TSX–Plus
User's Reference Manual

Sixth Edition—First Printing—January 1988

S&H Computer Systems, Inc.
1027 Seventeenth Avenue South
Nashville, Tennessee 37212-2299 USA
(615) 327-3670

Contents

Chapter 1

Introduction

TSX–Plus is a high performance operating system for Digital Equipment Corporation PDP–11 and LSI–11 computers, supporting as many as forty concurrent time-sharing users. TSX–Plus provides a multi-user programming environment that is similar to extended memory (XM) RT–11.

- TSX–Plus keyboard commands are compatible with those of RT–11.
- TSX–Plus supports most RT–11 system service calls (EMTs).
- Most programs that run under RT–11 will run without modification under TSX–Plus. This includes RT–11 utility programs such as PIP, DUP, DIR, LINK, and MACRO.
- TSX–Plus uses RT–11 XM version device handlers.
- TSX–Plus provides PLAS extended memory services such as virtual overlays and virtual arrays, as well as support for extended memory regions.

TSX–Plus can simultaneously support a wide variety of jobs and programming languages including COBOL–Plus, FORTRAN, BASIC, DIBOL, DBL, Pascal, C, MACRO, IND, TECO and KED. TSX–Plus is used in educational, business, scientific and industrial environments. It can concurrently support commercial users doing transaction processing, engineering users performing scientific processing, system programmers doing program development, and real-time process control. Numerous application software packages compatible with TSX–Plus are available from other vendors.

TSX–Plus supports RT–11 system service calls (EMTs) as its basic mode of operation. The result is low system overhead and substantially improved performance over systems that emulate RT–11 services. 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.

In addition to the basic RT–11 functionality, TSX–Plus provides extended features such as: process windows; shared file record locking; inter-job message communication; program performance monitoring; command file parameters; logon and usage accounting; directory and data caching; multitasking; and system I/O buffering.

This manual describes all the features unique to TSX–Plus as well as any differences from RT–11. Many of the special features of TSX–Plus are available as EMTs available to the MACRO programmer (see the *TSX–Plus Programmer's Reference Manual*). Access to these features from other languages requires the appropriate subroutine interface.

TSX–Plus will run on any PDP–11 or LSI–11 computer with memory management hardware and at least 128 Kb of memory. The system must also have a disk suitable for program swapping (the swapping disk can be used for regular file storage as well). Time-sharing lines and serial printers may be connected to the system through DH(U,V,Q)–11, DL(V)–11 or DZ(V,Q)–11 communication devices. Both hardwired and dial-up time-sharing lines are supported by TSX–Plus.

1.1 Summary of Chapter Contents

1.1.1 Basic Operations and Starting Time-sharing Sessions

Chapter 2 of this manual describes the procedure for starting and stopping a time-sharing session with TSX-Plus; process windows and subprocesses; the special functions of some control characters; use of the single line editor for correction of typing errors in command lines or data entry fields; and user defined keys.

1.1.2 Keyboard Commands

TSX-Plus responds to keyboard commands that are simple and natural (PRINT, RUN, COMPILE, TYPE, etc.). TSX-Plus also provides user definable commands, and includes support for a menu driven command interface. Chapter 3 discusses command interpretation sequence, user defined commands, the menu interface and lists all TSX-Plus keyboard commands. It provides full descriptions of commands unique to TSX-Plus, describes differences between TSX-Plus and RT-11 commands, and lists the RT-11 commands which are not supported by TSX-Plus. Many of the commands are identical to their RT-11 counterparts; the *RT-11 System User's Guide* should be consulted for detailed descriptions.

1.1.3 Command Files

Frequently used sets of commands may be stored in a disk file and executed by invoking the command file. Parameters may be passed to the command file and interpreted as though they were included at designated places in the file. This provides an alternative way for users to create their own commands.

1.1.4 Detached Jobs

Detached jobs provide a batch-like environment in which jobs may be started that run independently of a terminal. They may be started when TSX-Plus is started, with a keyboard command, or from within a program. Detached jobs inherit the environment characteristics (e.g., process name, logical assignments, logical subset disk mounts, privilege, etc.) of the job which starts them. Detached jobs are command files which may issue most keyboard commands and run programs. All terminal input to detached jobs must come from the command file; all terminal output from detached jobs is discarded unless sent to a log file. Programs run from detached jobs may accept and send messages to other processes via named message channels and may even acquire the file context of other processes.

1.1.5 Printer Spooling System

TSX-Plus provides a convenient and powerful facility for automatic spooling of output to line printers and other devices. The spooler may simultaneously drive several devices. If the line printer is spooled, then simply directing output to device "LP" from a program causes the output to be spooled. A spooled file may designate the name of a form on which it is to be printed. When a form change is required, the spooled device is suspended and a message is sent to the operator requesting the form. After mounting the new form, the operator may print a form alignment file. Once a form is mounted, all files requiring the form are printed without further operator intervention. The operator may also lock a particular form on the printer, preventing automatic form change requests. Keyboard commands are provided to check the status of spooled devices, delete current or pending files from the output queue, and reprint the most recent portion of the current file. Files sent to a spooled device may be released to the printer as data becomes available or held until the output file is closed.

1.1.6 Program Controlled Terminal Options

TSX-Plus allows the programmer to modify terminal handling characteristics during program execution. For example, a program may disable character echoing, use single character activation, use high efficiency output mode, enable lower case input, activate on field width, or disable automatic echoing of line-feed after carriage-return. These terminal options may be selected either by issuing the appropriate EMT request or by writing a special sequence of two or three characters to the terminal.

1.1.7 Differences from RT-11

Some inevitable differences exist between RT-11 and TSX-Plus. Chapter 3 describes the additional keyboard commands provided by TSX-Plus, the minor differences in some commands, and the RT-11 keyboard commands not supported by TSX-Plus. Some other differences between RT-11 and TSX-Plus may not be obvious. The FORMAT utility is not supported. A section is also included on special use and programming characteristics of some utility programs which have caused confusion because of their non-obvious features and their interaction with TSX-Plus.

1.1.8 Appendices

Appendix A describes the FILTIM utility which displays directory information about files, *including* the file creation time.

Appendix B describes the commands and techniques used to work with logical subset disks.

Appendix C describes the TSX-Plus startup error messages.

Appendix D describes the TSX-Plus fatal error messages.

Appendix E describes the TSX-Plus user error messages.

Chapter 2

Basic Operation

2.1 Logging on

Each user communicates with the TSX–Plus system through a time-sharing line (often referred to in this manual simply as a *line*). Lines are normally started by typing a carriage return at the terminal. (If a time-sharing line is set for automatic baud rate detection and the terminal speed is less than 1200 baud, two carriage returns are necessary. Automatic speed selection is allowed for baud rates 110, 300, 1200, 2400, 4800, 9600 and 19200.) The system manager may designate lines to automatically initiate time-sharing whenever the system is started.

When each line is first started, a greeting message will normally be displayed at the terminal. Some lines, especially dial-in lines, may require an additional system password before getting the greeting message. On those lines, the system will prompt for the password with a single exclamation point ("!"). The system password will not be echoed as you type it. If you do not correctly type the system password in two tries or in less than 30 seconds, then the system will hang up. (The actual time limit may vary on your system.) The system password is selected during system generation and may be obtained from your system manager.

The system manager may assign a start-up command file to be executed whenever a time-sharing line is started. A start-up command file contains initialization commands for a line. It can end by either starting execution of some program, or by waiting for a system command. (The default keyboard monitor prompt is a period.) It is possible for a start-up command file to lock a program to a line so that on termination of the program the line is automatically logged off and an optional logoff command file is executed. Typing CTRL–C will not abort start-up or logoff command files.

The system manager may also require *logon* authorization. In this case, the system manager assigns each user a user name and password. After the greeting message, the message "Logon please:" is printed. The user responds by typing the user name followed by a carriage return. TSX–Plus then requests the password. The password is not echoed to the terminal as it is typed. After the user name and password are validated, TSX–Plus types the message "Welcome to the system". The system manager may also designate a *start-up* command file for each account to control system access and set certain operating parameters.

The following example illustrates a typical logon sequence:

```
(carriage-return pressed)
(greeting message displayed)

23-Jan-84   13:39:50
Line #2

Logon please:JOHNSON
Password:MYPASS            (Password is not displayed.)
```

```
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 forward 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. Changing your password requires SETNAME privilege. The following example shows the password being changed from OLDP to NEWP. Note that neither the old nor the new password would actually be echoed.

```
(carriage return pressed)
(greeting message displayed)

20-Jun-83  14:17:43
Line #5

Logon please:JOHNSON
Password:MYPASS/NEWPASS      (Passwords are not displayed.)
Welcome to the system
.                    (TSX-Plus is now waiting for a system command.)
```

If the system manager has not required *logon* account authorization, then the system will either execute a start-up command file or simply display the monitor prompt (".") and wait for a system command to be entered.

2.2 Logging off

The OFF keyboard command is used to terminate (logoff) a time-sharing session.

The system manager may specify a *logoff* command file to be executed whenever a job logs off. CTRL–C will not abort a *logoff* command file.

2.3 Control characters

Certain control characters have special meaning to the system. These *control characters* control certain terminal operations. They are entered by holding down the "CTRL" key while pressing the selected character.

Some of these special characters may be redefined locally during system generation. (CTRL–B: the PRINTWINDOW command; CTRL–W: the subprocess switch.)

Note also that the function of some *control characters* may be altered when the single line editor is in use; see section 2.5 for more information on the Single Line Editor.

Some *control characters* can also be used as defined keys, which causes the defined text to be substituted when the key is typed. See section 2.6 for more information on defined keys.

NULL The null character is ignored when typed. (CTRL–@, CTRL–SPACE)

CTRL–A Is the special command character for the time-sharing line/CL line cross connection. See the SET HOST command.

CTRL–B Causes the current process window to be transmitted to the printer. See the SET PRINTWINDOW command. (May be redefined during system generation.)

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–Cs are required to interrupt its execution. CTRL–C will not interrupt start-up or logoff command files.

CTRL–D May interrupt the execution of the program and transfer control to the symbolic debugger. This depends on whether the debugging facility has been included during system generation and whether the SET CTRLD DEBUG command has been issued. See the description of the SET CTRLD DEBUG command in Chapter 3, and the description of the program debugger in the *TSX-Plus Programmer's Reference Manual*.

CTRL–O Suppresses program output to the terminal until one of the following conditions occurs:

- A second CTRL–O is typed
- The program returns to the monitor
- The running program issues a .RCTRLO EMT

CTRL–P Has no special meaning to TSX–Plus, but is used as the command signal character for the RT–11 VTCOM program and as the console break character on some types of processors.

CTRL–Q Resumes printing on the terminal from the point at which printing was previously suspended by CTRL–S.

CTRL–R Causes the current characters in the terminal input buffer to be displayed. This can be used to check the actual contents of an input line when rubout editing has been done on the line. Also used by KED, KEX and K52 to repaint the screen.

CTRL–S Temporarily suspends output to the terminal until CTRL–Q is typed.

CTRL–U Deletes the current input line.

CTRL–W Used to switch to a TSX–Plus subprocess. However, the subprocess switch character may be redefined during system generation. Typing two successive CTRL–Ws passes one CTRL–W to the program. Used by KED, KEX and K52 to repaint the screen.

CTRL–Z Indicates *end of file* when input is being read from device TT.

ESCAPE Used for terminal control sequences. The escape character is echoed as a dollar sign ("$"), thus preventing direct entry of terminal control sequences from the keyboard.

CTRL– Used to break a time-sharing line/CL line cross connection. Only significant to the system during a cross connection. See the SET HOST command.

DELETE Removes preceding character (on the same line) from the terminal input buffer. See also the description of the rubout filler character in the *TSX-Plus Programmer's Reference Manual*.

The normal function of these keys may be altered during program execution. The *TSX-Plus Programmer's Reference Manual* describes program controlled terminal options that influence these functions. When enabled, the single line editor defines additional special purpose keys; see section 2.5 for further information about the single line editor.

2.4 Subprocesses and Process Windows

TSX–Plus allows a single terminal to control several processes at once. Control may be switched from one process to another by typing a CTRL–W (hold down the CTRL key while pressing the W key), followed by a single digit (do not hold down CTRL while typing the digit). The digit identifies the number of the subprocess to be logically attached. Switching the terminal's logical connection between processes in this manner can be thought of as opening a *window* into each process. (A subset of the subprocess facility was known in earlier versions of TSX–Plus as *virtual lines*.) The ability to use subprocesses requires SUBPROCESS privilege.

When initially logging on to TSX–Plus, the terminal is connected to the *primary* process, also called process window number zero (0) or the primary time-sharing line. The terminal may be connected to a different

subprocess at any time by typing the CTRL–W–*digit* sequence. You may then proceed with ordinary time-sharing operations on the subprocess while processing continues on the primary line. When initiating the subprocess, the system will display a string to identify the subprocess number (same as <digit>) and the TSX-Plus job number. For example, switching to subprocess number one might result in something like the following:

```
.FORTRAN BIGJOB     [Start large compilation]
^W1                 [Switch to subprocess - this does not echo]
1>Line # 21         [Greeting for subprocess 1, job number 21]

.SYSTAT             [Perform normal time-sharing operations on subprocess]
Uptime: 01:20:21
System use:   Run=13%, I/O-wait=1%, Swap-wait=0%, Idle=84%
I/O Activity: User I/O=4%, Swapping I/O=0%

Job   Line   Pri   State    Size   Connect    CPU time  Program  Job name
---   ------ ---   -------  -----  ---------  --------  -------  ----------
1     1(0)    50   TI       62Kb   01:21:00   00:01:55  SRCCOM   HAIKU
2     2(0)    40   IN       62Kb   00:10:00   00:00:40  FORTRA   Pacific
16    Det.    50   SL       62Kb   01:21:00   00:00:01  RTSORT   SYSTEM
17    Det.    50   SL       30Kb   01:21:00   00:00:00  WINPRT
21*   2(1)    50   IN       36Kb   00:01:00   00:00:01  KMON     Pacific
.^W0              [Bell rings, return to primary - this does not echo]
.MAIN.            [Display continues where it left off]
SBRTN1
SBRTN2
SBRTN3
SBRTN4
SBRTN5
.
```

It is possible to specify during a start-up command file the name of a command file to execute when initiating a subprocess. See the description of the SET SUBPROCESS command in the *TSX-Plus System Manager's Guide* for more information on subprocess initiation command files.

The relative subprocess number is specified by the digit which follows the CTRL–W. The job number (process number) is assigned automatically by the system. Subprocesses are assigned the first available job number from among those reserved system-wide for subprocesses. Job numbers are allocated first for all primary processes, in the order in which they were declared during system generation. The next range of job numbers is reserved for detached jobs. Subprocesses are assigned the first unused job number following the last number reserved for detached jobs. The maximum number of subprocesses which may be active for all users is selected during system generation. In addition, each primary process may be restricted to a maximum number of subprocesses. If either of these quotas is filled, then the CTRL–W–*digit* sequence used to initiate a subprocess will be ignored. The SYSTAT command may be used to determine information about primary processes and subprocesses.

Switching to a subprocess by typing CTRL–W–*digit* has the effect of *logically* disconnecting the terminal from the primary process and connecting it to a different subprocess. The terminal may be reconnected to the primary process at any time by typing CTRL–W–0. (Zero is the subprocess number for the primary process.)

To stop a subprocess, connect the terminal to it and use the OFF command. The terminal will then be reconnected to the primary process. Subprocesses which are active when the primary process is logged off are aborted. It is not necessary to log off from a subprocess before connecting to a different subprocess. However, only a limited number of subprocess job slots are available in the system; if they are all busy, then no one will be able to initiate a new subprocess. In addition, restrictions can limit or prevent their use on

particular lines or by particular users.

Process windows have the option of retaining the screen display when they are disconnected and then refreshing the display when they are reconnected. Or, they may begin from the terminal's current cursor position. Using the window refresh option reduces the confusion which can result from working with multiple subprocesses from the same terminal. The SET WINDOW command is used to control process windowing. The simplest form of this command, which enables the process window refresh feature for the current process, is:

```
SET WINDOW
```

See Chapter 3 for a full description of the SET WINDOW command.

CTRL-W is also used by the KED, KEX and K52 editors to repaint the screen. Typing two successive CTRL-Ws passes one CTRL-W through to the program and will cause these editors to repaint the screen. Note that they will also accept CTRL-R to repaint the screen and only one need be typed. If windows with refresh are in effect, you may also refresh the screen by typing CTRL-W followed by the current relative subprocess number.

A single process may use multiple windows by using the EMTs to create and select windows described in the *TSX-Plus Programmer's Reference Manual*. Use of the window refresh option requires SYSGBL privilege.

Enabling window refresh also permits use of the PRINTWINDOW screen print facility. This allows an image of the current screen display to be sent to a printing device at any time a window with refresh is in effect by simply typing a control character. See the description of the SET WINDOW and SET PRINTWINDOW commands in Chapter 3 for a more complete description of screen refreshing when switching among different process windows with refresh and use of the PRINTWINDOW facility.

If a process is disconnected by switching to a different process window while a program is running, the program continues to execute just as it would if it were still connected. However, the priority of the disconnected process is reduced by an amount selected during system generation. If the disconnected process reaches a point at which it requires terminal input or it would begin to lose terminal output, then its execution is suspended and the terminal bell is rung to indicate that one of its disconnected subprocesses requires service. See also the description of the SCROLL option to the SET WINDOW command.

When a separate process window is initiated, it *inherits* most of its characteristics from those current for the primary process. (Detached jobs inherit their environment in much the same way, except that detached jobs inherit from the (sub)process which starts them, not always from the primary process and they always begin executing a command file.) The startup command file for the primary process is **NOT** executed for each process window. However, a command file may be specified to be executed when a subprocess is initiated; see the description of the SET SUBPROCESS command in the *TSX-Plus System Manager's Guide*. Each process may change its characteristics (within authorized ranges) subsequent to its initiation. If the process window screen refresh option is enabled, then the screen of a subprocess window is cleared during initiation and windowing is begun, as if an implicit SET WINDOW command had been issued for the subprocess. The characteristics inherited by the subprocess do not include an executing program; the subprocess is entered at the keyboard monitor, waiting for a command. The characteristics inherited when a subprocess (or detached job) is initiated include the following:

- Project/programmer numbers and process name
- Authorized and Set privileges
- File access controls
- Maximum authorized priority
- Logical assignments (ASSIGN command)
- Logical subset disk mounts (MOUNT LD command)

- Mounts of disks to enable caching
- User defined commands
- User command interface (SET KMON UCI command)
- Single line editor status (SET SL command)
- User defined terminal keys (DEFINE KEY command)
- Monitor prompt string (SET PROMPT command)
- Form name (FORM command)
- SET PRINTWINDOW device name and type

The terminal may be switched back and forth among any of its active processes without reinitiating them. While it is disconnected, each process retains whatever characteristics it has acquired; they are not re-established from the primary process for each connection, only when initiated.

2.5 Single Line Editor

It is often desirable to correct typing mistakes made in the entry of command input lines or to correct data entry fields while executing some program. This can be done to a limited extent by use of the DELETE and CTRL–U keys, however much more editing capability is available if the *Single Line Editor* facility is used. To use the single line editor, the system manager must enable it during the TSX–Plus system generation process. In addition, it must be turned on for each time-sharing line before use. To use the single line editor, issue the command:

```
SET SL ON
```

either in a start-up command file or as a keyboard command. The single line editor may be used only with VT200, VT100 or VT52 type terminals.

Several other SET options are available for use with the single line editor. These are described in Chapter 3 in the section on keyboard commands.

The single line editor accepts special key commands to direct its operation. There are two modes of operation of the single line editor: normal mode and KED mode. The following tables summarise the functions performed in each mode by the editing control keys.

The "RECALL" keyboard monitor command is used to display the set of saved commands and to recall a specific saved command either by specifying an index number or a string that matches the beginning of the command. These are described in Chapter 3 in the section on keyboard commands.

Sometimes a particular cycle of commands is used repeatedly. For example, EDIT, MACRO, LINK, and RUN. If you wish you can now define which commands constitute a cycle and then easily recall them in the order they should be executed. To do this, perform the commands in the cycle the first time. Then use <UP-ARROW> to recall the first command of the cycle and press <PF1><DOWN-ARROW> to mark the beginning of the cycle. After you execute each command in the cycle, press <PF1><UP-ARROW> to recall the next command of the cycle. The cycle is cleared when you enter any new command.

By typing CTRL–A you can now toggle between insert and overtype mode. In insert mode, any existing characters on the line to the right of the cursor are moved right to make room for the characters being inserted. In overtype mode, any characters you type in the middle of a line will replace existing characters as you type over them. Insert mode is automatically selected each time you begin to work on a new line.

Single Line Editor Key Functions

Normal (RT–11 compatible) Mode

Primary Key Functions (Not prefixed by PF1)	
Key	Function
Up arrow	Retrieve previous input lines
Down arrow	Retrieve line from save buffer
Left arrow	Move cursor left one character
Right arrow	Move cursor right one character
BACK SPACE	Exchange char under cursor with char to right
DELETE	Delete character to left of cursor
LINE FEED	Delete word to left of cursor
RETURN	Pass current line to running program
PF1	Used before other keys to perform second function
PF2	Not implemented
PF3	VT52: Delete from cursor to right end of line
PF4	VT100: Delete from cursor to right end of line
CTRL-U	Delete from cursor to left end of line
CTRL-R	Redisplay current line
Secondary Key Functions (Keys prefixed by PF1)	
Key	Function
Left arrow	Move cursor to left end of line
Right arrow	Move cursor to right end of line
BACK SPACE	Exchange char under cursor with char to left
DELETE	Retrieve last deleted character
LINE FEED	Retrieve last deleted word
RETURN	Truncate from cursor to end of line and execute
S	Save current line in *save buffer* for recall later
X	Retrieve line currently in the *save buffer*
PF1	Ignored
PF2	Not implemented
PF3	VT52: Retrieve last deleted line
PF4	VT100: Retrieve last deleted line
CTRL-U	Retrieve last deleted line

In addition to the normal mode of operation for the single line editor, the keypad or KED mode also permits use of some of the keypad functions in a fashion similar to KED or K52. These keypad operations are available in addition to the functions described above for the single line editor. In order to use the extended (KED) mode with the single line editor, issue the following command:

```
SET SL KED
```

When the KED mode is enabled, the following tables describe the additional functions available to the single line editor.

Single Line Editor Key Functions

KED Mode

Primary Key Functions (Not prefixed by PF1)	
Key	Function
0	Move cursor to left end of line
1	Move cursor one word in current direction
2	Move to end of line in current direction
3	VT100: Move cursor one char in current direction
4	Set direction forward (left to right)
5	Set direction backward (right to left)
6	VT52: Delete character under cursor
7	Not implemented
8	Not implemented
9	VT52: Delete word under cursor
-	VT100: Delete word under cursor
,	VT100: Delete character under cursor
ENTER	Pass current line to running program
Secondary Key Functions (Keys prefixed by PF1)	
Key	Function
0	Delete from cursor to right end of line
1	Change case of char under cursor
2	Delete from cursor to right end of line
3	Not implemented
4	Move cursor to right end of line
5	Move cursor to left end of line
6	VT52: Retrieve last deleted character
7	Not implemented
8	Not implemented
9	VT52: Retrieve last deleted word
-	VT100: Retrieve last deleted word
,	VT100: Retrieve last deleted character
ENTER	Truncate from cursor to end of line and execute

To return the single line editor to the normal mode, issue the command:

```
SET SL NOKED
```

To disable the single line editor altogether, issue the command:

