CHAPTER 11: CONFIGURING APPLICATIONS

Configuring and Running Applications

When certain applications are run in more than one partition at the same time, confusion can result. Such a multiple session usage could occur in either a multiuser or multitasking environment. For example, if an accounting package which was not designed to lock files and records were to be operated in multiple sessions, the data files could easily become corrupted.

The most secure approach would be to convert to the use of a multiuser package. However, if this is not possible you may find that the flag file technique, described in Chapter 4, can be of help. This approach will let you regulate access to the application so that only one user will be able to operate it at a time. Each user can work from their own workstation with the flag file system and be sure that all others wait their turn.

The RJE technique, also covered in Chapter 4, could also be used provided the application in question is not interactive. Both the flag file and RJE techniques impose one-at-a-time access to an application, so these methods will only work when simultaneous access is not a requirement. If your situation requires multiple sessions with a shared data file set, you must use an application which has been designed for a multiuser system.

There can be situations where you would like to make one copy of a program available for multiple session access where individual data files will be used in each session. If the application doesn't write to any common files, and can be directed to use indi- vidual data and configuration files for each session, there is no reason why the same code module cannot be loaded by more than one task at the same time.

In the case where a program uses a configuration file to control attributes such as screen colors, if there is no way to specify an individual configuration file for each session, all users will have to agree on the settings in the one CONFIG.SYS file. If an application needs to create scratch files, such as the overflow files some word processors use, you must be able to specify unique directories for each session or corrupted data will likely result.

Certain programs require customization based on the type of video display adapter you will be using. Supporting multiple sessions of such an application where one set of code modules will be used can present an interesting challenge in a co-resident VNA system. If the name of a configuration file or, more directly, the type of display adapter to be used, can be specified on the command line when the program is invoked, then simultaneous access may be possible.

Some programs may check for operating parameters in the environment or in a CONFIG.SYS file within the current directory, in which case supporting multiple sessions with individual requirements should be feasible. When re-configuring the program on the fly is not possible you will need to install multiple copies of the application. This can often happen in a co-resident system where the master console has an EGA, VGA, or CGA display adapter and the other workstations have Hercules™ and monochrome capability.

If you have set up a batch file to do a MOS VMODE command before starting a graphics based application, you should do an ERRORLEVEL test after the VMODE in case insufficient memory is available for the video buffers. For example, when the current video mode is MONO and a MOS VMODE HG2 is executed, an additional 64K of extended memory must be allocated to make up the video save and VIDRAM buffers. If the VMODE command fails because of too little memory, you wouldn't want the batch file to continue and start the graphics application. Entering a graphics application when the proper video mode support has not been established will produce unexpected results.

When a task which is in an EGA graphics mode is unwatched (PAMswitched away from), it must be suspended, since software emulation of an EGA adapter is not supported. However, if IRQ interrupts can occur for the task while it is suspended, they cannot be stopped without risking system lockups. Each IRQ situation is unique so MOS cannot just presume to ignore them. This can cause video problems if the IRQ handler is designed to update the video display.

Communications packages, tape backup utilities, and mouse oriented graphics programs are examples of applications which use IRQ interrupts. It is unlikely that a communications program or tape backup utility would use EGA graphics and, in the case of a mouse-based application, the best solution would be to not PAMswitch away while the program is active.

In the case of an application which must be run from within a certain directory, because it isn't sophisticated enough to find its overlay or data files in any other situation, use a batch file with the AUTOCD command. By placing this batch file in a directory which is within your command search path, you can bring up the application no matter what your current drive and directory.

Finally, don't forget about the %A batch file variable. This can be quite useful in forming directory or file names in the parameter line for a program. If you've had to install a copy of a certain application for each user, you could simplify access to this package by having one batch file which uses %A in the formation of an explicit program path name. For example, if each user has their own copy of XYZ.EXE then a batch file such as the following could be used:

```
c:\dir%a\xyz %1 %2 %3
```

For user #1, there would need to be a copy of XYZ.EXE in the directory DIR01, with user #2 using DIR02, etc. Note that this type of approach requires consistency in task numbering. You would be wise to use explicit task numbers in your ADDTASK commands rather than let MOS assign the next free number.

