# CHAPTER 6: ADDING TASKS & WORKSTATIONS

## Types of Tasks

A task is considered to be a "background task" if it is started without initially having a workstation console associated with it. Conversely, when the appropriate information is supplied to the ADDTASK command, a task known as a "workstation task" can be started -- one to which a console is initially attached.

In addition to this classification, tasks may also be categorized by the type of use to which they will be put. A general purpose task would be one in which a variety of different applications programs would be run such as word processors, data base managers, etc. A task may also be used in a specialized or dedicated manner, such as a one in which a communications gateway program or MOS's print spooling processor is run.

It is a good idea to keep a small spare task around, with a size of about 32 to 128K. The time you will need an extra task the most is when the one your are currently using is busy. Of course, if you have the memory to spare, there's no reason not to just add a background task of several hundred kilobytes, so it will be ready at all times.

However, by using a small task for this, the extended memory overhead required is minimal while the task is still large enough for a variety of utilitarian functions. When a larger spare task is needed, you can use your small spare task to invoke the ADDTASK command to start another, or use the MOS RESIZE command to enlarge the existing small one.

A new feature for workstation tasks, introduced with release 4.00 of MOS, is the ability to reboot just a specific workstation. This is accomplished by pressing the Ctrl-Alt-Del keys simultaneously from that workstation's keyboard. If any other workstations are viewing that task (through PAMswitching) they will automatically be returned to their home task before the task is restarted. The restart uses the same information that was used when the original ADDTASK command was issued, including the same startup batch file. Note that a workstation may only restart its original "home" workstation task.

For serial terminal workstations which are remotely connected with a modem, it is also possible to automatically invoke a task restart when the modem indicates a loss of carrier. This option is controlled on the parameter line of the serial driver. Consult the documentation for your specific serial driver for details. Also, see the section "Securing Remote Tasks" in Chapter 10 for more details on this feature.

There are two primary reasons for using this feature. First, in the event your online session is interrupted by a phone line failure, having the remote task automatically restarted will insure that the modem on the remote end is restored to its auto-answer state. This is typically taken care of by invoking the MODEM.COM utility from within the task's startup batch file. The second reason for configuring a remote task to automatically reboot when a loss of carrier is detected is to insure that an abnormal disconnection doesn't leave your remote task open for anyone else. Having the task restarted will insure that the startup batch file will re-issue the security signon prompt.

One consequence of using this feature is that you could not call up the computer at your office from home, start a long job of some sort, and then disconnect. As soon as you disconnect, the task would be re-started and your job would be terminated. If you find that you want to use the auto-restart on carrier drop feature, but also need to be able to start long jobs and then disconnect, start the job in a task other than the one you call in on. The RJE technique discussed in chapter 5 may be helpful in such a situation.

## Task Numbering

When working with more than one task it becomes important to be able to keep track of which task is doing which operation, or which one is associated with each workstation. In PC-MOS, each task has an associated task number. The foreground task is always task number 0. When the ADDTASK command is used, an explicit task number may be specified or you may elect to have MOS automatically assign the next available number in sequence. The range of numbers which may be used, regardless of whether they are assigned explicitly or by default, is 1 through 99. You may assign these numbers in any sequence you wish.

In some cases, there are advantages to starting tasks with specific numbers. For example, when a dedicated task is spawned in which to operate MOS's print spooling program, it can be useful to number it as task #90 (or something similar) to make it stand out as unique.

In a system supporting multiple users, where each user may want or require background tasks of their own, you may wish to use a patterned numbering system.

If the first workstation were assigned task #5 and the second #10, this would leave numbers 1 through 4 available for use by the operator at the master console, with 6 through 9 available for the first workstation, etc.

Labeling each workstation with their task number, and possibly with a directory of the other task numbers used in the system, can also help keep things clear. You might also want to create a small text file where this information can be reviewed on line, or a batch file menu which presents the user with a descriptive list of tasks to which they can switch by pressing one key. This feature will be covered in more detail later in this chapter.

## Types of Workstations

### Serial Terminals

A serial terminal type of workstation is one which communicates with the host system through an RS-232 type of interface. Each workstation is supported by the host system through a common RS-232 serial port (see Chapter 8 for setup information). Only three wires, or conductors, are necessary in the cables used for this type workstation interface. The cables should be shielded with the maximum length depending on the communications speed (baud rate) and the electrical noise conditions in your environment.

A baud rate of 38.4K is typically the maximum for a dumb terminal. This type of communications link may be extended through the use of modems, fiber optic repeaters, or any other type of data switcher or multiplexer which will faithfully reproduce the RS-232 interface in a full duplex mode.

When a serial terminal, often referred to as a "dumb" terminal, is used which provides a full PCTERM emulation, all of the special text symbols and keys specific to a PC style console will be available. Although other types of dumb terminals may be used with MOS, when the PCTERM emulation is not available, attempts by applications to display certain characters from the IBM extended character set will result in *'s being displayed instead. Likewise, in lieu of having the actual keytops for F1 through F10, PgUp, PgDn, etc, you will have to resort to what are known as escape sequences in order to supply equivalents to these IBM specific keystrokes.

It is also possible to run a terminal emulation software package on a stand alone computer which is linked to the host system through an RS-232 interface. In such a setup, the stand alone computer will act like a dumb terminal and be able to access the host system as a user. The Software Link's PC-Emulink™ package is one example of such a program.

Using a terminal emulator can be very useful in certain situations - especially when used in conjunction with LAN access to the host such as provided by LANLink (See Chapter 8 for more information on network connectivity). The ability to run CGA color graphics can be an additional reason for using PC-EmuLink.

Figures 6-1 and 6-2 illustrate some of the configurations possible with serial terminal workstations.