```
SET SL OFF
```

In addition to editing command lines, the single line editor may be used to edit data entry fields during program execution. However, there are some restrictions on such use. If any of the following conditions exist, then the single line editor may NOT be used to edit program data fields:

- The SET SL OFF command has been issued
- Bit 4 (EDIT$, mask 20) is set in the Job Status Word (JSW)
- Bit 12 (TTSPC$, mask 10000) is set in the JSW
- The program is using high efficiency terminal mode
- The program is using escape sequence activation
- Input is being accepted via the .TTYIN EMT (except as below)

In order to use the single line editor to edit data entry fields within programs which accept input via the .TTYIN EMT, the following command must be issued prior to execution of the program:

```
SET SL TTYIN
```

Since COBOL–Plus programs accept terminal input via the .TTYIN EMT, the following steps must be used to edit data entry fields while executing COBOL–Plus programs:

1. SET SL ON
2. SET SL TTYIN
3. From within the executing program, CALL "ESCAPE–OFF"

The TSX–Plus single line editor is generally compatible with that provided with RT–11, with the following exceptions:

- The HELP key (PF2) is not implemented.
- The SET SL LEARN mode is not implemented.
- The SET SL ASK option is ignored. (The current TSX–Plus terminal type information is used.)
- The SET SL SYSGEN option is ignored.
- The maximum width of a command line or input field that is accepted is 80 characters. The SET SL WIDTH=*n* option is ignored.
- Using the single line editor with .TTYIN input does not force a new line.
- The single line editor is implemented as a TSX–Plus overlay region and does not require the SL pseudo-device handler.
- A 300 byte buffer is used to store as many commands as will fit. Typically, about 10 commands can be saved, but this will vary with the length of the commands.

2.6 User defined terminal keys

It is possible to specify character strings which will be substituted for a character when that character is typed. This feature is most commonly used to associate frequently typed commands with some of the terminal function keys, but it may be used to associate a character string with almost any key.

Key substitutions are performed by the single line editor; hence, substitutions are only done on terminal input processed by the single line editor.

The DEFINE KEY keyboard command is used to specify a key substitution. The form of this command is:

```
DEFINE KEY[/qualifiers] key string
```

where *key* is the name of the key that is associated with *string*. For example, the following command associates the SHOW TIME command with the PF2 key:

```
DEFINE KEY PF2 "SHOW TIME"
```

String may be up to 64 bytes long. Current key definitions may be displayed by the SHOW KEYS command.

If *string* contains no leading or trailing spaces, no "@" symbol, and no control characters, the quotation marks may be omitted. Thus, the command may also be specified as:

```
DEFINE KEY PF2 SHOW TIME
```

If *string* contains an "@" symbol, then *string* must be enclosed in quotation marks. For example, the following key definition could be used to cause the PF2 key to invoke a command file:

```
DEFINE KEY PF2 "@DL3:CNDFIL"
```

If the key being defined is a control character, enclose the key in quotation marks and place a caret (^) in front of the letter that is to be converted to a control character. For example, the following command causes CTRL-T to display the current time of day:

```
DEFINE KEY/LETTER/NOECHO "^T" SHOW TIME
```

A key definition is removed by using a DEFINE KEY command with no substitution string. Thus, to remove the PF2 key definition use the following command:

```
DEFINE KEY PF2
```

The following table lists key names which may be used:

Key name	Actual key
The following keys may only be defined in KED mode	
KP0,KP1,...,KP9	digits 0 to 9 on numeric keypad
PERIOD	period key (.) on numeric keypad
COMMA	comma key (,) on numeric keypad
MINUS	minus key (-) on numeric keypad
ENTER	ENTER key on numeric keypad
The following keys are not affected by KED mode	
PF1,PF2,PF3,PF4	PF1,PF2,PF3,PF4
UP	up arrow key
DOWN	down arrow key
LEFT	left arrow key
RIGHT	right arrow key
FIND	FIND key on VT200
INSERT	INSERT HERE key on VT200
REMOVE	REMOVE key on VT200
SELECT	SELECT key on VT200
PREV	PREV SCREEN key on VT200
NEXT	NEXT SCREEN key on VT200
HELP	HELP key on VT200
DO	DO key on VT200
F6,F7,...,F20	Function keys on top row of VT200

Key substitutions may be disabled/enabled with the command:

```
SET SL [NO]SUBSTITUTE
```

The default mode is SUBSTITUTE which enables key substitutions by the single line editor.

If you define substitutions for any of the keys of the numeric keypad other than the PF keys, the single line editor must be in KED mode (SET SL KED).

Note that the PI handler which controls the console terminal on the Professional normally provides VT100 support for the console terminal, not VT200. Thus, the function keys on the top row and the DO, HELP, etc. keys may not normally be defined. However, it is possible to enable the Pro console for VT200 compatible operation by sending it the control sequence "<ESC>[?39h".

If a defined key appears not to be working, first verify that the definition is correct with the SHOW KEYS command. Then, use the SHOW SL command to verify that key substitution is in effect and that KED mode is enabled for numeric keypad keys.

You may specify a key definition for a key which is normally used to perform some single line editor function (e.g., up arrow). When this is done, the key definition takes precedence over the normal function.

The following qualifiers can be used with the DEFINE KEY command:

/[NO]ECHO Determines whether the substitution string will be printed at the terminal at the time the substitution takes place. The default mode is /ECHO.

/[NO]TERMINATE Determines whether a carriage-return and line-feed are appended to the end of the substitution string when it is substituted. Use /NOTERMINATE with a definition that specifies a part of a command. The default mode is /TERMINATE. You may not specify /NOECHO and /NOTERMINATE together. For example, the following definition associates the PF2 key with the word HELP followed by one space, without termination:

```
DEFINE KEY/NOTERMINATE PF2 "HELP "
```

/[NO]GOLD Determines whether the specified key must be prefixed by the GOLD (PF1) key. If /GOLD is specified then the substitution only takes place if you type the PF1 key followed by the specified key. If /NOGOLD is specified (the default case) then the substitution takes place when the specified key is pressed without being prefixed by the PF1 key. /GOLD definitions are not available if the PF1 key itself has been assigned a key definition. The following two definitions associate the SHOW TIME command with the unprefixed PF2 key and the SHOW DATE command with the two key combination PF1 PF2:

```
DEFINE KEY PF2 SHOW TIME
DEFINE KEY/GOLD PF2 SHOW DATE
```

/LETTER Specifies that the key being defined is not a function key, but is a key such as a letter, digit, or punctuation mark which is part of the typewriter portion of the keyboard. If /LETTER is specified, then *key* must be a single letter from the non-function-key portion of the keyboard. The /GOLD qualifier may be used in conjunction with /LETTER to cause the substitution to take place only if the specified letter is prefixed by the PF1 key. The default mode is /NOLETTER. For example, the following commands cause a pound sign (#) to be substituted for the dollar sign ($) and the SHOW JOBS command to be substituted for the "PF1 *dollar sign*" combination:

```
DEFINE KEY/LETTER/NOTERMINATE $ #
DEFINE KEY/LETTER/GOLD $ SHOW JOBS
```

Deleting or redefining a substitution for a letter which has already been defined requires that key substitution be turned off first. Otherwise, the defined string is substituted for the letter as the DEFINE KEY command is typed. If you need to delete or redefine a letter substitution, invoke the SET SL NOSUBSTITUTE command to turn off substitutions, then use the DEFINE command, then SET SL SUBSTITUTE.

Control characters may also be included in key substitution strings. To include control characters in a key definition string, enclose the substitution string in quotation marks and type a caret (^) in front of the letter that is to be a control character. For example, the following command defines numeric keypad key zero (0) as CTRL–U (SL must be in KED mode):

```
DEFINE KEY/NOTERMINATE KP0 "^U"
```

Control characters count as one character in the string even though two characters are used to specify them. If the DEFINE command is placed within a command file, two caret characters must be specified to denote each control character because command file processing translates two consecutive caret characters into a single caret. Thus, if the previous DEFINE command were placed in a command file, it would be written as:

```
DEFINE KEY/NOTERMINATE KP0 "^^U"
```

It is possible to define a key with a CTRL–W–*digit* sequence to cause a switch to a subprocess. However, this is less powerful than actually typing CTRL–W–*digit* because the switch with a defined key can only be used at a time when the single line editor is accepting input, not when a program is executing. The following key definition defines PF2 to switch to subprocess one:

```
DEFINE KEY/NOTERMINATE PF2 "^W1"
```

Key definitions are inherited from the primary process when subprocesses are initiated.

Key definitions for each job are stored in a local named region with the name KEYDEF. This region is created when the first key definition is made and is deleted when the last key definition is deleted or the job logs off. The size of the KEYDEF region (in bytes) is equal to KEYMAX*68.

Chapter 3

Keyboard Commands

3.1 Keyboard command interpretation

When a system command line is typed in, TSX–Plus attempts to interpret it by sequentially applying the following rules:

1. If the User Command Interface is in effect, then each time TSX–Plus is ready to accept a keyboard command it passes control to the current UCI program. The UCI program may be selected with the SET KMON UCI[=*filnam*] command. See the description of SET KMON command and the section in this chapter on the User Command Interface for more information.

2. If the command line begins with an at-sign ("@"), TSX–Plus attempts to execute a command file. If no device is specified with the command file name, device DK: is assumed. The default extension for command files is .COM. If the SET KMON IND command has been issued, command files will execute under the control of the IND program. Command files may be specified for execution as normal command files, regardless of whether KMON is set IND or NOIND, by starting the command with a dollar-sign (e.g., "$@filnam"). Conversely, command files can be forced to execute under the IND utility, regardless of whether KMON is set IND or NOIND, by typing: "IND filnam".

   See Chapter 4 for further information on the execution of command files by TSX–Plus.

3. If user-defined commands have been allowed during TSX–Plus generation, and if UCL is set FIRST either in TSGEN or subsequently by the keyboard SET UCL FIRST command, then commands are checked at this point to see if they are user-defined commands. However, if the first character on a command line is the underline character ("_") then the command on that line is not checked to see if it is a user-defined command. See the next section for more information on user-defined commands.

4. TSX–Plus next attempts to identify a 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.

5. If user-defined commands have been allowed during TSX–Plus generation, and if UCL is set MIDDLE either in TSGEN or subsequently by the keyboard SET UCL MIDDLE command, then commands are checked at this point to see if they are user-defined commands. However, if the first character on a command line is the underline character ("_") then the command on that line is not checked to see if it is a user-defined command.

6. If the command has not yet been identified, then 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:. In either case, if the SET KMON IND keyboard command has been issued, it will be executed under control of the IND program. 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 Chapter 4 for more information on command files.

7. If a 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, it is executed as if there were an R command in front of the program name. Note that while both RT–11 and TSX–Plus allow a line of text input to be passed to a program with the RUN command by specifying it after the program name, TSX–Plus also allows this to be done with the R command and with an implied R command line when the program name is specified as the command keyword. See example 6 below.

8. If user-defined commands have been allowed during TSX–Plus generation, and if UCL is set LAST either in TSGEN or subsequently by the keyboard SET UCL LAST command, then commands are checked to see if they are user-defined commands at this point. However, if the first character on a command line is the underline character ("_") then the command on that line is not checked to see if it is a user-defined command.

9. If a command has not been identified after all the above steps have been tried, then it is reported as an unrecognizable command. The error message is:

```
?KMON-F-Unrecognizable command
```

The following list summarizes the sequence of events which occur in the processing of keyboard commands. If a command can be recognized at any of these steps, then the appropriate command is executed and the remaining steps are skipped.

- Call user command interface if one is active.
- If command begins with "@", execute command file on DK:.
- If UCL FIRST, check for user-defined command.
- Check for system command.
- If UCL MIDDLE, check for user-defined command.
- Check for command file on DK:.
- Check for command file on SY:.
- Check for SAV file on SY:.
- If UCL LAST, check for user-defined command.
- Report unrecognizable command.

If user-defined command support has not been included in TSGEN, or if the SET UCL NONE command has been issued, or if a command line begins with the underscore character ("_"), then the command interpreter never attempts to interpret a command as a user-defined command. Attempts to issue user-defined commands under these conditions will also be reported as unrecognizable commands.

Examples

1. Execute the command file DK:PURGE.COM.

```
@PURGE
```

2. Run a program named DUMP on device SY:.

```
R DUMP
```

3. Run a program named PAYROL on device DX1:.

```
RUN DX1:PAYROL
```

4. List all files on RK1:

```
DIR RK1:
```

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

```
LOADIT PROG2
```

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

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

3.2 CCL Commands

Some keyboard commands are directly executed by the system, while others call utility programs (like PIP and DIR) to perform the task. The job of translating keyboard commands and passing the appropriate file specifications and switches to the utilities is done by a special program called CCL. Accordingly, these types of commands are called CCL commands. The CCL program converts the keyboard command passed to it into a pseudo command file which executes the utility and is always terminated with a control-C. The following examples for the TYPE and EXECUTE CCL commands demonstrate the type of pseudo command file constructed for them by CCL.

```
.TYPE AREA.CBL
_R PIP
TT:=DK:AREA.CBL
^C

.EXECUTE AREA
_R COBOL
DK:AREA=DK:AREA
^C
_R CBLINK
DK:AREA=DK:AREA
^C
_RUN DK:AREA
```

To determine whether or not a command is a CCL command simply use the SET CCL TEST command to see if the command has a CCL expansion.

3.3 User-defined Commands

It is possible to define your own keyboard commands in terms of system commands and other keyboard commands. New keyboard commands may be defined according to the following syntax:

```
name:==string
```

where *name* is a 1 to 11 character command keyword which must begin with a letter or digit and may consist of letters, digits, and the underscore symbol ("_"), and *string* is a string of up to 80 characters which defines the body of the command.

For example, the following command would define the keyword NOW as being equivalent to the TIME command:

```
NOW :== TIME
```

Multiple commands may be included in the body of a command definition by separating them with the backslash character ("\"). For example:

```
NOW :== DATE\TIME
```

An up-arrow character ("^") may be included in the command body to cause all text on the command line following the command keyword to be inserted in the command body at the position of the up-arrow. For example, if a command is defined as follows:

```
NAMES :== DIR/ORDER:NAME ^
```

Then, if this command is invoked by typing:

```
NAMES *.MAC
```

The resulting command will be equivalent to:

```
DIR/ORDER:NAME *.MAC
```

More than one up-arrow character may occur in the command body. The parameter string specified when the command is invoked is substituted for each occurrence of the up-arrow character.

A user defined command may invoke other user-defined commands within its definition provided that the definition does not call on itself and provided that the other commands are defined at the time the command is invoked. For example, the following command definitions would be legal:

```
NOW :== SHOW DATE\SHOW TIME
STATUS :== NOW\SYSTAT
```

However, the following pair of commands would cause an infinite loop:

```
ONE :== TWO
TWO :== ONE
```

It is possible to allow abbreviation of user-defined commands. To do this, place an asterisk character ("*") within the keyword at the point of the minimum length abbreviation. For example, the definition:

```
ST*ATUS :== NOW\SYSTAT
```

would allow the STATUS command to be abbreviated to STAT or ST but not to S. Note that the portion of the command typed must match the definition (except for any asterisk in the definition) up to the last character entered.

You can specify the order in which the TSX–Plus command interpreter checks for user-defined commands by use of the SET UCL command. This command has four forms:

```
SET UCL FIRST
SET UCL MIDDLE     (This is the recommended setting.)
SET UCL LAST
SET UCL NONE
```

If the SET UCL FIRST command is used, user-defined commands will be processed before system commands. This allows user-defined commands to replace system commands but makes the processing of system commands slower. This is the required setting if it is necessary to replace some system commands.

If the SET UCL MIDDLE command is used, user-defined commands are processed after system commands but before checking for command files and SAV files with names that match the command keyword. Using this setting, it is not possible to replace a system command with a user command, but both system commands and user-defined commands are processed relatively quickly. This is the recommended setting unless it is desirable to replace system commands.

If the SET UCL LAST command is used, a command will not be checked to see if it is a user-defined command until after it has been checked to see if it is a system command, the name of a command file on DK, the name of a command file on SY, or the name of a SAV file on SY. Using this setting, it is not possible to replace a system command with a user command and user commands cannot have the same name as command files or SAV files. System commands are processed quickly (the same speed as SET UCL MIDDLE), but the processing of user-defined commands is slow. This is the appropriate setting only if user-defined commands are desired, but command files already exist whose names would conflict with user-defined commands. Existing command files which are short and merely execute system commands should be replaced by user-defined commands.

If the SET UCL NONE command is used, user-defined commands are never interpreted. In this mode, attempts to invoke user-defined commands will result in the error:

```
?KMON-F-Unrecognizable command
```

The following list illustrates where the FIRST | MIDDLE | LAST setting causes the command interpreter to check for and process user-defined commands:

```
First   ⟶
                See if command is a system command
Middle  ⟶
                Look for command file on DK: with command name
                Look for command file on SY: with command name
                Look for SAV file on SY: with command name
Last    ⟶
```

See the beginning of this chapter for further information on the command interpretation process.

If an underscore character ("_") is typed in front of a keyboard command, this is a signal to the system that the command is *not* to be interpreted as a user-defined command. For example, if the NOW command has been defined as shown above, then the following command will be recognized as a user-defined command and will cause the date and time to be displayed:

```
NOW
```

However, the following command would not be recognized as a user-defined command and would be rejected as an undefined command since there is no NOW command defined by the system (unless there is a command file or SAV file on SY: with the name NOW):

```
_NOW
```

There are two reasons for using the underscore prefix. If UCL has been set FIRST, the underscore prefix can be used to speed up the processing of large numbers of system commands that could be present in frequently executed command files. For example, the NOW command could be defined as follows to cause it to execute faster:

```
NOW :== _DATE\_TIME
```

The second and more important reason for using the underscore prefix is to allow user-defined commands to be defined which replace system commands but which use the system commands in their definitions. (Note that it is necessary to SET UCL FIRST to allow system command replacement.) For example, the following definition replaces the system DIRECTORY command with a user-defined command that has the same name but which always orders the files alphabetically:

```
DIR*ECTORY :== _DIR/ORDER:NAME ^
```

If the body of this command were not preceded with the underscore character, then it would be a circular definition and cause a futile loop.

A user-defined command can cause a command file to be executed. However, the command file invocation must be the last (or only) command defined within the body of the command. For example, the following definition causes all .BAK files to be deleted, and a command file named QUIT to be executed when the OFF command is used (the QUIT command file could end with an _OFF command to do the actual logoff):

```
OFF :== _DEL *.BAK/NOQ\@QUIT
```

A user-defined command may be replaced at any time by simply entering a new definition for the command. A command may be deleted by entering its name and ":==" without a command body. For example, the following command deletes the definition of the NOW command:

```
NOW :==
```

The SHOW COMMANDS keyboard command may be used to display a list of all current user-defined commands.

Commands defined by one time-sharing user are *local* to that user and do not affect other users. However, when a subprocess is started the subprocess *inherits* the user-defined commands that are in effect for the primary line at the time that the subprocess is started. User-defined commands created for a subprocess are not available from other subprocesses or from the primary line. All user-defined commands for a job are reset (forgotten) when the job logs off.

When defining multiple commands from a command file (especially a start-up command file) it is useful to prevent repeated execution of the TSXUCL program and return to the keyboard monitor. The TSX-UCL program automatically determines whether it was called from TSKMON in response to a user-defined command definition or if it was run directly. When run directly, it accepts command definitions until it finds a blank line and then returns to the keyboard monitor. This is significantly faster only when defining multiple commands from within a command file. The following excerpts from a typical start-up command file demonstrate the repeated entry method and the direct definition method which is faster for multiple definitions:

Slower	Faster
!Slower method	R TSXUCL !Faster method
Q :== SPOOL LP,STAT	Q :== SPOOL LP,STAT
WHEN :== SHOW TIME	WHEN :== SHOW TIME
NEW :== DIR ^/D	NEW :== DIR ^/D
!End of command file	!Blank line required

The system manager must enable user-defined commands by setting the U$CL flag in TSGEN and the program TSXUCL.SAV must be on SY: in order to process user-defined commands. The maximum number of commands which can be defined by any job is set by the UCLMNC command in TSGEN. The default order for interpretation of user-defined commands during command processing is determined by the TSGEN parameter UCLORD. This may be overridden by the keyboard command SET UCL FIRST | MIDDLE | LAST | NONE for individual jobs.

3.4 User Command Interface

The User Command Interface (UCI) allows a user-provided program to take over the job of command acquisition from the TSX–Plus keyboard monitor. When UCI is enabled, the user-written program will be called by the TSX–Plus keyboard monitor each time it is ready to accept a new command. It is then the responsibility of the user-written program to prompt for a command and accept it from the terminal. The program may then perform the appropriate actions, chain to other programs or pass commands on to the keyboard monitor. This provides a mechanism for such applications as a command menu.

UCI is enabled with the command:

```
SET KMON UCI[=filnam]
```

which may be entered either from the keyboard or in a start-up command file. If the optional *filnam* is omitted, then the keyboard monitor passes command control to the program SY:UKMON.SAV. This is appropriate when a common command interface is desired for multiple lines. If different command interfaces are desired for different lines, then the file name of the appropriate user-written command interface should be specified.

After UCI is enabled, the TSX–Plus keyboard monitor will run the user-written UCI program each time it needs a new command. The program must prompt the user for a new command, accept the command, process it as desired and may optionally pass the command to the TSX–Plus keyboard monitor by doing a *special chain exit*. See the *TSX–Plus Programmer's Reference Manual*for more information on the UCI program interface.

Keyboard command control may be returned to the TSX–Plus keyboard monitor by the command:

```
SET KMON SYSTEM
```

Since the UCI program may optionally pass entered commands on to the TSX–Plus keyboard monitor, this command may be disable by the UCI.

3.5 Keyboard Commands

The keyboard commands accepted by TSX–Plus are listed below. Because many commands are identical or very similar to those of RT–11, full descriptions are only provided for TSX–Plus specific commands and for differences between TSX–Plus and RT–11 commands. Users should consult the *RT–11 System User's Guide* for more information on keyboard commands.

3.5.1 The ACCESS Command

The ACCESS command is used to restrict user access to a particular set of files or devices. It is only valid in start-up command files or if the job has SYSPRV privilege. Logical device names may be used in lieu of physical device names. Refer to the *TSX-Plus System Manager's Guide* for further information about this command.

3.5.2 The ALLOCATE Command

The ALLOCATE command is used to temporarily request exclusive access to a device. Use of this command requires ALLOCATE privilege. The form of the ALLOCATE command is:

```
ALLOCATE ddn[,ddn...] [lnm]
```

where *ddn* is the name of an allocatable device and *lnm* is an optional logical name to be assigned to the first device in the allocation list. Note that more than one device may be allocated with this command, but only the first may also be assigned a logical name in the same command. The device names *ddn* may also be substituted by previously assigned logical names. If an error occurs on the allocation of one device, an error message is printed and the system will continue to attempt to allocate any other specified devices. The most commonly used form of this command specifies a single device to be allocated and does not include a logical assignment. For example:

```
ALLOCATE MT0:
```

If no device names are specified, the command is equivalent to SHOW ALLOCATE.

While a device is allocated by one user, access to that device is restricted exclusively to the user who allocated it; no one else will be allowed to open files on that device, to mount the device for directory caching, or to mount logical subset disks on that device. Conversely, a device cannot be allocated while any other user has that device mounted or a file open on that device. The maximum number of devices which may be simultaneously allocated by all users combined is determined during system generation. All devices allocated by a job are deallocated when that job logs off. See the DEALLOCATE command for more information on device deallocation. The SHOW ALLOCATE command may be used to determine which devices are currently allocated by any user.

All jobs associated with a primary time-sharing line (the primary line and any subprocesses) have access to a device allocated by any other job associated with the same primary line. When a job connected to a primary line allocates a device, it overrides any allocation of the same device by a subprocess associated with that primary line. If two subprocesses associated with the same primary line attempt to allocate the same device, then the first allocation takes precedence.

Devices which are partitioned into multiple units may be allocated on a unit-by-unit basis. For example, allocating DL1 has no effect on use of DL0 by others; similarly, allocating CL0 has no effect on CL1. Only real devices may be allocated (those which are serviced by device handlers and CL). You cannot allocate the system device (SY:), the pseudo-device TT:, or any logical subset disks (LDn:).

Device allocation is useful when performing operations which have no other convenient mechanism for preventing disruptive interaction by multiple users. For example, if CL0 is attached to a communication port, undesirable confusion can arise if two users attempt to use the VTCOM program through CL0 at the same time. This can be overcome by allocating CL0 before running the VTCOM program. Another use for device allocation might be to prevent excessive rewinding delays when copying multiple files from magnetic tape while another user also attempts to access the same tape unit. It might even be desirable to allocate a removable device while it contains sensitive information.

The following example lists a command file which shows how the ALLOCATE command can be used to prevent conflicts in use of a communication line:

```
SET CLO LINE=6      !Take over time-sharing line
SET CLO NOLFOUT     !Inhibit line feed transmission
ASS CLO XL          !Assign for VTCON
ALLOCATE XL         !Prevent multi-user collisions
R VTCON             !Part of RT-11 5.01 distribution
DEALLOCATE XL       !Let others use CLO
DEASS XL            !Clean up assignment
SET CLO LINE=0      !Re-enable time-sharing line
```

3.5.3 The ASSIGN Command

The ASSIGN command is used to associate a logical I/O device name with a physical device. 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:

```
ASSIGN ddn lnm
```

or

```
ASSIGN ddn=lnm
```

where *ddn* is the physical device and unit specification (or another logical name previously assigned to a physical device) and *lnm* is a logical name (up to three characters long) by which the physical device may be referenced.

For example, to assign the logical device name BIN to physical device DX1 the command would be:

```
ASSIGN DX1 BIN
```

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 to which the previous assignment was directed. 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
```

The ASSIGN command is frequently used to assign FORTRAN I/O logical unit numbers to selected devices. To assign FORTRAN I/O logical unit number 1 to the terminal, the command would be:

```
ASSIGN TT 1
```

The TSX-Plus ASSIGN command provides a useful extension. In addition to being able to specify a physical device name, the user may specify a file name, extension, and size. If a file name and optional size are specified in addition to the physical device name, the file name and size follow the device name. For example, the following command assigns FORTRAN I/O logical unit number 1 to a file named PAYROL on device DX0 with a size of 43 blocks:

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

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

3.5.4 The BACKUP Command

The TSX-Plus BACKUP command has the same form and options as the RT-11 BACKUP command. Use of this command requires NFSREAD privilege.

