pressing RESET. You can use CTRL-S to temporarily stop or freeze the output on the screen. CTRL-F flushes the output: the program continues to run but its output is not sent to the screen or printer.

The keyboard remains open during the execution of an exec file. Thus characters that are entered while an exec file is running are saved and then used as console input once the exec file is closed.

If you use the Make command to create an exec file on a diskette that already has a file with the name you have given your exec file, the system will eliminate the original file when you close the new exec file.

It is not permissible to create an exec file from within another exec file. If you try, you will be warned

```
    NESTED EXEC COMMANDS ILLEGAL
```

After you see this message, you may continue entering commands into the exec file.

If you are using an Apple with a standard 4Ø-column screen, the control characters CTRL-S, CTRL-F, CTRL-@, CTRL-A, and CTRL-Z, cannot be read into an exec file. If you are using a Communications Card with your Apple to get upper and lowercase and an 8Ø column screen, CTRL-S, CTRL-F, and CTRL-@ cannot be read into an exec file. Although all of these characters will operate normally if used when an exec file is being created using the Make command, they will not be placed in the exec file and therefore will not be read when the file is executed.

If you run a Pascal program while you are making an exec file, typed responses to any UNITREAD procedures specifying unit 1 (CONSOLE:) or unit 2 (SYSTERM:) are used by the Pascal program, but are NOT stored in the exec file. The KEYPRESS function also will take its input from the console and not from the exec file. READ and READLN procedures will take their input from the exec file.

The system error routine will close an open exec file if an error occurs while the system is getting console input from the exec file.

An Example Exec File

Suppose you want to create an exec file that crunches whatever diskette is in Drive #5 and then sets the Prefix to the name of the diskette in that drive.

Prompt: NEW EXEC NAME:

Response: NEWSTUF:UPDATE

You have just created an exec file named UPDATE.TEXT that will be saved on diskette NEWSTUF: . Because you have not specified a size, the system will set aside eight blocks. Next you will be asked:

Prompt: TERMINATOR=%, CHANGE IT?

Response: N

Entering this response tells the system that the terminator character, the character used to signal the beginning and end of the exec file, will remain the percent sign.

You now are ready to enter the list of commands that will make up your exec file.

Keystroke	Explanation
F	Enter Filer
K	Execute Krunch command
#5:	Response to prompt CRUNCH WHAT VOLUME ? telling the system to crunch the diskette in drive #5
<cr>	Carriage Return ending the volume number
Y	Response to prompt FROM END OF DISK, BLOCK 28Ø ? (Y OR N)
P	Execute Prefix command
#5:	Response to prompt PREFIX TITLES BY WHAT VOL ?
<cr>	Carriage Return ending the volume number
Q	Exit from the Filer
%%	Terminator characters indicating to the system that the exec file is complete

Note that each of these commands is executed in the normal fashion when it is typed into the exec file. Thus, after carrying out the above list of commands, the diskette in drive #5 will be crunched and the Prefix will be set to the name of the diskette in drive #5.

When you are ready to execute the exec file, type X for eXecute followed by

```
    EXEC/NEWSTUF:UPDATE
```

You then will see the system enter the Filer, Krunch the diskette in Drive #5, set the Prefix, and return to the Command level.

If you later decide that you want to modify your exec file, you can read it into the Editor just as you would any textfile. The file NEWSTUF:UPDATE would look like this:

```
    %FK#5:
    YP#5:
    Q%%%%
```

Note that the percent sign that begins the file is automatically supplied by the system. Also note that the system supplies two additional terminator characters at the end of the exec file.

Suppose that, when creating the above exec file, you responded to the prompt

```
    KRUNCH WHAT FILE?
```

by typing a dollar sign instead of a number sign. After you realized your mistake, you backspaced once and typed the correct response (#5:). If you read this version of the exec file into the Editor, the display will look exactly like the display in the example shown above. In reality, however, the exec file contains these characters

```
    %FK$<bs>#5:<cr>
    YP#5:<cr>
    Q%%%%
```

where <bs> represents a backspace and <cr> represents a carriage return. Now suppose that you decide that you want to Krunch the diskette in Volume #4: instead of the diskette in Volume #5: . First you might decide to delete the 5. If you move the cursor so that it sits on the 5 , type D for Delete, and then move the right arrow one space, the first line of the display should now look like this:

```
    %FK$#5:
```

This same line in the exec file actually contains these characters:

```
    %FK$#5:<CR>
```

The character deleted, a backspace character, was the fifth character in the exec file, NOT the fifth character displayed on the screen. Notice that the dollar sign character that had been typed over by the backspace in the original exec file now appears on the display. Once the backspace character is removed, characters can be inserted and deleted normally.