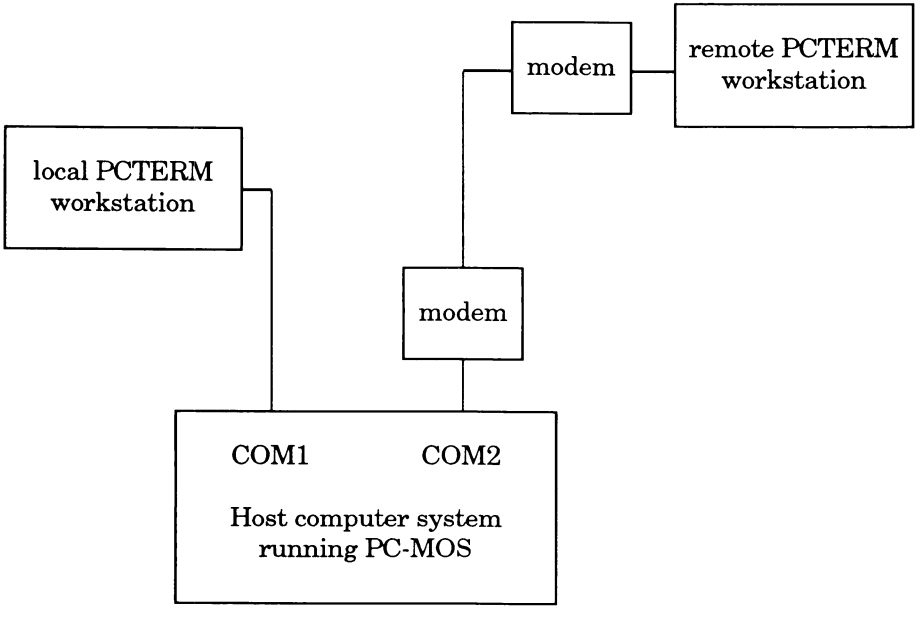


**Figure 6 - 1**

This is a simple multiuser configuration where the two serial ports which come as standard equipment on many systems have been used to connect the workstations. PCTERM type of serial terminal workstations are shown here. Other possibilities include other types of dumb ascii terminals and stand alone computers running a terminal emulation package such as Emulink. Note that since RS-232 serial communications is used to link the workstations, modems and similar devices may be used to distance workstations from the host.

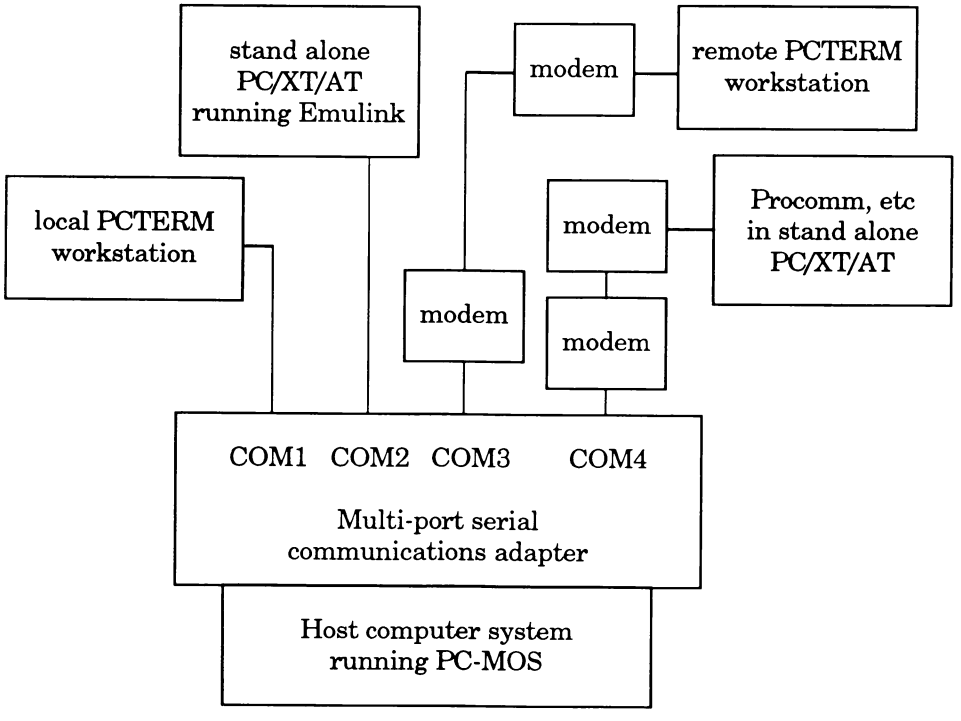


**Figure 6 - 2**

Through the use of multiport serial cards, up to a maximum of 24 serial ports may be installed in one system. See Chapter 8 for details on types of serial ports.

# CHAPTER 6

## VNA

The acronym VNA stands for Video Network Adapter. This technology provides each workstation with the hardware equivalent of its own monochrome and Hercules graphics master console. Each workstation's video display hardware is directly accessible to the host CPU so display updates do not incur the delays inherent to a serial terminal type of workstation. This makes it possible to support multiple users with graphics based workstations with video update speeds near that of standalone computers.

Each workstation is connected to the host through a 25-pin shielded cable and, given nominal electrical noise conditions, cable lengths of 400 feet may be used. Since this is not an RS-232 serial interface, modems and similar devices cannot be used to extend the reach of these workstations. Should a mixture of workstation types fill a need, there is no reason why serial terminal workstations cannot be added to a VNA system. Refer to Chapter 8 for a discussion of serial port configurations.