3.5.5 The BOOT Command

The BOOT command attempts to abort TSX-Plus and reboot RT-11. Due to the variations in hardware and bootstrap ROMs, this command is unsupported. The BOOT command may either reboot RT-11 on your system or simply halt depending on your particular hardware configuration.

Unlike the RT-11 BOOT command, no file name may be specified with the TSX-Plus BOOT command. TSX-Plus will attempt to start the monitor selected on the specified device by the COPY/BOOT command. If no device name is specified, TSX-Plus will attempt to initiate the hardware bootstrap from the current boot device. OPER privilege is required to use this command. The BOOT command is equivalent to the $STOP command.

3.5.6 The BYE Command

The BYE command is used to log off a time-sharing line. It is equivalent to the OFF command.

3.5.7 The COBOL Command

The COBOL command is used to compile a COBOL source program using the COBOL-Plus compiler. COBOL-Plus, a product of S&H Computer Systems, Inc., is sold separately. 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 for COBOL programs. TSX-Plus will implicitly invoke the COBOL-Plus compiler and CBLINK link program if the source program has the extension .CBL or the object program has the extension .CBJ. Switches that can be used with the COBOL command are listed below.

Switch	Meaning
/ALLOCATE:*size*	Specify size of list or object file.
/ANSI	Produce warning messages for non-ANSI feature use.
/CARD	Source program is in card sequence format.
/CREF	(Equivalent to /CROSSREFERENCE).
/CROSSREFERENCE	Produce a cross-reference of the source program.
/DCARDS	Compile lines with "D" in the indicator field.
/INFORMATION	Print additional information at compilation end.
/LINENUMBER	Enable source line information in object listing.
/LIST[:*name*]	Produce a source program listing.
/NARROW	Format the cross-reference for 80 column display.
/NOLINENUMBERS	Disable source line information in listing file.
/NOOBJECT	Do not produce an object file.
/NOWARNING	Suppress warning messages.
/OBJECT[:*name*]	Specify name of object file.
/ONDEBUG	Compile the program for use with the debugger.
/PRODUCTION	Omit line number tracing and subscript checking.
/RM	Compile RM/COBOL* programs.
/SEQUENCE	(Equivalent to /CARD).
/SUMMARY	Print only error messages on listing device.
/WARNING	Display warning messages.

* RM/COBOL is a trademark of Ryan-McFarland Corporation.

See the *COBOL-Plus Reference Manual* for further information about the COBOL command.

3.5.8 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. The default compiler for files with the extension .DBL is DBL; this may be changed with the SET LANGUAGE command.

3.5.9 The COPY Command

The TSX-Plus COPY command has the same form and options as the RT-11 COPY command. Use of this command requires NFSREAD privilege.

3.5.10 The CREATE Command

The TSX-Plus CREATE command has the same form and options as the RT-11 CREATE command. Use of this command requires NFSREAD privilege.

3.5.11 The DATE Command

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

3.5.12 The DEALLOCATE Command

The DEALLOCATE is used to release a temporary exclusive access restriction initiated by an ALLOCATE command for a device. This allows other users to again access the specified device. This command has three forms:

```
DEALLOCATE ddn[,ddn...]
```

or

```
DEALLOCATE
```

or

```
DEALLOCATE/ALL
```

where *ddn* is the name of the device to be released. Logical device names may be used in lieu of the physical device names. Individual devices may be deallocated with the first form of the command. Individual devices allocated from the primary line or any other subprocess associated with the same primary line may be deallocated from the primary or any other subprocess associated with the same primary line. The second and third, non-specific, forms of the command are equivalent. All devices allocated by a job may be deallocated with a non-specific form of the command. However, this only affects devices allocated from the same primary or subprocess issuing the DEALLOCATE command. Devices allocated from other lines, even

those associated with the same primary line, are not deallocated by non-specific forms of the DEALLOCATE command.

When a subprocess logs off, any devices allocated from that same subprocess are automatically deallocated, while those allocated from the primary line or a different subprocess are not deallocated. When a primary line logs off, any devices allocated from it or any of its associated subprocesses are automatically deallocated.

Logical names assigned to a device by using the ALLOCATE command are not deassigned by the DEALLOCATE command. See also the ALLOCATE and the SHOW ALLOCATE commands.

3.5.13 The DEASSIGN Command

The TSX-Plus DEASSIGN command is equivalent to the RT-11 DEASSIGN command. It is used to disassociate a logical unit assignment.

3.5.14 The DEFINE KEY Command

The DEFINE KEY command is used to specify a terminal key substitution. User defined keys cause a specified string to be substituted for a named key when that key is typed. Key substitutions are performed by the single line editor; hence, substitutions are only done on terminal input processed by the single line editor. Key definitions are inherited from the primary process when a subprocess is initiated. See Chapter 2 for more information on the single line editor and user defined keys.

The form of this command is:

```
DEFINE KEY[/qualifiers] key string
```

where *key* is the name of the key that is associated with the string. The substitution *string* may be up to 64 characters long.

The following table lists key names which may be used:

Key name	Actual key
The following keys may only be defined in KED mode	
KP0,KP1,...,KP9	digits 0 to 9 on numeric keypad
PERIOD	period key (.) on numeric keypad
COMMA	comma key (,) on numeric keypad
MINUS	minus key (-) on numeric keypad
ENTER	ENTER key on numeric keypad
The following keys are not affected by KED mode	
PF1,PF2,PF3,PF4	PF1,PF2,PF3,PF4
UP	up arrow key
DOWN	down arrow key
LEFT	left arrow key
RIGHT	right arrow key
FIND	FIND key on VT200
INSERT	INSERT HERE key on VT200
REMOVE	REMOVE key on VT200
SELECT	SELECT key on VT200
PREV	PREV SCREEN key on VT200
NEXT	NEXT SCREEN key on VT200
HELP	HELP key on VT200
DO	DO key on VT200
F6,F7,...,F20	Function keys on top row of VT200

For example, the following command associates the SHOW TIME command with the PF2 key:

```
DEFINE KEY PF2 "SHOW TIME"
```

If the associated string contains no leading or trailing spaces and no control characters, the quotation marks may be omitted.

The following qualifiers can be used with the DEFINE KEY command:

/[NO]ECHO determines whether the substitution string will be printed at the terminal at the time the substitution takes place. The default mode is /ECHO.

/[NO]TERMINATE determines whether a carriage-return and line-feed are appended to the end of the substitution string when it is substituted. Use /NOTERMINATE with a definition that specifies a part of a command. The default mode is /TERMINATE. You may not specify /NOECHO and /NOTERMINATE together. For example, the following definition associates the PF2 key with the word HELP followed by one space, without termination:

```
DEFINE KEY/NOTERMINATE PF2 "HELP "
```

/[NO]GOLD determines whether the specified key must be prefixed by the GOLD (PF1) key. If /GOLD is specified then the substitution only takes place if you type the PF1 key followed by the specified key. If /NOGOLD is specified (the default case) then the substitution takes place when the specified key is pressed without being prefixed by the PF1 key. /GOLD definitions are not available if the PF1 key itself has been assigned a key definition. The following two definitions associate the SHOW TIME command with the unprefixed PF2 key and the SHOW DATE command with the two key combination PF1 PF2:

```
DEFINE KEY PF2 SHOW TIME
DEFINE KEY/GOLD PF2 SHOW DATE
```

/LETTER specifies that the key being defined is not a function key, but is a key such as a letter, digit, or punctuation mark which is part of the typewriter portion of the keyboard. If /LETTER is specified, then *key* must be a single letter from the non-function-key portion of the keyboard. The /GOLD qualifier may be used in conjunction with /LETTER to cause the substitution to take place only if the specified letter is prefixed by the PF1 key. The default mode is /NOLETTER. For example, the following commands cause a pound sign ("#") to be substituted for the dollar sign ("$") and the SHOW JOBS command to be substituted for the "PF1 dollar-sign" combination:

```
DEFINE KEY/LETTER/NOTERMINATE $ #
DEFINE KEY/LETTER/GOLD $ SHOW JOBS
```

A key definition is removed by using a DEFINE KEY command with no substitution string. Thus, to remove the PF2 key definition use the following command:

```
DEFINE KEY PF2
```

Deleting or redefining a substitution for a letter which has already been defined requires that key substitution be turned off first. Otherwise, the defined string is substituted for the letter as the DEFINE KEY command is typed. If you need to delete or redefine a letter substitution, issue the SET SL NOSUBSTITUTE command to turn off substitutions, then use the DEFINE command, then SET SL SUBSTITUTE.

If you define substitutions for any of the keys of the numeric keypad other than the PF keys, the single line editor must be in KED mode (SET SL KED).

To include control characters in a key definition string, enclose the substitution string in quotation marks and type a caret (^) in front of the letter that is to be a control character. Control characters in substitution strings only count as one character, even though two characters are required to specify them. For example, the following command defines numeric keypad key zero (0) as CTRL–U (SL must be in KED mode):

```
DEFINE KEY/NOTERMINATE KP0 "^U"
```

If the DEFINE KEY command is placed within a command file, two caret characters must be specified to denote each control character because command file processing translates two consecutive caret characters into a single caret. Thus, if the previous DEFINE KEY command were placed in a command file, it would be written as:

```
DEFINE KEY/NOTERMINATE KP0 "^^U"
```

You may specify a key definition for a key which is normally used to perform some single line editor function (e.g., up arrow). When this is done, the key definition takes precedence over the normal function.

It is possible to define a key with a CTRL–W digit sequence to cause a switch to a subprocess. However, this is less powerful than actually typing CTRL–W digit because the switch with a defined key can only be used at a time when the single line editor is accepting input, not when a program is executing. The following key definition defines PF2 to switch to subprocess one:

```
DEFINE KEY/NOTERMINATE PF2 "^W1"
```

If a defined key appears not to be working, first verify that the definition is correct with the SHOW KEYS command. Then, use the SHOW SL command to verify that key substitution is in effect and that KED mode is enabled for numeric keypad keys.

The PI handler which controls the console terminal on a Professional, normally provides VT100 support for the console terminal, not VT200. Thus, the function keys on the top row and the DO, HELP, etc. keys may not be defined. However, it is possible to enable the PRO console for VT200 compatible operation by sending it the control sequence "<ESC>[?39h". These keys may then be defined.

3.5.15 The DELETE Command

The TSX–Plus DELETE command has the same form and options as the RT–11 DELETE command. Use of this command requires NFSREAD privilege.

3.5.16 The DETACH Command

The DETACH command is used to initiate execution of a command file as a *detached* job, to check the status of a detached job, or to abort a detached job. The system manager may restrict the use of this command. The process which initiates a detached job is known as the *parent* of the detached job; this makes it possible for the detached job to send a message to its *parent* without knowing its line number.

Use of the command to start a detached job requires DETACH privilege. The form of the command used to start a detached job is:

```
DETACH file [parameters]
```

where *file* is the name of a command file which is to be started as a detached job; *parameters* is any optional text to be passed to the detached command file. See Chapter 4 for more information on passing parameters to command files. The total length of the command beyond the DETACH keyword (including the name of the command file, parameters, and blank space) must not exceed 80 characters. The default command file extension is .COM. If a free detached job line is available, the system starts the command file and prints a message indicating the detached job line number. Detached job lines must be declared during TSX–Plus system generation.

Detached jobs inherit the full context of the process from which they are started. See Chapter 5 for more information on detached jobs. In addition, programs running as detached jobs may also use the TSX–Plus EMT to acquire a job's file context—see the *TSX–Plus Programmer's Reference Manual*.

In the following example a command file named SY:CRUNCH.COM is started as a detached job with the name of a data file passed as a parameter on the command line:

```
DETACH SY:CRUNCH DL2:DATFIL.DAT
```

Terminal output for detached jobs is normally discarded since they are not attached to any terminal. However, it is possible to collect the terminal output from a detached job by sending it to a log file. This can be done by using the SET LOG FILE=*filnam* command in the detached job command file. See the SET LOG command for more information on terminal output logging.

The DETACH/CHECK command is used to check the status of a specified detached job line. (The SYSTAT command also provides information on detached jobs.) For example, if a detached job is executing on line number 5:

```
DETACH/CHECK 5
```

Use of the command to kill a detached job requires DETACH privilege. Appropriate SAME, GROUP or WORLD privilege is also required to kill an unrelated detached job depending on the project-programmer number of the issuing and target jobs; these are not required for a *parent* to kill a *child* detached job. WORLD privilege is required to kill a detached job started as a result of the DETACH macro declared during system generation. The DETACH/KILL command is used to abort a detached job. (The KILL command may also be used to abort a detached job.) For example:

```
DETACH/KILL 5
```

3.5.17 The DIBOL Command

The DIBOL command is used to compile a DIBOL or DBL source program. The default compiler for programs with the extension .DBL is DBL. This may be changed with the SET LANGUAGE command. The TSX-Plus DIBOL command has the same form and options as the RT-11 DIBOL command, except that the options /BUFFERING, /LOG, /PAGE and /TABLES are not supported. Because of differences between compiler switch options, DIBOL switches are not supported for use with DBL.

3.5.18 The DIFFERENCES Command

The DIFFERENCES command is used to compare two files. The TSX-Plus DIFFERENCES command has the same form and options as the RT-11 DIFFERENCES command.

3.5.19 The DIRECTORY Command

The TSX-Plus DIRECTORY command has the same form and options as the RT-11 DIRECTORY command. Use of this command requires NFSREAD privilege.

3.5.20 The DISMOUNT Command

The DISMOUNT command has three functions:

- it tells TSX-Plus to stop directory caching on a particular device
- it tells the system to stop doing data caching on a device
- it dissociates a logical subset disk from its assigned file

The form of the DISMOUNT command is:

```
DISMOUNT[/NOWARN] ddn
```

where *ddn* is the real or logically assigned name of the device.

If the device is a physical device or a logical name for a physical device, then the DISMOUNT command removes the current job's entry from the mount table for that device. If no other jobs have mounted the device, then directory and data caching for that device are stopped. Files on a physical device may still be accessed after it is DISMOUNTed, but access may be slower since the directory is no longer cached. Note however, that if a job accesses a device which it has not mounted, and another user INITIALIZEs or SQUEEZEs that device, then reads will probably return garbage and writes will probably corrupt other files. ALWAYS MOUNT ANY DIRECTORY STRUCTURED DEVICE WHICH YOU INTEND TO USE!

The INITIALIZE and SQUEEZE operations cannot be performed on a device which is mounted by other users. If it is necessary to INITIALIZE or SQUEEZE a device which is mounted by other users, then they must first DISMOUNT that device. Remember that it is still possible for a job which has not mounted a device to access that device. After a device is INITIALIZEd or SQUEEZEd, then the directory and data caches are cleared for that device and if it is still mounted, then caching is resumed. Caching is not resumed if all jobs have DISMOUNTed a device until some job reMOUNTs that device.

The following information message is printed if a device is dismounted and the device is still mounted by other users:

```
?KMON-I-Device is still mounted by other users
```

This information message can be suppressed by specifying the /NOWARN qualifier with the DISMOUNT command. For example, the following command dismounts DL1, and suppresses the information message if the device is still mounted by other users:

```
DISMOUNT/NOWARN DL1
```

The SHOW MOUNTS command may be used to determine which jobs have devices mounted.

In the case of logical subset disk assignments (*ddn*=LD0–LD7 or logical names assigned to them), the effect of the DISMOUNT command is to stop directory caching on the logical subset disk as described above and to remove the device from the logical subset disk tables. In this case, files on the logical subset disk are no longer accessible until the logical subset disk is remounted. This form of the command only affects the logical subset disks belonging to the user who issues the command. If another job has mounted the same logical subset disk, then the rules for physical devices with regard to INITIALIZE and SQUEEZE also apply to the logical subset disk.

3.5.21 The DISPLAY Command

The DISPLAY command is used within a command file or user-defined command 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 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. See the section on user-defined commands in this chapter, and see Chapter 4 for more information on command files.

3.5.22 The DUMP Command

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

3.5.23 The EDIT Command

The TSX-Plus EDIT command has the same form and options as the RT-11 EDIT command. The editor which is invoked by this command is selected during system generation or with the SET EDIT command.

3.5.24 The EXECUTE Command

The TSX-Plus EXECUTE command has the same form and options as the RT-11 EXECUTE command. It also recognizes programs with the extension .CBL as COBOL source programs and automatically invokes the COBOL-Plus compiler and linker. Switches appropriate when using the COBOL-Plus compiler and linker are also valid with EXECUTE. See also the COBOL, COMPILE, DIBOL, FORTRAN, LINK and MACRO commands.

3.5.25 The FORM Command

The FORM command is used to specify the default form name for subsequent files sent to the spooler 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 files sent to the spooler until another FORM command is issued. The form name is converted internally to upper case. The initial default form name is STD. A form may also be requested within a file sent to the spooler. Subprocesses and detached jobs inherit their default form name from the primary line. See Chapter 6 for more information on spooled devices and forms.

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

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

3.5.26 The FORTRAN Command

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

3.5.27 The HELP Command

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

3.5.28 The INITIALIZE Command

The TSX-Plus INITIALIZE command has the same form and options as the RT-11 INITIALIZE command. However, you may not initialize a device that contains any of the following TSX-Plus system files:

- TSX-Plus swap file (SY:TSXSWP.TSX)
- PLAS region swap file (SY:TSXRSF.TSX)
- spool file (SY:TSXSPL.TSX)
- TSKMON.SAV
- CCL.SAV
- IND.SAV
- IND temp file (SY:TSXIND.TSX)

If you need to initialize a device containing any of these files, do it while running under RT-11. The following error message is printed if an attempt is made to initialize a device containing any of these files:

```
?KMON-F-Can't do this to device containing TSX system files
```

Use of the INITIALIZE command requires NFSWRITE privilege and may also require SPFUN privilege for some devices.

A device cannot be initialized if any other user has allocated the device. Attempts to INITIALIZE a device allocated by another user will result in an error message like:

```
?KMON-F-Device is allocated to job 14
```

The SHOW ALLOCATE command may also be used to determine the job numbers which have allocated any devices. Note that a device allocated from another subprocess started from the same primary line is not protected in this manner.

A device cannot be initialized if any other user has MOUNTed the device. Attempts to INITIALIZE a device which is mounted by another user will result in the error message:

```
?KMON-F-Device is mounted by another user
```

If the device is only mounted by the job which initializes it, then the directory and data caches are cleared after the operation and caching is resumed. The SHOW MOUNTS command may be used to determine which other users have mounted a device. However, remember that is still possible for a job which has not mounted a device to access that device. If such a job has a file open before you initialize, then severe problems can arise. Great circumspection is necessary before initializing a device in a multi-user environment. Do not initialize rashly. As a defensive measure, always MOUNT any disk device which you plan to use.

If terminal output logging is being done (see the SET LOG command) and the log file is open on the device being initialized, then the log file is closed before the device is initialized and the following warning message appears:

```
?KMON-W-Closing log file
```

3.5.29 The INSTALL Command

The INSTALL command is used to enable certain programs to be executed with special characteristics or privilege. It is *NOT* equivalent to the RT-11 INSTALL command which is used to enter device handler information in system tables. Note that the RT-11 compatible INSTALL *device* command is ignored by TSX-Plus. See the *TSX-Plus System Manager's Guide* for more information on the INSTALL command. Use of the INSTALL command requires SYSPRV privilege.

3.5.30 The KILL Command

The KILL command is used to abort a time-sharing job on another line. This has the effect of aborting the execution of the job and forcing the logoff of the line. SAME, GROUP or WORLD privilege is required to use the KILL command, depending on the project-programmer numbers of the issuing and target jobs. The form of this command is:

```
KILL line-number
```

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

3.5.31 The KJOB Command

The KJOB command is used to log off a time-sharing line. It is equivalent to the OFF command.

3.5.32 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 Manual* for further information.

3.5.33 The LINK Command

The TSX–Plus LINK command has the same form and options as the RT–11 LINK command. It also recognizes object files with the extension .CBJ as COBOL–Plus object files and then invokes the COBOL–Plus link program (CBLINK). The COBOL–Plus linker may also be explicitly specified with the /COBOL switch. See the *COBOL–Plus Reference Manual* for further information. Switches which are meaningful to COBOL–Plus are:

Switch	Meaning
/NOPAGE	Do not swap data segments.
/NOSHARED	Do not use shared COBOL–Plus run-time library.
/PAGE	Swap data segments.
/SHARED	Always use shared COBOL–Plus run-time library.
/SIZE	Report the size of the largest data segment.
/VM	Use VM for run-time and program segmentation.
/XM	Load entire program and run-time into extended memory.

3.5.34 The LOGOFF Command

The LOGOFF command is used to log off a time-sharing line. It is equivalent to the OFF command.

3.5.35 The MACRO Command

The TSX–Plus MACRO command has the same form and options as the RT–11 MACRO command.

3.5.36 The MAKE Command

The MAKE command is used to create a new file with the TECO editor. It is equivalent to the RT–11 MAKE command.

