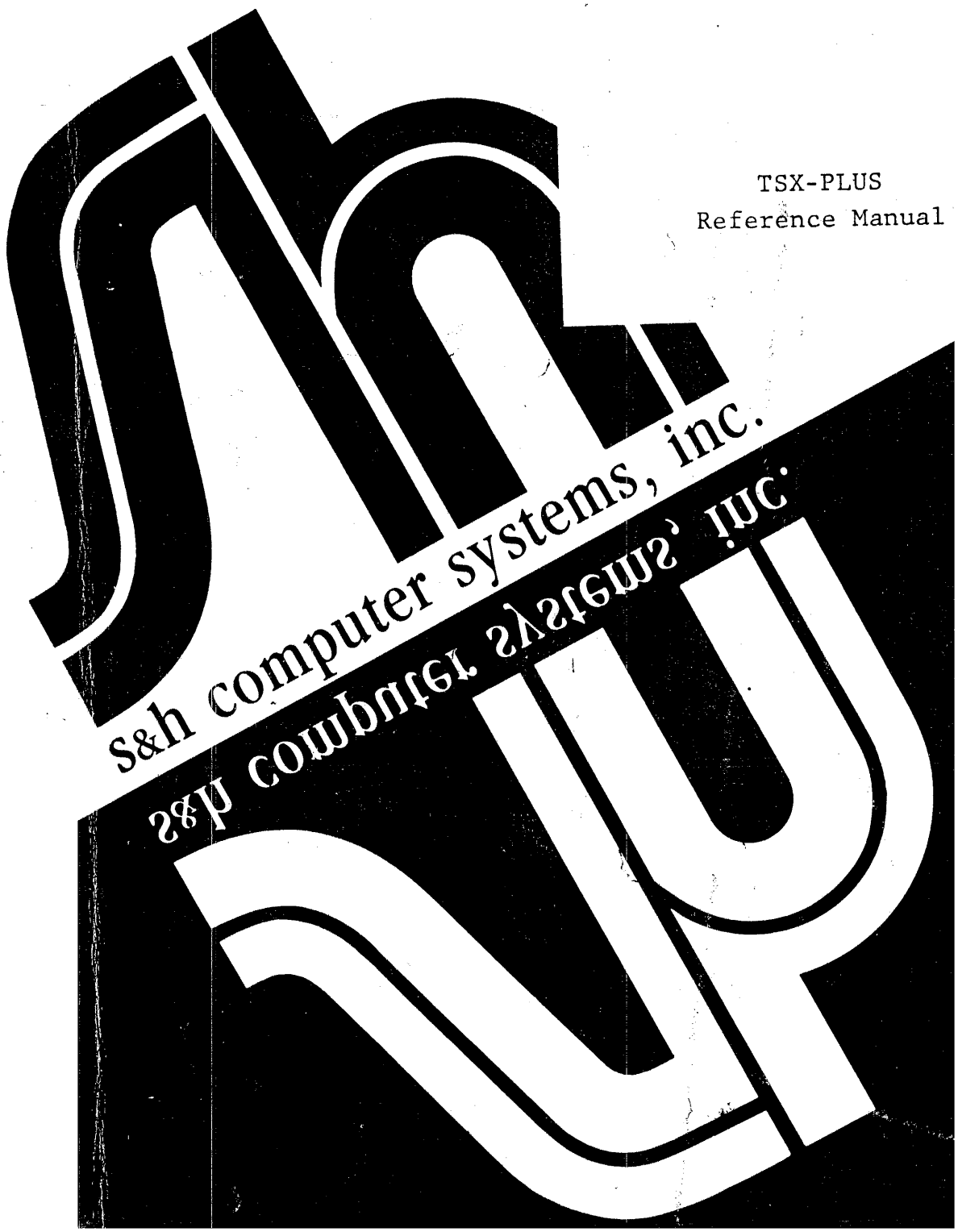

TSX-PLUS
Reference Manual

TSX-PLUS
Reference Manual

First printing -- December, 1980

S&H Computer Systems, Inc.
1027 17th Avenue South
Nashville, Tennessee 37212
USA
(615)-327-3670

Contents

1. INTRODUCTION

TSX-Plus is a high-performance, general purpose, time-sharing operating system for Digital Equipment Corporation PDP-11 and LSI-11 computers. TSX-Plus provides the functionality of the DEC RT-11 operating system to up to 20 concurrent time-sharing users.

TSX-Plus overlaps terminal interaction time, I/O wait time and CPU execution time for all jobs on the system. The result is a tremendous increase in the productivity of the computer system.

TSX-Plus was designed from the ground up to efficiently support a wide variety of RT-11 programs. TSX-Plus contains no RT-11 emulator as some other time-sharing systems do; rather, TSX-Plus supports RT-11 system service calls as its basic mode of operation. The result is lower system overhead and substantially improved performance over systems that must emulate RT-11 services. Most RT-11 programs can be used with TSX-Plus without change or even having to be relinked. TSX-Plus interfaces with standard RT-11 device handlers and supports RT-11 utility programs such as PIP, DIR, and DUP. The TSX-Plus keyboard commands are an extended set of those provided by RT-11. TSX-Plus fully supports CCL commands such as COMPILE and EXECUTE.

TSX-Plus is a general purpose time-sharing system. It can simultaneously support a wide variety of jobs and programming languages including COBOL-Plus, FORTRAN, BASIC, DIBOL, PASCAL, APL, MACRO, TECO and KED. TSX-Plus is in use in educational, business, scientific and industrial environments. It can concurrently support commercial users doing transaction processing, engineering users performing scientific processing, and system programmers doing program development.

In addition to the basic RT-11 functionality, TSX-Plus provides extended features such as a transparent line-printer spooling system; a shared file record locking facility; an inter-job message communication facility; a program performance monitor system; command files with parameters; and a logon and usage accounting system.

TSX-Plus will run on any PDP-11 or LSI-11 computer that has memory management facilities and at least 96Kb of memory. The system must also have a disk suitable for program swapping (the swapping disk can also be used for regular file storage as well). Time-sharing lines can be connected to the system through DL-11 or DZ-11 communication devices. Both hard-wired and dial-up lines are supported by TSX-Plus.

---

*DEC, RT-11, CTS-300, DIBOL and PDP-11 are trademarks of Digital Equipment Corporation.
**Unless otherwise indicated, all information applies to RT-11 and CTS-300.

2. BASIC OPERATION

2.1 Logging On

When TSX-Plus is generated, lines can be set up so that they are automatically initiated when TSX-Plus is started. If this is done, the greeting message "TSX-Plus version x.y" will be printed at each terminal when TSX-Plus is started. Lines which are not generated with the automatic start feature must be initiated by pressing carriage return at the terminal.

If TSX-Plus was generated requiring log on authorization, the message "Logon please:" will be printed. The user should respond by typing a project number, a comma, and a programmer number followed by carriage return. TSX-Plus will then request the password. For security reasons the password is not printed at the terminal as it is typed in. After the project-programmer numbers and password are entered, TSX-Plus checks them for validity and, if valid, types the message "Welcome to the system". TSX-Plus then either types a period indicating it is waiting for a command or executes a start-up command file that was specified by the system administrator.

A start-up command file contains initialization commands for a line. It can end by either leaving the line waiting for a system command (a period is printed) or by starting execution of some program. It is possible for a command file to lock a program to a line so that if execution of the program is terminated the line is automatically logged off. Any characters typed at the terminal during the execution of a start-up command file are ignored. Typing control-C will not abort a start-up command file.

The following example illustrates a typical log-on sequence. The information typed by the user is underlined.

```
(carriage return pressed)
     TSX-Plus version 1.5
     26-Sep-80  11:45:27
     Logon please:413,2107
     Password:MYPASS(password entered but not printed)
     Welcome to the system
     .(TSX-Plus is now waiting for a system command)
```

A user may adopt a new password while logging on. To do this, enter a slash and the new password immediately after typing the old password. The new password must then be used for future logons. Passwords may be from 1 to 7 characters in length and must be composed only of letters and digits. The following example shows the password being changed from "OLDP" to "NEWP". Note that neither the old nor the new password is echoed.

```
(carriage return pressed)
TSX-Plus version 1.5
26-Sep-80   4:17:43
Logon please:107,24
Password:OLDP/NEWP
Welcome to the system
.
```

If the TSX-Plus system is generated so that logon account authorization is not required, then instead of saying "Logon please:" it will either start executing a start-up command file or will simply print a period and wait for a system command to be entered.

The OFF system command (described below) is used to log off a timesharing line.

2.2 Control Characters

The following characters may be used to control terminal operations. They are entered by holding down the "CTRL" key while pressing the selected character.

CTRL-U - Deletes the current input line.

CTRL-O - Causes suppression of program output to the terminal until one of the following conditions occurs:

1) A second CTRL-O is typed
2) A return to monitor occurs
3) The running program issues a .RCTRLO EMT.

CTRL-S - Temporarily suspends output to the terminal until CTRL-Q is typed. CTRL-S has no special effect if SET TT NOPAGE has been used.

CTRL-Q - Resumes printing on the terminal from the point at which printing was previously suspended by CTRL-S or end of page (see SET TT LENGTH command). CTRL-Q has no special effect if SET TT NOPAGE has been used.

CTRL-R - Causes the current characters in the TTY input buffer to be typed out. This can be used to check the actual status of an input line when rubout editing has been done on the line.

CTRL-C - Interrupts the current program and returns control to the keyboard monitor. If the running program is not waiting for input, two successive CTRL-C's are required to interrupt its execution.

CTRL-Z - Indicates end-of-file when input is being read from device "TT".

CTRL-I - Equivalent to TAB.

CTRL-W - Used to switch to a TSX-Plus virtual line. (See chapter on virtual lines.)

2.3 Keyboard Commands

The TSX-Plus command interpreter is somewhat more powerful and flexible than the standard RT-11 command interpreter. When a system command line is typed in, TSX-Plus performs the following steps while trying to interpret it:

1. If the command line begins with an at sign ("@"), TSX-Plus interprets the command as a command file execution. If no device is specified with the command file name, device DK: is assumed. See the chapter on command files for further information on them.

2. TSX-Plus next attempts to identify the command as a standard system command such as COPY, RUN, EXECUTE, etc. Commands may be abbreviated to the minimum number of characters that uniquely specify the name. Thus "COP" would be an acceptable abbreviation for COPY, but "CO" would not because it could mean COPY or COMPILE. "COPX" also would not be identified as a system command.

3. If the command cannot be identified as a standard system command, TSX-Plus tries to find a command file on device "DK:" that has the same name as the command keyword. If no such command file is found on "DK:", TSX-Plus looks for the command file on device "SY:". If such a command file is found, its execution is started. When a command file is started in this fashion (rather than explicitly specifying an at-sign before its name), the command file listing is suppressed as if an implied "SET TT QUIET" command was executed during command startup. This implied listing suppression is temporary in effect and applies only to the command file started in this fashion. See example 5 below.

4. If a system command file cannot be found, TSX-Plus looks for an executable program (SAV file) on device "SY:" that has the same name as the command keyword. If such a program is found its execution is started as if there were an "R" command in front of the program name. Note that RT-11 and TSX-Plus allow a line of text input to be passed to a program by specifying it on the RUN command after the program name. TSX-Plus also allows this to be done with the "R" command as well as with an implied "R" command line when the program name is specified as the command keyword. See example 6 below.

Examples:

1. Run a program named "DUMP" on device "SY:".

```
.R DUMP
```

2. Run a program named "PAYROL" on device "DX1:".

```
.RUN DX1:PAYROL
```

3. Execute a command file named "PURGE" on device "DK:".

```
.@PURGE
```

4. List all files on "RK1:"

```
.DIR RK1:
```

5. Start a command file named "LOADIT" on "SY:" and pass it the parameter string "PROG2".

```
.LOADIT PROG2
```

6. Execute the program "PIP" on "SY:" and pass it the input line "A.TMP=B.TMP".

```
.PIP A.TMP=B.TMP
```

2.3.1 The "ACCESS" Command. The ACCESS command is used to restrict user file access to a particular set of files or devices. Refer to the TSX-Plus System Manager's Guide for further information about this command.

2.3.2 The "ASSIGN" Command. The RT-11 ASSIGN command is used to associate a logical I/O device name with a physical device. A frequent use is to assign FORTRAN I/O unit numbers to selected devices. The TSX-Plus ASSIGN command provides this facility in an identical fashion and also offers a useful extension. In addition to being able to specify a physical device name, the user may specify a file name, extension, and size.

To simply assign a logical device name to a physical device, the form of the ASSIGN command is the same as that under RT-11. For example, to assign the logical device name "BIN" to physical device "DX1" the command would be:

```
ASSIGN DX1 BIN
```

To assign FORTRAN I/O unit number 1 to the teletype, the command would be: ASSIGN TT 1

If a file name and optional size is to be specified in addition to the physical device name, the file name and size follow the device name. For example, to assign FORTRAN I/O unit number 1 to a file named "PAYROL" on device "DX0" with a size of 43 blocks, the following command could be used:

```
ASSIGN DX0:PAYROL[43]=1
```

The following command would assign logical device "BIN" to a file named "PROG1" on the system device:

```
ASSIGN SY:PROG1=BIN
```

It is also possible under TSX-Plus to assign a new logical name to a previously assigned logical name. The effect is to assign the new logical name to the same physical device that the previous assign was directed to. For example, the following sequence of ASSIGNs result in both logical devices "AA" and "BB" being assigned to "DL1".

```
ASSIGN DL1 AA
ASSIGN AA BB
```

A maximum of fifteen assignments may be in effect at any given time for each user.

2.3.3 The "BOOT" Command. The BOOT command causes TSX-Plus to abort all currently running jobs and to reboot RT-11. Unlike the RT-11 BOOT command no device or file name may be specified with the TSX-Plus boot command -- it always reboots from the system (SY) device. Operator command privilege is required to use this command. The BOOT command is functionally equivalent to the $STOP command.

2.3.4 The "COBOL" Command. The COBOL command is used to cause the COBOL-Plus compiler to compile a COBOL source program. The default extension for COBOL source programs is "CBL", the default extension for COBOL object files is "CBJ". The COMPILE, LINK and EXECUTE commands may also be used to compile and/or execute COBOL programs as TSX-Plus will invoke the COBOL-Plus compiler and CBLINK link program if the source program specified to compile has the extension "CBL" or the object program specified to link has the extension "CBJ". Switches that can be used with the COBOL command are listed below.