Unlike dumb serial terminals, VNA workstations are made up from individual keyboards and monitors. Once you've obtained the VNA specific pieces of the hardware network -- the cables and interconnection boxes, you are free to choose your own favorite type of keyboard and monitor. The monitor need only be capable of displaying monochrome text and Hercules graphics. The only requirement for the keyboard is that it be XT-compatible (many keyboards are switch configurable).

With the VNA system, it is possible for the master console itself to use one unit of the VNA's hardware as the foreground workstation's display adapter. In such a configuration, it is possible to support a maximum of 16 workstations per system (including the master console). Each VNA motherboard requires an expansion slot in the host system and supports four workstations. Therefore, an additional consideration in planning an installation is the availability of free expansion slots.

In such a configuration, a VTYPE of 5 must be used to provide a large enough video save area to accommodate the needs of Hercules graphics support. The "F" option may be included if it is desired to fill out the foreground's TPA up to a ceiling of B0000. A VTYPE of 2 or 2F could be selected if no Hercules graphics applications will ever be used. This will reduce the FREEMEM allocation incurred by the video save area from 16K to 4K.

In an alternate configuration, an EGA or VGA display adapter is used for the master console in conjunction with one or more VNA workstations. Such a system can support CGA text mode, CGA graphics, EGA graphics, and VGA graphics (when a VGA card is used) at the master console and monochrome text and Hercules graphics at each VNA workstation. It would also be possible to install a pure CGA card as the master console's display card in conjunction with VNA workstations but, due to hardware conflicts, this combination precludes the use of any Hercules graphics applications which make use of the 2nd graphics memory page. As it is not always easy to determine which applications make use of this 2nd page, using a pure CGA adapter is not advised.

When the master console's display adapter is of the EGA or VGA type, the system configuration is referred to as a co-resident one. In such a case, the maximum number of VNA workstations that may be installed is reduced to 15 and, as stated above, a VTYPE of 5 must be used to support Hercules graphics. However, one significant difference in the co-resident case is that the "F" option must not be used since this would cause a conflict with the EGA/VGA hardware installed for the master console and the foreground task's TPA space.

Figures 6-3 and 6-4 illustrate the two different types of VNA system configurations.

Figure 6-3 shows a VNA system where the master console is made up from VNA equipment. The item labeled TU is a Translation Unit and the box labeled IU is an Interface Unit. These are connection boxes which are part of the VNA architecture. A separate serial card is used to provide ports for a mouse at each workstation.

Figure 6-4 shows a co-resident VNA system -- where the master console is made up from an EGA (or VGA) display adapter.

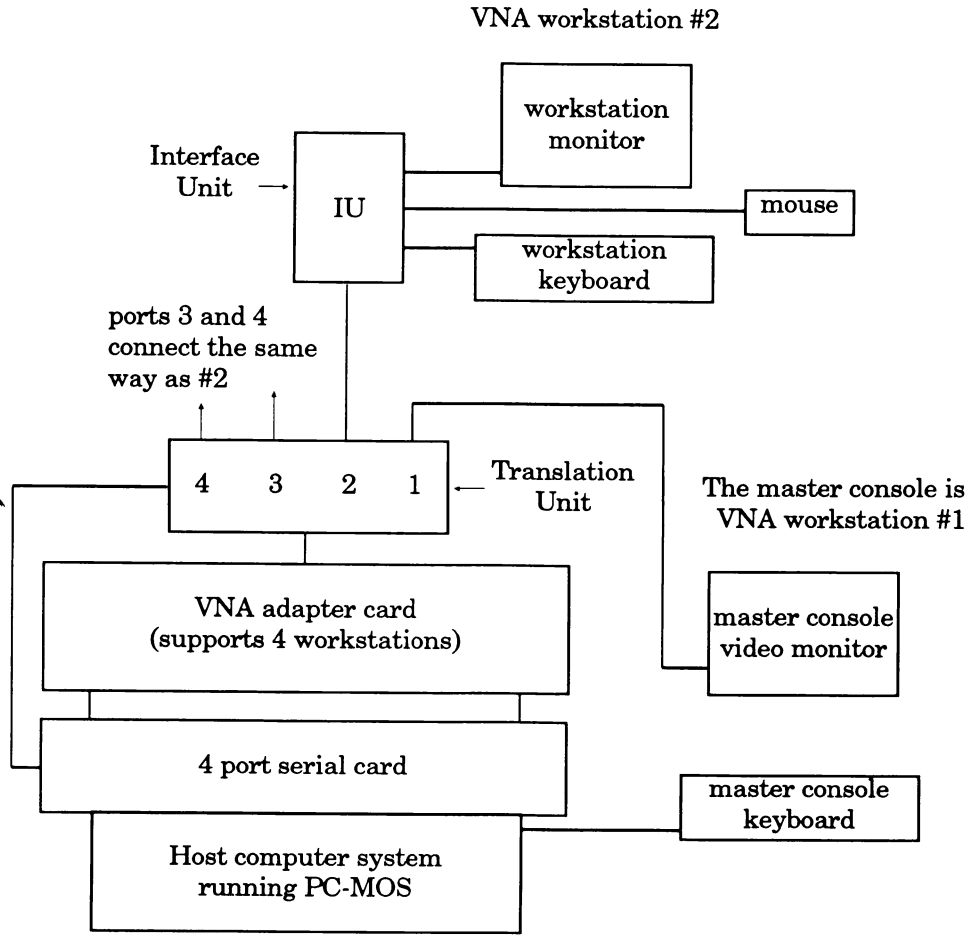


**Figure 6 - 3**

Shown here is a VNA system where the master console is made up from VNA equipment. (A separate serial card is used to provide ports for a mouse at each workstation).