3.5.37 The MEMORY Command

The MEMORY command is used to control the amount of memory available to a job. When a job initially *logs on* it receives a default memory allocation set by the system manager. The MEMORY command can be used to change the allocation for the job. Use of the MEMORY command to change a job's memory allocation is not valid if the system has been generated to disallow job swapping. The form of the MEMORY command is:

```
MEMORY nn
```

where *nn* is the number of kilobytes (Kb, 1024. bytes) of memory to be allocated for the job. The maximum memory size that a job may use is set by the system manager, but never exceeds 64 Kb. 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 EMTs do not actually affect the amount of memory allocated to a job—only the MEMORY command and a TSX-Plus EMT described in the *TSX-Plus Programmer's Reference Manual* do that.

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

Programs are only allowed to use more than 56 Kb of memory if they are *virtual*, meaning they do not directly access the RMON area, although they may still access it indirectly by use of the .GVAL and .PVAL EMTs (see the *TSX-Plus Programmer's Reference Manual*and the *RT-11 Programmer's Reference Manual*). Programs may indicate that they are virtual by any of the following techniques:

- Set bit 10 (mask 2000) in the job status word (location 44) of the SAV file. See the *TSX-Plus Programmer's Reference Manual* for information about how the SETSIZ program can be used to do this.
- Use the /V LINK switch (/XM switch for the LINK keyboard command) which stores the RAD50 value for VIR in location 0 of the SAV file.
- Store a memory allocation size greater than 56 Kb in location 56 of the SAV file. See the *TSX-Plus Programmer's Reference Manual* for information about how the SETSIZ program can do this.

If none of these conditions is met the program is restricted to 56 Kb even if a larger value is specified with the MEMORY command. See the *TSX-Plus Programmer's Reference Manual* for more information on memory usage and virtual images.

A program may dynamically control the amount of memory allocated for the job by use of the TSX-Plus EMT with function code 141 (described in the *TSX-Plus Programmer's Reference Manual*). See also the description of the SETSIZ program in the *TSX-Plus Programmer's Reference Manual* for information about how the amount of memory to be allocated for a particular program can be stored in the SAV file for the program.

3.5.38 The MONITOR Command

The MONITOR command is used to cause TSX-Plus to begin a performance analysis. See the *TSX-Plus Programmer's Reference Manual* 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, and *cell-size* is the number of bytes to group per histogram cell. The only valid switch is /IO which causes I/O wait time to be included in the analysis.

3.5.39 The MOUNT Command

The MOUNT command is used to:

- begin directory caching on a file-structured device
- enable data caching on a device
- associate a logical subset disk with a disk file

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. Data caching is a technique used to speed up disk reads by keeping memory resident copies of recently used file blocks. Both directory and data caching are enabled during TSX–Plus system generation and activated when a device is MOUNTed. The form of the MOUNT command to activate directory and data caching is:

```
MOUNT ddn
```

where *ddn* is a physical device name such as DL1:. The effect of this type of MOUNT command is to tell TSX–Plus that it should begin directory and data caching for the device being mounted. The system device is automatically MOUNTed for each user. If caching is not wanted, then no MOUNT should be performed, or the DISMOUNT command should be used to halt caching on a previously mounted disk. If the device being mounted is allocated by another user, then the system will report an error similar to:

```
?KMON-F-Device is allocated to job 12
```

The INITIALIZE and SQUEEZE operations cannot be performed on a device which is mounted by other users. If it is necessary to INITIALIZE or SQUEEZE a device which is mounted by other users, then they must first DISMOUNT that device. Remember that it is still possible for a job which has not mounted a device to access that device.

Once a MOUNT command is issued, caching is enabled for all users who access files on the device (including users who have not MOUNTed the device). The system maintains a table of all users who have mounted each device. See also the DISMOUNT and SHOW MOUNTS commands.

Warning: If directory caching is enabled for a device, it is crucially important that the DISMOUNT command be used to dismount the device before a disk is replaced on the same drive. If a new disk pack is inserted in the drive without issuing the DISMOUNT command, TSX–Plus would try to access files on the new pack according to the locations stored in the directory cache for the old pack.

Directory caching causes a dramatic improvement in the speed of 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 second form of the MOUNT command is used to associate a logical subset disk with a disk file. The form of the command is:

```
MOUNT[/[NO]WRITE] LDn filnam [logical-name]
```

where *LDn* is the logical subset disk, and n is in the range 0–7, *filnam* is the name of a disk file containing the subset directory and files, and *logical-name* is an optional logical name to be assigned to the logical subset disk. The [NO]WRITE option controls access to the logical subset disk. WRITE allows full access, whereas NOWRITE allows read-only access. WRITE access is allowed by default unless the NOWRITE option is used. The default extension for the disk file to be associated with a logical subset disk is .DSK. File protection is automatically set on the file specified to be a logical subset disk. When a logical subset disk is mounted, directory and data caching are also begun for it. If the device which holds the file to be mounted as a logical subset disk is allocated to another job, then the following error will be reported:

```
?KMON-F-File not found
```

Each user may mount up to 8 logical subset disks at any time. The association between a logical subset disk (LD0–LD7) and a file is *local* to each user. For example, one user may associate her LD2 with file DL1:MYDISK.DSK, while a different user associates his LD2 with file DL1:YORDSK.DSK.

Logical subset disks may be *nested*, allowing one or more subsets to be defined within other subsets. However, if this is done, the MOUNT commands must be executed sequentially from outer-most to inner-most logical subset disk and the unit numbers must be assigned sequentially from LD0 to LD7.

Example

```
MOUNT LD0 MANUAL MAN
CREATE LD0:SUB.DSK/ALLOCATE:100.
MOUNT/WRITE LD2 MAN:SUB
INIT/NOQ LD2:
```

Other commands which refer to logical subset disks are: DISMOUNT, SET LDn CLEAN | WRITE | NOWRITE and SHOW SUBSETS | MOUNTS. The ACCESS command may also be used with logical subset disks in start-up command files. The restrictions on INITIALIZE and SQUEEZE operations also apply when another user has mounted the same logical subset disk.

Logical subset disk support is integral to the file management functions of TSX-Plus, and does not require the LD pseudo-device handler. See Appendix B for more information on the use of logical subset disks.

3.5.40 The MUNG Command

The MUNG command is used to start a file of TECO commands. The MUNG command is equivalent to the RT-11 MUNG command.

3.5.41 The OFF Command

The OFF command is used to log off a time-sharing line, and to release a subprocess (see the discussion of process windows in Chapter 2). The accumulated connect time and CPU time used during the session are printed during the logoff processing.The BYE, KJOB and LOGOFF commands are synonyms for the OFF command. TSX-Plus automatically logs off dial-up lines if the telephone connection is broken. A special logoff command file may also be designated to execute whenever a job logs off; see the *TSX-Plus System Manager's Guide* for more information.

Example

```
.OFF
Connect time=01:43:00  CPU=00:12:03
```

If the off command is issued from the primary line, a check is made to see if there are any active subprocesses. If there are, the following warning message is printed:

```
You have n active subprocesses.
Are you sure you want to log off (Y/N):
```

You can suppress the warning message by specifying the /NOWARN qualifier with the OFF COMMAND.

3.5.42 The OPERATOR Command

The OPERATOR command is used to send a message to the operator's console. The OPERATOR command works like the SEND command, but it is not necessary to know the line number of the operator's terminal. This command is equivalent to the SEND,OPERATOR command. Use of this command requires SEND privilege. The form of the OPERATOR command is:

```
OPERATOR message
```

For example, to send a disk mount message to the operator:

```
OPERATOR PLEASE MOUNT PAYROLL MASTER DISK ON RK1
```

3.5.43 The PAUSE Command

The PAUSE command is used within command files (see Chapter 4) to temporarily suspend processing of 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 suspended until carriage return is pressed. This gives the operator an opportunity to perform manual operations such as mounting disks or tapes.

3.5.44 The PRINT Command

The TSX-Plus PRINT command has the same form and options as the RT-11 PRINT command. Use of this command requires NFSREAD privilege.

3.5.45 The PROTECT Command

The TSX-Plus PROTECT command has the same form and options as the RT-11 PROTECT command.

3.5.46 The R Command

The R command is used to start a program. The form of the command is:

```
R[/switch] dev:filnam [input-data]
```

If no device name is specified with the program name, TSX-Plus attempts to find the specified program on SY:. The amount of memory available to a program can be controlled with the MEMORY command, or size information can be included in its disk image. See the description of the SETSIZ program in the *TSX-Plus Programmer's Reference Manual* for more information about how the amount of memory allocated for a program may be controlled. See also the *TSX-Plus Programmer's Reference Manual* for more information on the operating environment for programs under TSX-Plus.

A line of input may be passed to a 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 CTRL-C as its second line of input. (See example 2 below.) Note that only programs which accept input with the .GTLIN, .CSISPC, and .CSIGEN requests can accept input in this manner. Single character input (.TTYIN) does not normally accept data from a command line; however, the program can be invoked with a command file and accept *all* terminal input from the command file—see the section on command file control characters in Chapter 4. Note also that text passed on the command line to be accepted by the program via a .GTLIN request will be reorganized if it contains multiple words separated by spaces. The first word will be placed as the input to a CSI command string and the remainder of the line will be placed on the output side of the equal sign. (See example 7 below.) The unscrambled input parameters are available in the chain data area. This behavior is compatible with RT-11.

The valid switches for the R (and RUN) command are:

Switch	Meaning
/BYPASN	Run program without logical device assignments
/DEBUG	Run program under debugger control
/HIGH	Run program in high efficiency TTY mode
/IOPAGE	Run program with PAR 7 mapped to I/O page
/LOCK	Lock program to line
/MEMLOCK	Run program in low memory
/NONINTERACTIVE	Run program in non-interactive mode
/NOWINDOWING	Suspend window processing during program execution.
/ODT	Equivalent to /DEBUG
/SCCA	Suppress CTRL–C aborts during program execution.
/SINGLECHAR	Run program with single-character activation
/TRANSPARENT	Run program in *transparent* terminal mode

All switches can be abbreviated to a single character.

Note that the functions of these switches are also available to installed programs. See the *TSX–Plus System Manager's Guide* for more information on installed programs and the INSTALL command.

/BYPASN causes all logical device references to be ignored while the program is running. This includes any reassignment of SY and DK which will be directed to the system disk (device from which RT–11 was booted). Programs run with the /BYPASN switch must not access any logical device names (other than SY and DK which are directed to the system device). Only physical device names should be used. The following example shows a potential effect of bypassing logical device assignments:

```
.RUN SRT:TEST
*SRT:TEST.SAV
.RUN/BYPASN SRT:TEST
*SRT:TEST.SAV
?CSI-F-Cannot find file
*^C
```

/DEBUG causes the program being started to execute under control of the TSX–Plus program debugging facility. Use of this switch requires DEBUG privilege. Note that the debugger is a system generation option which must be included in order to use the debugging facility (see your System Manager). Since the debugger does not share memory space with the program being debugged, there are no restrictions on the size of program which can be debugged. It is not necessary to link the debugger with the program being debugged, although a link map of the program is essential to make effective use of the debugger. When a program is started under the debugger, the program is loaded and control is passed to the debugger. At this point, the starting address of the program is loaded into R0 and the debugger is set in such a way that the ";G" command will begin execution of the program. Execution of a program under debugger control may be interrupted as though a break point had been hit by typing a CTRL–D (see the SET CTRLD DEBUG command). If you wish a program to stop at a breakpoint caused by execution of a BPT instruction included in the program image, then *DO NOT* run the program with the /DEBUG switch. Debugger commands are similar to those of RT–11 ODT. See the *TSX–Plus Programmer's Reference Manual* for more information on the TSX–Plus program debugger.

/HIGH automatically enables the program to use high efficiency terminal I/O. This disables much of the character testing done during terminal operations and can increase terminal throughput. See the description of the "R" program controlled terminal option in the *TSX–Plus Programmer's Reference Manual* for more information on high efficiency terminal mode.

/IOPAGE causes PAR 7 for a program (virtual addresses in the range 160000 to 177777) to be mapped to the physical I/O page. Use of this switch requires MEMMAP privilege. Under TSX–Plus a program

usually has PAR 7 mapped to a simulated RMON; this allows old programs which directly access RMON fixed offsets to operate without being rewritten. However, when PAR 7 is mapped to a simulated RMON, access to hardware control registers in the I/O page is not allowed without special coding. When a program is started with the /IOPAGE switch, then direct I/O page access is possible without recoding; however, direct RMON access is not available. See the *TSX-Plus Programmer's Reference Manual* for more information on the TSX-Plus job environment. See the *TSX-Plus Programmer's Reference Manual* for more information on special system service calls which can be used to control job virtual address mapping, especially RMON vs. I/O page mapping.

/LOCK causes the program that is being started to be *locked* to the time-sharing line so that the line is automatically logged off when the program exits. 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 executing a particular program. If a locked program chains to another program, the program that was chained to then becomes the locked program.

/MEMLOCK is used to force a program to be loaded into a low memory area located directly above the low portion of the operating system and any other memory locked jobs. The program is locked in memory while it is executing. Use of this switch requires PSWAPM privilege. Forcing the program to be locked in low memory minimizes the memory fragmentation likely to occur if the program were simply locked in the current memory location for the process. This switch should be used for programs which always have an I/O completion pending. This is necessary to prevent the program from hanging when swapping is attempted. For example, VTCOM is installed with the /MEMLOCK switch. This switch might also be used for a real-time program that needed assured rapid I/O response. Not only is rapid response assisted by always having the program in memory (rather than swapped on disk), but I/O to devices that require the MAPIO facility may complete more quickly, as no mapping occurs when the program requesting I/O resides in the low 256 Kb memory region. Note that this switch does not assure the program of residing at or below any physical boundary in memory.

/NONINTERACTIVE prevents a program from receiving the priority boost normally given to jobs on the completion of terminal input. This should be used with programs which do heavy terminal I/O but which are not really interactive jobs, such as file transfer programs. A program run with this switch will execute at a lower priority and will not interfere with interactive jobs. The effect of this switch is cleared when a program exits to KMON, but the switch remains in effect if the program chains to another program.

/NOWINDOW suspends process windowing (i.e., characters sent to the terminal are not monitored by the process windowing system). The /NOWINDOW switch is especially useful for file transfer programs that transfer large volumes of characters through terminal lines. /NOWINDOW is automatically invoked when TRANSF runs.

/SCCA suppresses CTRL-C aborts while the program is executing.

/SINGLECHAR causes a program to execute in *single character activation* mode. (See the discussion of activation characters in the *TSX-Plus Programmer's Reference Manual.*) 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 true even if 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, the /SINGLECHAR switch causes TSX-Plus to honor bits 6 and 12 of the Job Status Word, allowing the program to activate on each character and test for terminal input without stalling. A program can also cause TSX-Plus to honor JSW bits 6 and 12 by using the "U" and "S" terminal control commands (see the *TSX-Plus Programmer's Reference Manual*). The example program STEALS in the section on requesting exclusive system control in the *TSX-Plus Programmer's Reference Manual* uses single character activation in this way. KED and K52 are automatically run in single character activation mode.

/TRANSPARENT is used to minimize terminal output character checking by the system during execution of a program. This enables the program to transmit the program controlled terminal option *lead-in* character to the terminal. There is no way to turn off transparent mode from within a program until it exits to the monitor. See the *TSX-Plus Programmer's Reference Manual* for a discussion of this and other program controlled terminal options.

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. (If no system command, UCL command, or command file name conflicts with a program name, then a program on SY: may be run simply by name. See the beginning of Chapter 3 for more information on command interpretation.)

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

3. Run the program named SAMPLE on RK2:.

```
R RK2:SAMPLE
```

4. Start the execution of BASIC and force logoff on exit.

```
R/LOCK BASIC
```

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

```
R/SINGLE PLANE
```

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

```
R/DEBUG TRIAL
TSX-Plus debugger
DBG:
```

7. Start the program SY:GETLIN and pass it some input text.

```
R GETLIN INPUT OUTPUT
```

A .GTLIN request in the program will receive the following text:

```
OUTPUT=INPUT
```

3.5.47 The RECALL Command

The "RECALL" keyboard monitor command is used to display the set of saved commands and to recall a specific saved command either by specifying an index number or a string that matches the beginning of the command.

The "RECALL/ALL" command is used to display a list of the saved commands. An example of the display is shown below:

```
.RECALL/ALL
  1: RUN TEST
  2: LINK TEST
  3: MACRO TEST
  4: EDIT TEST
```

This is the normal display mode, with the most recent command displayed first. You may select reversed ordering, with most recent displayed last, with the SET RECALL REVERSE command. The SET RECALL NORMAL command re-establishes the normal (default) ordering.

Each saved command is assigned an index number, the most recent being number 1. To recall a specific command using its index number, use a command of the form "RECALL *number*".

You can also recall a command by specifying a string of characters that match the beginning characters of the command to be recalled. The form of this command is "RECALL *string*". For example, the following command would recall the MACRO TEST command: "RECALL MAC". If more than one saved command matches the specified string, the most recent saved command is recalled.

3.5.48 The REMOVE Command

The TSX-Plus REMOVE command is only partially similar to the RT-11 REMOVE command. Device handlers are permanently loaded under TSX-Plus, so use of the REMOVE command to clear a device name from system tables is not applicable.

The TSX-Plus REMOVE command is used to eliminate a local or global named region. SYSGBL privilege is required to remove named global regions. See the *TSX-Plus Programmer's Reference Manual* for more information on named regions. The form of the command is:

```
REMOVE region-name
```

Region names may be determined from the SHOW REGIONS command.

3.5.49 The RENAME Command

The TSX-Plus RENAME command has the same form and options as the RT-11 RENAME command. Use of this command requires NFSREAD privilege.

3.5.50 The RESET Command

The RESET command is used to reset the system usage statistics that are displayed with the SYSTAT command and the SYSMON utility. Data caching statistics are also reset by the RESET command. This is useful when you want to monitor system performance during a particular part of the day. SYSPRV privilege is required to use the RESET command. The TSX-Plus RESET command is *NOT* equivalent to the RT-11 RESET command.

3.5.51 The RESUME Command

The RESUME command is used to continue the execution of a process which was suspended with the SUSPEND command. SAME, GROUP or WORLD privilege is required to use this command depending on the project-programmer numbers of the current and target processes. The form of the command is:

```
RESUME job-number
```

where *job-number* is the number of the process to be resumed. Job numbers can be obtained from the SYSTAT command.

3.5.52 The RUN Command

The RUN command is equivalent 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.

3.5.53 The SEND Command

The SEND command is used to send messages between processes and from detached jobs to processes. Use of this command requires SEND privilege. The form of the command is:

```
SEND,line-number message
```

where *line-number* is the number of the process to which the message is to be sent. One of the following special names may be specified in lieu of *line-number*:

Name	Destination
ALL	All logged on lines
OPERATOR	Terminal designated as operator during system generation
PARENT	Process from which sender was started

The special name PARENT is useful when a detached job needs to send a message to the process which started it. A message sent to any primary or subprocess will always be delivered, regardless of which process is currently attached to the terminal. Thus, sending to PARENT from a subprocess will result in the message being delivered to the same terminal which sent the message.

If the receiving process has SET TT GAG and is executing a program other than KMON, then the message will not be delivered. Messages sent to a process which has windowing enabled are not preserved in the window memory for that process. Thus, if the window is refreshed, the message will be lost.

Examples

1. Send a message to all logged on users:

```
SEND,ALL Bob, Call me when you get a chance.
```

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 and the user name currently associated with that line. For example, a message from line 1 might be displayed as follows:

```
01 (SYSNGR) -- Bob, Call me when you get a chance.
```

Keep in mind that several characters are used to identify the sending line and user, with the result that a long message may be truncated.

3.5.54 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 RT–11, the TSX–Plus SET command is used to specify options for devices such as line-printers and 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 device handler). OPER privilege and either SYSPRV or BYPASS privilege is required to set an option in a device handler. The device name may be either the physical device name or a logical name assigned to the physical device. For example:

```
ASSIGN CL1 XX
SET XX LENGTH=66
```

The SET command may use logical names for physical devices and for CL, LD and VM devices, but not for pseudo-devices like TT and SL, nor for system parameters like QUAN1, CACHE or PRIORITY. 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 issued, the change will be made; otherwise, a warning message:

```
?KNON-F-Handler active -- Can't update running copy
```

will be printed and the running copy of the handler is not altered. If a SET is applied to a handler which was not loaded when TSX–Plus was started, then the following warning message also appears:

```
?KNON-W-Handler not installed. Set applied to handler disk file only.
```

Some device handlers do not allow alterations to the memory copy of the handler. These handlers will produce the following message when a SET is specified:

```
?KNON-W-Set performed on handler -- must reboot to take effect
```

When a SET is done to a device handler, blocks 0 and 1 of the handler are read from the disk, the SET option is applied and then block 1 is written back to the disk and moved over the copy of block 1 that is in memory (when appropriate). If the vector of a device is changed with the SET *dd* VECTOR=*nnn* command, TSX–Plus must be restarted to function correctly. See the *RT–11 System User's Guide* for device handler SET options.

3.5.55 SET CACHE

The SET CACHE command is used to alter the number of blocks which may be held in the generalized data cache. This command does not alter the amount of memory reserved for the data cache, but only controls the number of blocks within the limits allowed by the CACHE parameter selected during system generation. This command is used by the system manager to determine the effect of varying cache sizes on system performance. SYSPRV privilege is necessary to use this command. The form of this command is:

```
SET CACHE blocks
```

where *blocks* may range from 0 to the number of blocks reserved by the CACHE parameter during system generation.

3.5.56 SET CCL

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 translated into the appropriate low level commands before execution. 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. The form of this command is:

```
SET CCL [NO]TEST
```

When TSX-Plus is in CCL TEST mode, it will display at the terminal the low level commands that are generated by a CCL command, but not execute them. The SET CCL NOTEST command turns this mode off and TSX-Plus resumes 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.

Example

```
.SET CCL TEST
.DIR/OUTPUT:DIR.DAT/OCTAL/BLOCKS DL1:
R DIR
DK:DIR.DAT=DL1:*.*/O/B
^C
.SET CCL NOTEST
```

3.5.57 SET CL

The SET CL command is used to control the functions of a CL (Communication Line) pseudo-device. The CL facility is a device handler which may be used to service most serial devices such as printers, plotters and dial-out lines. Up to 16 CL units may be declared during system generation. Units 0 through 7 are referenced as CL0 through CL7 respectively; units 8 through 15 are referenced as C10 through C17 respectively. CL units may either be assigned to dedicated lines which are to be used only as I/O devices or may be assigned to *take over* inactive time-sharing lines on a temporary basis. This might be useful when a particular line is to be used as a *dial in* time-sharing line at some times and as a *dial out* communication device at other times. The CL handler can be used to replace the LS and XL device handlers. The CL handler can serve as a communication interface with virtual terminal or file transfer utilities, such as the RT-11 VTCOM program. CL lines, as well as time-sharing lines, may be connected through DH(Q,U,V)11, DL(V)11 and DZ(Q,V)11 type interfaces. See the *TSX-Plus Programmer's Reference Manual* for specific programming information for the CL device handler.

TERMINAL privilege is required to issue SET CL commands.

SET options are issued for each individual CL unit. The number of available CL units is determined during TSX-Plus system generation. CL unit numbers range from zero to one less than the number of CL units defined during system generation. For example: if a system is generated with a total of 5 CL units, then the units would be designated CL0 through CL4. If a system were generated with a total of 9 CL units, they would be designated CL0 through CL7 plus C10 (which would be the ninth unit). The unit CL: is equivalent to CL0:.

SET options for CL units are not permanent; they return to the default parameters and must be reset each time TSX-Plus is restarted. A detached job start-up command file is a convenient way to set CL options each time TSX-Plus is restarted. If an unattached CL unit is assigned to take over a time-sharing line, then that CL unit acquires the default options. The SHOW CL command may be used to determine current settings for any CL unit attached to a line.