In general, it is quite difficult to edit exec files. If you must do so, remember that certain characters, such as backspace, CTRL-C, and cursor-moving characters, will not show up on the display even though they are present in an exec file.

THE SWAPPING OPTION

Apple Pascal now includes a Swapping option that allows you to maximize the space available in the Apple's memory. The following chart illustrates memory availability for a codefile at execution time on a standard Apple II with 4Ø column screen:

	Swapping On	Swapping Off
Approximate maximum user program space in bytes	399ØØ	377ØØ

The Swapping option is set to OFF each time the system is booted. To change the Swapping option, type S from the Command level. You will either see the message

```
    SWAPPING IS ON
```

or

```
    SWAPPING IS OFF
```

depending on Swapping's current state. Below this will be the question

    TOGGLE SWAPPING?

Typing a Y changes Swapping from ON to OFF or vice versa. Typing an N leaves Swapping in its current state. When Swapping is turned on, a portion of the operating system must be read in from the boot diskette each time you open or close a file. This results in a slight slowdown of the system.

Two new procedures, SWAPON and SWAPOFF, make it possible to turn swapping on and off from within a program. These procedures are part of the CHAINSTUFF unit and are described in the section on chaining in the Addendum to the Apple Pascal Language Reference Manual.

Using a standard Apple II with 4Ø column screen, the maximum size file that can be read into the Editor with Swapping set to OFF is 34 blocks. The maximum size file that can be read into the Editor with Swapping set to ON is 4Ø blocks. Before reading a file that is longer than 34 blocks into the Editor, make sure that Swapping is turned on. If you try to read a file into the Editor that exceeds the Editor's limit, you will see the message:

```
     ERROR:  BUFFER OVERFLOW!!!! PLEASE PRESS <SPACEBAR> TO CONTINUE
```

When you press the spacebar, as much of the file as can fit will be read into the Editor. At this point you should use the Quit and Exit commands to leave the Editor; next set Swapping to ON. Now when you read the file into the Editor, the complete file will be available for editing.

The Pascal Operating System Manual says on page 98 that the maximum size Editor file is about 184ØØ bytes or 38 diskette blocks. This figure does not include the 2 blocks of information that are included in the first two blocks of any text file. To find out the actual amount of space that a file occupies, use the Filer's List directory or Extended directory list command.

One drive users: When Swapping is set to ON, all operations must be performed with the boot diskette in the disk drive. The only exception is the execution of Filer commands. The Filer, once called, causes the operating system to remain entirely inside the Apple's memory. Thus, once the Filer has been called, the boot diskette can be replaced by any other diskette containing files that need to be accessed.

UPPER AND LOWERCASE CAPABILITY

In normal operation, all letters typed on the Apple II keyboard are uppercase letters and all letters are displayed on the screen as capitals, regardless of whether they are actually uppercase or lowercase. Version 1.1 of Apple Pascal allows you to alter this normal mode in two ways:

You can shift the keyboard into lowercase (and back into uppercase) at any time.

You can cause uppercase letters to be displayed on the screen in reverse video (black on white) to distinguish them from lowercase letters.

Note that only letters are affected by these features. Also, note that you can now create a file containing lowercase letters as well as uppercase; if you then print this file on a printer that has upper and lowercase capability, true upper and lowercase characters will be printed.

Keyboard Case Control

By turning the "reverse video mode" ON, you cause uppercase letters to be displayed in reverse video while lowercase letters are displayed in normal video. The control characters CTRL-E and CTRL-W are used to turn reverse video ON and change the case of letters typed on the keyboard. They can be used from any level of the system, including the Editor.

- Typing CTRL-E (for "Enable") on the keyboard turns the reverse video mode ON, and changes the case of the keyboard from upper to lowercase or vice versa. Thus it is like the "shift lock" key on a typewriter.

- Typing CTRL-W (for "Word") on the keyboard turns the reverse video mode ON, and forces the keyboard into uppercase for the next character typed. After the next character is typed, the keyboard is forced into lowercase. This is useful when you want to capitalize the first letter of a word.

Notice that since CTRL-E and CTRL-W turn reverse video ON, they provide all the control you need except for turning reverse video OFF again.

Other Control Characters

The control characters CTRL-R and CTRL-T control the video mode from any level of the system except the Editor:

- Typing CTRL-R (for "Reverse") on the keyboard turns the reverse video mode ON. Note that this does not change the keyboard case; all letters typed on the keyboard are still uppercase unless CTRL-E or CTRL-W is used to change case.

- Typing CTRL-T (for "Turn off") on the keyboard restores normal operation: that is, it turns reverse video mode OFF and also forces the keyboard into uppercase.

Note that CTRL-T and CTRL-R have no effect when you are in the Editor.