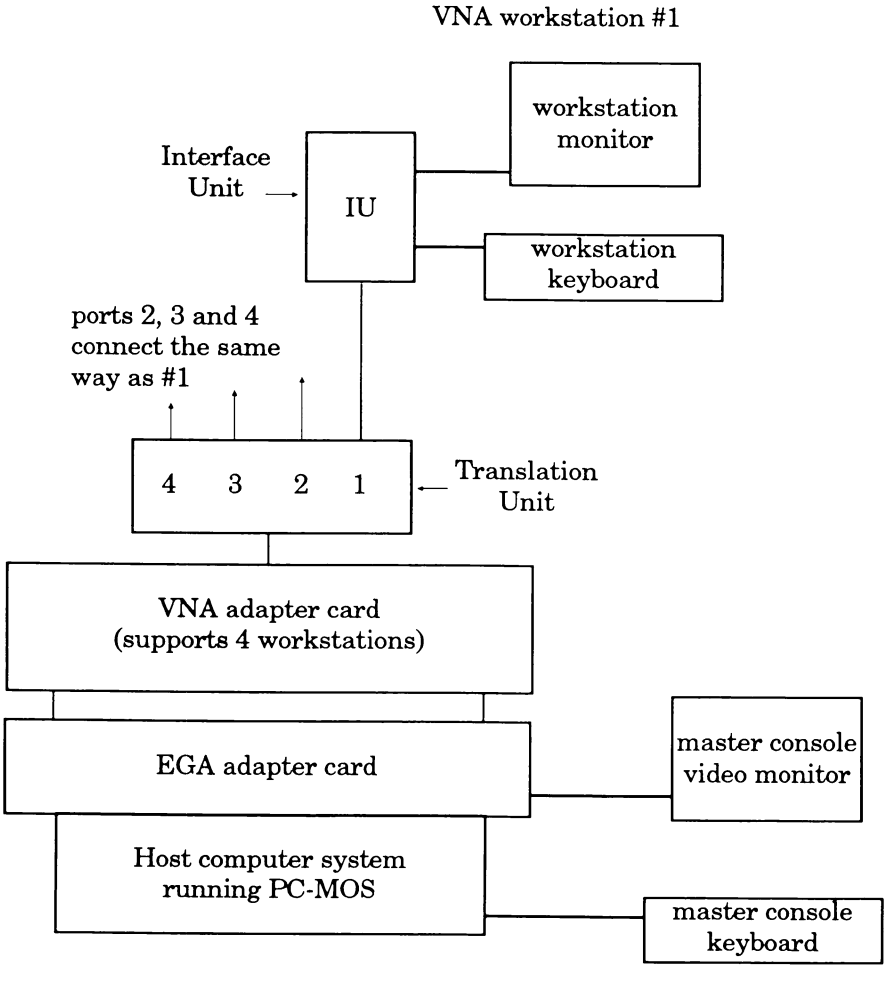


**Figure 6 - 4**

This is a co-resident VNA system - where the master console is made up from an EGA (or VGA) display adapter.

Beginning with release 4.00 of MOS, support for an adapter known as the IONA card was introduced. The IONA adapter card is designed to work in conjunction with the VNA workstation adapter supplementing the VNA system through the addition of a parallel port, a serial port, and a bell for each workstation. Each IONA card contains hardware to provide four serial ports, four parallel ports and the hardware to drive four beepers. The signals for driving the serial and parallel ports and beeper are transmitted to each workstation's interface unit through the same 25 line cable which supports the monitor and keyboard.

The VNA.SYS device driver has been expanded to include operating code for the IONA's hardware. To support the serial ports, the equivalent of a $SERIAL.SYS has been incorporated within VNA.SYS. In order to prevent the waste of SMP memory, this extra code and the requisite data buffers are only retained within the SMP by VNA.SYS when IONA support is indicated in the parameter line. Otherwise, only the code required for the VNA hardware is retained. To use this new version of VNA.SYS in a system where no IONA cards exist, use the same parameter line as presently documented for VNA.

In order to utilize the parallel port the command MOS ROUTE LPTx TO TERM must be issued from within the workstation task. This is the same as is required to activate the printer port on a serial terminal workstation such as a Wyse 60®.

The primary use of an IONA serial port is to support a serial mouse. The INT14 mouse interface method which uses $MOUSE.SYS must be used with IONA serial ports (see Chapter 9). It is also possible to drive a serial terminal workstation or a serial printer through one of these ports. However, a tele- communications program which is designed to directly access a standard COM1 or COM2 port would not be able to use an IONA port.

Within each IONA interface unit is a small beeper which allows a VNA workstation to hear a beep when a bell code is sent. Note that this is only a beeper and not a speaker in the sense of a master console's speaker. Music and sirens are not supported - just beeps.

Figure 6-5 illustrates a VNA system using the IONA companion adapter. Each workstation has its own serial and parallel port and a beeper. Although not shown here, a special cable can be used to access these ports for the master console.