The SET CL command has two forms, one for on/off options that accept a NO prefix, and another for options that accept a value. Those which accept a NO prefix are used to toggle between two possible states. Those which accept a value are used to control variable features like page length. Examples of these two forms are:

```
SET CL1 NOLFOUT
```

or

```
SET CL0 LINE=6
```

A single command may specify multiple parameters and options. If there is more than one qualifier, the qualifiers may be separated by a comma, a space, or a slash. For example:

```
SET CL2 SPEED=1200,BITS=7,PARITY=EVEN
```

The SHOW CL and SHOW TERMINALS commands and the SYSMON utility may be used to obtain information about the current status of CL units.

A typical use of CL is to redefine a *dial up* time-sharing line for temporary use as a *dial out* communication device using the RT-11 VTCOM utility. A typical command file to do this might be:

```
SET CL0 LINE=6      !Take over time-sharing line
ASSIGN CL0 XL       !Assign for VTCOM (use XC on the PRO)
ALLOCATE XL         !Prevent multi-user collisions
SET XL NOLFOUT      !Inhibit line feed transmission
R VTCOM             !Part of RT-11 5.01 distribution
...                 !Communicate with remote system using VTCOM
DEALLOCATE XL       !Let others use CL0
DEASS XL            !Clean up assignment
SET CL0 LINE=0      !Re-enable time-sharing line
```

A second common use for CL is to drive serial printers, as a replacement for the LS device handler. In some situations with fixed system resources it can be useful to alternate a single interface port between use as a time-sharing line and as a printer port. The printer port on a Professional 350 is one example. A typical command file to make this redefinition is:

```
SET CL2 LINE=2      !Redefine printer port as spooled CL unit
SET CL2 SPEED=1200  !1200 baud printer
SET CL2 LENGTH=66   !Set page length
SET CL2 NOFORM      !Printer does not have form feeds
SET CL2 FORM0       !Start new page for each file
ASS CL2 LP          !Assign so PRINT command works
```

The following are a list of SET CL commands:

7BIT is equivalent to NOEIGHTBIT.

[NO]8BIT is equivalent to [NO]EIGHTBIT.

[NO]BININ controls binary input mode. In binary input mode: 8-bit data characters are passed directly to the user's data buffer; and received control characters (such as CTRL-S and CTRL-Q) are treated as normal data characters. BININ overrides the current selection of EIGHTBIT for input. Note that in order to receive 8-bit characters, BITS=8 must also be selected. The default mode is NOBININ.

[NO]**BINOUT** controls binary output mode. In binary output mode: 8-bit data characters are sent to the CL unit without modification; the CL handler does not send XON or XOFF characters to control the speed of the remote device, but it will respond to XON/XOFF characters received from the remote device unless binary input mode is also in effect. BINOUT overrides the current selection of EIGHTBIT for output. Note that in order to transmit 8-bits characters, BITS=8 must also be selected. The default mode is NOBINOUT.

BITS=*n* selects the number of data bits to be used by the hardware line controller. If the hardware does not support programmable data bit selection, then this command has no effect. The number of data bits may be 7 or 8. Note that in order to transmit and receive 8 data bits both BITS=8 and EIGHTBIT (or BININ/BINOUT) mode must be selected.

[NO]**CR** controls transmission of carriage-return characters to the output device. If a CL unit is set NOCR, carriage return characters are discarded and not sent to the output device. Carriage-return characters received from the CL device are not affected. The default setting is CR.

[NO]**CTRL** controls transmission of control characters to a CL output device. Control characters are those ASCII characters from NUL to US (octal values 0 to 37; decimal values 0 to 31). When a CL unit is set NOCTRL, control characters are not sent to the output device. This includes the ESC character. Carriage and paper movement characters are exceptions; BS, HT, LF, FF and CR are always transmitted, regardless of the setting of CTRL. (Note, however, that other parameters may affect transmission of these characters, such as the LFOUT and FORM settings.) Characters received from the CL device are not affected by the CTRL setting. The default setting is CTRL.

[NO]**DTR** causes the Data Terminal Ready (DTR) signal to be asserted on interface devices capable of modem control. NODTR causes the DTR signal to be dropped. The DTR signal is automatically asserted if a read or write operation is directed to a CL unit, even if that unit has been set NODTR. In this case, DTR remains asserted even after the operation has completed until it is specifically turned off. The default (initial) setting is NODTR.

[NO]**EIGHTBIT** If the EIGHTBIT option is selected, all eight bits of received characters are passed to programs. This option does not bypass control character checking as do the BININ and BINOUT options. If NOEIGHTBIT is specified, characters are masked to seven bits. Note that in order to transmit or receive 8 data bits, both BITS=8 and EIGHTBIT (or BININ/BINOUT) must be selected.

ENDPAGE=*n* specifies the number of form-feed characters to be appended to the end of the output file when the channel to the CL unit is closed. This is required by some laser printers to cause the last page buffer to be printed. The default is zero. The ending form feeds are only appended if a write operation has been performed to the CL unit. If both ENDPAGE and ENDSTRING are specified for a CL unit, then the ENDPAGE option is performed first.

ENDSTRING=*string* specifies a character string to be appended to the end of the output file when the channel to the CL unit is closed. The string may be enclosed in either single (') or double (") quote marks. The string may be up to seven characters in length. The endstring is only appended to the output file if a write operation has been performed to the CL unit. If both ENDPAGE and ENDSTRING are specified for a CL unit, then the ENDPAGE option is performed first.

Control characters are specified in the string by a caret (^) followed by the character whose ASCII value is 100 (octal, 64 decimal) larger than the control character desired. Each control character counts as one character in the string even though two characters are used to specify it. For example, the following command causes EOF to be printed, followed by carriage-return, line-feed, CTRL–Z:

```
SET CL2 ENDSTRING="EOF^M^J^Z"
```

If the SET CL*n* ENDSTRING command is included in a command file, then each control character requires a double caret (^^). For example:

```
SET CL2 ENDSTRING="EOF^^M^^J^^Z"
```

[NO]**FORM** controls the transmission of form-feed characters to CL output devices. If a device's hardware can utilize form feed characters, the CL unit should be set FORM. If a CL unit is set NOFORM, then form feeds sent to the unit are translated to enough line feed characters to advance to the top of the next form, based on the setting of the LENGTH parameter. The default setting of this parameter for dedicated CL units is determined during TSX-Plus system generation. For extra CL units, the default setting is NOFORM.

[NO]**FORM0** controls paging at the beginning of print files. More precisely, if a CL unit is set FORM0, then a form feed (or enough line feeds) will be sent each time a write is issued to the device with a block number of 0. The default setting is NOFORM0.

[NO]**GRAPH** implicitly controls whether line width truncation is performed. The GRAPH option is equivalent to WIDTH=0; no line width trunction is performed. The NOGRAPH option is equivalent to WIDTH=132; lines are truncated after 132 characters.

[NO]**LC** controls case conversion of output characters. If a CL unit is set NOLC, then lower case characters (a-z) will be translated to upper case (A-Z) before transmission. Characters received from the CL unit are never converted to upper case. The default setting is LC.

LENGTH=*n* defines the number of lines per page. This option is used together with the NOFORM option to replace form feed characters with enough line feeds to advance to the top of a page. If LENGTH=0 and NOFORM are selected, no form feed simulation occurs and form feeds are discarded. This option is also used with the SKIP option to define bottom margins. The default setting is LENGTH=66.

[NO]**LFIN** determines the response of the CL handler to incoming line feed characters. If a unit is set NOLFIN, then line feed characters received from the CL device are discarded (except when reading with .SPFUN 203). The default setting is LFIN.

[NO]**LFOUT** controls the transmission of line feed characters to a CL device. If a unit is set NOLFOUT, then line feed characters sent to the CL device are discarded. When a CL unit is used with the RT-11 VTCOM utility, it should be set NOLFOUT. The default setting is LFOUT.

LINE=*n* is used to redirect a CL unit to use a time-sharing line or a dedicated CL line. The parameter *n* must refer to a physical time-sharing line number or to the line number assigned to a dedicated CL line. You may not assign a CL unit to a time-sharing line which is in use or to a dedicated CL line which currently has another CL unit assigned to it. A CL unit may be disassociated from any line by setting it to LINE=0. Time-sharing lines may be temporarily redefined as output devices or *dial-out* ports by assigning a CL unit to them. They may then be restored as time-sharing lines by disassociating the CL unit from them. A CL unit may not be assigned to a line which is flagged as DEAD. If the CL unit being assigned currently has output pending, then the system will wait up to 20 seconds for pending output to be transmitted before reassigning the line. The SHOW CL and SHOW TERMINALS commands may be used to determine the current assignments of CL units.

PARITY=*xxxx* selects parity control for the terminal hardware line controller to which the CL unit is connected. This option has no effect if the hardware does not support programmable parity control. The valid options are: EVEN, ODD or NONE. The alternate form NOPARITY is equivalent to PARITY=NONE.

RESET is used to clear the CL handler. A handler .SPFUN call (265) is also available to clear CL from within a program. It performs the following operations:

1. clears the input ring buffer of received characters
2. clears the output ring buffer of transmitted characters
3. stops sending a break if one is in progress
4. resets the internal *CTRL-S received* flag
5. transmits CTRL-Q if CTRL-S has previously been sent

6. clears the internal *end-of-file* flag

SKIP=*n* is used together with the LENGTH setting to control bottom margins. When only *n* lines remain on a page, a form feed (or enough line feeds) is issued to advance to the top of the next form. This feature is normally used to skip over page perforations between forms. The default setting is SKIP=0, which does not skip lines at the bottom of a page.

If a page is ejected because the skip count is reached and the next character to be sent is a form feed, then that form feed is ignored. This reduces paper waste when using page formatting programs which implement their own skip counts.

SPEED=*n* controls the transmit/receive baud rate for a CL unit. Split transmit/receive speeds are not supported. This option can only be used for hardware interfaces which support programmable baud rates, such as DZ11, DZQ11, DZV11, DH11, DHQ11, DHU11, DHV11, and DLV11-E interfaces and the Professional 350 printer and communication ports. The available speed values are: 75, 110, 134.5, 150, 300, 600, 1200, 1800, 2000, 2400, 3600, 4800, 7200, 9600, and 19200. DEC DZ(Q,V)11 interfaces do not support the 19200 baud rate. DEC DH(Q,V)11 interfaces do not support 3600 or 7200 baud. DEC DH11 interfaces do not support 2000, 3600 or 7200 baud. The PRO quad SLU does not support 3600 or 7200 baud.

[NO]TAB controls transmission of horizontal tab characters to CL units. If the device can support the tab character, the TAB setting should be used. If the unit is set NOTAB, then tab characters sent to the unit are replaced with enough spaces to advance to the next tab position. (Substituted tabs are at columns 1, 9, 17, 33, 41, ...) The default setting for dedicated CL lines is determined during TSX-Plus system generation. The default setting for extra CL units is NOTAB.

TOP is used to tell the CL handler that the printer is now at the top of form. This is necessary when repositioning forms on a printer which does not support hardware form feeds.

TRANSLATE=*n* is used to translate characters from *external* values as they are received to *internal* values as they are processed by the system; a reverse translation from *internal* values to *external* values occurs as the characters are transmitted. This feature is primarily intended to aid TSX-Plus users that have terminals with non-English character sets.

VERSION=*n* is used to set the version number assigned to the CL handler. This is useful in solving the problem encountered when the error message "?VTCOM-F-Wrong version of XL:" is reported while running VTCOM.

WIDTH=*n* specifies the maximum line width. Characters sent to a CL unit are discarded if they would exceed the maximum line width. The current column count is cleared after each carriage return. The default setting is WIDTH=0, which allows unlimited line widths.

XON clears the *CTRL-S received* flag (allowing the CL unit to resume transmission), and sends a CTRL-Q (XON) character. It also clears the end-of-file status for the CL unit. Issuing this command is equivalent to performing a .SPFUN 201 operation.

3.5.58 SET CORTIM

The SET CORTIM command is used to adjust the value of the CORTIM system control parameter. This parameter controls the minimum memory residency time for jobs just swapped into memory. See the *TSX-Plus System Manager's Guide* for further information about the CORTIM parameter. The form of this command is:

```
SET CORTIN value
```

where *value* is the time value specified in 0.1 second units. The current value of the CORTIM parameter may be determined with the SHOW CORTIM command. SYSPRV privilege is required to use this command.

3.5.59 SET CTRLD

The SET CTRLD command is used to enable or disable automatic entry to the TSX–Plus program debugger when CTRL–D is typed. Use of the program debugger and this command requires DEBUG privilege. Any time a program is running under control of the debugger, then CTRL–D may be used to interrupt the program execution and pass control to the debugger as if a breakpoint had been hit at the current location. Normally, this is only possible if the program was started under debugger control (/DEBUG switch to the R or RUN command). However, it is also possible to enable CTRL–D to interrupt programs which were not started under the debugger and pass control to the debugger. To enable this feature, issue the command:

```
SET CTRLD DEBUG
```

To disable CTRL–D interruption of programs which were not started under the debugger, issue the command:

```
SET CTRLD NODEBUG
```

See the description of the /DEBUG switch to the R command and the description of the debugger in the *TSX–Plus Programmer's Reference Manual* for more information.

3.5.60 SET EDIT

The SET EDIT command is used to select which edit program will be invoked when the system EDIT command is used. The form of this command is:

```
SET EDIT option
```

where *option* may be EDIT, TECO, KED or K52. The options KED and K52 are actually synonymous; the KED editor is used if the terminal type has been specified to be a VT100 or VT200 and the K52 editor is used if the terminal type has been specified to be a VT52. VT200 terminals must be set to 7-bit control characters to function with KED. The terminal must be SET TT LC in order to use KED or K52.

3.5.61 SET EMT

The SET EMT command is used to control tracing of EMT calls during the execution of a user program. Use of this command requires DEBUG privilege. The form of the command is:

```
SET EMT [NO]TRACE
```

When SET EMT TRACE is specified, a line of information about the EMT call is displayed at the terminal each time an EMT is executed. The .TTYIN, .TTYOUT and .PRINT EMTs, however, are not included in EMT traces. EMT tracing is disabled with the SET EMT NOTRACE command. Each line of information which is displayed during EMT tracing contains: the virtual address of the EMT call, the EMT code, function code, channel number (or sub-function code), and the first 5 words in the EMT argument block. Only the first two items are defined for all EMTs. The other items are defined only if used for the EMT currently being traced. As an example of EMT tracing, the LNTT program which displays the current line number and terminal type (see the *TSX–Plus Programmer's Reference Manual*) was traced as follows:

```
.SET EMT TRACE
.RUN LNTT

001004  374  004  000  000000  000000  000610  000000  000000

001012  375  110  000  057400  001252  001262  001270  001277
```

```
001026  374  005  000  057400  001252  001262  001270  001277
TSX-Plus line number: 2
Terminal type:
001060  375  137  000  001252  001262  001270  001277  001311
VT-100

001072  350  016  010  001270  001262  001270  001277  001311
.SET ENT NOTRACE
```

Note that the .PRINT calls are not traced, since .TTYIN, .TTYOUT, .PRINT, shared run-time and PLAS EMTs are never traced. The *TSX-Plus Programmer's Reference Manual* contains a list of both RT-11 compatible and TSX-Plus specific EMTs.

3.5.62 SET ENDSTARTUP

The SET ENDSTARTUP command is used within a start-up command file to denote the end of the portion of the start-up command file that runs with start-up privilege. See the *TSX-Plus System Manager's Guide* for more information about this command.

3.5.63 SET ERROR

The SET ERROR command is used to specify the level of error which will abort command file execution. The form of this command is:

```
SET ERROR option
```

where *option* may be FATAL, ERROR, WARNING or NONE. Command files being executed under the IND program are not normally aborted if errors occur during execution; the SET IND ABORT command can be used to cause IND command files to abort according to the same rules as for normal command files. The TSX-Plus SET ERROR command functions in the same way as the RT-11 SET ERROR command.

3.5.64 SET HIPRCT

The SET HIPRCT command is used to set the value of the HIPRCT system control parameter. See the *TSX-Plus System Manager's Guide* for more information on the effect of this parameter. The form of this command is:

```
SET HIPRCT value
```

where *value* sets the non-interactive job I/O counter. SYSPRV privilege is necessary to use this command. The HIPRCT parameter may be referenced by the SET SIGNAL and SHOW commands. Refer to the *TSX-Plus System Manager's Guide* for more information on job scheduling and performance optimization.

3.5.65 SET HOST

The SET HOST command is used to cross connect a time-sharing line with a CL (communication line) unit in such a fashion that all characters received from the time-sharing terminal are transmitted directly to the CL unit and all characters received from the CL unit are transmitted directly to the time-sharing terminal. This is useful to allow a time-sharing line on one TSX-Plus system to be connected through a CL line to another system and used as a remote terminal to it.

This function is similar to using RT-11's VTCOM to communicate through a CL line but is more direct and more efficient. On the other hand, this internal cross-connection feature does not provide the file transfer capabilities of VTCOM/TRANSF.

The form of this command is:

```
SET HOST/PORT=ddn
```

where the *ddn* is the name of the CL unit to which your terminal will be cross connected. For example, the following commands would *take over* terminal line 4 with CL unit 1 at 9600 baud and then cross connect the current terminal with the CL unit (CL1):

```
SET CL1 LINE=4,SPEED=9600
ALLOCATE CL1
SET HOST/PORT=CL1
```

See the SET CL command for more information on taking over time-sharing lines as CL units.

A logical name assigned to a CL unit may also be used as the port name. For example:

```
SET C11 LINE=21,SPEED=9600
ALLOCATE C11 VAX
SET HOST/PORT=VAX
```

While a cross connection is in effect, the program name assigned to the process is "$HOST$".

TERMINAL privilege is required to use the SET HOST command.

The cross connection can be released at any time by typing CTRL-\ (control backslash). (You should normally log out from the remote system before releasing the connection.) When this is done, the Data Terminal Ready (DTR) signal is dropped causing a dial-up line to be hung up, and the message "Cross connection released" is displayed. Control returns to the keyboard monitor. See the description below of the "X" function for a method of releasing the cross connection without dropping DTR.

When a cross connection is established, characters typed at your terminal are transmitted to the CL line. Control characters such as CTRL-C, CTRL-W, etc. are not interpreted by the local system. There are only four control characters which are interpreted by the local system:

CTRL-A	Signal special command
CTRL-Q	Resume terminal output (XON)
CTRL-S	Stop terminal output (XOFF)
CTRL-\	Release connection

(Different characters may be selected during system generation (see the *TSX-Plus Installation Guide*) as the special command and release connection characters.)

Typing CTRL-A signals the cross connection manager that one special character follows. When typed immediately after CTRL-A, the following characters perform the specified functions:

B	Send a 0.5 second break signal.
D	Raise DTR signal.
H	Drop DTR signal (hangup).
R	Reset flag that says an XOFF (CTRL-S) character has been received, and transmit an XON (CTRL-Q) character.
X	Release cross connection without dropping DTR signal.

If any other character is typed after CTRL–A, then that character is sent to the CL line without interpretation. This provides a way to transmit the CTRL–A and CTRL–\ characters. If you type CTRL–A twice, one CTRL–A character will be transmitted to the CL line. Typing CTRL–A CTRL–\ will cause a CTRL–\ character to be transmitted rather than releasing the cross connection.

The DTR signal is automatically raised when a SET HOST command is issued. The "D" and "H" function characters can be used to control the DTR signal after the connection is made. The DTR signal is automatically dropped when the cross connection is broken using a CTRL–\. If you wish to break the cross connection without dropping DTR, you may type CTRL–A X (CTRL–A followed immediately by the letter X).

3.5.66 SET IND

The SET IND [NO]ABORT command is used to control the execution of command files under the control of the IND program when there is an error. Normally, command files under the control of IND are not aborted when an error occurs, regardless of the current SET ERROR level. However, if the SET IND ABORT command is issued, then command files under the control of the IND program will abort under the same conditions as would normal command files. The SET IND NOABORT command restores the default abort processing under IND control (no abort).

3.5.67 SET INTIOC

The SET INTIOC command is used to set the value of the INTIOC system control parameter. See the *TSX-Plus System Manager's Guide* for more information on the effect of this parameter. The form of this command is:

```
SET INTIOC value
```

where *value* is the interactive job I/O counter. SYSPRV privilege is necessary to use this command. The INTIOC parameter may be referenced by the SET SIGNAL and SHOW commands. Refer to the *TSX-Plus System Manager's Guide* for more information on job scheduling and performance optimization.

3.5.68 SET IO

The SET IO [NO]ABORT command is used to select the method of handling I/O abort requests. If the SET IO ABORT command is issued, then an I/O abort request will call device handler abort entry points. If the SET IO NOABORT command is issued, then I/O abort requests will proceed through I/O rundown; that is, all pending I/O will complete before the job is aborted. The initial setting of this parameter is selected during TSX–Plus system generation. SYSPRV privilege is required to use this command. The method selected affects all lines, not just the line from which the command is issued.

3.5.69 SET KMON

The SET KMON command is used to direct the processing of commands from the keyboard and from command files. The form of this command is:

```
SET KMON option
```

where the valid *options* are: [NO]IND, UCI[=*filnam*] or USER[=*filnam*], and SYSTEM. The chain of events in command processing by TSX–Plus is described at the beginning of this chapter.

The processing of indirect command files may either be controlled by TSX–Plus or by the IND utility provided with RT–11. To cause command files to be processed by IND, use the command: SET KMON

IND. To return to the normal mode of TSX-Plus command file processing: SET KMON NOIND. Note that command files which result in errors are not normally aborted, regardless of the SET ERROR level, when executed under control of IND. The SET IND [NO]ABORT command can used to control error abort of IND command files. Command files may be forced to the normal TSX-Plus mode of execution regardless of whether KMON is set IND or NOIND by calling them as:

```
$@filnam
```

Conversely, command files may be forced to execute under the IND utility regardless of whether KMON is set IND or NOIND by calling them as:

```
IND filnam
```

See the *TSX-Plus Programmer's Reference Manual* for more information on command file processing.

The other function of the SET KMON command is to control the user command interface. It is possible for user written programs to accept and pre-process keyboard commands before the TSKMON program. Command control is local to each user. That is, each job may select its own command processing method. The User Command Interface may be enabled by the command:

```
SET KMON UCI[=filnam]
```

or

```
SET KMON USER[=filnam]
```

The option USER is equivalent to UCI. When UCI is in effect, then each time TSKMON is ready to accept a command it passes control to the current UCI program. If the optional *=filnam* has been omitted, then TSKMON passes command acquisition control to the program SY:UKMON.SAV. If a file has been specified, then TSKMON passes control to that program. It is the responsibility of the user-written command interface program to prompt for and accept command input lines. Commands may be further passed on to TSKMON through the chain-data area by doing a special chain exit.

Command processing control is returned to TSKMON by the command:

```
SET KMON SYSTEM
```

See the example program in the *TSX-Plus Programmer's Reference Manual* for more information.

3.5.70 SET LANGUAGE

The SET LANGUAGE command is used to select DBL or DIBOL as the default compiler for programs with the extension DBL. This also affects the COMPILE and EXECUTE commands. The form of the command is:

```
SET LANGUAGE option
```

where *option* is DBL or DIBOL. The default setting of this parameter is DBL.

3.5.71 SET LD

The SET LD command is used to control writing to a logical subset disk and to verify logical subset disk assignments. The form of the command is:

```
SET LDn option
```

where *LDn* is in the range of LD0 to LD7, and *option* may be CLEAN, EMPTY, FREE, WRITE or NOWRITE.

The following *options* are valid:

CLEAN is used to verify and correct logical subset disk assignments. An implicit SET LDn CLEAN is done by TSX-Plus whenever the DUP utility program is run (e.g., INIT and SQUEEZE operations).

EMPTY dismounts *ALL* logical subset disks for the process and deassigns logical device names assigned to them. The LD unit number is ignored with the EMPTY option.