Switch	Meaning
/CROSS	Produce a cross-reference of the source program.
/CREF	(Equivalent to /CROSS).
/ONDEBUG	Compile the program for use with the interactive debugger.
/PRODUCTION	Omit line number tracing and subscript checking code.
/ANSI	Produce warning messages for any non-ANSI standard feature
/LIST	Produce a source program listing.
/NARROW	Format the cross-reference for 80 column display.
/INFORMATION	Print additional information at end of compilation.
/CARD	Source program is in card sequence format.
/SEQUENCE	(Equivalent to /CARD).
/SUMMARY	Print only error messages on listing device.
/OBJECT:name	Specify name of object file.
/ALLOCATE:size	Specify size of object file.

See the COBOL-Plus reference manual for further information about this command.

2.3.5 The "COMMENT" Command. This command may be used to place comments inside command files. The form of the COMMENT command is:

```
COMMENT comments
```

where "comments" may be any string of characters.

2.3.6 The "COMPILE" Command. The COMPILE command invokes the appropriate language processor to compile the specified source file. The TSX-Plus COMPILE command is the same as the RT-11 COMPILE command except that it also recognizes programs with the extension "CBL" as COBOL source programs and calls the COBOL-Plus compiler. When compiling a COBOL program, the switches that are legal with the COBOL command may also be used with the COMPILE command. It is also possible to explicitly specify that the COBOL-Plus compiler is to be called by using the "/COBOL" switch with the COMPILE command.

2.3.7 The "COPY" Command. The TSX-Plus COPY command has the same form and options as the RT-11 COPY command.

2.3.8 The "DATE" Command. The TSX-Plus DATE command has the same form and options as the RT-11 DATE command. Operator command privilege is required to set the date.

2.3.9 The "DEASSIGN" Command. The TSX-Plus DEASSIGN command is identical to the RT-11 DEASSIGN command. It is used to disassociate a logical unit number assignment.

2.3.10 The "DELETE" Command. The TSX-Plus DELETE command has the same form and options as the RT-11 DELETE command.

2.3.11 The "DETACH" Command. The DETACH command is used to initiate execution of a command file as a "detached" job, to abort a detached job or to check the status of a detached job. The use of this command requires the user to be authorized to use detached jobs.

The form of the command used to start a detached job is:

```
.DETACH file
```

where "file" is the name of a command file which is to be started as a detached job. If a free detached-job line is available the file will be started and a message will be printed indicating on which line the detached job was started. Detached-job lines must be declared when TSX-Plus is generated.

In the following example a command file named "CRUNCH" is started as a detached job.

```
.DETACH CRUNCH
Job started on line #5
```

The form of the DETACH command used to abort a detached job is:

```
.DETACH/KILL line-number
```

where "line-number" is the number of the detached-job line where the job is executing.

The form of the DETACH command used to check the status of a detached job line is:

```
.DETACH/CHECK line-number
```

In response to this command TSX-Plus will indicate if a job is still executing on the line.

See the chapter on virtual lines and detached jobs for further information.

2.3.12 The "DIBOL" Command. The DIBOL command is used to compile a DIBOL source program. The TSX-Plus DIBOL command has the same form and options as the RT-11 DIBOL command.

2.3.13 The "DIFFERENCES" Command. The DIFFERENCES command is used to compare two source or object files. The TSX-Plus DIFFERENCES command has the same form and options as the RT-11 command.

2.3.14 The "DIRECTORY" Command. The TSX-Plus DIRECTORY command has the same form and options as the RT-11 DIRECTORY command.

2.3.15 The "DISMOUNT" Command. The DISMOUNT command is used to tell TSX-Plus that it should stop doing directory caching on a particular device. The form of the DISMOUNT command is:

```
DISMOUNT ddu
```

where "ddu" is the name of the device. The only effect of the DISMOUNT command is to cause TSX-Plus to stop doing directory caching for the device. Files on the device may still be accessed after the DISMOUNT command is issued. When one user issues a DISMOUNT command, the device is dismounted for all users. An automatic dismount is done when an INIT or SQUEEZE command is done on a device.

2.3.16 The "DISPLAY" Command. This command can be used within a command file to cause a line of text to be displayed on the terminal when the command is executed. This is useful in command files that are not being listed to keep track of the progress through the command file. The form of the DISPLAY command is:

```
.DISPLAY comments
```

where "comments" can be any text string to be displayed on the terminal.

2.3.17 The "DUMP" Command. The TSX-Plus DUMP command has the same form and options as the RT-11 DUMP command.

2.3.18 The "EDIT" Command. The TSX-Plus EDIT command has the same form and options as the RT-11 EDIT command.

2.3.19 The "EXECUTE" Command. The TSX-Plus EXECUTE command has the same form and options as the RT-11 EXECUTE command. It will also recognize programs with the extension "CBL" as COBOL source programs and will invoke the COBOL-Plus compiler and linker.

2.3.20 The "FORM" Command. The FORM command is used to specify the default form name to be used by subsequent spool files generated by the user. The form of the FORM command is:

FORM name

where "name" is the one to six character default form name to be used for all spool files generated by the user until another FORM command is issued. See section 4.3 for further information. The initial default form name for each user is "STD".

In the following example a FORTRAN listing will be generated for printing on a form called "2-PART".

```
.FORM 2-PART
.COMPILE/LIST TEST
.FORM STD
```

2.3.21 The "FORTRAN" Command. The TSX-Plus FORTRAN command has the same form and options as the RT-11 FORTRAN command.

2.3.22 The "HELP" Command. The TSX-Plus HELP command has the same form and options as the RT-11 HELP command.

2.3.23 The "INITIALIZE" Command. The TSX-Plus INITIALIZE command has the same form and options as the RT-11 INITIALIZE command. If the MOUNT command was used to mount the device being initialized, the device is automatically dismounted before the initialization takes place.

2.3.24 The "KILL" Command. The KILL command can be used to kill a timesharing job. This has the effect of aborting the execution of the job and forcing the logoff of the line. Operator command privilege is required to use the KILL command. The form of this command is:

```
.KILL line-number
```

where line-number is the number of the job to be killed.

2.3.25 The "LIBRARY" Command. The TSX-Plus LIBRARY command has the same form and options as the RT-11 LIBRARY command. It also can be used to build COBOL-Plus object program libraries; see the COBOL-Plus reference for further information.

2.3.26 The "LINK" Command. The TSX-Plus LINK command has the same form and options as the RT-11 LINK command. It will also recognize object files with the extension "CBJ" as COBOL-Plus object files and will invoke the COBOL-Plus link program (CBLINK).

2.3.27 The "MACRO" Command. The TSX-Plus MACRO command has the same form and options as the RT-11 MACRO command.

2.3.28 The "MEMORY" Command. The MEMORY command is used to control the amount of memory made available to a job. When a job initially logs on it receives a default memory size allocation that was set by the system manager. The MEMORY command can be used to change the allocation for the job. The form of the MEMORY command is:

MEMORY nn

Where "nn" is the number of k-bytes of memory to be allocated for the job. The maximum memory size that a job may expand to is set by the system manager but is never greater than 56. When a running program performs a .SETTOP emt the top of memory address corresponds to the size last specified by a MEMORY command. Note that .SETTOP emt's do not actually affect the amount of memory allocated to a job -- only the MEMORY command does that.

If the MEMORY command is entered without specifying a size, the current memory allocation for the job is displayed. See also the description of the SHOW MEMORY command.

2.3.29 The "MONITOR" Command. The MONITOR command is used to cause TSX-Plus to begin doing a performance analysis. See chapter 9 for complete information about the TSX-Plus performance analysis feature. The form of the MONITOR command is

MONITOR base-address,top-address[,cell-size]/switches

where "base-address" is the lowest address in the program region being monitored, "top-address" is the highest address in the region, "cell-size" is the number of bytes to group per histogram cell. The only available switch is "/I" which, if specified, causes I/O wait time to be included in the analysis.

2.3.30 The "MOUNT" Command. The MOUNT command is used to tell TSX-Plus that a file-structured device is to have its directory cached. Directory caching is a technique that speeds up file lookups by keeping information about files in memory so that it is not necessary to access the directory on the device each time a file is opened. The form of the MOUNT command is:

```
MOUNT ddu
```

where "ddu" is a device name such as "DL1". The only effect of the MOUNT command is to tell TSX-Plus that it should do directory caching for the device being mounted. If directory caching is not wanted it is quite possible to mount a device and access files on it without using the MOUNT command. Once a MOUNT command is issued, directory caching is enabled for all users who access files on the device. If the MOUNT command is used to cause TSX-Plus to do directory caching for a device, it is crucially important that the DISMOUNT command (see below) be used to dismount the device before a new disk is mounted on the same drive. If a new pack is mounted without telling TSX-Plus, it would try to access files on the new pack according to their positions on the old pack whose directory information is in its directory cache.

Directory caching has a dramatic speed effect on file lookups but does not speed up file enters, deletes or renames. This is because TSX-Plus always updates the directory on the device when it is altered. The maximum number of devices whose directories may be cached and the number of file entries that are kept in the directory cache are specified when TSX-Plus is generated. The system disk is always cached and need not be mounted.

Example:

```
MOUNT DL2
```

2.3.31 The "OFF" Command. The OFF command is used to log off a terminal. It is also used to release a virtual line (see the discussion of virtual lines on page 17). On dial-up lines, an automatic log off is performed by TSX-Plus if the telephone connection is broken. The accumulated connect time and CPU time used during the session are printed during the logoff processing.

Example:

```
.OFF
Connect time=01:43:00   CPU=00:12:03.7
```

2.3.32 The "OPERATOR" Command. The OPERATOR command is used to send a message to the terminal which was specified to be the operator's console when TSX-Plus was generated. The OPERATOR command works like the SEND command, but it is not necessary to know the line number of the operator's terminal. The form of the OPERATOR command is:

OPERATOR message

Example:

Send a disk mount message to the operator:

OPERATOR PLEASE MOUNT PAYROLL MASTER DISK ON RK1

2.3.33 The "PAUSE" Command. The PAUSE command is used within command files (see chapter 9) to cause a pause in processing the file. The form of the PAUSE command is:

PAUSE comments

where "comments" may be any string of characters. When a PAUSE command is encountered within a command file, the PAUSE command is printed on the terminal followed by ">>". Execution of the command file is then suspended until carriage return is pressed. This gives the operator an opportunity to perform manual operations such as mounting disks or tapes.

2.3.34 The "PRINT" Command. The TSX-Plus PRINT command has the same form and options as the RT-11 PRINT command.

2.3.35 The "R" Command. The "R" command is used to start execution of a program. The default device which is searched for the program is "SY:"; a different device name may be specified with the name of the program. A line of input may be passed to the program by specifying it as part of the "R" command following the program name. If this is done the program will receive the text string as its first line of input and will receive control-C as its second line of input.

The /LOCK switch may be specified following the "R" command keyword. If specified, this switch causes the program that is being started to be "locked" to the timesharing line so that when the program exits the line is automatically logged off. If the "R/LOCK" command occurs within a command file the command file is terminated as the program is started and any additional information in the command file is ignored. The most frequent use of this feature is in start-up command files where a line is to be restricted to accessing a particular program. If a locked program chains to another program, the program that was chained to then becomes the locked program. See example 4 below.

The /DEBUG switch may be used with the "R" or "RUN" commands to cause the program being started to be executed under control of the TSODT debugging program. If the /DEBUG switch is specified, a relocatable copy of the TSODT program ("SY:TSODT.REL") is loaded into the upper most portion of the available memory space (reducing the memory space available to the running program by about 4Kb). The program being started is then loaded into the low memory space below TSODT and control is passed to TSODT. TSODT responds by printing a greeting message and waiting for a command from the terminal. At this point register 0 ("$0") contains the address of the starting point of the program. TSODT may be used to display or examine locations in the program being started or set breakpoints. The program being run is then started by using the TSODT "xxxxx;G" command (where "xxxxx" is the starting address of the program). The /DEBUG switch allows TSODT to be used to debug a program without having to link the program with TSODT.

The /SINGLECHAR switch may be used with the "R" and "RUN" commands to cause the program being run to execute in "single character activation" mode. Normally when programs are run under TSX-Plus they do not receive terminal input until an "activation" character such as carriage-return has been entered. This is normally true even in the case where the program sets bit 12 in the Job Status Word. Also, TSX-Plus does not normally allow a program to test for terminal input without stalling on the .TTYIN EMT. However, if the /SINGLECHAR switch is used with the R or RUN command, TSX-Plus honors bits 6 and 12 of the Job Status Word and allows the program to activate on each character and test for terminal input without stalling. A program can also achieve this effect by using the "S" and "U" terminal control commands -- see chapter 5. KED and K52 are automatically run in single character activation mode.

Examples:

1. Run the program named "DUMP" on device "SY:".

```
.R DUMP
```

2. Run the program named "PIP" on "SY:" and pass to it the input line "A.TMP=B.TMP.

```
.R PIP A.TMP=B.TMP
```

3. Run the program named "SAMPLE" on "RK2:".

```
.R RK2:SAMPLE
```

4. Start the execution of BASIC and lock it to the line.

```
.R/LOCK BASIC
```

5. Start a program named PLANE and allow it to use single character activation mode.

```
.RUN/SINGLE PLANE
```

6. Start a program named TRIAL in debug mode so that it will be run under TSODT.

```
.RUN/DEBUG TRIAL
*TSX-ODT-V3*
*
```

2.3.36 The "RENAME" Command. The TSX-Plus RENAME command has the same form and options as the RT-11 RENAME command.

2.3.37 The "RESET" Command. The RESET command can be used to reset the system usage statistics that are displayed with the SYSTAT command. This is useful when you want to monitor system performance during a particular part of the day. Operator command privilege is required to use the RESET command.

2.3.38 The "RUN" Command. The RUN command is identical to the "R" command except that the default device is "DK:" instead of "SY:". See the description of the "R" command for information about available switches.

2.3.39 The "SEND" Command. The SEND command is used to send messages between timesharing terminals. The form of the command is:

SEND,Line# message

where line# is the number of the line to which the message is to be sent. If no line number is specified, the message is broadcast to all logged-on lines.