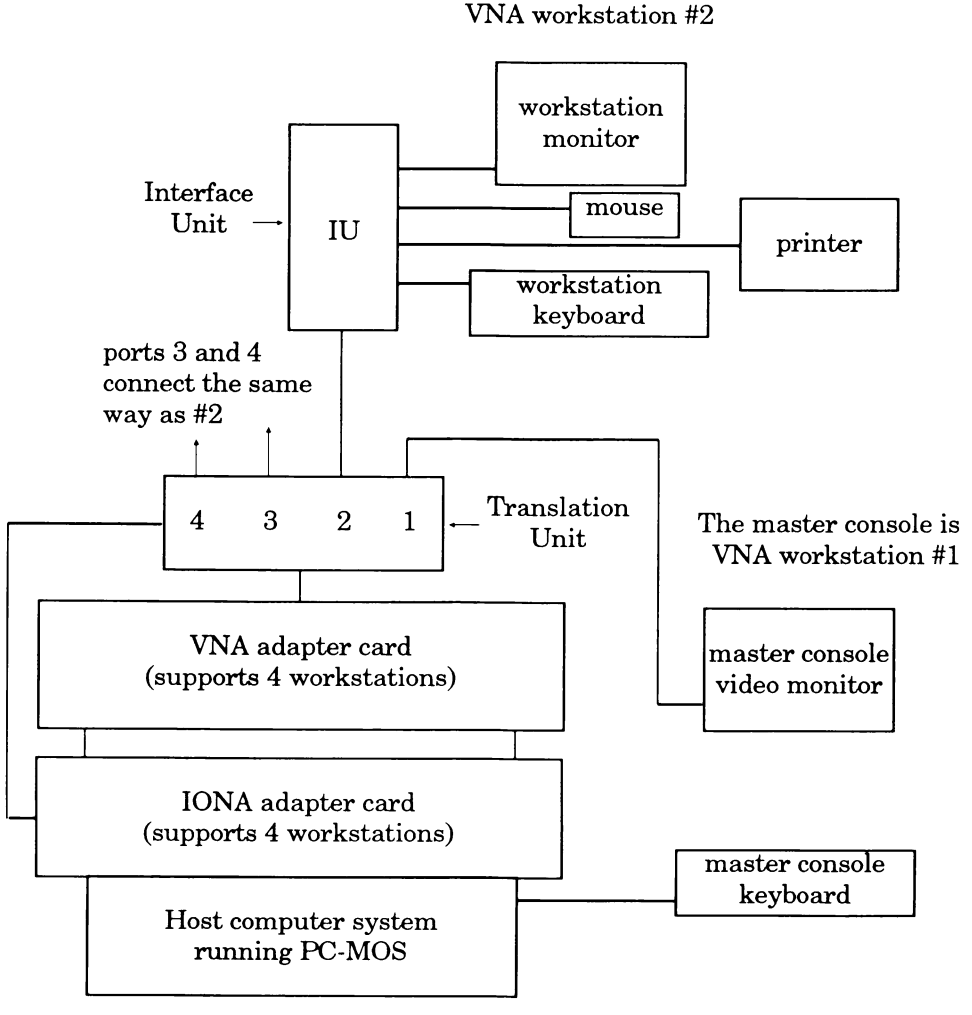


**Figure 6 - 5**

VNA system using the IONA companion adapter

# CHAPTER 6

## SUNRIVER

Beginning with release 4.00, PC-MOS supports a workstation known as the SunRiver® Fiber Optic Station™. These stations can display EGA and VGA graphics and are linked to the host system through a pair of fiber optic cables. With a 32 Megabit-per-second transmission rate, approximately ten full graphics screen refreshes per second are possible.

Each SunRiver adapter card can support up to four workstations. The total number of workstations which can be supported within one system does not have an arbitrary limit as with VNA. The number of available expansion slots in your system, processor speed, and extended memory size are the limiting factors.

A SunRiver workstation is a complete keyboard/monitor combination. This is more similar to a dumb serial terminal and in contrast to the VNA case where you must add your own keyboard and monitor. Through an adapter card known as the PC Fiber-Optic Station Emulator, an existing PC or AT machine with an EGA display can be converted to the equivalent of a SunRiver style workstation. In this configuration, it will still be possible to use the PC or AT as a stand alone machine. If you wish to convert an existing hardware base to remote graphics workstations, this can be a very cost effective approach.

Each workstation unit contains one serial port and one parallel port. With the parallel port available for a local printer, the serial port will typically be used for a mouse but other serial devices could also be used. Since a SunRiver serial port is not directly accessible to the host CPU, most tele- communications programs will be able to use it. This port must be accessed through the COMn device name interface or through the INT14 system call interface. A maximum of 115K baud is possible.

With the present software design, the master console of a system in which SunRiver style workstations are to be used must match the workstation type. When EGA type workstations are used, an EGA master console card must be installed, with a VGA adapter being installed when SunRiver's VGA workstations are attached.

The terminal device driver which supports these workstations is named SRTERM.SYS. This driver requires a 12K mapping window which means that a gap must be reserved somewhere within the FREEMEM memory region. Since EGA/VGA video types are involved, no VTYPE declaration can be used to enlarge the TPA size and the memory area from B4000 to B8000 should not be used for either $RAMDISK.SYS's memory window, or that of SRTERM.SYS.

Certain restrictions on PAMswitching exist within a system using SunRiver stations. No task may be watched by more than one workstation at the same time. A SunRiver workstation task can spawn background tasks and PAMswitch to them but the workstation cannot be PAMswitched into another workstation task. However, if another workstation is not currently watching its home task, another workstation could switch in to watch it since this fits within the rule of no more than one workstation viewing a task at one time.

Regarding the use of fiber optic cables, the maximum run permissible is 1000 ft. Shorter cables are available and up to 3 separate ones may be connected end to end to make up a longer run. The standard cable lengths available are 25, 50, 100, 150, 250, 500 and 1000 foot.

Figure 6-6 shows an illustration of two possible workstation configurations. The lower host system has two different types of SunRiver workstations attached. The first is a simple workstation and the second is a standalone PC or AT computer system which is using a PC Fiber Optic Station Emulator to make it appear as a SunRiver terminal to the main host.