FREE dismounts the specified logical subset disk and deassigns any logical name assigned to it.

[NO]**WRITE** is used to prevent/enable writing to a logical subset disk. WRITE/NOWRITE control is also an option of the MOUNT command.

See the MOUNT command and Appendix B for further information on the use of logical subset disks.

3.5.72 SET LOG

It is possible to copy terminal input and/or output to a log file. When terminal logging is enabled, all terminal I/O may be written to the log file. The only exception to this is high-efficiency terminal output which is never logged. To initiate terminal logging issue the following command:

```
SET LOG FILE=name
```

where *name* is the file specification for the log file. The default extension is .LOG. Optionally, any of the following may be specified to control the contents of the log file:

/ALL	-	Write both input and output to the log file (default selection).
/INPUT	-	Write only input to the log file.
/OUTPUT	-	Write only output to the log file.

For example, the following command would copy terminal output to a file named DK:RUNLST.LOG:

```
SET LOG FILE=RUNLST/OUTPUT
```

The following command causes terminal I/O to be copied to the line printer:

```
SET LOG FILE=LP:
```

Logging of terminal input and/or output may be stopped and the log file closed by the command:

```
SET LOG CLOSE
```

The log file is automatically closed when another log file is opened or the job logs off. The log file is also automatically closed if the device containing the log file is initialized or squeezed; the following warning message appears:

```
?KMON-W-Closing log file
```

It may be desirable to suspend and resume terminal logging during some operations without closing and reopening another log file. To temporarily suspend terminal logging, issue the command:

```
SET LOG NOWRITE
```

To resume terminal logging, issue the command:

```
SET LOG WRITE
```

It may be desirable at some times to reset the terminal log file. To clear the contents of the log file without closing and deleting the file, issue the command:

```
SET LOG CLEAN
```

This command resets the highest block written information for the file.

Terminal logging is especially useful with detached jobs. Terminal output to detached jobs is normally discarded since the job is not attached to any terminal. However, when terminal logging is enabled for a detached job, then the terminal output may be directed to a file or to a printer.

3.5.73 SET LOGOFF

The SET LOGOFF command is used to associate a *logoff* command file with a job. This command is only valid within start-up command files or if the job has SYSPRV privilege. The form of this command is:

```
SET LOGOFF FILE=name
```

where *name* is the file specification of a command file to be executed when the job logs off. Logoff command files, like start-up command files, cannot be aborted by CTRL–C. See the *TSX-Plus System Manager's Guide* for further information about logoff command files.

3.5.74 SET MAXPRIORITY

The SET MAXPRIORITY command may be used to reduce the maximum priority allowed to a job. When placed in a start-up command file, this command restricts the maximum available priority for that job. This permits restriction of priority on lines which do not use the LOGON facility. The form of this command is:

```
SET MAXPRIORITY value
```

where *value* is in the range of 0 to 127. If the current maximum job priority is less than 127, then the current maximum priority is the upper limit for the valid range of *value*. See the *TSX-Plus System Manager's Guide* for further information about restricting job priority.

3.5.75 SET NUMDC

The SET NUMDC command is used to control the number of buffers used for data caching. The form of this command is:

```
SET NUMDC value
```

where *value* is the number of buffers to use. The initial value of the NUMDC parameter is specified when the system is generated. The SET NUMDC command may be used to restrict the number of data cache buffers actually used to a value less than the number of buffers specified when the system was generated, but may not exceed that value. The SET NUMDC command does not alter the memory space allocated for cache buffers, it merely controls the number of buffers actually used. SYSPRV privilege is required to use this command.

The primary use of the SET NUMDC command is to determine the optimum number of data cache buffers to include in a system. To do this, the system can be generated with a large number of data cache buffers and the SET NUMDC command can be used to determine the minimum number which are actually needed to provide effective performance. When using this procedure, all files that are being cached should be closed before the SET NUMDC command is issued. See the *TSX-Plus Programmer's Reference Manual* for a discussion of shared-file data caching.

3.5.76 SET OFFTIM

The SET OFFTIM command is used to alter the amount of time a user may remain connected to a dial-up link after logging off before Data Terminal Ready (DTR) will be dropped causing the phone to hang up.The form of the command is:

```
SET OFFTIM n
```

where *n* is length of time allowed to elapse before DTR is dropped specified in 0.5 second units.

3.5.77 SET PRINTWINDOW

The SET PRINTWINDOW command is used to enable the printwindow control character to send an image of the current process window to a printer. The printwindow control character is normally CTRL-B, but it may be changed during system generation. The printwindow feature is only available if: the process has enabled windows with refresh with the SET WINDOW command; the SET PRINTWINDOW device has been specified; and the WINPRT program is running (usually as a detached job).

The form of this command is:

```
SET PRINTWINDOW [/DEVICE=ddn] [/TYPE=type] [/LETTER or /DRAFT]
                [/[NO]BELL] [/[NO]WIDTH] [/[NO]KEYPRINT]
```

where the parameters mean:

/DEVICE=*ddn* specifies the name of the printer device to be used when the printwindow control character is typed. This may be a printer name such as LP or LS, a CL unit such as CL2, or some other device name to which the window image is to be sent. A logical device name may also be used. The output device is usually spooled, but it is not required. The device must be specified at least once in order to enable the printwindow feature, but the SET PRINTWINDOW command may be issued without the device name at a later time to change the current printwindow device characteristics. A disk file may also be specified as the printwindow device. The window contents are written to a file on the specified device with a name of the form "WINjji", where *jj* is the job number and *i* is a letter corresponding to the window ID number (e.g., A=1).

/TYPE=*type* specifies the type of the printer. This enables the system to simulate the actual screen display as closely as possible by taking advantage of special printer character sets and features. *Type* may be: LA12, LA36, LA50, LA100, LA120, LA210, LQP02, LQP03, or LN03. If some other printer is used, specify FOREIGN as the type. The default type is FOREIGN.

/LETTER specifies that letter-quality print mode is to be used on those printer which support it. This qualifier is ignored on those printers which do not support letter-quality mode.

/DRAFT specifies that draft-quality mode is to be used. This is the opposite of /LETTER. /DRAFT is the default.

/[NO]BELL specifies that the bell is to be rung at the terminal that is requesting a printwindow when the WINPRT program has successfully captured the window image. When the bell is rung, a *snapshot* of the window contents has been taken so new data may be displayed without affecting the window image that will be printed. /NOBELL is the default mode.

/[NO]WIDTH specifies that control characters are to be sent to the printer to control character width spacing. This only applies to DEC terminals that support character width settings. /NOWIDTH suppresses character width control. /WIDTH is the default mode.

/[NO]KEYPRINT The /NOKEYPRINT qualifier disables the print window function from being triggered by typing the printwindow control character (normally CTRL–B). If this is done, the control character that causes a window print operation is passed through to the running program rather than being intercepted by the window management system. Note that the control character is also passed through if windowing is turned off for the process. The /NOKEYPRINT qualifier only affects window printing triggered by typing the control character—the window print EMT is not affected. The window print control character may be reenabled by use of the /KEYPRINT qualifier. /KEYPRINT is the default mode.

The SET PRINTWINDOW command applies only to the process issuing the command. Thus, different jobs may select different printwindow devices or modes.

3.5.78 SET PROCESS

The SET PROCESS command is used to alter job characteristics. The SET PROCESS command may be used to:

- change the priority of a job (within its authorized range)
- suspend or resume the execution of a job
- change the user name for a job
- or change the privileges of a job

The form of this command is:

```
SET PROCESS [/PRIORITY=value]
            [/SUSPEND]
            [/RESUME]
            [/IDENTIFICATION=value]
            [/NAME=string]
            [/PRIVILEGES=(privlist)[/AUTHORIZED]]
```

where the parameters mean:

/PRIORITY=*value* is used to specify a new job execution priority for the current job or for another job. The /IDENTIFICATION qualifier may be used with the /PRIORITY qualifier to cause the priority to be set for a job other than the current job.

The specified priority value must be in the range authorized for the job. The default priority assigned to a job when not otherwise specified or restricted is 50. The maximum priority allowed to a job may be restricted through the logon mechanism or the SET MAXPRIORITY command. The SHOW PRIORITY command can be used to display the current job priority and the maximum authorized

priority. The SYSTAT command also displays current job priorities. Job priorities may also be assigned from within an executing program with an EMT. See the *TSX-Plus Programmer's Reference Manual* for more information on setting job priority from within a program. When a job is disassociated from the terminal by switching to a different subprocess, the job is reduced in priority by an amount determined during system generation (the PRIVIR parameter).

/SUSPEND is used to suspend the execution of a job. The /IDENTIFICATION qualifier must be used to identify the job which is to be suspended.

/RESUME is used to resume the execution of a suspended job. The /IDENTIFICATION qualifier must be used to identify the job which was suspended.

/IDENTIFICATION=*value* is the number of the job whose status is to be set. This qualifier may only be used with the /PRIORITY, /SUSPEND, and /RESUME qualifiers. SAME, GROUP or WORLD privilege is required, depending on the project-programmer numbers of the target and issuing jobs.

/NAME=*string* can be used to specify a new user name for the job. The user name is displayed by the SYSTAT command. SETNAME privilege is required to change the job name. The name of another job may not be changed.

/PRIVILEGES=*privlist* is used to specify a list of privileges. If more than one privilege is being specified, the privilege keywords are enclosed in parentheses and separated by commas. If only a single privilege keyword is specified, the parentheses may be omitted. The word NO may be concatenated with a privilege keyword to cause the privilege to be removed from the job. For example, the following command grants the privilege represented by *priv1* and removes the privilege represented by *priv2*:

```
SET PROCESS/PRIV=(priv1,NOpriv2)
```

Job privileges are grouped into three sets:

Authorized Those for which the job is authorized.

Set Those which have been specified with the SET PROCESS command or the EMT to set privileges.

Current Those currently in effect for the job, based on the set privileges and program-dependent privileges for installed programs.

Unless the job has SETPRV privilege, the set privileges will not exceed the authorized privileges. See the *TSX-Plus System Manager's Guide* for more information on job privileges.

/AUTHORIZED may be used in conjunction with the /PRIVILEGE qualifier to cause the authorized privilege flags to be affected as well as the set and current privileges. If the /AUTHORIZED qualifier is not specified, only the set and current privileges are affected. The following command sets the selected privileges as the authorized, set, and current privileges for the job:

```
SET PROC/PRIV=(priv1,priv2,...)/AUTHORIZED
```

The SET PROCESS command can always be used to remove a privilege from the authorized, set, and current privileges for the job. Specific privileges can only be granted for a job if the job is authorized to set those privileges. SETPRV privilege is required to set privileges not specifically included in the authorized set. See the *TSX-Plus System Manager's Guide* for more information on job privileges. Specific privilege requirements are also described individually for system commands and system service calls.

3.5.79 SET PROMPT

The SET PROMPT command is used to change the keyboard monitor prompt character. The default character is a period (“.”). The form of the command is:

```
SET PROMPT string
```

where *string* is a quoted string containing from 1 to 8 characters. The quoted string will subsequently be used as the keyboard monitor prompt string. For example, in order to set the monitor prompt to a dollar-sign followed by a space, use the following command:

```
SET PROMPT "$ "
```

To restore the standard RT–11 prompt string, issue the command:

```
SET PROMPT "."
```

3.5.80 SET QUANxx

The SET QUANxx command is used to set the value of the system time-slice control parameters (QUAN0, QUAN1, QUAN1A, QUAN1B, QUAN1C, QUAN2, and QUAN3). See the *TSX–Plus System Manager's Guide* for information about the effect of the these parameters. The form of this command is:

```
SET QUANxx value
```

where *value* is the time value specified in 0.1 second units. SYSPRV privilege is required to use this command. The value selected takes effect for all jobs on the system, not just the job issuing the command. These parameters may be referenced by the SET SIGNAL and SHOW commands. See the *TSX–Plus System Manager's Guide* for more information on job scheduling and performance optimization.

3.5.81 SET RECALL

The SET RECALL command is used to determine the order the in which the commands are displayed when using RECALL/ALL command. The form of the command is as follows:

```
SET RECALL option
```

where the options are as described below:

NORMAL is used to cause the most recent command to be displayed first

REVERSE is used to cause the most recent command to be displayed last

3.5.82 SET SHUTDOWN

The SET SHUTDOWN command is used to begin a system shutdown sequence. This command is equivalent to the $SHUTDOWN command. The SET SHUTDOWN command sets a flag which prevents any new users from logging on and then waits for all active users to log off. When the last user logs off, TSX–Plus is stopped and control may return to RT–11. A SET NOSHUTDOWN command can be used to cancel a shutdown request between the time a SET SHUTDOWN (or $SHUTDOWN) command is issued and the system is shut down.

3.5.83 SET SIGNAL

The SET SIGNAL command is used as an aid to system performance tuning. The form of the SET SIGNAL command is:

```
SET SIGNAL [NO]parameter
```

where *parameter* may be one of the system scheduling parameters: HIPRCT, INTIOC, QUAN0, QUAN1, QUAN1A, QUAN1B, QUAN1C, QUAN2, or QUAN3. Only one parameter may be selected by each SET SIGNAL command, although with multiple SET SIGNAL commands more than one parameter may be selected for signaling at once. Signaling may be disabled for any individual parameter by prefacing the parameter with NO. Signaling may be halted for all parameters with the command SET SIGNAL OFF.

When signaling has been enabled for a system tuning parameter, the bell will be rung at the terminal of the job for which signaling has been selected each time the job changes state because it exceeds the value of the selected parameter. The SET SIGNAL command functions on a line-by-line basis and only affects the line from which the command is issued.

The signaling feature is intended as an aid to the system manager in determining appropriate values of the system tuning parameters for a given job. See the *TSX-Plus System Manager's Guide* for more information on use of the signaling feature.

3.5.84 SET SL

The SET SL command is used to control the ability to edit keyboard input lines. The TSX-Plus SL facility is generally compatible with that provided with the RT-11 SL editor. It allows editing of the current input line or field and recall of the previous input line or field. Logical names may not be used in lieu of SL. See the section on the single line editor in Chapter 2 for information on the use of the single line editor. The form of this SET command is:

```
SET SL option
```

where the options are described below:

ASK This command is ignored. The terminal type is determined from the current TSX-Plus terminal type.

[NO]KED This command disables/enables KED-like extensions to the single line editor. Note that the numeric keypad is reset/set in application mode. SET SL NOKED is equivalent to SET SL RT11. The alternate forms KEX and K52 are also accepted, although the NO forms are not.

[NO]LEARN This command is ignored. The single line editor LEARN mode is not implemented.

[NO]LET This command is ignored. The LET utility may not be used with the single line editor.

OFF This command turns off the single line editor.

ON This command turns on the single line editor. Note that this places the process in single-character activation mode.

KMON This command is the same as SET SL ON.

[NO]RECALL This command is ignored. When the single line editor is active, the RECALL ability is always enabled.

RT11 This command disables the KED-like extensions to the single line editor. This command is equivalent to SET SL NOKED.

[NO]SUBSTITUTE This command disables/enables defined terminal key substitutions. See the DEFINE/KEY command.

SYSGEN This command is ignored. The single line editor is implemented as a TSX–Plus system overlay region and does not require the SL pseudo-device handler.

[NO]TTYIN This command either enables (TTYIN) or disables (NOTTYIN) editing of keyboard input being accepted via the .TTYIN EMT from within programs.

WIDTH=*n* This command is ignored. The maximum input line width that can be used with the single line editor, either as a command line or in a program data field, is 80 characters.

3.5.85 SET SUBPROCESS

The SET SUBPROCESS command is used to specify a command file to be executed whenever a subprocess is initiated. This command is only valid in start-up command files or if the job has SYSPRV privilege. See the *TSX–Plus System Manager's Guide* for more information on the SET SUBPROCESS command.

3.5.86 SET SYSPASSWORD

The SET SYSPASSWORD command is used to temporarily change the system password for those lines which require it prior to the greeting message. The change does not remain in effect if the system is rebooted. See the *TSX–Plus System Manager's Guide* for more information on the system password. SYSPRV privilege is required to use this command.

3.5.87 SET TERMINAL

The SET TERMINAL command is used to set various terminal parameters. It is equivalent to the SET TT command.

3.5.88 SET TT

The SET TT command is used to set various terminal and job parameters, and is equivalent to the SET TERMINAL command. Logical names may not be used in lieu of TT. The form of this SET command is:

```
SET TT [line-number] [NO]option
```

to turn an option off and on. For example:

```
SET TT FORMO
```

Some options require a numeric parameter, in which case they are specified as:

```
SET TT [line-number] option=value
```

For example:

```
SET TT 6 SPEED=9600
```

A single command may specify multiple parameters and options. If there is more than one qualifier, the qualifiers may be separated by a comma, a space, or a slash. For example:

```
SET TT SPEED=9600,BITS=8,NOPARITY
```

The optional *line-number* parameter refers to the TSX-Plus line number, which is determined by the order in which time-sharing lines were declared during system generation. The hardware interface to which each line is attached may be determined from the SHOW TERMINALS command. Only physical line numbers may be specified with the SET TT command, not detached jobs or subprocesses.

If *line-number* is omitted from the command, the setting is applied to the terminal from which the command is issued. TERMINAL privilege is required to issue a SET TT command for any line other than your own or to make a permanent SET to your own line.

If *line-number* is specified, then the setting becomes permanent and will remain in effect until changed or until the system is restarted. If *line-number* is omitted, then the setting is temporary and will be reset when the line logs off. Terminal SPEED setting is an exception to this rule. Line speed is additionally affected by the AUTOBAUD option. If a line is set AUTOBAUD, then a temporary change reverts to AUTOBAUD when the line logs off, whereas a permanent change clears AUTOBAUD and remains in effect across logoffs. If a line is not set AUTOBAUD, then any SPEED setting, either with or without *line-number* specified, is permanent and remains in effect across a logoff.

The following are valid options for the SET TT command:

7BIT This command is equivalent to SET TT NOEIGHTBIT.

[NO]8BIT This command is equivalent to SET TT [NO]EIGHTBIT.

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.

[NO]AUTOBAUD Allows automatic baud rate selection for terminals during log on. This is only functional for lines connected through interface cards which support programmable baud rates. TERMINAL privilege is required to use this command even for your own line. Only the following speeds are supported with automatic baud rate selection: 110, 300, 1200, 2400, 4800, 9600, 19200. Two carriage-returns are necessary to start an AUTOBAUD line if the terminal speed is less than 1200 baud. The NOAUTOBAUD option disallows automatic baud rate selection. Use of the SET TT *line-number* SPEED command also cancels automatic baud rate detection. Specifying AUTOBAUD also selects 8 bit characters and no parity; terminals should be set accordingly, although character length and parity may be changed after the connection is established. When the connection to an AUTOBAUD line is broken, the line reverts to 8 bits and no parity and the speed becomes indeterminate.

BITS=*n* Specifies the number of bits to be used by the hardware line controller for the line. Note that not all line controllers support programmable character lengths; this command has no effect if not supported by the hardware line controller. TERMINAL privilege is required to use this option even for your own line. This option is not the same as the EIGHTBIT option which specifies whether or not the operating system masks characters to seven bits. In order to transmit and receive eight bit characters, both EIGHTBIT and BITS=8 must be specified.

[NO]DEAD Disables further logons on the line. An active line cannot be set DEAD. The NODEAD option re-enables logons. A CL unit may not be assigned to a line flagged as DEAD.

DECWRITER Equivalent to the LA36 option.

[NO]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 echoed to the terminal until the program is ready to accept them. Under standard RT-11, EDIT, BASIC, and DIBOL programs simulate deferred mode. Most other programs use immediate echoing. Deferred echoing is the preferred mode under TSX-Plus. See the *TSX-Plus Programmer's Reference Manual* for information about how a program can control deferred echo mode.

DIABLO Has the effect of SET TT NOSCOPE, FORM, NOTAB, PAGE. The QUME type is equivalent. When doing plotting or printing with proportional spacing, SET TT TAB.

DTR Controls the raising or lowering of the DTR (data terminal ready — pin 20 on an EIA/RS232-C connector) signal on a line. This is only effective if the interface supports modem control signals (e.g., this has no effect on a DLV11J port). This can be useful in situations where a modem or data PBX requires the DTR signal.

The DTR signal is also manipulated by the system when connecting and disconnecting "phone" lines.

The form of the command is:

```
SET TT n [NO]DTR
```

where *n* is the line number to which the command is issued.

[NO]ECHO Controls echoing of characters to the terminal. See the *TSX-Plus Programmer's Reference Manual* for information about how a program can control character echoing.

[NO]EIGHTBIT Controls 8-bit I/O for a terminal line. When in 8-bit mode, all 8 bits of each character are passed to and from a terminal. The null character (000) may never be received from the terminal. In addition, in EIGHTBIT mode, if VT*xxx* escape sequences are defined as a terminal input activation condition, then the character 377 (octal) may not be received. In 8-bit mode, terminals should be set for 8-bit characters with no parity. VT2*xx* terminals should be set to VT200 mode with 7-bit control characters. The EIGHTBIT option controls operating system masking of characters to seven bits; it is not the same as the BITS=*n* parameter which selects the number of data bits in the hardware line controller. In order to transmit and receive eight bit characters, both EIGHTBIT and BITS=8 must be specified.

[NO]FORM Controls conversion of form feed (FF) characters to line feeds. FORM should be set with terminals whose hardware can respond to form feed characters. NOFORM should be used for terminals whose hardware cannot handle form feed characters. When NOFORM is set, form feed characters are replaced by eight line feed characters.

[NO]FORM0 The FORM0 option causes TSX-Plus to first issue a form-feed 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. The NOFORM0 option disables special handling of block zero writes.

[NO]GAG The GAG option inhibits messages sent from another line from being displayed at the terminal. This is only in effect when the job is executing a program. Messages are not inhibited while in the keyboard monitor. The NOGAG option allows messages to be displayed at any time. The GAG option is useful for preventing messages from interrupting jobs on hardcopy terminals, such as printers or plotters, where a message could spoil the format. This command only affects messages sent either with the SEND command or the EMT which sends a message to another job. This does not apply to message communication channels.

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

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

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

[NO]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. If the terminal is set NOLC *or* the job status word bit 14 is clear, lower case characters are translated to

upper case. See the *TSX-Plus Programmer's Reference Manual* for information about how a program can control lower-case character conversion. Note that in order to use the keypad editors KED and K52 the terminal must be SET TT LC, otherwise some of the keypad functions will not work.

[NO]**PAGE** Selects XON/XOFF flow control for the line.. This is the only flow control method allowed for use with TSX-Plus. PAGE turns on flow control, and NOPAGE turns it off. PAGE is automatically selected if any of the following terminal types is set: VT100, VT200, VT52, LA120, Diablo, Qume. The SET TT NOPAGE command can be used to override this setting after the terminal initialization has been done. This command only affects the primary line. Attempts to change this mode from a subprocess are ignored.

PARITY=*xxxx* Selects parity control for the terminal hardware line controller. This option has no effect if the hardware does not support programmable parity control. TERMINAL privilege is required to use this option even for your own line. The valid options are: EVEN, ODD or NONE. The alternate form, NOPARITY, is equivalent to PARITY=NONE.

[NO]**PHONE** Selects modem control. NOPHONE disables modem control, indicating that the line is hardwired to a terminal. See the *TSX-Plus System Manager's Guide* for more information on modem control under TSX-Plus.

[NO]**QUIET** Setting the terminal QUIET suppresses the listing of command files as they are executed. When the terminal is set NOQUIET, command file lines are listed as they are executed. See the *TSX-Plus Programmer's Reference Manual* for additional information on controlling command file listing.

QUME Equivalent to the DIABLO type. Has the effect of SET TT NOSCOPE, FORM, NOTAB, PAGE. When doing plotting or printing with proportional spacing, SET TT TAB.