Examples:

1. Send a message to all logged-on users:
   SEND BOB-ARE YOU LOGGED ON?

2. Send a message to line number 2:
   SEND,2 WILL YOU BE ON TONIGHT?

When a SEND message is printed at a terminal, the message is preceded by the number of the line that originated the message. For example, a message from line 1 might be printed as follows:

01 BOB-ARE YOU LOGGED ON?

2.3.40 The "SET" Command. The SET command is used to set various options controlling system operation. The general form of the SET command is:

.SET device option

As with standard RT-11, the TSX-Plus SET command is used to specify options for devices such as line-printers, card-readers as well as setting certain system parameters such as terminal control characteristics. When used to set device options, the SET command has the same form as under RT-11 and may set the same options (they are specified in the handler). The SET command causes the copy of the device handler on the disk to be altered so that the effect of the command becomes "permanent" (until another SET changes the parameter back). The SET command also attempts to make the change to the copy of the handler that is in memory with

TSX-plus. If the handler is idle when the SET is done, the change will be made; otherwise, a warning message will be printed and the running copy of the handler is not altered. Operator command privilege is required to set an option in a device handler.

2.3.40.1 Options for the "SET TT" command

The form of this SET command is "SET TT option" to turn an option on and "SET TT NO option" to turn the option off. A few options require a numeric parameter, in which case they are specified as "SET TT option=value".

OPTION	MEANING WHEN SET ON
SCOPE	The terminal is a CRT device. Setting this option on has two effects: 1) Pressing the rubout key causes a backspace-space-backspace sequence to be echoed, erasing the previously typed character. 2) TSX-Plus counts the number of lines of output sent to the terminal. When as many lines have been sent as the screen size, TSX-Plus suspends output to the terminal (just as if CTRL-S had been pressed). Output may be continued by pressing CTRL-Q. This feature may be disabled by setting the page length to zero lines. See the "LENGTH" option below.
ECHO	Causes characters to be echoed to the terminal.
LC	Allows lower case characters to be passed to a program. If LC is set and bit 14 of the job status word is set to 1, input of lower case characters from the terminal will be passed to the running program. Otherwise, lower case characters are translated to upper case.
FORM	Causes form feed (FF) characters not to be converted to line feeds. FORM should be set on with terminals whose hardware can respond to form feed characters. If set off, form feed characters are replaced by an appropriate number of line feed characters.
TAB	Causes TAB characters not to be converted to multiple spaces. TAB should be set on with terminals whose hardware can respond to TAB characters. If set off, TAB characters are replaced by an appropriate number of spaces when being sent to the terminal.
PAGE	Allows CTRL-S and CTRL-Q characters to suspend and restart terminal output. If set off, CTRL-S and CTRL-Q have no special effect and are passed directly to the running program.

HOLD — May be used with VT50 and VT52 terminals to set "hold screen" mode on. This allows the SCROLL key to control output. Generally, the normal TSX-Plus end of page halt for scopes is more convenient than hold screen mode. The HOLD option may also be used with terminals other than VT50s. In this case, when a form-feed character is sent the terminal's bell is rung and output is suspended. Output is restarted when CTRL-Q is typed.

DEFER — TSX-Plus offers two modes of character echoing--"deferred" and "immediate" (NODEFER). If the user only types input to programs while they are waiting for input, the two modes function identically. However, if the user types input before the program finishes processing the previous line of input, the two modes are different. In immediate (NODEFER) mode the input characters are echoed immediately and may be printed even before the program prints the response to the previous line. In deferred echo mode, the characters that are typed ahead are accepted and held for the program, but are not printed until the program is ready to accept them. Under standard RT-11, EDIT, BASIC, and DIBOL programs run in deferred mode. Most other programs use immediate echoing. Deferred echoing is the preferred mode under TSX-Plus.

QUIET — Setting the QUIET option on suppresses the listing of command files as they are being executed. See the chapter on command files for additional information on controlling their listing.

FORM0 — Setting the FORM0 option on causes TSX-Plus to advance to the top of the page on the terminal when a write is done to the terminal with a block number of zero. This is convenient when producing multiple program listings to cause each listing to begin at the top of a new page.

LENGTH — The LENGTH option is used to set the number of lines on a page. The form of this option is:

"SET TT LENGTH=value"

TSX-Plus uses this line count for two purposes. When a form feed (FF) character is output, TSX-Plus replaces the form feed by as many line-feeds as required to upspace the paper to the top of the next page. This operation can be prevented by setting the "FORM" flag on (see the SET command), or by setting the page size to zero lines. The initial top of the page is defined each time the user types carriage return. Thus, when starting a program that will be doing paging, position the paper to the top of a page before pressing the carriage return.

The second use of the page size is to determine when to suspend output when printing on a CRT (SCOPE) terminal. When TSX-Plus prints the last line of the page on a CRT terminal, it suspends output and waits for the user to press CRTL-Q. This automatic suspension can be suppressed by setting the page size to zero lines.

FILLER The FILLER option is used to set the number of delay (filler) characters that are to be transmitted after carriage-return/line feed. The form of this option is "SET TT FILLER=value1,value2,value3".

where
value1 = Number of delay characters to send. (Specify in decimal).
value2 = Character after which delay characters are required. (Specify in octal). Carriage-return = 15, line feed = 12.
value3 = Character to use as a filler. (Specify in octal.)

Value 2 and value 3 may be omitted from the command if only value 1 needs to be changed. The appropriate sets of values for various terminals are listed below.

Terminal	DELAY Parameters
LA36	= 0
LA30 @ 300 Baud	= 10,15,0
LA30 @ 150 Baud	= 4,15,0
LA30 @ 110 Baud	= 2,15,0
VT05 @ 2400 Baud	= 2,12,0
VT05 @ 1200 Baud	= 4,12,0
VT50	= 0
VT52	= 0

WAIT Normally TSX-Plus blocks the execution of a program that does a .TTYIN EMT if no activation character has been received even if the program sets bit 6 in the Job Status Word which is supposed to mean that the program can do non-blocking .TTYIN character tests. This was done to prevent programs from burning up CPU time by constantly looping back to test for terminal input. If the NOWAIT option is specified with the SET TT command, TSX-Plus will honor bit 6 in the Job Status Word and allow the program to do non-blocking .TTYIN's.

VT100 Tells TSX-Plus that the terminal being used is a VT100 (which must be operated in VT100 mode -- not VT52 compatible mode) and also has the effect of SET TT SCOPE, TAB, PAGE, NOFORM, LENGTH=24.

VT52 Tells TSX-Plus that the terminal being used is a VT52 and also has the effect of SET TT SCOPE, TAB, PAGE, NOFORM, LENGTH=24.

LA36 Tells TSX-Plus that the terminal being used is an LA36 and also has the effect of SET TT NOSCOPE, NOTAB, NOFORM, NOHOLD, LENGTH=66.

LA120 Tells TSX-Plus that the terminal being used is an LA120 and also has the effect of SET TT PAGE, TAB, FORM, NOSCOPE, NOHOLD, LENGTH=66.

ADM3A Tells TSX-Plus that the terminal being used is a Lear Siegler ADM3A and also has the effect of SET TT SCOPE, NOTAB, NOFORM, NOHOLD, LENGTH=24.

HAZELTINE Tells TSX-Plus that the terminal being used is a Hazeltine brand terminal and has the effect of SET TT SCOPE, NOTAB, NOFORM, NOHOLD, LENGTH=24.

DIABLO Tells TSX-Plus that the terminal being used is a Xerox Diablo terminal and has the effect of SET TT NOSCOPE, NOTAB, FORM, NOHOLD, LENGTH=66. If you are doing plotting or proportional spacing printing, you should do a SET TT TAB command.

QUME Tells TSX-Plus that the terminal being used is a Qume brand terminal and has the effect of SET TT NOSCOPE, NOTAB, FORM, NOHOLD, LENGTH=66. If you are doing plotting or proportional spacing printing, you should do a SET TT TAB command for the terminal.

All of these options can be given initial settings for each line when the TSX-Plus system is generated. Each time a user logs onto a line, the initial option settings are used. The options may be altered by using the SET command, but when the user logs off, the options revert to their initial (sysgen) setting. When a user initiates a virtual line, the initial flag settings for the virtual line are copied from the current flag settings for the user. Subsequent flag changes for a virtual line do not affect flag settings for the user's other lines.

2.3.40.2 Options for the "SET CCL" command

There are two types of system commands, low level commands such as RUN, SET, ASSIGN and high level commands such as EXECUTE, COPY, DELETE, and DIRECTORY. Low level commands are executed directly by TSX-Plus. High level commands are first translated into the appropriate low level commands and then these commands are executed. The set of high level commands is known as the Concise Command Language (CCL). The "SET CCL" command can be used to observe the low level commands that are produced by translating CCL commands. There are two options to the SET CCL command: "SET CCL TEST" and "SET CCL NOTEST". When TSX-Plus is in CCL TEST mode, it will print at the terminal the low level commands that

are generated when a CCL command is entered. In this mode the low level commands are only printed and not executed. The "SET CCL NOTEST" command turns this off and TSX-Plus goes back to executing CCL commands. Test mode is very useful if you are having trouble getting some complex CCL command to work and want to examine the low level commands that are being generated.

2.3.41 The "SHOW" Command. The SHOW command is used to display information about the state of the system. Each form of the SHOW command is described below.

2.3.41.1 SHOW ASSIGNS. The SHOW ASSIGNS command displays information about all logical device assignments that are currently in effect.

2.3.41.2 SHOW DEVICES. The SHOW DEVICES command displays information about which devices were specified as being available when TSX-Plus was generated.

2.3.41.3 SHOW JOBS. The SHOW JOBS command displays information about jobs that are currently logged onto the system. The information displayed by this command is identical to that displayed by the SYSTAT command.

2.3.41.4 SHOW MEMORY. The SHOW MEMORY command displays information about memory usage including the total installed memory on the machine, the size of TSX-Plus and handlers, the memory space available to user jobs and the current job memory allocation and maximum authorized size. It also lists the size of the swappable job context area which is a system table associated with each job.

2.3.41.5 SHOW QUEUE. The SHOW QUEUE command displays information about print files in the spool queue. The following information is displayed for each print file in the queue: name of the device the file is queued for; an asterisk if the file is currently being printed; the name of the file if a file name was specified with the enter -- otherwise the name of the program that created the file; the name of the form on which the file is to be printed; the number of blocks in the file remaining to be printed -- this will decrease as the file is printed.

2.3.41.6 SHOW USE. The SHOW USE command causes display of the usage statistics for the current job since the logon. The statistics displayed include the connect time and CPU time.

2.3.42 The "SQUEEZE" Command. The TSX-Plus SQUEEZE command has the same form and options as the RT-11 SQUEEZE command. If the MOUNT command has been used to mount the device being squeezed, it is automatically dismounted before the squeeze takes place.

2.3.43 The "SYSTAT" Command. The SYSTAT (SYstem STATus) command displays information about the performance of the system and information about each logged on job. The first line of the system performance information shows a break down of the percent of total time spent running user jobs, waiting for user I/O, waiting for swapping I/O and waiting for something to do (idle time). These percentages should add up to approximately 100% (less rounding errors). The second line of system performance information shows the percent of the total time that some user I/O and swapping I/O was being performed. The percentages on this line are not expected to sum to 100 because they simply indicate how much of the time some I/O was taking place without considering whether jobs were running while the I/O was going on. The difference between the I/O percentages on the second line and the I/O wait percentages on the first line is a measure of the amount of overlap of job execution with I/O that took place. A "RESET" keyboard command (see above) may be used to reset these statistics so that the statistics may be gathered over a desired interval of system execution.

The information displayed for running jobs consists of one line per job. The line number is printed first, followed by an asterisk if the line is the one you are logged onto. A letter "V" is printed next if the line is a virtual line; a letter "D" indicates the line is a detached job line. A two-character state code is printed next indicating the current state of the job. The state codes and their meanings are as follows:

code:	meaning:
TI	Waiting for input from the terminal.
TO	Waiting for the terminal to print output.
RN	Program is running.
SL	Job is doing a timed wait (.TWAIT).
SF	Job is waiting for access to a shared file.
MS	Job is waiting for a message.
IO	Job is waiting for I/O to finish.
US	Job is waiting for access to USR file management module.
SP	Job is waiting for free spool block or file entry.

The characters "-SWP" are printed following the state code if the job is currently swapped out of memory.

The number of K-bytes of memory space used by the job is shown next, followed by the connect time and CPU time used so far by the job. The name of the program being run by the line is shown next. If the job logged on the Project,Programmer number is listed last.

Example:

```
.SYSTAT
Uptime: 04:27:32
System use:   Run=61%, I/O-wait=12%, Swap-wait=5%, Idle=22%
I/O activity: User I/O=28%, Swapping I/O=11%
1*    IO      11KB Connect=00:31:00 CPU=00:01:14 TECO    PPN=3,4
2     RN      36KB Connect=01:12:00 CPU=00:17:03 COBOL   PPN=5,9
6 V   RN      20KB Connect=00:04:00 CPU=00:02:17 KED     PPN=5,9
7 D   MS-SWP  20KB Connect=01:00:00 CPU=00:23:13 RTSORT
```

2.3.44 The "SPOOL" Command. The SPOOL Command is used to control the operation of the spooling system. It may only be used if spooling is specified when the system is generated. The form of the SPOOL command is:

.SPOOL device, function, parameter

Where "device" is the name of a device that was specified to be spooled when the system was generated. "Function" denotes what function is to be performed and "parameter" provides additional information for some functions. See Chapter 4 for further information on the SPOOL command. Operator privilege is required to be able to use this command.

2.3.45 The "TIME" Command. The TSX-Plus TIME command has the same form and options as the RT-11 TIME command. Operator command privilege is required to set the time.

2.3.46 The "TYPE" Command. The TSX-Plus TYPE command has the same form and options as the RT-11 TYPE command.

2.3.47 The $STOP" Command. The $STOP command is used to halt the execution of TSX-Plus and to reboot RT-11.

```
$STOP
```

The $STOP command forces an immediate logoff of all users before stopping TSX-Plus. Operator privilege is required to be able to use this command.

2.3.48 The "$SHUTDOWN" Command. The $SHUTDOWN command is similar to the $STOP command in that it is used to stop TSX-Plus and return control to RT-11. However, it differs from $STOP in the manner in which it stops TSX-Plus. $STOP forces the immediate logoff of all users; $SHUTDOWN does not. Rather, it sets a flag which prevents any new users from logging on and then waits for all logged on users to log off. When the last user logs off, TSX-Plus is stopped and control returns to RT-11. The form of this command is:

```
$SHUTDOWN
```

Operator privilege is required to be able to use this command.