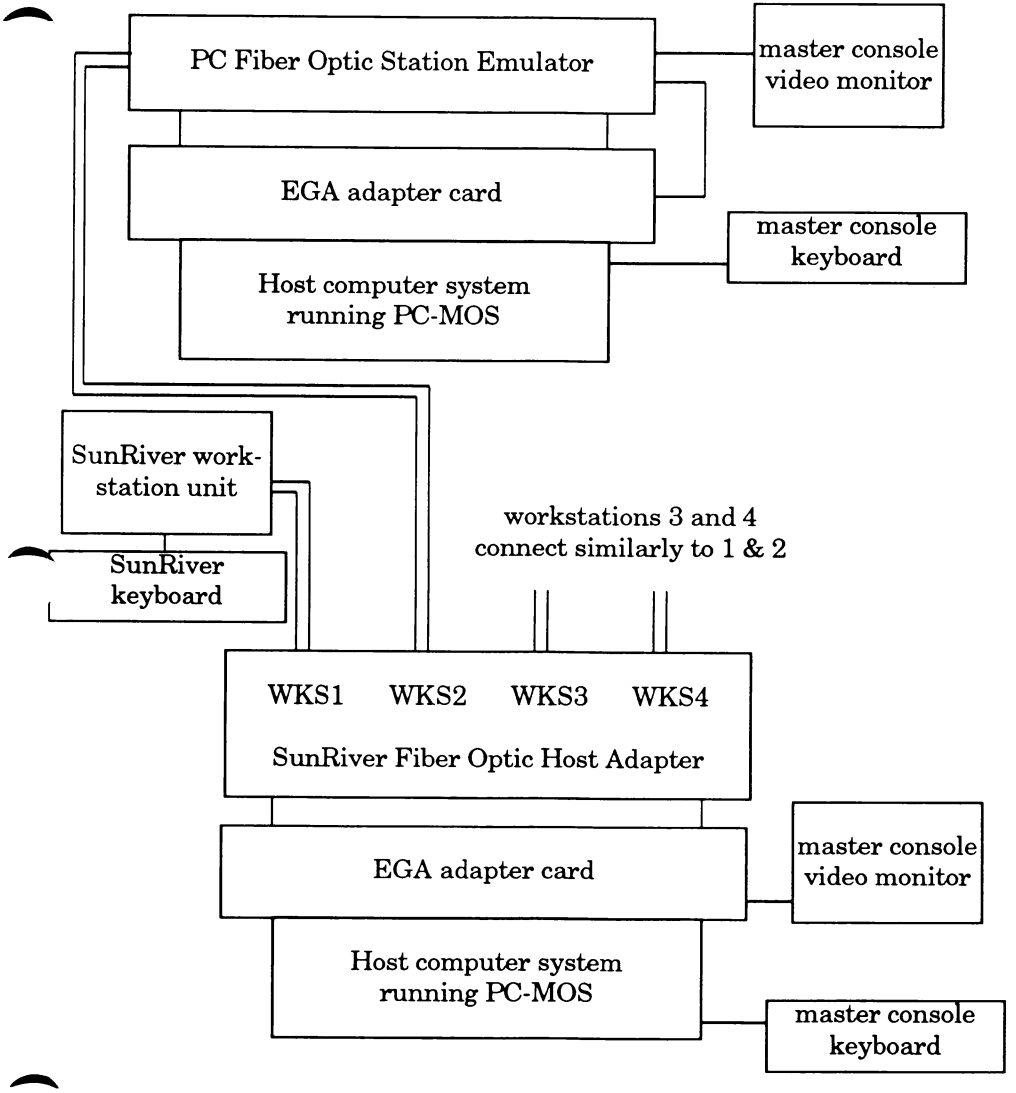
PC Fiber Optic Station Emulator
master console video monitor
EGA adapter card
master console keyboard
Host computer system running PC-MOS
SunRiver work-station unit
SunRiver keyboard
workstations 3 and 4 connect similarly to 1 & 2
WKS1 WKS2 WKS3 WKS4
SunRiver Fiber Optic Host Adapter
EGA adapter card
master console video monitor
Host computer system running PC-MOS
master console keyboard

**Figure 6 - 6**

## Adding Background Tasks

When the ADDTASK command is used without any terminal driver being specified, the task spawned is what is referred to as a background task. The only required command line parameter for ADDTASK in this case is the memory size. Including a task number, security class designator and/or start up batch file name is optional. Such a task will begin running without any physical console being associated with it, and the startup batch file for such a task may invoke one or more application programs.

Processes invoked in this manner will run in an unwatched state, just as if an actual console was attached, due to the support of MOS's virtual machine strategy. The only exception to this would be if an unwatched application was invoked in an EGA or VGA specific graphics mode. In this event, the task would be suspended until a compatible workstation was attached to it.

If your system installation will always involve the use of more than one task, the ADDTASK commands may be placed in your AUTOEXEC batch file. If you find a periodic need for an additional task to perform a specific function, it can be useful to setup a pair of batch files. For example, the file DOSORT.BAT could contain an ADDTASK command which specifies the startup batch file DOSORT2.BAT. DOSORT2.BAT would then contain the batch file instructions required to cause the new task to invoke a database sorting operation. Note that it is possible for the startup batch file to end with the command "REMTASK *" which would cause this background task to delete itself when it had completed its function.

## Adding Workstation Tasks

When the ADDTASK command is invoked with a parameter known as a terminal device driver, the result will be the creation of a workstation task. Each type of workstation has a corresponding device driver and each driver also requires one or more additional parameters following its name. This additional information is necessary to specify which workstation is to be activated. The exact syntax of these additional parameters are dependent on the particular terminal device driver used.

Following the PCTERM driver, a port number and baud rate must be specified. The port number must match the serial port to which the workstation is connected and the baud rate must match the speed to which the workstation will be configured. In the case of a VNA workstation, only a workstation unit number needs to follow the device driver name in the ADDTASK parameter line, since there is no such thing as a baud rate with VNA.

The function of a terminal device driver is to insulate the MOS kernel from the peculiar details of interfacing with each type of workstation. When its time for a display update, the MOS kernel can "know" that it needs to update the character at column 4 of row 10. Knowing the particular cursor addressing command sequence that must be used to get the job done is the function of the terminal driver.