[NO]**SCOPE** Tells TSX-Plus whether the terminal is a CRT device. Setting SCOPE causes the DELETE key to echo as a backspace-space-backspace sequence, erasing the previously typed character. See the *TSX-Plus Programmer's Reference Manual* for information about how a program can specify an alternate *rubout filler* character that will be used to overwrite characters being erased when DELETE is typed. When the terminal is set NOSCOPE, the DELETE key causes preceding characters to be echoed in reverse order as they are removed from the input buffer, and CTRL-U echoes carriage return, line feed. CTRL-R can be used to check the current contents of the input buffer when doing rubout editing. See Chapter 2 for information on other special control characters.

SEVENBIT This command is equivalent to SET TT NOEIGHTBIT.

[NO]**SINGLE** Controls automatic single character activation for programs. If the SET TT SINGLE command has been issued, then programs may control single character activation on terminal input by toggling bit 12 in the job status word. The /SINGLE switch to the R[UN] command and the "S" program controlled terminal option also allow individual programs to control single character activation. The SET TT SINGLE command applies to all programs until the SET TT NOSINGLE command is issued. If the SET TT NOSINGLE command is in effect, then programs may still individually control single character activation with the R[UN]/SINGLE switch or the "S" program controlled terminal option. In all cases, the program must still set bit 12 in the job status word to achieve single character activation. The SET TT SINGLE command differs from the R[UN]/SINGLE switch in that it only affects setting of single character activation (JSW bit 12), not terminal no-wait input (JSW bit 6). The SET TT NOWAIT (see below) command may be used to allow no-wait terminal input.

SPEED=*n* Specifies the transmit/receive speed for a terminal line. The available speed values are: 75, 110, 134.5, 150, 300, 600, 1200, 1800, 2000, 2400, 3600, 4800, 7200, 9600, and 19200. This command is only functional for those lines connected through interface cards which support programmable baud rates (DLV11-E, DH(Q,U,V)11, DZ(Q,V)11 and the PRO-350 printer and communications ports and Quad Serial Line multiplexer). TERMINAL privilege is required to set this option even for your own line. DEC DZ(Q,V)11 interfaces do not support 19200 baud. DHV11 interfaces do not support 3600 or

7200 baud. DH11 interfaces do not support 2000, 3600 or 7200 baud. The PRO QSL multiplexer does not support 3600 or 7200 baud. SET TT *line-number* SPEED cancels automatic baud rate selection, which may be restored with the SET TT AUTOBAUD command.

START The SET TT *n* START command initiates a time-sharing line as if a carriage-return was received from the terminal. OPER and TERMINAL privilege are required to use this command. If the line being started is set for autobaud speed recognition, it defaults to 9600 baud as it is started. If the terminal is known to be set to another speed, combine the speed setting with the START option to avoid speed mismatch errors. For example:

```
SET TT 7 SPEED=19200,START
```

[NO]SYSPASSWORD Requires the line to enter the current system password before logging on.

[NO]TAB Controls conversion of TAB characters to multiple spaces. TAB should be selected with terminals whose hardware can respond to TAB characters. When set NOTAB, TAB characters being sent to the terminal are replaced by an appropriate number of spaces.

[NO]TAPE Causes TSX-Plus to ignore received line-feed characters. See the *TSX-Plus Programmer's Reference Manual* for information about how a program can turn tape mode on and off.

TRANSLATE Causes translation of characters from *external* values as they are received to *internal* values as they are processed by the system; a reverse translation from *internal* values to *external* values occurs as the characters are transmitted. This feature is primarily intended to aid TSX-Plus users that have terminals with non-English character sets but it may have other uses as well.

VT50 Equivalent to the VT52 option.

VT52 Tells TSX-Plus that the terminal being used is a VT52 (or VT100 in VT52 mode) and also has the effect of SET TT SCOPE, TAB, PAGE, NOFORM.

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

VT200 Tells TSX-Plus that the terminal being used is in the VT2*xx* series and also has the effect of SET TT SCOPE, TAB, PAGE, NOFORM, EIGHTBIT. The terminal type may also be specified as VT220, VT240 or VT241, all of which are equivalent to VT200.

[NO]WAIT Normally, TSX-Plus blocks the execution of a program that does a .TTINR EMT if no activation character has been received. This occurs even if the program sets bit 6 (inhibit terminal wait bit) in the Job Status Word which is supposed to mean that the program can do non-blocking .TTINR character tests. This is 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 .TTINRs if bit 6 is set. The example program STEALS in the section on requesting exclusive system control in the *TSX-Plus Programmer's Reference Manual* demonstrates the program controlled terminal option to allow NOWAIT input.

XON Causes a reset of the XOFF (CTRL-S) state for a terminal line and to start output to the line. This only affects XOFFs received by the system from the terminal, not XOFFs sent to the terminal by the system. This can be used when a terminal gets hung in XOFF sent and XOFF received state. This condition usually causes a terminal to be permanently hung in the TO (terminal output pending) state and cannot be killed. Successive use of the commands SET TT *n* XON and KILL *n* can usually be used to clear the line.

Note that when you type CTRL-S on a buffered terminal such as a VT100 or VT200, this does not immediately send a CTRL-S to the computer but rather stops the terminal from displaying more characters. The terminal itself generates CTRL-S when its input silo is almost full. In this case,

clearing the XOFF state within TSX–Plus will allow characters to be transmitted but the terminal may generate another CTRL–S as characters are received until CTRL–Q is typed at the keyboard reenabling its output. Thus, the SET XON command resets the *system* XOFF state but does not reset the *terminal* XOFF state. The form of the command is:

```
SET TT n XON
```

where *n* is the job number to which this command is directed.

Most of these terminal 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 setting specified in TSGEN. When a user initiates a subprocess, the initial flag settings for the subprocess are copied from the flag settings for the primary process. Subsequent flag changes for a subprocess do not affect flag settings for the user's other lines. The SYSMON utility may be used to determine the current settings for any time-sharing line.

3.5.89 SET TIMOUT

You can specify the amount of time that carrier must be lost on a dial-up line before the connection is assumed to have been broken. This value is also used to time-out lines which ring and do not raise carrier. The form of the command is:

```
SET TIMOUT n
```

where *n* is the amount of time allowed to elapse before the line is logged off specified in 0.5 seconds units.

3.5.90 SET UCL

You can specify the order in which the TSX–Plus command interpreter checks for user-defined commands by use of the SET UCL command. This command has four forms:

```
SET UCL FIRST
SET UCL MIDDLE
SET UCL LAST
SET UCL NONE
```

If the SET UCL FIRST command is used, user-defined commands will be processed before system commands. This allows user-defined commands to replace system commands but makes the processing of system commands slower. This is the required setting only if it is necessary to replace some system commands.

If the SET UCL MIDDLE command is used, user-defined commands are processed after system commands but before checking for command files and SAV files with names that match the command keyword. Using this setting, it is not possible to replace a system command with a user command, but both system commands and user-defined commands are processed relatively quickly. This is the recommended setting unless it is desirable to replace system commands.

If the SET UCL LAST command is used, a command will not be checked to see if it is a user-defined command until after it is checked to see if it is a system command, the name of a command file on DK, the name of a command file on SY, or the name of a SAV file on SY. Using this setting, it is not possible to replace a system command with a user command and user commands cannot have the same name as command files or SAV files. System commands are processed quickly (the same speed as SET UCL MIDDLE), but the processing of user-defined commands is slow. This is the appropriate setting only if user-defined commands are desired,

but command files already exist whose names would conflict with user-defined commands. Existing command files which are short and merely execute system commands should be replaced by user-defined commands.

If the SET UCL NONE command is used, user defined commands are never interpreted. In this mode, attempts to invoke user-defined commands will result in the error:

```
?KMON-F-Unrecognizable command
```

The following list illustrates where the FIRST | MIDDLE |LAST setting causes the command interpreter to check for and process user-defined commands:

First ⟶
See if command is a system command
Middle ⟶
Look for command file on DK: with command name
Look for command file on SY: with command name
Look for SAV file on SY: with command name
Last ⟶

See the beginning of this chapter for further information on the command interpretation process.

3.5.91 SET [NO]VERIFY

The SET VERIFY command is used to control command file listing. It is the reciprocal of the SET TT QUIET command. SET VERIFY causes command files to be listed as the commands are executed. SET NOVERIFY inhibits command file listing. Also see Chapter 4 for more information on command files. The following commands are equivalent:

```
SET VERIFY ≡ SET TT NOQUIET

SET NOVERIFY ≡ SET TT QUIET
```

3.5.92 SET VM

The SET VM command is used to control the amount of memory available to the VM pseudo-device. The SET VM command is normally not necessary as VM will automatically calculate the correct base address to use, starting just above the last address used by TSX–Plus, and using all physical memory installed above that address. The form of this command is:

```
SET VM BASE=nnnnnn
```

where *nnnnnn* represents the octal value of bits 6 through 22 of the base memory address which VM is allowed to use. For example, in a system with 1.5 megabyte of memory, to reserve the top 256 Kb of memory for use by VM the value of *nnnnnn* should be 50000. This corresponds to a base address for VM of 5000000. If this command is issued, then VM will use the higher of the address specified by *nnnnnn* or the top of TSX–Plus. That is, it is not permitted to set the base of VM to overlap memory regions reserved for use by TSX–Plus.

Normally, VM will calculate the top address to use as the absolute top of physical memory. You may decrease this top address. The form of the command to adjust the top address used by VM is:

```
SET VM TOP=nnnnnn
```

where *nnnnnn* represents bits 6 through 22 of the top memory address (in octal) which VM is allowed to use. For example, if you wish to set the top address of VM to end at 1280 kilobytes, then *nnnnnn* should be 50000 since the memory address is 5000000 (octal). Any time a new base or top address is defined, VM should be initialized.

3.5.93 SET WILDCARDS

The SET WILDCARDS command is used to control substitution of missing parts of file names. The form of the command is:

```
SET WILDCARDS option
```

where *option* is either IMPLICIT or EXPLICIT. If IMPLICIT is chosen, then file specifications which are incomplete will be treated as though the "*" character had been typed for the missing part of the specification. The only parts of file specifications affected by this substitution are the file name and the extension. The device cannot be specified as a wildcard. Most keyboard commands which accept file names also accept wildcards. The following table demonstrates the interpretation of various file specifications when wildcards have been set IMPLICIT or EXPLICIT.

Command	IMPLICIT	EXPLICIT
cmd TEST.FOR	cmd TEST.FOR	cmd TEST.FOR
cmd TEST.*	cmd TEST.*	cmd TEST.*
cmd TEST	cmd TEST.*	cmd TEST
cmd TEST.	cmd TEST.	cmd TEST.
cmd *.FOR	cmd *.FOR	cmd *.FOR
cmd .FOR	cmd *.FOR	cmd .FOR
cmd *.*	cmd *.*	cmd *.*
cmd *	cmd *.*	cmd *
cmd .*	cmd *.*	cmd .*
cmd	cmd *.*	cmd

where *cmd* is one of the keyboard commands which accepts wildcards in file specifications, such as: COPY, DIRECTORY, PRINT, PROTECT, etc.

The system-wide default setting for wildcard interpretation may be selected in TSGEN; see the *TSX-Plus Installation Guide*.

3.5.94 SET WINDOW

The SET WINDOW command is used to enable the system to remember entire screens of information when switching among subprocesses. Use of this command requires SYSGBL privilege. When windowing is first enabled, or when switching to a new subprocess, the screen is cleared. Subsequent terminal output or input echoing is remembered by the system independently for the window connected to each subprocess. The entire screen is then refreshed when the terminal is logically connected to a subprocess which has windows enabled. See Chapter 2 for more information on screen windows. Screen windows may only be used with VT200, VT100 and VT52 series terminals. The COLUMNS, DARK, LIGHT, WIDE, and NARROW qualifiers are not valid in VT52 mode. Color is not supported. Note, be sure the terminal type (VT52, VT100, or VT200) declared to TSX-Plus, either during system generation or by use of the SET TERMINAL command, matches the actual terminal type before enabling windows.

The form of the SET WINDOW command is:

```
SET WINDOW [ON] [OFF] [COLUMNS=n] [LIGHT] [DARK]
           [WIDE] [NARROW] [SCROLL[=n]]
```

SYSGBL privilege is required to use windows since they create global memory regions.

The following qualifiers may be used with the SET WINDOW command:

COLUMNS=*n* specifies the number of columns per line which the window can hold. (VT100 and VT200 only.) The value should not exceed 132.

DARK specifies that the window is to be in normal (not reverse) video (i.e., light characters on dark background). (VT100 and VT200 only.)

DELETE is equivalent to OFF.

LIGHT specifies that the window is to be in reverse video (i.e., dark characters on light background). (VT100 and VT200 only.)

NARROW specifies the window is to hold 80 columns (the COLUMNS=80 option is implied) and causes the terminal to be placed into 80 column mode. (VT100 and VT200 only.)

OFF deletes all windows for the job.

ON creates a new window number 1 for the job. See the *TSX-Plus Programmer's Reference Manual* for more information on multiple windows for the same job.

SCROLL[=*n*] specifies the maximum number of lines which are allowed to scroll off the window while you are disconnected from it by switching to a different subprocess. If you specify zero (0), then no lines are allowed to scroll off the screen. If you specify a value in the range 1 to 127, the job writing to the window will be suspended, and the bell will ring at your terminal, if more than the specified number of lines are attempted to be scrolled. Values greater than 127 are divided by 128 and the remainder used as the scroll value. If the number of lines is unspecified, then an unlimited number of lines is allowed to scroll off the screen.

WIDE specifies the window is to hold 132 columns (the COLUMNS=132 option is implied) and causes the terminal to be placed into 132 column mode. (VT100 and VT200 only.)

The default qualifiers are: ON, COLUMNS=80, DARK, SCROLL=16. Thus, you can initialize a window with 80 column mode by simply typing:

```
SET WINDOW
```

The SET WINDOW OFF command deletes all windows for the job.

Display windows remember all characters sent to the screen by programs or from echoing of input characters. However, messages sent to the terminal by use of the SEND command (or equivalent EMT) are not processed by the message manager. Consequently, received messages will not appear if a window is redrawn. You can cause the window for the current job to be redrawn without switching to a different subprocess by typing CTRL-W followed by the number that selects your current line. For example, if you are running on your primary line and wish to have the current window redrawn, type CTRL-W 0. Terminal attributes such as line width, reverse/normal video, application keypad mode, etc. are saved along with line attributes (double wide, double high), and character attributes. The character attributes are: blinking, bold, underlined, reverse video, and character set information (ascii, U.K. national, DEC supplemental, or graphics).

In order to use DEC multi-national character sets with process windowing, the terminal must be in VT200 mode with either 7-bit or 8-bit controls, and must transmit 8-bit characters (we recommend 8 bit, no parity). TSX-Plus must recognize the terminal as a VT200 type and must be set for both software and hardware eight-bit handling. For example:

```
SET TT (n) VT200,8BIT,BITS=8,NOPARITY
```

The SET WINDOW command must be issued after the terminal is known to be a VT200 type. If these conditions are not met, then a typical symptom would be that accented characters displayed on one window would be incorrectly redisplayed as ASCII after switching to another window and then switching back.

3.5.95 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.

3.5.96 SHOW

The SHOW command without qualifiers is equivalent to SHOW ASSIGNS.

3.5.97 SHOW ALL

The SHOW ALL command is equivalent to specifying all of: SHOW VERSION, SHOW DEVICES, SHOW ASSIGNS, SHOW ALLOCATE, SHOW JOBS, SHOW TERMINALS, SHOW CL, SHOW MEMORY, SHOW SPOOL, SHOW SUBSETS, SHOW MOUNTS, SHOW RUN-TIMES, SHOW REGIONS, SHOW INSTALL, SHOW PRIVILEGE, SHOW SL, SHOW VM.

3.5.98 SHOW ALLOCATE

The SHOW ALLOCATE command displays a list of all devices allocated for exclusive use by any job on the system. For example:

```
.SHOW ALLOCATE
Device  Job   User name
------  ---  ------------
  DB5     7   MINSY
  MT0    15   BOROGROVES
```

3.5.99 SHOW ASSIGNS

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

```
.SHOW ASSIGNS
Assignments:
SY  --> DL0:
CBL --> RK0:
TMP --> DL3:
DK  --> LD0:
```

3.5.100 SHOW CACHE

The SHOW CACHE command reports the total number of blocks available in the generalized data cache buffer. For example:

```
.SHOW CACHE
Number of blocks in data cache = 1000
```

3.5.101 SHOW CL

The SHOW CL command is used to determine the current characteristics and assignments of all CL units included during TSX–Plus system generation. CL units are typically used either to control serial devices like printers and plotters or as communication lines to other computer systems. CL units may either be dedicated only for use as CL ports, or may be generated as extra units that can be used to take over inactive time-sharing lines or dedicated CL lines which do not have any currently assigned CL unit. CL units may

also be spooled (see Chapter 6 for more information on device spooling). Several parameters may be set for each individual CL unit and the SHOW CL command is used to determine the current settings for each unit. See the SET CL command for descriptions of the various parameters which may be set for CL units. The SHOW TERMINALS command also indicates which lines are currently assigned as CL units. The *Line* column indicates which TSX-Plus line number each CL unit is assigned to. The *Job* column shows the line number of any job which is either currently using or has allocated each CL unit. CL units are numbered from 0 up to the highest number generated (16 maximum). CL units are numbered CL0 to CL7, followed by C10 to C17 if more than eight units are generated. For example:

```
.SHOW CL
CL version number 18.

Unit  Line   Job          Options
----  ----   ----  --------------------------
 CL0    6      2   [TAB,LC,LFIN,CR,CTRL]
 CL1   10   none   [LC,LFOUT,LFIN,CR,CTRL]
 CL2 none   none   (spooled)
```

3.5.102 SHOW COMMANDS

The SHOW COMMANDS command is used to display the currently available user-defined command definitions. See the beginning of this chapter for more information on declaring and using user-defined commands. Note that escape characters embedded in commands are represented by a dollar sign ("$"). For example:

```
.SHOW COMMANDS
C100      :== _DISPLAY $[H$[J
DEF       :== _ASS ^ DK
EXAM      :== _R KED ^/I
KPAD      :== _DISPLAY $=
NEW       :== _R DIR ^/D
NOKPAD    :== _DISPLAY $>
Q         :== _SPOOL LP,STAT
SUB       :== _SH SUB
TOME      :== _DISMO LDO\_ASS DL2 DK\_SH SUB
WORK      :== _DISMO LDO\_MOU LDO DL2:WORK DK\_SH SUB
```

3.5.103 SHOW CONFIGURATION

The SHOW CONFIGURATION causes a display of certain hardware and operating system characteristics. It does not invoke the RT-11 RESORC utility, which is unsupported for use with TSX-Plus. SHOW CONFIGURATION is equivalent to SHOW DEVICES and SHOW TERMINALS.

3.5.104 SHOW CORTIM

The SHOW CORTIM command displays the current value of the system parameter CORTIM. See the *TSX-Plus System Manager's Guide* for more information on the significance of the CORTIM parameter. For example:

```
.SHOW CORTIM
   2
```

3.5.105 SHOW DEVICES

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

```
.SHOW DEVICES
        Active                Handler  Handler  Handler
Device    I/O    Status       V. base  P. base  size     CSR     Vector
------  ------   ------       -------  -------  -------  ------  ------
  TT       0     000004
  LD       0     102446
  CL       0     006057       047102   000000    612
  DL       0     102405       120000   054075   1380     174400  160
  DY       0     102406       100710   000000    746     177170  264
  DM       1     102423       102262   000000   1516     177440  210
  DU       0     102450       105236   000000    846     172150  154
  NL       0     000025       120000   054074     58
  LS       0     020041       120000   054062    624     176510  310 314
```

Device is the name of the device handler. *Active I/O* is the total number of I/O operations for all units queued to the handler but not yet completed. *Status* is the device status word (see the *RT-11 Programmer's Reference Manual*, .DSTATUS request, for a description of this value). *Handler V. base* is the device handler virtual base address (load address reported by .DSTAT request; 120000 for mapped handlers). *Handler P. base* is the device handler physical base address (22-bit physical address divided by 100 octal; 0 for unmapped handlers). *Handler size* is the device handler size (decimal bytes). *CSR* is the device CSR (Control and Status Register) address. *Vector* is the device interrupt vector(s) address.

The CSR and vector values are only displayed if the user issuing the SHOW DEVICES command has BYPASS or SYSPRV privilege. Some of this information is not displayed for pseudo-devices such as TT, NL, and LD.

3.5.106 SHOW HIPRCT

The SHOW HIPRCT command displays the current value of the system parameter HIPRCT. See the *TSX-Plus System Manager's Guide* for more information on the significance of the HIPRCT parameter. For example:

```
.SHOW HIPRCT
  40
```

3.5.107 SHOW INSTALL

The SHOW INSTALL command is used to display information on programs which have been installed for execution with certain characteristics or privilege. Use of this command requires SYSPRV privilege. See the *TSX-Plus System Manager's Guide* for more information on installed programs.

3.5.108 SHOW INTIOC

The SHOW INTIOC command displays the current value of the system parameter INTIOC. See the *TSX-Plus System Manager's Guide* for more information on the significance of the INTIOC parameter. For example:

```
.SHOW INTIOC
  30
```

3.5.109 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.