2.4 RT-11 Commands Not Supported by TSX-Plus

The following keyboard commands are not supported by TSX-Plus: REENTER, CLOSE, START, SAVE, DEPOSIT, EXAMINE, BASE, GET, LOAD, UNLOAD, GT ON/OFF.

3. VIRTUAL TIMESHARING LINES AND DETACHED JOBS

3.1 Virtual Lines

TSX-Plus provides a facility known as "virtual lines" that allows one timesharing user to control several simultaneously running programs from a single terminal. When a user initially logs onto TSX-Plus, that person is said to be connected to the primary line that is also called virtual line number 0 (zero) for that user. At any time the user may switch to a different virtual line. This has the effect of logically disconnecting the timesharing terminal from the current virtual line and reconnecting it with a different logical line.

If a program is running at the time the switch is made, its execution is not affected (but it is given a lower CPU priority). When a running program that is not currently connected to a terminal writes output to the terminal, the output is stored in a terminal output buffer. When this buffer is filled, the program's execution is suspended (and the program is swapped out of core) until the terminal is reconnected to the virtual line. When any virtual line belonging to a user, but not currently connected to the user's terminal, enters an input or output wait state the bell is rung on the user's terminal to signify that a virtual line needs service.

A request to switch to a virtual line is indicated by typing control-W (hold down CTRL key and press W) followed by a single digit (do not hold down CTRL while typing the digit). The digit identifies which virtual line the user wishes to access. This line number represents the relative virtual line for that user. Other users may be using virtual lines of the same number without conflict. The CTRL-W digit sequence may be entered at any time - even in the middle of a line of input. The command takes effect immediately and leaves the old line in an undisturbed state. TSX-Plus responds to CTRL-W, digit by printing "n>" on the terminal, where "n" is the number of the virtual line that has just been accessed.

If a start-up command file was associated with the primary line, the same command file will be executed when the virtual line is initiated. If the LOGON program is run as part of the start-up command file for a virtual line, it will automatically log the user on with the same project-programmer number as was specified when logging onto the primary lines.

The user logs off a virtual line by typing the "OFF" command while connected to the virtual line. When a user logs off a virtual line other than the primary line, that virtual line is returned to the pool of free lines and the user is automatically connected to the primary line. When a user logs off the primary line, not only is the primary line released, but any virtual lines the user may have been using are freed, and the user is disconnected from the system. Note that it is possible (and

frequently desirable) to switch back and forth between the primary line and several virtual lines without logging off any.

The total number of virtual lines and the maximum number of virtual lines which any user may utilize at a given time are specified when the TSX-Plus system is generated. If a user attempts to log onto a virtual line and all the virtual lines are busy or the user is already connected to the maximum number of virtual lines, TSX-Plus will not respond to the CTRL-W.

Virtual lines are quite useful in situations where the user wishes to start a long "number crunching" job without tying up a line. Once the long job is started, the user can switch to another line. Programs that are running on a line that is not connected to a terminal are given a lower CPU priority than programs that are on lines connected to terminals. This means that the low priority programs will run only when no high priority programs are running.

3.2 Detached Jobs

The TSX-Plus Detached job facility is very similar to the virtual line facility; they both allow a timesharing user to initiate execution of several simultaneously executing jobs. The major difference between detached jobs and virtual lines is that virtual lines allow a user to switch terminal communication between several running tasks, whereas detached jobs do not. Detached jobs operate more like a "batch" facility. ALL terminal input for a detached job must come from a command file. Any terminal output generated by a detached job is discarded.

The differences between virtual lines and detached jobs are summarized below.

1. Using virtual lines, terminal communication may be switched between several running jobs. A detached job must receive all its terminal input from a command file and any terminal output it generates is discarded.

2. Virtual lines are "owned" by a physical line. When the physical line logs off, the virtual lines also log off. Detached jobs are not associated with any physical line. Once started, any or all timesharing users may log off without affecting detached jobs.

3. Detached jobs may be started automatically when TSX-Plus is initiated. Virtual line jobs must be started by timesharing users after TSX-Plus is running.

4. When a detached job reaches the end of its command file and asks for more terminal input, the job is aborted and the detached job slot is freed. Virtual lines wait for more input.

3.2.1 The DETACH command

The DETACH command is used to start a detached job, check its status and abort the job. The system manager may restrict the use of the DETACH command to selected users.

3.2.1.1 Starting a Detached Job The form of the DETACH command used to start a detached job is:

```
DETACH file-spec
```

where "file-spec" is the name of a command file that is to be executed as a detached job. The default device is "DK:" and the default extension is ".COM". When a request is made to start a detached job TSX-Plus searches for a free detached job slot. The total number of such job slots is established when TSX-Plus is generated. If a free slot is found the job is started and a message is printed saying which job slot was used. This number may be used later to reference the detached job.

Examples:

1. Start a command file named "PURGE" as a detached job.

```
.DETACH PURGE
Job started on line #4
```

2. Start a command file named "RK1:STATS.NEW".

```
.DETACH RK1:STATS.NEW
Job started on line #5
```

3.2.1.2 Checking the status of a detached job.

The form of the DETACH command used to check the status of a detached job is:

```
.DETACH/CHECK line-number
```

where "line-number" is the job slot number listed when the job was started.

Examples:

1. Check the status of the job on line #4.

```
.DETACH/CHECK 4
Line is active
```

2. Check the status of the job on line #5.

```
.DETACH/CHECK 5
Line is free
```

3.2.1.3 Aborting a detached job

The form of the DETACH command used to abort a running detached job is:

```
DETACH/KILL line-number
```

where "line-number" is the job slot number listed when the job was started.

Example:

1. Kill the job on line #4.

```
.DETACH/KILL 4
Job aborted
```

3.2.2 Detached Job Control Emt's

TSX-Plus provides a set of emt's that can be used to control the operation of detached jobs. Using these emt's it is possible to start a detached job, kill a detached job and check the status of a detached job.

3.2.2.1 Starting A Detached Job

This emt can be used to start the execution of a detached job. The form of the emt is:

```
EMT  375
```

with R0 pointing to the following argument block:

```
.BYTE     0,132
.WORD     .name-address
```

where ".name-address" is the address of an area containing the name of the command file to be started as a detached job. The command file name must be stored in ASCIZ form and may contain an extension. If a free detached job line is available, the specified command file is initiated as a detached job and the number of the detached job line is returned in R0.

errors:

Code	Meaning
1	No free detached job lines

3.2.2.2 Killing A Detached Job

This emt may be used to abort a detached job. The form of the emt is:

```
EMT  375
```

with R0 pointing to the following argument block:

```
.BYTE     2,132
.WORD     .job-number
```

where ".job-number" is the job number of the detached job to be killed.

errors:

Code	Meaning
1	Invalid job number

3.2.2.3 Checking The Status Of A Detached Job

This emt may be used to check the status of a detached job. The form of the emt is:

```
EMT  375
```

with R0 pointing to the following argument block:

```
.BYTE     1,132
.WORD     .job-number
```

where ".job-number" is the number of the detached job to be checked. If the detached job is still active the emt returns with the carry-flag cleared. If the detached job has terminated and the detached job line is free, the emt returns with the carry-flag set.

4. DEVICE SPOOLING

4.1 The Concept of Device Spooling

Device spooling is a technique that provides more efficient use of slow peripheral devices by buffering data directed to the slow devices to a high speed disk file where it is stored; later it is processed by the slow speed device without holding up the operation of the program that is generating the data. TSX-Plus optionally provides automatic spooling to output devices such as printers, card punches and plotters. Output may also be spooled to a timesharing terminal if an appropriate device handler is provided. Several devices may be spooled on a system.

When a running program directs output to a spooled device, the output is diverted by TSX-Plus to a spool disk file. An entry is made in a spool control table indicating a spool file is ready for the spooled device. When the spooled device becomes free, the spool file is copied by TSX-Plus to the device. All of the processing is automatic and the user does not have to be concerned with its operation. In fact a user can run programs without waiting for them to be printed. Devices that are to be spooled must be declared when the system is generated.

Spooled device handlers such as LP must be set to the "HANG" mode of operation for them to work properly with the TSX-Plus spooler. This can be done with the SET command. For example:

```
SET LP:HANG
```

The SET command need only be issued once as it sets flags in the copy of the handler on the disk.

4.2 Directing Output to Spooled Devices

Output is directed to a spooled device in exactly the same way it would be directed to the same device if it were not spooled. For example, if the line printer (LP) were spooled, the following commands would send a FORTRAN listing to the printer:

```
.R FORTRA
*TEST,LP:=TEST
```

The name of a spooled device may be used in a MACRO .ENTER command the same as a non-spooled device would be.

Any number of users may be writing output to a spooled device simultaneously without conflict. TSX-Plus stores the output from each user in a separate spool file and prints them in an orderly fashion. A user may direct output through several I/O channels to the same or different spooled devices. Output records directed through separate channels to the same device are separated into different spool files.

All output directed to spooled devices is stored in a common disk file. This total file space is allocated dynamically on a block by block basis as needed. The space is returned when output is processed by the spooled devices. For each logical output file generated by a user, a table is created in TSX-Plus called the Spool File Control Block. This table retains information about the spool file and reserves an entry in the ordered list of files waiting to be processed by a spooled device. The total file space that is available for spool files is specified when a TSX-Plus system is generated. If the file space is totally filled, running programs will be suspended when they attempt to write to the spool file. The programs are allowed to continue as file space is freed.

The total number of spool files that may be in existence is also specified when TSX-Plus is generated. A spool file is created when an I/O channel is opened to a spooled device; the file remains in existence until all of the output is finished being processed by the spooled device. If a channel is opened to a spooled device and the maximum number of files is already in existence, the execution of the program is suspended until a spool file is printed and deleted.

4.3 Use of Special Forms with Spooled Devices

Output files directed to spooled devices are queued and held until the spooled device becomes free. Because of this, a special procedure is required to synchronize the mounting of special forms with the printing of a file that requires the form.

If the first character of the first line in a file directed to a spooled device is a right square bracket ("]"), TSX-Plus will interpret the following one to six characters in the file as the name of the form that must be mounted when the file is printed. Form names may be from one to six characters in length and must be specified immediately following the initial square bracket character. The form name must be terminated with a carriage return, line feed. Square bracket characters are not significant to TSX-Plus in other than the first character position of the file.

If a spooled file does not begin with a right square bracket character, TSX-Plus uses the form name that was last specified by the user with a FORM command (Section 2.3.20). If no FORM command has been issued by the user, TSX-Plus uses the form name "STD" for the file.

Each time TSX-Plus attempts to select a file to be printed on a spooled device, it first looks for a waiting file that requires the same form that is currently mounted on the spooled device. If several such files are available, the oldest one is selected and started. If no file can be found that requires the currently mounted form, TSX-Plus selects the oldest file requiring a different form and issues a form mount request to the operator's terminal. The message appears as:

"Mount 'XXXXXX' form on ZZ"

The terminal to which the message is directed is the one that was declared to be the operator's console when the TSX-Plus system was generated.

Once the form mount request message is printed, the spooler for the device requiring the form mount is suspended. In order to restart the spooler the operator must enter a SPOOL-FORM or SPOOL-LOCK command. These commands tell the spooler that a particular form has been mounted and is ready for use. The operator does not have to mount the form that was called for in the form mount request message. He may mount any form he desires, in which case TSX-Plus will search for a file that needs the mounted form.

The SPOOL-FORM and SPOOL-LOCK commands are both used by the operator to indicate which form has been mounted; however, there is a difference in the effect of the two commands. When TSX-Plus has processed all files that need the currently mounted form, it checks to see if there are files requiring a different form. If there are any, it checks to see if the current form was mounted using a SPOOL-FORM or SPOOL-LOCK command. If a SPOOL-FORM command was used, TSX-Plus issues a form mount request message. If a SPOOL-LOCK command was used, TSX-Plus considers the current form to be locked on the printer and does not issue a form mount message; rather, it waits for new files to be created that need the currently mounted form.

4.4 Form Alignment Procedure

When mounting a new form it is necessary to have a way to verify the correct positioning of the form before starting production printing on the form. The SPOOL-ALIGN command provides this facility.

The SPOOL-ALIGN command allows the TSX-Plus operator to specify a form alignment file to be printed on the indicated spooled device. Form alignment files are printed immediately without regard to the name of the currently mounted form. The SPOOL-ALIGN command may be issued repeatedly if several attempts are required to mount a form.

Alignment files are created by the user and may contain any desired information. Typically they contain a short sample output file that matches a particular form. Alignment files should not contain a form name specification. The normal sequence of operations involving a form mount is as follows:

1) TSX-Plus issues a form-mount request message and suspends the spooler.
2) The operator mounts the desired form and issues one or more SPOOL-ALIGN commands to verify its positioning.
3) Once the form is correctly mounted, the operator issues a SPOOL-FORM or SPOOL-LOCK command to tell TSX-Plus which

form has been mounted.
4) TSX-Plus begins printing the oldest file that needs the currently mounted form.

The SPOOL-ALIGN command may be issued at any time, but it is typically used between the time a form-mount message is issued and the SPOOL-FORM or SPOOL-LOCK command is entered.

4.5 The SPOOL Command

The SPOOL command is used to control the operation of the spooling system. The form of the SPOOL command is:

```
.SPOOL device,function,parameter
```

where "device" is the name of a spooled device, "function" indicates the operation to be performed and "parameter" is an optional item of information used by some functions. Each of the available functions is described below.

4.5.1 The FORM and LOCK Functions. The FORM and LOCK functions are used to specify the name of the currently mounted form. The form name is specified in the "parameter" field of the command. The FORM function allows TSX-Plus to request a form mount when a different form is needed. The LOCK function specifies that the form is to be locked on the printer and form mount request messages are not to be generated. See Section 4.3 for further information.

Examples:

```
.SPOOL LP,FORM,BILLS
.SPOOL LP,LOCK,BILLS
.SPOOL LP,FORM,STD
```

Note the difference between the FORM command (Section 2.3.20) and the SPOOL command with the FORM or LOCK functions. The FORM command is used by the TSX-Plus user to specify the default form name to be used with subsequently generated spool files. The SPOOL FORM/LOCK commands are used by the TSX-Plus operator to tell the spooling system which form is currently mounted.