## Startup Batch Files

When adding a task, the name of a startup batch file may be included on the ADDTASK command line. This can provide for the automatic definition of the new task's operating characteristics such as the command search path, the PROMPT string, and the drive and directory the user will be in. Though some aspects of batch files will be covered here, please refer to Chapter 4 for more detail.

Basically, anything you would do within an AUTOEXEC batch file can be done within the startup batch file for a new task. In fact, through the use of a special batch file variable, %A, it is possible to use the same AUTOEXEC file that the foreground uses as the startup file for any new background or workstation task. This special batch language variable allows a batch file to determine the number of the current task.

Listing 6-1 shows an example of this where a group of operations common to all tasks is done first, followed by any specific operations required for task 0, the foreground task. There is also, in this example, a group of commands which is only executed for any task other than the foreground. Through careful planning, unique action could be taken for each specific task number involved in your system.

### Listing 6-1

Shown here is an AUTOEXEC.BAT which can be used both for the booting of the foreground task as well as the spawning of background and terminal tasks. After the label :startbg you would add any instructions required to configure your additional tasks.

```
.batecho off
.dot off
envsize 384
insert
path c:\batch;c:\mosfiles;c:\utils
prompt [$i $P]
if not %a == 00 goto startbg
rem ================== commands specific to the foreground partition
addtask 512,,,autoexec,vna,1
abort
rem ================== commands specific to background partitions
:startbg
```

When a startup batch file is to be used with the ADDTASK command, only the file's name need be specified. The extension of .BAT is assumed, and the file must be located in the root directory of the current drive.

Since the specification of a task's environment size is usually something done within a startup batch file (using the "ENVSIZE n" command), you should note that when a batch file is called from within another batch file, a ceiling on the environment's size is in effect for the duration of the nested call. For this reason, do not place the ENVSIZE command within a common batch file that will be called in a nested manner. This command must be placed in the first level batch file: AUTOEXEC.BAT for the foreground, or the batch file named in the ADDTASK parameter line for background tasks.

In cases where a startup batch file other than AUTOEXEC must be used, but it would be advantageous to include a call to AUTOEXEC within it to set up common system parameters, you can simply preface the nest call with an ENVSIZE statement. This will make the ENVSIZE statement within the AUTOEXEC file redundant but it will guarantee the environment's minimum size.

One aspect of a task's operating environment which is typically defined within a startup batch file is the command search path. When setting up a command search path, it is important to order the entries within it by the frequency of use. When MOS's command processor uses this list to search for a command or batch file, it does so by searching each of the directories specified in your PATH statement. If the application you use the most is in a directory near the end of a long PATH statement, the program load time will suffer. Another approach to speeding the loading of often used programs is to create a batch file which will start that application with an explicit path. This can also keep your command search PATH statement from getting too long.

Another environment string which is often defined within a startup batch file is the command prompt. In a PC-MOS system with multiple tasks it can be especially useful to include a $I within your PROMPT specification. This will cause the task number to appear within the command prompt as a constant reminder of where you are. In addition, if you are using MOS's security features, including a $U will cause the name of the currently logged-on user to appear in the command prompt. There are many other useful specifications which may be included in a prompt -- refer to the PC-MOS User's Guide for more details.

## PAMswitching

It has already been mentioned in previous discussions that the task which a workstation is currently viewing is not a fixed assignment. A workstation may be detached from its current task and re-attached to another task providing that the workstation is compatible with the video mode of the new task and that MOS's security feature has not been configured to prohibit the switch. The term "PAMswitching" has come to be associated with switching a console from one task to another. It is derived from the phrase "Partition Access Method".

It is possible, within a PC-MOS system, for more than one console to be watching the same task at the same time. In this situation, you will notice some degradation in the video update speed since each console watching must be updated for each new character displayed. If the ability for one user to be able to watch what others are doing is unacceptable, MOS's security system may be used to selectively control PAMswitching. Refer to Chapter 10 for more details on security.

In order to invoke a PAMswitch, hold down the ALT key while you press the number of the new task on the numeric keypad. When the Alt key is released, the switch will occur. This alternate use of the numeric pad can be suppressed if you need to use such keystrokes to enter extended character codes. By holding down the ALT key and then pressing 999 (on the numeric keypad), this special use of the keyboard can be toggled on and off.

An alternative way to invoke a PAMswitch is to use MOS's SWITCH command. The number of the task to switch to is entered after the keyword SWITCH. This feature makes it possible to create a batch file menu which allows the use of more descriptive names. An example of such a menu is presented in listing 6-2. See Chapter 4 for more information on the KEY command.

**Listing 6-2**

SW.BAT - an example of using the SWITCH command within a batch file menu:

```
echo The current task is %a
text

    0.  Foreground

    1.  Mary's workstation

    2.  The PRINT.COM partition

endtext
key %%a in (0-2) do switch %a
```

One peculiarity that must be noted concerning the SWITCH command is that when it is invoked within a task that currently has more than one workstation switched in to watch, all workstations involved will be switched to the new task. This is because there is no way for MOS to infer the intent. To illustrate this more clearly, consider the case where one user enters the switch keyword and another enters the task number. Which one should be switched?

## Task Switching

One primary function of a multitasking system is the scheduling and switching of tasks -- determining when each task is activated and for how long. MOS is what is referred to as a preemptive multitasker meaning that an application can be switched in and out at virtually any time. The task switching process happens in a manner which is transparent to whatever applications are running (with the exception of certain time critical programs). It is not up to the application to explicitly release the processor as in some multitasking environments.