Using Upper and Lowercase

The best way to get acquainted with this capability is to experiment with it. Try entering the Editor and typing I for Insert. Now type CTRL-E followed by some text. The letters you are typing will appear in normal video. Now type CTRL-E again. All subsequent alphabetic characters appear in reverse video. Next type CTRL-W followed by a word. The first letter appears in reverse video while the remaining letters appear in normal video. If, at this point, you save your file and then print it, all characters except those that appeared in reverse video will be printed in lowercase. After you exit from the Editor, the system will still be in reverse video mode. To return the system to standard uppercase only mode, type CTRL-T.

RESET

The discussion of RESET that appears on page 1Ø of the Apple Pascal Operating System Reference Manual is not correct. Pressing the Apple's RESET key now does a "cold boot" of the system. It is equivalent to turning off the power and turning it on again. After you have pressed RESET, you will have to reboot the system using the normal startup procedure.

THE FILER: THE PREFIX COMMAND

The Prefix command is normally used to set the Prefix to a specified volume name. You can, however, use the Prefix command to set the Prefix to the volume number of a disk drive. This feature allows you to change diskettes in a given drive without having to change the Prefix each time a different diskette is placed in that drive. To use the Prefix command in this way, the Prefix must be set when no diskette is placed in the drive that you wish to designate as the Prefix disk drive. If, when you are setting the Prefix, a diskette is in the drive that is designated as the Prefix disk drive, the Prefix will be set to the name of the diskette in that drive.

EXAMPLE

Suppose that you want to set the Prefix to Volume #5. With your boot diskette in the boot drive and no diskette in Volume #5, enter the Filer and type P for Prefix.

```
Prompt:     PREFIX TITLES BY WHAT VOL ?

Response:   #5
```

You have just set the Prefix to whatever diskette is placed in Drive #5. Suppose that you now place diskette LISTA: in drive #5 and type E for Extended directory list followed by a colon. The directory of LISTA: will appear on the screen. Now remove diskette LISTA: and replace it with diskette LISTB: . If you again type E for Extended directory list followed by a colon, the directory of LISTB: will appear on the screen.

THE EDITOR: FIND AND REPLACE

The Find and Replace commands now have the capability to locate a string of characters within text regardless of whether the characters in the text are of the same case as the characters in the target string that you have specified. To utilize this option, type F for Find or R for Replace followed by U for Upper and Lowercase. Next enter the string, set off by delimiters, that you want to Find or Replace. To use the Find command to find the next occurrence of the previously specified string without regard to case, type

F for Find. Then, when the Find prompt line appears, type US . To use the Replace command to replace the next occurrence of the previously specified string without regard to upper and lowercase, type R. Then, when the Replace prompt line appears, type US .

This option may be used to locate strings of characters that were entered using the reverse video option described earlier.

EXAMPLE 1 :

Suppose you have a file containing the following text:

```
Program String;
Begin
    write ('The rain in Spain');
    writeln (' falls mainly on the plain.')
End.
```

and that you want to use the Find command to move the cursor to each occurrence of the word "the". When at the Edit level, type F for Find. When you see the Find prompt, type the following

```
    U/the/
```

When you type the last / , the cursor will move immediately to the space following the "e" in the first occurrence of the word "the" even though, in this instance, the first occurrence of the word "the" begins with an uppercase letter. To move the cursor to the next occurrence of "the" type F again. When you see the Find prompt line this time, type U S . This tells the system to find the same string as specified previously, regardless of whether the characters in the string are in upper or lowercase. If you were only to type S when the Find prompt line was on the screen, the system would find the next occurrence of "the" containing only lowercase characters since that is the string you specified in the original Find command.

Example 2:

Suppose you have the following program

```
program chocolate;
begin
     writeln (' Checkmate Chocolate is the best ');
     writeln (' chocolate in the whole wide world.')
end.
```

and that you want to replace every occurrence of the word "chocolate" with the word "peanut butter". From the Editor type R for Replace. When the Replace prompt line appears type the following:

```
    u/chocolate//peanut butter/
```

All three occurrences of the word "chocolate" will be replaced by the word "peanut butter". Note that "peanut butter" will appear in lowercase regardless of whether the word it replaces is in upper or lowercase.

THE 6502 ASSEMBLER

ERROR MESSAGES

The following new Assembler error message has been added:

```
     5ØØ:  General assembler error
```

ASSEMBLY LISTINGS

Note that you can use the Editor to examine (or edit) a listing file produced by the Assembler; after the Editor loads the listing file, you should do a Replace operation on the entire file to replace all CTRL-L's with CTRL-M's. To do this, type the following characters:

```
     /R/CTRL-L//CTRL-M/
```

(The CTRL-L's that the Assembler puts in the listing are "page eject" characters intended for controlling output on a printer.)