4.5.2 The ALIGN Function. The ALIGN function is used to cause a form alignment file to be printed on a spooled device. The name of the file to be printed is specified in the "parameter" field of the command. The default file extension for form alignment files is "ALN". See Section 4.4 for further information.

Examples:

```
.SPOOL  LP,ALIGN,BILLS
.SPOOL  LP,ALIGN,RK1:PAYROL.DAT
.SPOOL  LP,ALIGN,DX:RPORT2
```

4.5.3 The DEL Function. The DEL function causes the file currently being printed on the indicated spooled device to be deleted. The DEL function has no effect if no file is currently being printed.

Examples:

```
.SPOOL  LP,DEL
.SPOOL  LX,DEL
```

4.5.4 The SKIP Function. The SKIP function causes the spooler to skip over the next n blocks in the spool file that is currently being printed, where n is specified in the parameter field of the instruction. Each block in the spool file contains 508 characters. Printing of the file continues after the indicated number of blocks have been skipped.

Examples:

```
.SPOOL  LP,SKIP,10
.SPOOL  LP,SKIP,100
```

4.5.5 The BACK Function. The BACK function causes the spooler to skip backward in the spool file a number of blocks and then resume printing at that point. The number of blocks involved is specified when TSX-Plus is generated. This function is particularly useful for recovering from paper tears or remounts. The spooler will finish printing the current block before doing the backup.

Examples:

```
.SPOOL  LP,BACK
.SPOOL  LX,BACK
```

4.5.6 The STAT Function. The STAT function is used to determine the status of a spooled device. Information returned includes the condition of the spooler--active, idle, waiting for a form mount; the name of the currently mounted form; and information about files waiting to be printed on the device. The SHOW QUEUE command may also be used to display information about files in the spooler queue.

Examples:

```
.SPOOL  LP,STAT
.SPOOL  LX,STAT
```

4.5.7 The SING and MULT Functions. The SING and MULT functions control how the spooler will handle multiple files queued for the same form. In "MULT" mode (the initial setting) no form mount request message is generated if a spool file is found that needs the currently mounted form. Processing of the file begins automatically.

In "SING" mode a form mount request is generated for every file even if the file needs the currently mounted form. This is useful where equipment setup or form alignment is needed for every file.

Examples:

```
.SPOOL  LP,SING
.SPOOL  LP,MULT
```

4.5.8 The HOLD and NOHOLD Functions. A spooled device that is in the HOLD mode will not begin to process a spool file until the file is completely created and the I/O channel associated with the file is closed. A spooled device that is in NOHOLD mode will begin to process a spool file as the file is being created.

In NOHOLD mode, the spooler will begin to process a file sooner; however, if the file is being created slowly, the spooled device will remain busy (and unavailable to other users) for as long as it takes to finish generating the file. If the spool storage file is completely filled, the spooler will attempt to free space by beginning to process open spool files even if HOLD is in effect.

The default HOLD/NOHOLD mode is established when the TSX-Plus system is generated. A HOLD/NOHOLD command remains in effect until another HOLD/NOHOLD command is issued or the system is restarted.

Examples:

```
.SPOOL  LP,NOHOLD
.SPOOL  LP,HOLD
```

5. Program Controller Terminal Options

TSX-Plus provides a facility whereby a running program can dynamically alter some parameter settings relating to the user's timesharing line. This facility is invoked by having the running program output a special lead-in character followed by one or more function characters. The lead-in character alerts TSX-Plus to the fact that the following one or two characters are not to be sent to the user's terminal, but rather are to be interpreted as a command to TSX-Plus. The character that is to be used as the lead-in character may be specified when TSX-Plus is generated; its normal value is octal 35(decimal 29, CTRL-SHIFT-M). The character that immediately follows the lead-in character specifies the function to be performed. The available functions are listed below.

function character	meaning
A	Set rubout filler character.
B	Enable VT52 & VT100 escape-letter activation.
C	Disable VT52 & VT100 escape-letter activation.
D	Define new activation character.
E	Turn on character echoing.
F	Turn off character echoing.
H	Disable virtual lines.
I	Enable lower case input.
J	Disable lower case input.
K	Enable deferred character echo mode.
L	Disable deferred character echo mode.
M	Set transparency mode for output.
N	Suspend command file input.
O	Restart command file input.
P	Reset activation character.
Q	Set activation on field width.
R	Turn on high-efficiency TTY mode.
S	Turn on single-character activation mode.
T	Turn off single-character activation mode.
U	Enable non-wait TT input test.
V	Set field width limit.

These functions have a temporary effect in that they are automatically reset to their normal values when a program exits to KMON. They are not reset if a .CHAIN is performed.

5.1 "A"-function--Set rubout filler character

When a scope type terminal is being used, the normal response of TSX-Plus to a rubout character is to echo backspace-space-backspace which replaces the last character typed with a space. TSX-Plus responds to a CTRL-U character in a similar fashion, echoing a series of backspaces and spaces. Some programs that display forms use underscores or periods to indicate the fields where the user may enter values. In this case it is desirable for TSX-Plus to echo backspace-character-backspace for rubout and CTRL-U where "character" is either period or underscore as used in the form. The character to use as a rubout filler is specified by sending the character to TSX-Plus following the lead-in and A, that is, a three character sequence is sent to TSX-Plus: lead-in, "A", filler character.

5.2 "B" and "C" functions--Set VT52 & VT100 escape-letter activation

VT50 and VT52 terminals are equipped with a set of special function keys marked with arrows and other symbols. When pressed they transmit escape (octal 33) followed by a letter. The "B" function tells TSX-Plus to consider escape-letter to be an activation sequence. The escape character and the letter are not echoed to the terminal, but are passed to the user. The "C" function disables this processing and causes escape to be treated as a normal character (initial setting).

5.3 "D" - function--Define new activation character

Under normal circumstances TSX-Plus only schedules a job for execution and passes it a line of input when an "activation" character such as carriage return is received. The "D" function provides the user with the ability to define a set of activation characters that are used in addition to carriage return.

To define a new activation character the running program sends to TSX-Plus a lead-in character followed by the letter D, followed by the new activation character. The maximum number of activation characters that a program may define is specified when the TSX-Plus system is generated.

Using this technique, any character may be defined as an activation character, including such characters as letters, rubout, CTRL-U, and CTRL-C. When a user defined activation character is received, it is not echoed but is placed in the user's input buffer which is then passed to the running program.

By specifying CTRL-C as an activation character, a program may lock itself to a terminal in such a fashion that the user may not break out of the program in an uncontrolled manner. If carriage return is specified as a user activation

character, neither it nor a following line feed will be echoed to the terminal. TSX-Plus will also not add a line feed to the input passed to the program.

5.4 "E" and "F" functions--Control character echoing. The "E" and "F" functions are used to turn on and off character echoing. The "E" function turns it on, and the "F" function turns it off. An example of a possible use is to turn off echoing momentarily while a password is being entered.

5.5 "H" function--Disable virtual line use. The "H" function disables the virtual line facility for the timesharing line.

5.6 "I" and "J" functions--Control lower case input. The "I" function sets TSX-Plus to allow lower case characters to be passed to the running program. The "J" function causes TSX-Plus to translate lower case letters to upper case letters.

5.7 "K" and "L" functions--Control character echoing. The "K" function causes TSX-Plus to enter "deferred" character echo mode. The "L" function causes TSX-Plus to enter immediate character echo mode. See description of the DEFER option of the SET command for an explanation of deferred echo mode.

5.8 "M" function--Set transparency mode of output. If transparency mode is set on, TSX-Plus will pass through each transmitted character without performing any special checking or processing. Transparency mode allows the user's program to send any 7-bit character to the terminal. Note that once transparency mode is set on, TSX-Plus will no longer recognize the leading character (octal 35) which means a program control function follows. The only way to turn off transparency mode is to exit to KMON.

5.9 "N" and "O" Functions--Control input from command files. When a command file is being used to run programs (see Chapter 9), all input which would normally come from the user's terminal is instead drawn from the command file. Occasionally, it is desirable to allow a program running from a command file to accept input from the user's terminal rather than the command file. The "N" function suspends input from the command file so that subsequent input operations will be diverted to the terminal. The "O" function redirects input to the command file. These functions are ignored by TSX-Plus if the program is not being run from a command file. See also the description of the PAUSE keyboard command which can also be used to control command files.

5.10 "P"-function--Reset activation character. The "P" function performs the complement operation to the "D" function. The "P" function is used to remove an activation character that was previously defined by the "D" function. To reset an activation character the running program sends to TSX-Plus a

lead-in character followed by the letter "P" followed by the character that is to be removed from the activation list. Only activation characters that were previously defined by the "D" function may be removed by the "P" function.

5.11 "Q"-function--Set activation on field width. The "Q" function allows the user to define the width of an input field so that activation will occur if the user types in as many characters as the field width, even if no activation character was entered. The field width is specified by sending to TSX-Plus a lead-in character followed by the letter "Q" followed by a character whose binary value is the width of the field. If an activation character is entered before the field is filled, the program will be activated as usual. Each time activation occurs the field width is reset and must be set again for the next field by reissuing the "Q" function. For example, the following sequence of characters would be sent to TSX-Plus to establish a field width of 43 characters: "<lead-in>Q+". Note that the character "+" (plus) has the ASCII code of 053 (octal) which is 43 decimal.

5.12 "R"-function--Turn on high-efficiency TTY mode. The "R" function causes TSX-Plus to place the line in "high efficiency" tty mode. The effect of this is to disable most of the character testing overhead that is done by TSX-Plus as characters are transmitted and received by the line. Once a program has entered high-efficiency mode characters sent to the terminal are processed with minimum system overhead: tab characters are not expanded to spaces and form-feed characters are not treated specially. Also, TSX-Plus does not check to see if the character being sent is the TSX-Plus terminal control "leadin" character. This means that no further program controlled terminal commands may be issued until the program exits. Characters received from the terminal are passed to the program with minimum processing: they are not echoed, and control characters such as rubout, control-U, control-C, control-W and carriage-return are all treated as ordinary characters and passed directly to the program. Before entering high-efficiency mode the program must declare a user defined activation character that will signal the end of an input record. High-efficiency mode of TTY I/O is designed to facility machine-to-machine communication; it is also useful for dealing with buffered terminals that transmit a page of information at a time.

5.13 "S"-function--Turn on single-character activation mode. The "S" function causes TSX-Plus to allow a program to do single-character activation by setting bit 12 in the Job Status Word. Normally TSX-Plus stores characters received from the terminal and only activates the program and passes the characters to it when an activation character such as carriage-return is received. It does this even if bit 12 in the Job Status Word is set, which under RT-11 causes the program to be passed characters one-by-one as they are received from the terminal. The "S" function can be used to

cause TSX-Plus to honor bit 12 in the Job Status Word. If bit 12 is set and the program is single-character activation mode, TSX-Plus passes characters one-by-one to the program as they are received and does not echo the characters to the terminal. The /SINGLECHAR switch for the RUN command can also be used to do this.

5.14 "T"-function--Turn off single-character activation mode. The "T" function is the complement of the "S" function. It turns off single-character activation mode.

5.15 "U"-function--Enable non-wait TT input testing. The "U" function causes TSX-Plus to allow a program to do a .TTYIN EMT that will return with the carry bit set if no terminal input is pending. Normally TSX-Plus suspends the execution of a program if it attempts to obtain a terminal character by doing a .TTYIN EMT and no input characters are available. It does this even if bit 6 of the Job Status Word is set which under RT-11 would enable non-blocking .TTYIN's. This is done to prevent programs from burning up CPU time by constantly looping back to see if TT input is available. The "U" function causes TSX-Plus to honor bit 6 in the Job Status Word and allows a program to do a .TTYIN to check for pending TT input without blocking if none is available. The SET TT NOWAIT command and the /SINGLECHAR option to the RUN command can also be used to perform this function.

5.16 "V"-function--Set field width limit. The "V" function is used to set a limit on the number of characters that can be entered in the next terminal input field. Once the "V" function is used to set a field limit, the .TTYIN or .GTLIN EMT's may be used to accept an input field. If the user types in more characters to the field than the specified limit, the excess characters are discarded and the bell is rung rather than echoing the characters. The field width is specified by sending to TSX-Plus a lead-in character followed by the letter "V" followed by a character whose binary value equals the desired field width. The field size limit is automatically reset after each field is accepted and must be respecified for each field to which a limit is to be applied. Note the difference between the "Q" and "V" functions. The "Q" function sets a field size which causes automatic activation when the field is filled; the "V" function sets a field size which causes characters to be discarded if they exceed the field size.

6. TSX-Plus EMT'S

The following EMT service calls are available to jobs running under TSX-Plus.

6.1 Determining if a job is running under TSX-Plus. The proper way to determine if a job is running under TSX-Plus is to do a .SERR EMT (to suppress error aborts) and then to do the EMT to determine the TSX-Plus line number (see below). If the job is running under TSX-Plus the EMT will return without error; if not under TSX-Plus the carry bit (indicating an error) will be set on completion of the EMT.

6.2 Determining the TSX-Plus line number. The following EMT will return in R0 the number of the line to which the job is attached. Real lines are numbered consecutively starting at 1 in the same order they are specified when TSX-Plus is generated. Detached job lines occur next and virtual lines are numbered last.

The form of the EMT is:

```
EMT 375
```

with R0 pointing to the following argument area:

```
.BYTE     0,110
```

6.3 Determining the TSX-Plus license number. The following EMT will return in R0 as a 16-bit binary value the license number of the TSX-Plus system. The form of the EMT is:

```
EMT 375
```

with R0 pointing to the following argument block:

```
.BYTE     0,124
```

6.4 Determining the terminal type. The following EMT will return in R0 a value that indicates what type of time-sharing terminal is being used with the line. The form of the EMT is:

```
EMT 375
```

with R0 pointing to the following argument block:

```
.BYTE     0,137
```

The terminal type is specified either when the TSX-Plus system is generated or by use of the SET TT command (e.g., SET TT VT100). The currently returned terminal type codes are listed below.

Terminal-type	Code
(Unknown)	0
VT100	2
Hazeltine	3
ADM3A	4
LA36	5
LA120	6
Diablo	7
Qume	8

A type code of 0 (zero) is returned if the terminal type is unknown.