To effect a task switch, MOS suspends the operation of the current task and records all necessary information so that this process can be resumed at a later time. Then, MOS determines which task should be activated next and gives it its share of execution time. In a system with a paging-capable memory management feature, a task switch basically involves remapping the new task's TPA memory into place along with the associated context and video buffers. When there is no memory management each task's TPA and video buffer is already located within the base memory area so only the context buffer needs to be updated. In this case the current context area must be copied into an SMP save buffer and then the new task's context copied into place from its save buffer.

A task switch can occur when a system call is made to the MOS kernel for services such as reading or writing data from a console or disk. A task switch can also occur from what is known as a timer tick interrupt. This is an event which occurs approximately 18 times a second from a clock circuit built into the computer.

At these times the computer sets aside whatever it is doing and processes a time counting routine. In addition, the MOS kernel also performs certain other activities such as task switching and workstation display updating. However, not every timer tick interrupt will cause a task switch. Sometimes, when the timer interrupt occurs, the machine is in a critical state where a task switch cannot be done.

Another term often used to describe the type of multitasking done in PC-MOS is time slicing. The period of time from one timer tick to the next, roughly 1/18th of a second, is referred to as one slice of execution time. When MOS is first booted the foreground task will, of course, receive all time slices. As new tasks are added, the initial default value used for slicing is one which means that each task will receive at least one slice of execution time before being suspended in favor of the next task.

Through the use of the CONFIG.SYS statement SLICE=n the system default slice count may be increased from its default of 1. The effect of this is global in that each new task added, and the foreground task, will inherit this slice value. It is also possible to adjust the time slice count on a per task basis with the command "MOSADM SLI n". These two methods may be used independent of one another or in combination.

As a larger time slice value is given to each task, more of the computer's time will be spent actually processing each user's program as opposed to spending time performing the system overhead involved in task switching. However, there is a practical limit to increasing the slice count since the longer that each task is active, the longer that all other tasks will be suspended while waiting their turn.

In a system where serial terminal workstations are being used, where there is already some delay due to the serial communications speed, an intelligent serial card can improve the overall response time and possibly compensate somewhat for the effects of a larger time slice. Refer to Chapter 8 for more information on serial communications options.

As stated above, MOS switches from one task to the next roughly 18 times per second. This is because the timer chip within a PC/AT type of computer is designed to interrupt the processor at this rate. These timer ticks are responsible for almost all time keeping processes which go on in a system. One consequence of timer ticks is that they do have an effect on overall system throughput. Eighteen times a second, the CPU is forced to set aside whatever it is doing and handle certain system overhead processes. The time required by these different processes is, of course, kept as small as possible and, at only 18 times per second, it represents a fairly insignificant amount of overhead.

Another consequence of timer ticks, and their 18 tick per second frequency, is that the granularity of time slicing can be somewhat coarse. When a system is supporting several busy tasks, video display updating can occur with a noticeable jumpiness. This will be most noticeable when a display is scrolling continuously, as in a directory listing, or when drawing with a mouse.

In order to smooth out this video staccato effect MOS provides a means whereby the system timer's frequency can be increased. When this is done, MOS applies an internal compensation factor so any applications which use clock ticks to measure time will not be affected. This feature was introduced with release 4.00 of MOS.

A subcommand of the MOSADM.COM utility called TMFACTOR is used to change or review this slice rate multiplication factor. By entering MOSADM TMFACTOR 8 the timer tick frequency, and, therefore the task switching rate, will be 8 times faster -- or roughly 144 times per second (8 times 18). By simply entering MOSADM TMFACTOR with no number, the current value of this speed-up factor will be displayed. To reset slicing to the default rate, use a time factor of 1. The allowable range of time factor values is from 1 to 40.

Again, the primary use for this command is to improve the visual effects of task switching. Time factor values in the range of 4 to 10 should be suitable for most situations. Using a value larger than necessary will only reduce your systems throughput since you are introducing certain portions of the time slice system overhead more often (It makes no sense, of course, to use a faster slicing rate in a single tasking situation). The SLICE=n directive and MOSADM SLI n command still effect the relative time slice balance, but their count value represents the number of faster slices when MOSADM TMFACTOR n is used.

In normal operation, when all tasks are busy in the execution of user programs, each task is receiving its allotted share of time slices before being switched out in favor of another task. There are, however, times when a task may be explicitly suspended before its slice of time has expired. This means that the operating system immediately puts the current task on hold and searches for the next task to activate. Some of the times this may happen are when the program currently running is waiting for an event such as a keystroke from the user, when data is being sent to a printer which is momentarily busy, or when an EGA graphics mode has been entered and no physical console is currently attached.

Through MOS's priority control feature, the user may influence the way in which MOS selects tasks for execution. As soon as the current task is put to sleep, the next task in priority order will be tested to see if it can be activated. A task's availability for activation depends on whether it is suspended due to some condition that has not yet been resolved, such as when a task is idle waiting on a keystroke.

MOS uses a relative priority numbering scheme which means that if all tasks have a priority of 2, there will be no favoring of one task over another. Likewise, if all tasks have a priority of 7 there will be no preferential treatment. However, if one task's priority value is 1 and another's is 3 and yet another's is 5, the task with the highest priority value will always be tested first when a task switch is to be done.

## Securing Tasks and Remote Tasks

A final pair of topics that bears mention in this chapter is the setup of remote tasks and secured tasks. A remote task is one where a serial terminal type of workstation is connected to the host system through a modem and a secured task is one which can only be accessed by entering a password. You can certainly implement MOS's security system on tasks other than remote ones and you could setup a remote task without using security. However, unless the transmission media which connects the modems is guarded in some other fashion common sense dictates the use of security with remote tasks. These topics are more appropriately detailed in Chapters 8 and 10.