Shelling vs Spare Tasks

Many application programs allow you to "shell out" to the operating system. This involves a temporary exit from the application to a point where you can enter commands such as COPY and DIR from a command prompt. When you are ready to re-enter your application, you enter the EXIT command to terminate the shell.

A consequence of shelling out is that the memory available to run programs is often greatly reduced. Using a utility task rather than shelling out of an application has several benefits. You will always have a full size task available which is setup as you want it and stays as you leave it in between uses. In addition, it is available even when you are running a program in your main task which does not support shelling out.

Maintaining one or more spare tasks can also be beneficial when you wish to use certain TSR style utilities. If you have a TSR which isn't compatible with one of your main applications, put it in a small task by itself. Also, keeping TSR's out of your main task helps keep more TPA memory free for your main applications. Finally, when a TSR's inability to un-install itself becomes a problem, you can always remove and re-add your spare task.

Without doubt, the greatest productivity boost obtainable from maintaining one or more spare tasks is the ability to run more than one process at the same time. Jobs such as large file updates and telecommunications file transfers consume your entire machine in a single tasking environment. With MOS, you can switch to your spare task, start a long job, and then switch back to your previous work before your train of thought even begins to fade.

The most important thing about a spare task is to pre-plan for it. The time when you'll want to have a spare task the most is when your current task is busy and running ADDTASK.COM is not possible. There is one exception to this rule however. Beginning with release 4.00, a new utility called MONITOR.COM was included with the PC-MOS package.

The MONITOR.COM utility performs many of the same functions as the ADDTASK.COM, MOS.COM and MOSADM.COM utilities, but in TSR format. With it, you can pop up a menu in the middle of an application and add or remove tasks and review the system's status just as with MOS MAP. Since this TSR utility does include the capability to perform MOSADM functions such as setting the slice and priority, you should consider whether access to it should be restricted.

Command Entry and Recall

MOS's command recall and command line editing features can be a great boon to productivity. As with any new skill to be learned, the best way to get comfortable with it is to practice. If you've ever used a word processor or text editor, you'll absorb this in no time. Don't wait until you need to know how to do each function to look it up and try it out, letting yourself play a bit. A few minutes spent gaining familiarity will more than pay for itself later.

The command recall buffer can be quite useful when you've entered a command that should have been entered in another directory. For example, if you want to see all the .DOC files in your WORK directory and enter DIR *.DOC but forgot that you were in the BIN directory, just enter a CD \WORK and press the up arrow twice to bring back the DIR *.DOC command.

If you've just typed in a command line - and a thought occurs to you "is the file I'm about to process ready?" rather than aborting the current line you've just typed, especially if it is a long one, use Ctrl-PgUp to save the line in the command recall buffer. Then you can check the file with DIR or TYPE. When ready, use the up arrow to bring back the line you just typed. It will be there as long as you didn't execute a FLUSH command or SIGNON/SIGNOFF. (The SIGNON and SIGNOFF commands contain an intrinsic flush to prevent someone from reviewing your recent command history when you leave your machine).

When you've just finished typing in a long command that you'd like to have in a batch file, there is a way to use the command recall buffer and MOS's command line editing features to simplify the job. After bringing the command back with the up arrow, type a ">" symbol after it, followed by the name of the batch file you wish to create. For example: >XYZ.BAT Then press the HOME key to move to the start of the line and, with insert mode active, type in the word ECHO followed by one space. When you press enter, you will have created a one line batch file with a minimum of keystrokes. The final form of this command line should appear as:

```
ECHO your command line >XYZ.BAT
```

Finally, if you get interrupted while entering a sequence of commands and you want to review what you've done when the interruption is over, press the up arrow key a few times to see what you were doing.