6.5 Sending a message to another line. The following EMT can be used to cause a message to display on another line's terminal. The form of the emt is:

```
EMT 375
```

with R0 pointing to the following argument block:

```
.BYTE     0,127
.WORD     Line-number
.WORD     Message-address
```

where "line-number" is the number of the line to which the message is to be sent and "message-address" is the address of the start of the message text that must be in ASCIZ form.

6.6 Mount a file structure. This EMT is used to tell TSX-Plus that a file structure is being mounted and that TSX-Plus should begin doing file directory caching for the device. The effect of this EMT is the same as doing a system MOUNT keyboard command. The form of the EMT is:

```
EMT  375
```

with R0 pointing to the following argument block:

```
.BYTE     0,134
.WORD     Device-spec-address
.WORD     0
```

where "device-spec-address" is the address of a word containing the RAD50 form of the name of the device on which the file structure is being mounted. If there is no room left in the table of mounted devices, the carry bit is set on return and the error code returned is 1.

6.7 Dismounting a file structure. This emt can be used to tell TSX-Plus to stop doing file directory caching for a file structure mounted on a particular drive. The effect of this emt is the same as doing a DISMOUNT keyboard command. The form of the emt is:

```
EMT  375
```

with R0 pointing to an argument block of the following form:

```
.BYTE     0,135
.WORD     Device-spec-address
.WORD     0
```

where "device-spec-address" is the address of a word containg the RAD50 value of the name of the device on which the file structure is mounted.

6.8 Set Terminal Read Time-out Value. This emt can be used to specify a time-out value that is to be applied to the next terminal input operation. This emt allows you to specify the maximum time that will be allowed to pass between the time that you issue a command to get input from the terminal (.TTYIN) and the time that an activation character is received to terminate the

input field. You also specify with this emt a special activation character that is returned as the terminating character for the field if the input operation times out without receiving an activation character from the terminal. The form of the emt is:

EMT 375

with R0 pointing to the following argument block:

```
.BYTE     0,117
.WORD     time-value
.WORD     activation-character
```

where "time-value" is the time-out value specified in 0.5 second units and "activation-character" is a single character value that is to be returned as the last character of the field if a time-out occurs. The time value specified with this emt only applies to the next terminal input field. The time value is reset when the next field is received from the terminal or the time-out occurs. A new time-out value must be specified for each input field that is to be time controlled.

6.9 Establishing Break Sentinal Control.

The following EMT can be used to declare a completion routine that will be triggered when the "Break" key is pressed. The form of the EMT is

EMT 375

with R0 pointing to the following argument block:

```
.BYTE     0,133
.WORD     .brkchr
.WORD     .cplrtn
```

where .brkchr is a user defined character that is to be declared the "Break" character and .cplrtn is the address of the completion routine that is to be called when the break character is received from the terminal. The specified completion routine will be called if the user presses either the key labeled "BREAK" (which transmitts a long space) or types the character that is declared as the user-specified break character (.brkchr). If no user-specified break character is wanted, specify the value 0 (zero) for .brkchr in the argument block and only the real "BREAK" key will be activated. Note that on some systems the console terminal "BREAK" key causes entry to the hardware ODT module and for this reason cannot be used with this TSX-Plus function. Only one break routine may be specified at a time for each user. If a break routine was previously specified, it is cancelled when a new

routine is declared. If an address of 0 (zero) is specified as the address of the completion routine (.cplrtn), any previously specified break routine is cancelled and the break key connection is cancelled. A break routine can be used to signal an asynchronous request for service to a running program. A good example of its use would be to trigger entry to an interactive debugging program.

6.10 Checking for Terminal Input Errors

The following EMT can be used to determine if any terminal input errors have occured. The form of the emt is

```
EMT  375
```

with R0 pointing to the following argument block:

```
.BYTE     0,116
```

On return from the emt, the carry-flag is set if an input error has occured since the line logged on or since the last time a check was made for input errors. The two types of errors that are monitored by this emt are hardware reported errors (parity, silo-overflow, etc.) and characters lost due to TSX-Plus input buffer overflow.

6.11 Checking for Activation Characters

The following emt can be used to determine if any activation characters have been received by the line but not yet accepted by the program. The form of the EMT is

```
EMT  375
```

with R0 pointing to the following argument block:

```
.BYTE     0,123
```

If there are pending activation characters, the carry-flag is set on return from the emt; otherwise the carry-flag is cleared.

6.12 Sending a Block of Characters to the Terminal

The following emt can be used to efficiently send a block of characters to the terminal. The form of the emt is

```
EMT  375
```

with R0 pointing to the following argument block

```
.BYTE     0,114
.WORD     .buffer
.WORD     .count
```

Where ".buffer" is the address of the buffer containing the characters to be sent and ".count" is a count of the number of characters to be sent. This emt is much more efficient to use than a series of .TTYOUT emt's -- it has the same efficiency as a .PRINT emt but it uses a count of the number of characters to send rather than having the character string in ASCIZ form.

6.13 Accepting a Block of Characters From the Terminal

The following emt can be used to accept all characters from the terminal input buffer up to and including the last activation character entered. The form of the emt is

```
EMT  275
```

with R0 pointing to the following argument block

```
.BYTE     0,115
.WORD     .buffer
.WORD     .size
```

where ".buffer" is the address of the buffer where the characters are to be stored and ".size" is the size of the buffer (number of bytes). This emt causes a program wait until an activation character is entered and then returns all characters received up to and including the last activation character. On return R0 contains a count of the number of characters received. If the specified buffer overflows, the carry-flag is set on return. This emt is substantially more efficient than doing a series of .TTYIN emt's; it is particularly well suited for accepting input from page buffered terminals.

6.14 Turning High-efficiency Terminal Mode On and Off

TSX-Plus offers a "high-efficiency" mode of terminal operation that eliminates a substantial amount of system overhead for terminal character processing by reducing the amount of processing that is done on each character. When in high-efficiency mode, characters are sent directly to the terminal with minimum handling by TSX-Plus; operations such as expanding tabs to spaces and form-feeds to line-feeds are omitted as well as input processing such as echoing characters and recognizing control characters such as rubout, control-U and control-C. The only characters treated specially on input are user defined activation characters and the user specified break character. At least one user specified activation character must be declared if high-efficiency mode is to be used. This form of terminal I/O is designed to facilitate high-speed machine-to-machine communication. It can be used effectively to communicate with buffered mode terminals. The form of the emt used to control high-efficiency mode is:

```
EMT  375
```

with R0 pointing to the following argument block:

```
.BYTE     .code,120
```

where ".code" is 1 to turn high-efficiency mode on and 0 to turn it off.

6.15 Determing number of free blocks in spool file

The following EMT will return in R0 the number of free blocks in the spool file. The form of the EMT is:

```
EMT 375
```

with R0 pointing to the following argument area:

```
.BYTE     0,107
```

6.16 Set/Reset ODT activation mode

The following EMT can be used to set TSX-Plus to activate on characters that are appropriate to ODT. In this mode TSX-Plus considers all characters to be activation characters except digits, ',', '$', and ';'. The form of the EMT is:

```
EMT 375
```

with R0 pointing to the following argument area:

```
.BYTE     .code,111
```

where .code is 1 (one) to turn on ODT activation mode and 0 (zero) to reset to normal mode.

7. SHARED FILE RECORD LOCKING

TSX-Plus provides a record locking facility that is useful in situations where programs being run from several terminals wish to update a common data file. Through the record locking facility a program may gain exclusive access to one or more blocks in a file by locking those blocks. Other users attempting to lock the same blocks will be denied access until the first user releases the locked blocks.

The usual protocol for updating a shared file being accessed by several users is as follows.

1) Open file.
2) Tell TSX-Plus that file is "shared".
3) Lock all blocks in file which contain desired record.
4) Read locked blocks into core.
5) Make update to record.
6) Write updated blocks to file.
7) Unlock blocks.
8) Repeat steps 3-7 as needed.
9) Close file.

DIBOL record locking procedures

See Appendix A for information on performing record locking from DIBOL programs.

Assembly language record locking EMT's

At the assembly language level TSX-Plus provides 4 EMT's to control record locking.

7.1 Opening a shared file

Before a file can be used with shared access it must be opened by using a standard .LOOKUP emt. After the lookup has completed successfully the following emt may be used to declare the file to be opened for shared access. The form of this emt is:

```
EMT 375
```

with R0 pointing to the following argument area:

```
.BYTE     .chan,125
.WORD     .access-code
```

where .chan is the number of an I/O channel that has previously been opened to the desired file and .access-code is a value that indicates the type of access protection desired for the file.

The following access codes are recognized:

Code	Protection	Access
0	Exclusive	Input
1	Exclusive	Update
2	Protected	Input
3	Protected	Update
4	Shared	Input
5	Shared	Update

The access-code specifies two things: The type of access that you intend to make to the file (input only or update) and the type of access that you are willing to grant to other users of the file. There are three protection classes that you may specify: Exclusive, Protected and Shared. Exclusive access means that you demand exclusive access to the file and will allow no other users to access the file in any fashion (input or update). Protected access means that you will allow other users to open the file for input but wish to prohibit any other users from opening the file for update. Shared access means that you are willing to allow other users to open the file for both input and update type access. The access code you specify also indicates your intended access to the file: input only or update.

When this emt is executed, TSX-Plus checks your specified protection mode and access type with that previously declared for the file by other users. If an access conflict arises because of your specified access characteristics an error code of 4 is returned for the emt. If no access conflict is detected, your specified access code is saved with the file and will be used to check for conflicts with future access requests issued by other users.

It is possible to have several channels simultaneously open to different shared files. The exact number of channels that can be open to shared files and the total number of shared files that may be opened are specified when the TSX-Plus system is generated.

Once all access to the file is completed, the I/O channel should be closed using the standard .CLOSE or .PURGE emt's. See below for information about saving the status of a channel that has been opened to a shared file.

The error codes that can be returned by this emt are listed below:

errors:	code	meaning
	1	Channel has not been opened to a file.
	2	Too many channels opened to shared files.
	3	Too many shared files open.
	4	File protection-access conflict

7.2 Saving the Status of a Shared File Channel

A standard .SAVESTATUS emt may be used to save the status of a shared file channel. If this is done, all blocks that are being held locked in the file remain locked until the channel is reopened and an unlock emt (see below) is done.

When using a single channel number to access several shared files it is convenient to initially do a .LOOKUP on each file then declare the file to be shared (emt above) and then do a .SAVESTATUS. The channel being used to access the set of files can then be switched from one file to another by doing a .PURGE followed by a .REOPEN. However, before doing the .PURGE, TSX-Plus must be told that you wish to save the shared-file status of the file (otherwise all locked blocks will be unlocked and the file will be removed from the shared-file list). The form of the emt used to perform this function is

```
EMT  375
```

with R0 pointing to the following argument block:

```
.BYTE     .chan,122
```

where .chan is the I/O channel number. The effect of this emt is to suspend the connection between the shared file information table and the I/O channel. Any blocks that are currently locked in the file remain locked until the channel is reopened to the file (by using a standard .REOPEN emt). After doing this emt the channel may be freed by using a .PURGE emt.

7.3 Waiting for a Locked Block

The following emt can be used to lock a specific block in a file. If the requested block is locked by another job, the requesting user's job will be suspended until the desired block becomes available. The form of the emt is

```
EMT  375
```

with R0 pointing to the following area:

```
.BYTE     .chan,102
.WORD     .block
```

where .chan is the number of an I/O channel that has previously been declared to be open to a shared file; .block is the number of the block in the file to be locked. Other blocks in the file which were previously locked remain locked. The maximum number of

blocks which may be simultaneously held locked is specified when TSX-Plus is generated. A block number of -1 (octal 177777) can be used to request that all blocks in the file be locked. If several users request the same block, access will be granted sequentially in the order that the requests are received.

errors:	code	meaning
	1	Channel is not open to a shared file
	2	Request to lock too many blocks in file

7.4 Trying to Lock A Block

This emt is similar in operation to the previous emt. It too is used to request that file blocks be locked. The difference is that if the requested block is already locked by another user the previous EMT suspends the requesting program whereas this EMT does not suspend the program but rather returns an error code. If the block is available it is locked for the requesting user and no error is reported. The form of this EMT is:

```
EMT 375
```

with R0 pointing to the following argument area:

```
.BYTE     .chan,103
.WORD     .block
```

errors:	code	meaning
	1	Channel is not open to a shared file.
	2	Request to lock too many blocks in file.
	3	Requested block is locked by another user.

7.5 Unlocking A Specific Block

This emt is used to unlock a specific block in a file. The form of the emt is

```
EMT  375
```

with R0 pointing to the following argument block:

```
.BYTE     .chan,113
.WORD     .block-number
```

where ".chan" is the number of the I/O channel opened to the shared file and ".block-number" is the number of the block to be unlocked.

errors:	code	meaning
	1	Specified channel not opened to a shared file

7.6 Unlocking All Locked Blocks In A File

This emt is used to unlock all blocks held locked for a file. The form of the emt is

```
EMT  375
```

with R0 pointing to the following argument area:

```
.BYTE     .chan,101
```

where .chan is an I/O channel number that is open to a shared file. When this EMT is executed all blocks previously locked by the user on the shared file are unlocked. Blocks locked by the user on other files are not released nor are blocks of the same file that are locked by other users.

error:	code	meaning
	1	Channel is not open to a shared file.

7.7 Checking For Writes To A Shared File

The following emt can be used to determine if any other user has written to a shared file. The form of the emt is

EMT 375

with R0 pointing to the following argument block

```
.BYTE     .chan,121
```

where ".chan" is the I/O channel number opened to the shared file. If no other user has written to the file since the file was opened by the user issuing this emt or since that last time this emt was issued for the file, the carry-flag is cleared on return from the emt. If the file has been written to by some other user since the last check was made, the carry-flag is set on return and an error code of 2 is returned.

This emt can be used to advantage in a situation where data from some block in the file is being held in a buffer and it is desired to determine if the data is valid or if it may be invalid because some other user might have altered it by writing to the file. The usual sequence of operations in this situation is to first lock the block whose data is in the in-memory buffer, then do the emt to see if the file has been written to; if the file has not been modified the data in memory is valid and can be used, otherwise the block must be reread from the file.

8. MESSAGE COMMUNICATIONS FACILITIES

TSX-Plus provides an optional facility that allows running programs to send messages to each other. This message communication facility allows programs to send messages through named channels, check to see if messages are pending, and suspend execution until a message is received.

8.1 Message Channels

Messages are transferred to and from programs by using TSX-Plus "Message Channels". A message channel accepts a message from a sending program, stores the messsage in a queue associated with the channel and delivers the message to a receiving program that requests a message from the channel. Message channels are totally separate from I/O channels.

Each active message channel has associated with it a one to six character name that is used by the sending and the receiving programs to identify the channel. The total number of message channels is defined when TSX-Plus is generated. The names associated with the channels are defined dynamically by the running programs. A message channel is said to be "active" if any messages are being held in the queue associated with the channel or if any program is waiting for a message from the channel. When message channels become inactive they are returned to a free pool and may be reused by another program.

Once a message is queued on a channel, that message will remain in the queue until some program receives it or the TSX-Plus system is restarted. A program's exiting to the keyboard monitor does not remove any pending messages that it queued. This allows one program to leave a message for another program that will run later.

8.2 Sending a Message

An EMT is provided for the assembly language programmer to use to queue a message on a named channel. If other messages are already pending on the channel, the new message is added to the end of the list of waiting messages. The sending program continues execution after the EMT and does not wait for the message to be accepted by a receiving program. During processing of the EMT the message is copied to an internal buffer, and the sending program is free to destroy its message on completion of the EMT. The form of the EMT is:

EMT 375

with R0 pointing to the following argument area:

```
.BYTE     0,104
.WORD     .chadr
.WORD     .msadr
.WORD     .mssiz
```

where .CHADR is the address of a six byte field containing the name of the message channel in ASCII with trailing blanks if the name is less than six characters. .MSADR is the address of the beginning of the message text. .MSSIZ is the message length in bytes.

errors	code	meaning
	1	All message channels are busy. (Re-gen TSX-Plus and increase the value of the MAXMC parameter.)
	2	Maximum allowed number of messages are being held in message queues. (Re-gen TSX-Plus and increase the value of the MAXMSG parameter.)
	4	The transmitted message is too long. (Message is truncated to maximum length.)

Note that the maximum message length is defined during system generation by the MSCHRS parameter. If a message longer than this is sent, only the first part of the message will be delivered and error code 4 will be returned.

8.3 Checking for Pending Messages

The second EMT is used to receive a message from a named channel if a message is pending on the channel. If no message is pending, an error code (3) is returned, and the program is allowed to continue execution. The form of the EMT is:

EMT 375

with R0 pointing to the following argument area:

```
.BYTE     0,105
.WORD     .chadr
.WORD     .msadr
.WORD     .mssiz
```

where .CHADR points to a field with a six character channel name; .MSADR points to the buffer to which the message is to be placed and .MSSIZ is the size of the message buffer (bytes).

If a message is received, its length (bytes) is placed in R0 on return from the EMT. If the received message is longer than the message buffer (.MSSIZ), only the first part of the message will be received.

errors	code	meaning
	0	No error. A message was received.
	3	No message was queued on the named channel.
	4	Message was longer than the receiving buffer.

8.4 Waiting for a Message

The third EMT is used to suspend execution of a program until a message becomes available on a named channel. The form of the EMT is:

```
EMT 375
```

with R0 pointing to the following argument area:

```
.BYTE     0,106
.WORD     .chadr
.WORD     .msadr
.WORD     .mssiz
```

where .CHADR points to a field with a six character channel name; .MSADR points to the buffer where the message is to be placed; and .MSSIZ is the size of the message buffer (bytes).

The length of the received message (bytes) is placed in R0 on return from the EMT.

errors	code	meaning
	0	No error. A message was received.
	1	All message channels are busy. (Re-gen TSX-Plus and increase the value of the MAXMC parameter.)
	4	Message was longer than the receiving buffer.

9. COMMAND FILES

The TSX-Plus command file facility is significantly more powerful than that provided by standard RT-11. Parameter strings may be specified when a command file is started; the parameters are stored by TSX-Plus and inserted in the text of the command file at selected points as the command file is processed. Unlike RT-11 command files, TSX-Plus command files allow program data as well as system commands to be placed in a command file. Under TSX-Plus it is possible to set up a command file so that any request for data from device "TT" comes from the command file. This allows EDIT and TECO commands to be placed in a command file.

A command file is invoked by typing:

```
@filename param1 param2 ... param5
```

where "filename" is the name of the command file and "param1", "param2", etc. are parameter string arguments to the command file. The default extension for a command file is ".COM".

When the name of a command file does not conflict with a system command, the command file may be invoked by typing:

```
filename param1,param2, ... param5
```

that is, the @-sign may be left out. There are two differences between starting a command file with and without the @-sign. With the @-sign the default device searched for the command file is "DK". Without the @-sign the default device is "SY". With the @-sign the command file listing is initially set according to the last executed SET TT QUIET or SET TT NOQUIET command. Without the @-sign the command file listing is initially suppressed but may be turned on by putting a listing control command in the file.

Parameter strings are normally delimited by spaces. Thus in the command:

```
@TSTRUN ABC 123.45 2/3
```

the string "ABC" is parameter 1, "123.45" is parameter 2 and "2/3" is parameter 3. It some cases it may be desirable to include spaces as part of a parameter string. If this is to be done, the first parameter must begin with the character "\" (left leaning slash), and the left-slash must be used as the parameter delimiter rather than spaces. For example, in the command line:

```
@TSTRUN \A STRING\OF PEARLS
```

parameter 1 is "A STRING" and parameter 2 is "OF PEARLS".

Up to five parameter strings may be specified when the command file is started. The total number of characters in the parameter string may not exceed 60.

To insert a parameter string in a command file put an up-arrow ("^") character followed by a digit in the range of 1 to 5 at the desired point of insertion. The digit indicates which parameter string is to be inserted at that point. The up-arrow-digit sequence is replaced by the appropriate parameter string as the command file is being executed. If a parameter string is called for that was not specified when the command file was invoked, the up-arrow-digit will be deleted and no characters will be inserted in their place. For example, consider the following command file:

```
R ^1
^3=^2
```

If it is executed by use of the following command:

```
@TEST FORTRAN PROG
```

The result will be:

```
R FORTRAN
=PROG
```

Command files may be nested, that is, one command file may call another command file. When such a call occurs, the parameter strings for the outer level (calling) command file are stored on a stack in TSX-Plus and then the parameter strings for the called command file are set up. When the called command file finishes, the parameters for the calling command files are restored. The maximum depth of nesting is governed by the size of the parameter string stack and the length of the actual parameter strings. A nesting depth of 3 can be reached even with very long parameter strings. If no parameters are specified, the nesting depth may go to about 7 levels.

The command file listing status (SET TT QUIET) is also stacked as command files are nested. This means that an inner nested command file may have its listing turned on or off; when it exits the listing control will be reset to the state it was in when the command file was called. This is not done for the outer-most command file, however, so if the outer level command file contains a SET TT QUIET command, it will have a permanent effect.

Several combinations of characters take on special meaning when they are found within a command file. The up-arrow character ("^") followed by a letter is replaced by the control character corresponding to the letter specified. Thus "^C" becomes control-C. The escape character (alt mode) may be represented by up-arrow, dollar sign ("^$").

Several combinations of characters within a command file perform special functions when they are encountered. These control character functions are carried out immediately by TSX-Plus as the characters are found. These control characters are not passed to the program.

control character	function
^(	Stop listing command file. This has the same effect as SET TT QUIET, except it only applies to the current command file.
^)	Start listing command file. This has the same effect as SET TT NOQUIET except that it only applies to the current command file.
^!	Suppresses all TT output. Both the command file listing and any program output to TT are suppressed. The "^(" and "^)" commands restart program generated output.
^>	Accept all TT input from the command file. Initially command file data is only passed to programs when they do .GTLIN, .CSISPC or .CSIGEN EMT's. This is the way RT-11 handles all command files. Requests for data such as .TTYIN or .READ bypass the command file and go to the terminal. The "^>" command causes all subsequent requests for data from the terminal to come from the command file regardless of which EMT is used. This allows data for application programs to be placed in a command file. It also allows commands for TECO and EDIT to be placed in a command file. This command only affects input requests that occur after TSX-Plus reads the "^>" sequence.
^<	Return to standard data mode. The following command file data will only be passed to programs which do .GTLIN, .CSISPC or .CSIGEN EMT's (standard RT-11 mode).

A PAUSE command is provided to allow the execution of a command file to be suspended while the operator performs some manual operation. The form of the PAUSE command is:

PAUSE comments

where "comments" may be any string of characters. When a PAUSE command is encountered within a command file, the PAUSE command is printed on the terminal followed by ">>". Execution of the command file is then suspended until carriage return is pressed. See also the description of the "N" and "O" program controlled terminal options that affect command file input.

A DISPLAY command may be placed in a command file to cause a line of text to be displayed at the terminal when the DISPLAY command is executed. The form of the DISPLAY command is:

DISPLAY comments

where "comments" may be any line of text to be printed at the terminal. This is useful when running command files in "quiet" mode so that they are not being listed.

10. TSX-Plus Performance Monitor Feature

TSX-Plus includes a performance analysis facility that can be used to monitor the execution of a program and determine what percentage of the run time is spent at various locations within the program. When the performance analysis facility is being used TSX-Plus examines the program being monitored when each clock tick occurs (50 or 60 times per second) and notes at what location in the program execution is taking place. Once the analysis is completed the TSX-Plus performance reporting program (TSXPM) can be used to produce a histogram showing the percentage of time spent at various locations during the monitored run.

There are three steps involved in performing a performance analysis on a program:

1) Use the MONITOR command to begin the analysis.
2) Run the program to be monitored.
3) Run the TSXPM program to print a histogram of the result.

10.1 Starting a Performance Analysis

The first step in doing a performance analysis is to use the MONITOR keyboard command to tell TSX-Plus that a performance analysis is to be done on the program that will be run next. The form of the MONITOR command is

```
MONITOR base-address,top-address[,cell-size]/switches
```

where "base-address" is the lowest address in the region to be monitored, "top-address" is the highest address in the region to be monitored and "cell-size" is an optional parameter that specifies the number of bytes of address in the region being monitored to group together into each histogram cell. If the cell-size parameter is not specified, TSX-Plus calculates the cell size by dividing the number of bytes in the region being monitored (base-address to top-address) by the total number of histogram cells available (specified when TSX-Plus is generated). This gives the finest resolution possible. The only available switch is "/I" which, if specified, causes I/O wait time to be included in the analysis. If this switch is not specified, only CPU execution time is included in the analysis.

Examples:

```
MONITOR 1000,13000/I

MONITOR 20000,40000,10

MONITOR 2000,6000
```

The effect of the MONITOR command is to set up parameters within TSX-Plus which will be used to monitor the next program run -- It does not actually begin the analysis so there is no rush in running the program to be monitored.

Only one user may be doing a performance analysis at a time. This is because the performance analysis histogram buffer is a common memory area that may not be in use by more than one user at a time. An analysis is in effect for a user between the time the MONITOR command is issued and the TSXPM program is run to display the results of the analysis. Running the TSXPM program terminates the performance analysis and allows other users to perform analyses. Note also that space for the performance analysis data buffer must be reserved when TSX-Plus is generated.

Once the MONITOR command has been issued, the program to be monitored is run by using the standard "RUN" or "R" commands. A link map of the program should be available to determine the addresses in the program that are appropriate to monitor. If the program is overlayed and the region to be monitored is in the overlay area, the analysis technique is more complex. It is necessary to use the performance monitor emt's (described below) to control when the analysis is turned on and off to monitor a particular overlay segment running in the region being monitored. If a program being monitored does a .CHAIN to another program, the analysis continues and the times reported will be the composite of the programs run.

10.2 Displaying The Results Of The Analysis

After the program being monitored has been run and it has exited and returned control to the keyboard monitor, the TSXPM performance reporting program is used to generate a histogram of the time spent in the region being monitored. The TSXPM program is started by typing "R TSXPM"; it responds by printing an asterisk ("*"). In response to the asterisk, enter the file specification for the device/file where the histogram is to be written. Optionally a switch of the form "/M:nnn" may be specified following the file specification. This switch is used to specify the minimum percentage of the total run-time that a histogram cell must contained in order to be included in the display. If this switch is not specified, the default cut-off percentage is 1%.

After receiving the file specification TSXPM prompts for a title line. Enter a line of text which will be printed as a page title in the histogram file. Press return if you wish no title.

The next item of information requested by TSXPM is a set of base offset values. The base offset values are optional. Base offsets are useful in the situation where you have several modules making up a program being monitored and you want the addresses displayed on the performance analysis histogram to be relative to the base of each module. You may specify up to 10 offset values. Each offset value is specified as an offset module number (in the range 0 to 9) followed by a comma and the base address of the module (see example below). If offset values are specified, TSXPM

determines in which module each cell of the histogram falls and displays the address as a module number and offset within the module. After you enter all desired module offsets, enter return without a value.

After the base offset values are entered the histogram will be produced and written to the specified device and file. After the histogram is generated TSXPM prints the asterisk prompt again at which point you may enter the name of another device/file and produce the histogram again if desired or you may type control-C to return to the keyboard monitor.

Example use of TSXPM

```
.R TSXPM
*LP:/M:5
Title:PERFORMANCE ANALYSIS OF EIGENVALUE CALCULATION
Base offsets:
>1,1000
>2,2134
>3,5212
>
[Histogram is produced at this point]
*control-C
```

The histogram produced by TSXPM consists of one line per histogram cell. Each line contains the following information: 1) the base module offset number (if offsets were specified); 2) The address range covered by the histogram cell (relative to the module base if base offsets were used); 3) The percentage of the total execution time spent at the address range covered by the histogram cell; 4) A line of stars presenting a graphic representation of the histogram.

10.3 Performance Monitor Control EMT's

For most applications the method described above can be used to do a performance analysis. However, in special cases (such as analizing the performance of an overlayed program) it is necessary to have more explicit control over the performance analysis feature as a program is running. The following set of emt's may be used to control a performance analysis.

10.3.1 Initializing A Performance Analysis

This emt is used to set up parameters that will control a performance analysis. It does not actually begin the analysis. The form of the emt is

```
EMT  375
```

with R0 pointing to the following argument block:

```
.BYTE     0,137
.WORD     .base-address
.WORD     .top-address
.WORD     .cell-size
.WORD     .flags
```

where .base-address is the address of the base of the region to be monitored, .top-address is the address of the top of the region to be monitored and .cell-size is the number of bytes to group in each histogram cell. If 0 (zero) is specified as the cell size, TSX-Plus calculates the cell size to use by dividing the number of bytes in the region being monitored (.top-address minus .base-address) by the number of cells available in the histogram data area (specified when TSX-Plus is generated). The .flags parameter is used to control whether I/O wait time is to be included in the analysis or not. If a value of 1 is specified as the .flags parameter I/O wait time is included in the analysis; if a value of 0 (zero) is specified I/O wait time is not included in the analysis.

errors:	Code Meaning
0	Performance analysis being done by some other user.
1	Performance analysis feature not genned into TSX-Plus.

10.3.2 Starting a Performance Analysis

This emt is used to begin the actual collection of performance analysis data. The previous emt must have been executed to set up parameters about the performance analysis before this emt is done. The form of the emt is

```
EMT  375
```

with R0 pointing to the following argument block:

```
.BYTE     1,136
```

Errors:	Code Meaning
0	Performance analysis has not been initialized yet.

10.3.3 Stopping A Performance Analysis

The following emt can be used to suspend the data collection for a performance analysis. The data collection can be restarted by using the start-analysis emt described above. This emt could, for example, be used to suspend the analysis when an overlay module is loaded that is not to be monitored. The start-analysis emt would then be used to reenable the data collection when the overlay of interest is reloaded. The form of this emt is

```
EMT  375
```

with R0 pointing to the following argument block:

```
.BYTE     2,136
```

Errors:	Code	Meaning
	0	Performance analysis has not been previously initialized.

10.3.4 Terminating A Performance Analysis

This emt is used to conclude a performance analysis. It has the effect of returning into a user supplied buffer the results of the analysis and freeing the performance analysis feature for use by other users. The form of this emt is:

```
EMT  375
```

with R0 pointing to the following argument block:

```
.BYTE     3,136
.WORD     .parameter-buffer
.WORD     .histogram-buffer
.WORD     .buffer-size
```

where .parameter-buffer is the address of a 4 word buffer into which will be stored some parameter values describing the analysis that was being performed; .histogram-buffer is the address of the buffer that will receive the histogram count values; .buffer-size is the size (in bytes) of the histogram buffer area.

Errors:	Code	Meaning
	0	This job is not doing a performance analysis.
	1	Area provided for histogram count vector is too small.

The parameter values returned consist of the following 4 words: 1) Base address of monitored region; 2) Top address of the monitored region; 3) Number of bytes per histogram cell; 4) Control and status flags. The control and status flags are a set of bits that provide the following information:

Flag	Meaning
1	I/O wait time was included in the analysis.
100000	Some histogram cell overflowed during the analysis.

The histogram data returned consists of a vector of 16-bit binary values -- one value for each cell in the histogram. The first value corresponds to the histogram cell that starts with the base address of the region that was being monitored.

11. TSX-Plus RESTRICTIONS

11.1 EMT's Not Supported by TSX-Plus

The following EMT's (monitor service requests) are not supported by TSX-Plus and are treated as NOP's: .FETCH, .RELEAS, .QSET, .CDFN, .INTEN and .SYNCH (.INTEN and .SYNCH may be used within device handlers). Since the .CDFN EMT is not allowed, programs must use only channels 0-17. Both version 1 and version 2 type EMT's are supported by TSX-Plus. TSX-Plus provides full timer support including month and year date roll-over and the .TWAIT and .MRKT emt's. TSX-Plus does not provide the RT-11 style multi-terminal support emt's.

11.2 Programs Not Supported by TSX-Plus

Most programs which run under RT-11 will run under TSX-Plus without change. However, a modified version of ODT ("TSODT"--supplied with TSX-Plus) must be used to debug programs under TSX-Plus. The BATCH RT-11 facility is not supported by TSX-Plus.

APPENDIX A

DIBOL TSX-Plus SUPPORT SUBROUTINES

A set of subroutines is provided with TSX-Plus to perform DIBOL record locking and message transmission functions. Note that if these TSX-Plus features are to be used, they must be enabled when the TSX-Plus system is generated.

1. Record Locking Subroutines

The record locking subroutines coordinate access to a common file being shared and updated by several TSX-Plus users. The five subroutines parallel the operation of the DIBOL statements: OPEN, CLOSE, READ, WRITE and UNLOCK. The normal DIBOL I/O statements cannot be used to perform record locking under TSX-Plus.

1.1 Opening the file

The first subroutine is used to open a shared file in update mode. The form of the call is:

```
XCALL FOPEN(chan,devlbl,errflg)
```

where

chan = A decimal expression that evaluates to a number in the range 1-15. This is the channel number used in associated calls to FREAD, FWRIT, FUNLK and FCLOS subroutines.

devlbl = The name of an alphanumeric literal, field or record that contains the file specification in the general form: dev:filnam.ext

The file size must not be specified with the file name. An optional "/W" switch may be appended to the file name to cause the "WAITING FOR dev:file" message to be printed.

errflg = A numeric variable capable of holding at least two digits into which is stored an indication of the result of the FOPEN call. The following values are returned:

value	meaning
0	No error. File is open and ready for access.
17	File name specification is invalid.
18	File does not exist or channel is already open.
72	Too many channels are open to shared files. (Re-gen TSX-Plus and increase the value of MAXSFC parameter).
73	Too many shared files are open. (Re-gen TSX-Plus and increase the value of MAXSF parameter).

The FOPEN subroutine should only be used to open files that will be updated by several users. The normal DIBOL OPEN READ/WRITE sequence should be used for other files. Several files may be opened for update by calling FOPEN with different channel numbers. The ONERROR DIBOL statement does not apply to these record locking subroutines. Instead the "errflg" argument is used to indicate the outcome of the operation.

1.2 Locking and reading a record

The FREAD subroutine is used to lock and read a record. The form of the call is:

```
XCALL FREAD(chan,record,rec #,'T' or 'W',errflg)
```

where

chan = Decimal expression in the range 1-15 that identifies a channel previously opened by FOPEN.

record = Name of the record or alphanumeric field in which the record read is to be placed.

rec # = Decimal expression that specifies the sequence number of the record to be read. This value must be between 1 and the total number of records in the file.

'T'/'W' = If 'T' is specifed as the fourth parameter, FREAD will return a value of 40 in errflg if the requested record is locked by some other user. If 'W' is specified, FREAD will wait until the record is unlocked by all other users and will never return the record-locked error code.

errflg = Decimal variable into which is stored one of the following values:

value	meaning
0	No error. Record has been locked and read.
1	End-of-file record has been read.

22	I/O error occurred on read or channel is not open.
28	Invalid record number (possibly beyond end of file).
40	Record locked by another user. (Only returned if 'T' is specified as fourth argument.)
71	Channel was not opened by calling FOPEN.
72	Request to lock too many blocks in file. (Re-gen TSX-Plus and increase value of MXLBLK parameter.)

The FREAD subroutine functions like the DIBOL READ statement. However, whereas the DIBOL READ statement always returns an error code (40) if the requested record is locked, FREAD offers the user a choice: If 'T' is specified as the fourth argument to FREAD, a code of 40 will be returned in errflg if the record is already locked. If 'W' is specified as the fourth argument and the record is locked, FREAD does not return an error code, but rather waits until the requested record is unlocked. It is much more efficient to wait for a locked record by using the 'W' option rather than re-executing the FREAD with the 'T' option. It may be desirable to perform the first FREAD using the 'T' option. If the record is locked a "WAITING FOR RECORD..." message can be displayed on the user's console and another FREAD can be issued with the 'W' option to wait for the record. On return from this FREAD the "WAITING" message can be erased.

Note that although record locking is requested on a record-by-record basis, the actual locking is done on a block-within-file basis. (A block contains 512 characters). The result of this is that a record is locked if any record contained in the same block(s) as the desired record is locked.

Once a record is locked and read using FREAD, the record remains locked until the program performs one of the following operations:

1. Issues an FWRIT to the channel from which the record was read.
2. Issues another FREAD to the channel.
3. Issues an FUNLK to the channel.
4. Issues an FCLOS to the channel.
5. Terminates execution by use of the STOP statement or because of an error.

The same set of rules that applies to the DIBOL READ statement apply to FREAD.

1.3 Writing a record

The FWRIT subroutine is called to write a record to a shared file. The form of the call is:

```
XCALL FWRIT(chan,record,rec #,errflg)
```

where

chan = Channel number associated with the file.

record = Name of the record or alphanumeric field that contains the record to be written.

rec # = Decimal expression that specifies the sequence number of the record to be written.

errflg = Decimal variable into which is stored one of the following values.

value	meaning
0	No error.
22	I/O error occurred during write or channel is not open.
28	Bad record number specified.

The FWRIT subroutine writes the indicated record to the file then unlocks any blocks that were locked by the program. FWRIT appends a <CR><LF> to the end of the written record as does the DIBOL WRITE statement. The rules for the DIBOL WRITE statement also apply to FWRIT.

1.4 Unlocking records

The FUNLK subroutine is used to unlock records that were locked by calling FREAD. The form of the call is:

```
XCALL FUNLK(chan)
```

chan = Channel number.

1.5 Closing a shared file

The FCLOS subroutine is called to close a channel that was previously opened to a shared file by calling FOPEN. The form of the call is:

```
XCALL FCLOS(chan)
```

chan = Channel number.

FCLOS unlocks any locked records and closes the file. Other users accessing the file are unaffected. After calling FCLOS, the channel may be reopened to some other file.

1.6 Record Locking Example

In the following example a program performs the following functions:

1. Opens a shared file named "INV.DAT" on channel 2.

2. Reads a record whose record number is stored in RECN into the field named ITEM and waits if the record is locked by another user.

3. Updates the information in the record.

4. Rewrites the record to the same position in the file.

5. Closes the shared file.

```
XCALL FOPEN(2,'INV.DAT',ERRFL)
XCALL FREAD(2,ITEM,RECN,'W',ERRFL)
;<update record>
XCALL FWRIT(2,ITEM,RECN,ERRFL)
XCALL FCLOS(2)
```

1.7 Modifying programs for TSX-Plus

It is a straightforward process to modify DIBOL programs to use the TSX-Plus record locking subroutines. OPEN, CLOSE, READ, WRITE, and UNLOCK statements that apply to shared files must be replaced by the appropriate subroutine calls. Error conditions must be tested by IF statements following the subroutine calls rather than by using the ONERROR statement.

2. Message Transmission Subroutines

Three subroutines are included in the DIBOL support package to allow programs to transfer messages to each other. When running under TSX-Plus these subroutines must be used instead of the DIBOL SEND and RECV statements.

2.1 Message Channels

Messages are transferred to and from programs by using TSX-Plus "Message Channels". A message channel accepts a message from a

sending program, stores the message in a queue associated with the channel and delivers the message to a receiving program that requests a message from the channel. Message channels are totally separate from I/O channels.

Each active message channel has associated with it a one to six character name that is used by the sending and receiving programs to identify the channel. The total number of message channels is defined when TSX-Plus is generated. The names associated with the channels are defined dynamically by the running programs. A message channel is said to be "active" if any messages are being held in the queue associated with the channel or if any program is waiting for a message from the channel. When message channels become inactive they are returned to a free pool and may be reused by another program.

The DIBOL SEND command directs a message to a program by using the name of the receiving program. Under TSX-Plus, a sending program transmits a message using an arbitrary channel name. Any program may receive the message by using the same channel name when it requests a message.

2.2 Sending a Message

The MSEND subroutine is called to queue a message on a named channel. If other messages are already pending on the channel the new message is added to the end of the list of waiting messages. The form of the call is:

```
XCALL MSEND(chan,message,errflg)
```

where

chan = alphanumeric literal or variable that contains the channel name (1 to 6 characters).

message = alphanumeric or decimal literal, field or record that contains the message to be sent.

errflg = Decimal variable into which will be stored one of the following values:

value	meaning
0	No error. Message has been sent.
1	All message channels are busy. (Re-gen TSX-Plus and increase the value of MAXMC parameter).
2	Maximum allowed number of messages are being held in message queues. (Re-gen TSX-Plus and increase the value of MAXMSG parameter).

Note that the maximum message length that may be transferred is defined during system generation by the MSCHRS parameter. If a message longer than this is sent, only the first part of the message will be delivered.

.3 Checking for Pending Messages

The MSGCK subroutine may be called to determine if any messages are pending on a named channel. The form of the call is:

```
XCALL MSGCK(chan,message,errflg)
```

where

chan = alphanumeric literal or variable that contains the name of the channel (1 to 6 characters).

message = alphanumeric or decimal field or record where the received message is to be placed.

errflg = decimal variable into which will be stored one of the following values:

value	meaning
0	No error. A message has been received.
3	No message was queued on the named channel.

If a received message is shorter than the receiving message field the remainder of the field is filled with blanks. If the message is longer than the field, only the first part of the message is received.

2.4 Waiting for a Message

The MSGWT subroutine is used by a receiving program to suspend its execution until a message is available on a named channel. It is much more efficient for a program to wait for a message by calling MSGWT rather than repeatedly calling MSGCK. The form of the call is:

```
XCALL MSGWT(chan,message,errflg)
```

where the arguments have the same meaning as for MSGCK, and the following values may be returned in errflg.

value	meaning
0	No error. A message has been received.
1	All message channels are busy.

.5 Message Examples

In the following example a program sends a message to another program by using a message channel named "SORT" and then waits for a reply to come back through a message channel named "REPLY".

```
XCALL MSEND('SORT','DK:PAYROL.DAT',ERRFL)
IF(ERRFL.NE.0)GO TO ERROR
XCALL MSGWT('REPLY',MSGBF,ERRFL)
IF(ERRFL.NE.0)GO TO ERROR
```

3. Using the subroutines

The subroutines described above are part of the MACRO program called "DTSUB.MAC". Once assembled, the object file for DTSUB (DTSUB.OBJ) may be linked with DIBOL programs that use the record locking or message facilities. An example of a LINK command is shown below.

```
.R LINK
*PROG=PROG,DTSUB,DIBOL
```

4. Miscellaneous Functions

4.1 Determining the TSX-Plus line number

The TSLIN subroutine can be called to determine the number of the TSX-Plus timesharing line from which the program is being run. Real lines are numbered consecutively starting at 1 in the same order they are specified when TSX-Plus is generated. Detached job lines occur next and virtual lines are numbered last.

The form of the call of TSLIN is:

```
XCALL TSLIN(lnum)
```

where "lnum" is a numeric variable capable of holding at least two digits into which is stored the TSX-Plus line number value.