3.5.110 SHOW KEYS

This command is used to display all current defined key definitions. For example:

```
.SHOW KEYS
Gold J          R TALK 5
     PF2        HELP
```

3.5.111 SHOW MAXMC

The SHOW MAXMC command displays the maximum number of message communication channels which may be simultaneously in use. For example:

```
.SHOW MAXMC
  5
```

3.5.112 SHOW MAXMRB

The SHOW MAXMRB command displays the maximum number of requests for messages that may be simultaneously held in queue. For example:

```
.SHOW MAXMRB
  5
```

3.5.113 SHOW MAXMSG

The SHOW MAXMSG command displays the maximum number of messages which may be simultaneously held in queue. For example:

```
.SHOW MAXMSG
  6
```

3.5.114 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. For example:

```
.SHOW MEMORY
Total installed memory = 1536Kb
Size of unmapped TSX and handlers = 37Kb
Size of mapped TSX system regions = 75Kb
Total size of TSX and mapped data = 132Kb
Size of sharable run-time systems = 2Kb
Size of data cache buffer area = 516Kb
Space available for user jobs = 830Kb
Swappable job context area = 6Kb
Current job memory limit = 56Kb
Maximum job memory limit = 64Kb
```

3.5.115 SHOW MOUNTS

The SHOW MOUNTS command displays the names of those devices that have been mounted by use of the MOUNT command and whose directories are being cached. TSX-Plus maintains a list of which jobs have mounted each device. This list is used with the INITIALIZE and SQUEEZE commands to reduce the chance of data destruction by altering a device directory while another job has a file open on the device. For logical subset disks, the name of the file which contains the device is also shown. Note that with nested logical subset disks, the table formatting is relaxed. For example:

```
.SHOW MOUNTS
  Device          Associated jobs
---------- ------------------------------
DL0:         1  2  6  8  9 10 11
DL1:         1  6  9
DL2:SMADA    8
DL3:         2 11
DL3:WORK    11
DL3:WORK:TEMP 11
DL3:MANUAL   2
```

3.5.116 SHOW NUMDC

The SHOW NUMDC command displays the current value of the system parameter NUMDC. See the *TSX-Plus System Manager's Guide* for more information on the significance of the NUMDC parameter. For example:

```
.SHOW NUMDC
  0
```

3.5.117 SHOW PRIDEF

The SHOW PRIDEF command displays the default priority for an interactive job. See the *TSX-Plus Programmer's Reference Manual* for further information about job execution priorities. For example:

```
.SHOW PRIDEF
  50
```

3.5.118 SHOW PRIHI

The SHOW PRIHI command displays the maximum priority for an interactive job. See the *TSX-Plus Programmer's Reference Manual* for further information about job execution priorities. For example:

```
.SHOW PRIHI
  80
```

3.5.119 SHOW PRILO

The SHOW PRILO command displays the lowest priority for an interactive job. See the *TSX-Plus Programmer's Reference Manual* for further information about job execution priorities. For example:

```
.SHOW PRILO
  19
```

3.5.120 SHOW PRIORITY

The SHOW PRIORITY command displays the current and maximum priority values for a job. In addition, the range of priority values which are used for high and low fixed priority jobs is shown. See the *TSX-Plus Programmer's Reference Manual* for further information about job execution priorities. For example:

```
.SHOW PRIORITY
Current priority = 50; maximum authorized priority = 127
Low priority range = 0 to 19
High priority range = 80 to 127
```

3.5.121 SHOW PRIVILEGES

The SHOW PRIVILEGES command is used to display both the authorized and current privileges for the job. Current privileges will always be a subset of authorized privileges. See the *TSX-Plus System Manager's Guide* for more information on job privileges. For example:

```
.SHOW PRIVILEGES
Authorized privileges: ALLOCATE,DEBUG,DETACH,MESSAGE,NFSREAD,NFSWRITE,
                       SEND,SETNAME,SPFUN,RLOCK,GROUP,SAME,UP1,
                       UP2
Current privileges:    ALLOCATE,DEBUG,DETACH,MESSAGE,NFSREAD,NFSWRITE,
                       SEND,SETNAME,SPFUN,RLOCK,SAME
```

3.5.122 SHOW PRIVIR

The SHOW PRIVIR command displays the value an interactive job is reduced by when disconnected from by switching to a subprocess. See the *TSX-Plus Programmer's Reference Manual* for further information about job execution priorities. For example:

```
.SHOW PRIVIR
  10
```

3.5.123 SHOW QUANxx

The SHOW QUANxx command is used to display the current value of various system tuning parameters. The parameters which may be shown are: QUAN0, QUAN1, QUAN1A, QUAN1B, QUAN1C, QUAN2, and QUAN3. Related tuning parameters which may also be shown are: CACHE, CORTIM, HIPRCT, INTIOC, and NUMDC. See the *TSX-Plus System Manager's Guide* for more information on the significance of the various system tuning parameters. For example:

```
.SHOW QUAN3
  20
```

3.5.124 SHOW QUEUE

The SHOW QUEUE command displays information about print files in the spool queue. For example:

```
.SHOW QUEUE
 ID   Dev  Job   File    Form   Blocks
----  ---  ---  ------  ------  ------
  12  LP *   1  PLT50   STD      244
  13  LP     5  EXTERN  STD        5
  15  LP     5  PAYROL  FORM3     35
```

ID is the id-number assigned by the system to each entry. *Dev* is the name of the device for which the file is queued. An asterisk is displayed if the file is currently being printed. *File* is the name of the file, if a file name was specified with the .ENTER, otherwise the name of the program that created the file. *Form* is the name of the form on which the file is to be printed. *Blocks* is the number of blocks in the file remaining to be printed; this will decrease as the file is printed.

For more information on printer spooling, see Chapter 6.

3.5.125 SHOW REGIONS

The SHOW REGIONS command shows information about named regions local to the job and about all global named regions. For example:

```
.SHOW REGIONS
Region   Size    Type    Job  Use  Shared  AGE
------  ------  ------  ---  ---  ------  ---
KEYDEF     0.5  Local     2    0    No     No
WIN01A     3.8  Global    1    1    No     No
WIN02A     3.8  Global    2    1    No     No
WIN10A     3.8  Global   10    1    No     No
```

Region specifies the region name. *Size* shows the region size in kilobytes (and tenths). *Type* indicates whether the region is local or global. *Job* shows the number of the process which created the region. *Use* shows the number of jobs that are currently attached to the region. *Shared* is Yes if the region may be accessed by processes other than the one creating the region and No if the region is reserved for exclusive use (note that local named regions will always be No). *AGE* is Yes if the Automatic Global Elimination option was specified for the region.

See the *TSX-Plus Programmer's Reference Manual* for more information on named regions.

3.5.126 SHOW RUN-TIMES

The SHOW RUN-TIMES command causes the display of the names of the shared run-time systems that were loaded with the system. For example:

```
.SHOW RUN-TIMES
CBRTS
RTCON
```

3.5.127 SHOW SL

The SHOW SL command lists the current status of the single line editor. See Chapter 2 for a description of the single line editor. For example:

```
.SHOW SL
SL status: ON,KED,NOTTY,SUBSTITUTE
```

The meaning of status options are as follows:

ON Single line editor is in effect.

OFF Single line editor is not in effect.

KED Numeric keypad is in alternate keypad mode. KED is mode required for some SL functions and to use numeric keypad keys as defined keys.

NOKED Numeric keypad is in normal (RT11) mode. KED mode SL functions are not valid and defined key substitutions are not made for keys on the numeric keypad.

TTYIN The single line editor can edit input being accepted by programs via the .TTYIN request. (FORTRAN, COBOL and DBL programs accept most terminal input via the .TTYIN request.)

NOTTYIN The single line editor cannot be used on input to programs using the .TTYIN request.

SUBSTITUTE User defined key substitutions are enabled.

NOSUBSTITUTE User defined keys are disabled.

3.5.128 SHOW SPOOL

The SHOW SPOOL command lists the devices which have been declared as spooled during system generation. See Chapter 6 for more information on device spooling. For example:

```
.SHOW SPOOL
Spooled devices:  LP   CL2
```

3.5.129 SHOW SUBSETS

The SHOW SUBSETS command is used to obtain information about logical subset disks which have been mounted. Only the logical subset disks which have been mounted by the user are shown; those mounted by other users are not included. The name of the logical subset disk, the file with which it is associated, the file size, and an optional message are displayed. The optional message indicates when the logical disk is

not available for access either because the logical subset disk is missing or an outer level logical subset disk containing it has been dismounted. See the MOUNT and DISMOUNT commands and Appendix B for more information on logical subset disks. For example:

```
.SHOW SUBSETS
LD0 --> DL1:MANUAL.DSK[2601]
LD2 --> LD0:SUB.DSK[100] (not available)
```

3.5.130 SHOW SYSPASSWORD

The SHOW SYSPASSWORD displays the current system password. SYSPRV privilege is required to use this command. For example:

```
.SHOW SYSPASSWORD
TRANSGLUTAMINASE
```

3.5.131 SHOW TERMINALS

The SHOW TERMINALS command gives a display of the parameters of all primary time-sharing terminal lines defined to TSX–Plus during system generation along with some current characteristics. For example:

```
.SHOW TERM
Line  Type     Vector      CSR     Terminal  Speed      Active  Line Name
----  -----  ----------  ------  --------  --------   ------  ------------
 1    Oper.  DL     060  177560  VT100      9600 8N     Yes    Console
 2*   Local  DH - 0 330  160060  VT100      9600 8N     Yes    Boris
 3    Local  DH - 4 330  160060  VT100      Auto 8N     No     VAX link
 4    Local  DH - 5 330  160060  VT100      9600 8N     No     Natasia
 5    Local  DH - 6 330  160060  LA120      Auto 8N     No     AJ Terminal
 6    Local  DH - 7 330  160060  VT200      9600 8N     Yes    VT220
 7    Phone  DZ - 0 300  160040  VT100      Auto 8N     No     Dial up line
 8    Local  DZ - 6 300  160040  VT100      9600 8N     CLO    PRO Link
```

Line indicates the number assigned to each line (determined by the order of declaration during system generation). The line which issued the SHOW TERMINALS command is marked with an asterisk. The *type* of line may be:

Oper. The line was declared as the operator's console during TSX–Plus system generation.

Phone The line was declared to use modem control.

CL The line was declared as a dedicated CL line.

Local The line was declared as a normal time-share line.

The *Vector* column identifies the type of interface card to which the line is assigned. The possible types are:

DL a DL11 or DLV11 type interfaces, followed by the vector address for the line.

DZ a DZ11 or DZ(Q,V)11 type multiplexer, followed by the port number assigned to the line and the common interrupt vector for the multiplexer.

DH a DH11 type multiplexer, followed by the port number assigned to the line and the common interrupt vector for the multiplexer.

DHV a DH(Q,U,V)11 type multiplexer, followed by the port number assigned to the line and the common interrupt vector for the multiplexer.

The following special types are provided to identify the ports on a Professional series computer:

PC Professional Console

PP Professional Printer Port

CP Professional Communications Port

QP Quad serial line multiplexer Port

The Vector and CSR addresses are those declared during system generation.

The *Terminal* identification is that which has been declared to TSX–Plus either during system generation or via a later SET TT *type* command. Note that only the type currently specified for the primary line is shown here. If a SET TT *type* command is issued from a subprocess, that setting does not affect the primary line and is not shown by the SHOW TERMINALS command. The terminal type setting for a subprocess may be identified from the SYSMON utility.

The *Speed* column indicates the baud rate specified for lines for which the baud rate is known. The speed for DL lines is undefined unless the speed has been declared during system generation or a later SET TT *n* SPEED=*speed* command. The speed is followed by the number of data bits and parity control selected for the line.

Lines which are currently logged on are marked with a "Yes" in the *Active* column and the owner is identified in the Job Name column of the SYSTAT command, if known. If a time-sharing line has been *taken over* by a CL unit, then the CL unit currently attached to the line is shown. See also the SHOW CL command.

Line Name is the same as the description used for the NAME macro when defining the line in the system generation.

3.5.132 SHOW USE

The SHOW USE command causes display of the accumulated connect time and CPU usage for the current job since logon. The SHOW USE command is equivalent to the USE command. For example:

```
.SHOW USE
Connect=00:50:00  CPU=00:00:23
```

3.5.133 SHOW VERSION

The SHOW VERSION command causes the current TSX–Plus system version number to be displayed. For example:

```
.SHOW VERSION
TSX-Plus version=6.31
```

3.5.134 SHOW VM

The SHOW VM command causes the current base, top, and size of VM to be displayed. For example:

```
.SHOW VM
VM Base=044000 Top=050000 Size=256
```

3.5.135 The SPOOL Command

The SPOOL Command is used to control the operation of the spooling system. It may only be used if device spooling is enabled 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. A logical device name may be used in lieu of the physical device name. All functions except DELETE and STATUS require OPER privilege. The available functions are:

Function	Priv	Meaning
ALIGN	OPER	Print form alignment file
BACKUP	OPER	Skip backward and resume printing
DELETE	none	Delete current or pending file from queue
FORM	OPER	Mount form name
HOLD	OPER	Wait for file closure before printing
NOHOLD	OPER	Do not wait for file closure before printing
LOCK	OPER	Mount and lock form name
MULTIPLE	OPER	Print any file available for current form
SINGLE	OPER	Request form mount for every file
SKIP	OPER	Skip forward and resume printing
STATUS	none	Display status of spooled device

The SPOOL,DELETE command takes a slightly different form:

```
SPOOL device,DELETE [id-number]
```

which is used to delete entries from the list to be printed. Note that there is a space between DELETE and *id-number*. When no *id-number* is specified, then the entry currently being printed on *device* is deleted. When *id-number* is specified, then the entry associated with that *id-number* is deleted and *device* may be any spooled device. The id-number assigned to an entry may be determined from the SHOW QUEUE or SPOOL *device*,STAT commands.

See Chapter 6 for further information on the SPOOL command.

3.5.136 The SQUEEZE Command

The TSX-Plus SQUEEZE command has the same form and options as the RT-11 SQUEEZE command. However, you may not squeeze a device that contains any of the following TSX-Plus system files:

- TSX-Plus swap file (SY:TSXSWP.TSX)
- PLAS region swap file (SY:TSXRSF.TSX)
- spool file (SY:TSXSPL.TSX)
- TSKMON.SAV
- CCL.SAV
- IND.SAV
- IND temp file (SY:TSXIND.TSX)

If you need to squeeze a device with any of these files, do it while running under RT-11. The following error message is printed if an attempt is made to squeeze a device containing any of these files:

```
?KMON-F-Can't do this to device containing TSX system files
```

The command SQUEEZE/OUT:*ddn* SY: is also restricted, although you may run the DUP program with the appropriate command line. When running the DUP program (rather than the SQUEEZE command) all of the standard checking for system files, mounts, and opened channels is bypassed. Be certain that the device does not contain system files, that no users have the device mounted, that no command file is being read from the device, and that no executing programs have files which are opened on the device before issuing squeeze operations by running the DUP utility.

You must have NFSWRITE privilege to use the SQUEEZE command.

A device which has been allocated by another user cannot be squeezed. An error message will result, similar to:

```
?KMON-F-Device is allocated to job 13
```

A device cannot be squeezed if any other user has MOUNTed the device. Attempts to SQUEEZE a device which is mounted by another user will result in the error message:

```
?KMON-F-Device is mounted by another user
```

The SHOW MOUNTS command may be used to determine which other jobs have the device mounted. If the device is mounted only by the job which squeezes it, then the directory and data caches are cleared after the operation and caching is resumed.

If terminal logging is being done (see the SET LOG command) and a log file is open on the device being squeezed, the log file is automatically closed and the following warning message appears:

```
?KMON-W-Closing log file
```

Warning: It is still possible to write to a device without first mounting it. If one user has a file open on a device while another user squeezes the device, the possibility of severe data corruption exists. If the user with the open file writes to the disk during or after the squeeze operation, the data will be directed to locations on the disk according to the previous directory information. When the disk is squeezed, these locations are no longer appropriate and write operations will probably corrupt other files. **Never** squeeze a disk to which another user may be writing data. As a defensive measure, always MOUNT any device which you plan to use. You should also ALLOCATE physical devices before squeezing.

Warning: If the device from which a command file is being read is squeezed during execution of the command file, then the command file pointer will be invalid and unpredictable behavior will result. The behavior depends on the contents of the disk after the squeeze at the location of the command file prior to the squeeze. This is yet another good reason for mounting devices prior to using them. The SQUEEZE command checks that no other users have mounted or allocated the device and that no other jobs have I/O channels open to the device.

Warning: The system checks for device mounts and open files only at the level of the SQUEEZE command. Running the DUP utility directly bypasses system checking and presents a possibility of severe device corruption. Exercise extreme caution whenever squeezing a device.

3.5.137 The SUSPEND Command

The SUSPEND command is used to temporarily suspend the execution of another process. The suspended process may be allowed to continue at a later time with the RESUME command. The form of the command is:

```
SUSPEND job-number
```

where *job-number* is the number of the process to be suspended. Job numbers may be determined with the SYSTAT command. SAME, GROUP or WORLD privilege is required to use this command depending on the project-programmer numbers of the issuing and target jobs.

3.5.138 The SYSTAT Command

The SYSTAT (SYstem STATus) command displays information about the performance of the system and information about each logged on job. The SHOW JOBS and WHO commands are synonyms for the SYSTAT command. For example:

```
.SYSTAT
Uptime: 05:34:29
System use:   Run=10%, I/O-wait=8%, Swap-wait=0%, Idle=81%
I/O Activity: User I/O=10%, Swapping I/O=0%

Job   Line   Pri   State    Size   Connect  CPU time  Program   User name
---  ------  ---  -------  -----  --------  --------  -------  ------------
  1    1(0)   50   TI-Swap   33Kb  05:35:00  00:03:40  KMON     HAIKU
  2    2(0)   40   TI        36Kb  01:53:00  00:01:36  KED      Pacific
  7    7(0)   84   SL-Lock   16Kb  01:00:00  00:03:17  LAB3     SHORE
  8    8(0)   50   RN        34Kb  05:24:00  00:00:33  COBOL    Fast
 10    Det.   50   MS-Swap   62Kb  05:35:00  00:00:00  RTSORT
 11    Det.   50   SL        30Kb  01:54:00  00:00:00  WINPRT
 13*   2(1)   50   IN        38Kb  01:53:00  00:01:00  KMON     Pacific
```

Uptime indicates the amount of time since TSX-Plus was started or the RESET command was issued. *System use* shows a breakdown 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%. *I/O Activity* shows the percentage 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 of *System Use* and the I/O wait percentages of *I/O Activity* is a measure of the amount of overlap of job execution with I/O that took place. The RESET keyboard command may be used to reset these statistics so that statistics may be gathered over a desired interval of system execution.

The information displayed for running jobs consists of one line per job. *Job* is the job number, followed by an asterisk for the line to which you are currently attached. *Line* indicates the primary line and subprocess line number for each job. If the job is a primary line, then the line number will correspond to the job number and the number in parentheses will be zero—indicating relative subprocess number 0. If the job is a subprocess, the line number will indicate the primary line from which it was started, and the subprocess number relative to the primary line in parentheses. In the example above, job number 13 is the first subprocess of job number 2. Detached jobs will be marked by "Det.". *Pri* is the current priority for each job. *State* is a two-character code indicating the current state of the job. The state codes and their meanings are as follows:

State	Meaning
HI	High priority non-interactive run state
IN	Interactive run state
IO	Waiting for non-terminal I/O to finish
LO	Fixed-low-priority run state
MI	Waiting for mapped I/O buffer
MS	Waiting for a message
RN	Normal priority non-interactive run state
RT	Fixed-high-priority run state (real-time)
SF	Waiting for access to a shared file
SL	Doing a .TWAIT, .SPND or suspended
SP	Waiting for free spool block or file entry
TI	Waiting for input from the terminal
TO	Waiting for the terminal to print output
US	Waiting for access to file management module
WT	Waiting for miscellaneous system resource

The characters "-Swap" follow the state code if the job is currently swapped out of memory. "-Lock" is displayed if the job has locked itself in memory. *Size* is the number of kilobytes of memory space used by the job including ordinary job space, extended memory regions, and the job context block (currently 6 Kb). *Connect* is the length of time the job has been connected. *CPU time* is the time used so far by the job. *Program* is the name of the program being run by the job. *User name* indicates the user's name, except for detached jobs which are not associated with any particular user or line. The user name may be blank. The optional switch **/PPN** may be used with the SYSTAT command (or with the synonymous WHO and SHOW JOBS commands) to display the current project-programmer number for each line rather than the user name.

3.5.139 The TECO Command

The TECO command is used to initiate an editing session with the TECO editor. The TSX-Plus TECO command is equivalent to the RT-11 TECO command.

3.5.140 The TIME Command

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

3.5.141 The TYPE Command

The TSX-Plus TYPE command has the same form and options as the RT-11 TYPE command. Use of this command requires NFSREAD privilege.

3.5.142 The UCL Command

The UCL command is equivalent to the SHOW COMMANDS command.

3.5.143 The UNPROTECT Command

The UNPROTECT command is used to clear file protection status. The TSX-Plus UNPROTECT command has the same form and options as the RT-11 UNPROTECT command. Use of this command requires NFSREAD privilege.

3.5.144 The USE Command

The USE command displays the accumulated connect time and CPU usage since log-on. For example:

```
.USE
Connect=02:25:00  CPU=00:02:19
```

3.5.145 The WHO Command

The WHO command is used to display information about the system status and logged on jobs. The WHO command is equivalent to the SYSTAT command.

3.5.146 The $SHUTDOWN Command

The $SHUTDOWN command is similar to the $STOP command in that it is used to stop TSX-Plus and may 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 the $STOP command is executed. The form of this command is:

```
$SHUTDOWN
```

OPER privilege is required to use this command.

3.5.147 The $STOP Command

The $STOP command is used to halt the execution of TSX-Plus. On some hardware configurations, this may also reboot RT-11. The form of the command is:

```
$STOP
```

The $STOP command forces an immediate logoff of all users before stopping TSX-Plus. OPER privilege is required to use this command. If other users are logged on, the system will request verification before stopping. Any response beginning with "Y" or "y" will halt TSX-Plus; any other response will return to the monitor. This command is equivalent to the BOOT command; see its description for more information. For example:

```
.$STOP
Other users are logged on.
Are you sure you want to stop the system?Y

Connect=01:14:22  CPU=00:11:15
```

3.5.148 The YELL Command

The YELL command allows a message to be sent to a line that has TT GAG in effect.. The YELL command has the same form and function as the SEND command but overrides the TT GAG setting, whereas SEND does not. OPER and SEND privileges are required. For example, the following command forces a message to line 7:

```
YELL,7 The computer is on fire \\
```

3.6 RT-11 Commands not supported by TSX-Plus

The following RT-11 keyboard commands are not supported by TSX-Plus and are ignored if issued: CLOSE, GT, LOAD, UNLOAD.

The following RT-11 keyboard commands are not supported by TSX-Plus and will result in the "Unrecognizable command" error message: ABORT, B, D, E, FRUN, GET, REENTER, SAVE, SRUN, START.

The following keyboard commands are partially equivalent between RT-11 and TSX-Plus: INSTALL, REMOVE

The following keyboard commands have different effects under RT-11 and TSX-Plus: BOOT, RESET.

The following switches are not supported by TSX-Plus with the DIBOL, COMPILE/DIBOL or EXECUTE/DIBOL commands: /BUFFERING, /LOG, /PAGE, or /TABLES.

The following SET options are not supported by TSX-Plus: EL, EXIT, or USR. The following SET TT options are not supported by TSX-Plus: CONSOL, [NO]CRLF, [NO]FB, or WIDTH.

Chapter 4

Command Files

Certain series of keyboard commands may be executed together regularly, or may be so complex that it is desirable to edit and proofread them before execution. These needs are met by the facility known as indirect command files. These are files containing a set of keyboard commands to be executed together. Command files are invoked by providing the file name preceded by an "@" symbol in response to the keyboard monitor prompt. Command files may also be invoked without the "@" if their names do not conflict with system commands. See the description of keyboard command interpretation in Chapter 3 for more information on how command files may be initiated.

TSX–Plus allows the specification of parameter strings to be inserted in a command file. 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. TSX–Plus also allows 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 (but not KED) commands to be placed in a command file.

Command files may be executed either using normal TSX–Plus interpretation or under the control of the IND utility. Either type of control may be specified either implicitly or explicitly. The keyboard command SET KMON [NO]IND selects the desired implicit control of command files. If KMON is set IND, then command files will normally execute under the control of the IND program. If KMON is set NOIND, then command files will normally execute under TSX–Plus control. Regardless of the setting of KMON, a file may be executed under control of IND by typing:

```
IND file
```

Regardless of the setting of KMON, a file may be started as a normal command file (not under the control of IND) by typing:

```
$@ file
```

A program may determine whether it is running from an indirect command file by testing bits in the fixed offset location 366 (octal) in the simulated RMON. If the program is being run from a command file, then bits 8, 12, and 15 (mask 110400) will be set in RMON fixed offset 366. If the program is not running from a command file, then these three bits will be clear (0).

Example

- Execute the command file DK:DAILY.COM under control of the IND utility regardless of the setting of KMON:

```
IND DAILY
```

- Execute the command file DL1:WEEKLY.COM under normal TSX-Plus control regardless of the setting of KMON:

```
$@DL1:WEEKLY
```

4.1 Invoking command files

A command file is invoked by typing:

```
@filename [param1 [param2 ... [param6]]]
```

where *filename* is the name of the command file and *param1*, *param2*, etc. are optional 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,...param6]
```

That is, the "@" may be left out. With or without the "@" sign, if no device is specified device DK is first searched for the command file. Then, only without the "@" sign, if the named file is not found, then TSX-Plus attempts to locate the command file on device SY. See Chapter 3 for a description of the steps used by TSX-Plus to interpret keyboard commands. Listing control for command files differs whether the "@" sign is used or not. If the "@" is used, then listing is controlled by the current value of the SET TT [NO]QUIET option. If no "@" is used, then listing is disabled—as if a SET TT QUIET had been issued. In either case, the command file itself can control its own listing status with a SET TT [NO]QUIET command or with the control characters described later in this chapter.

4.2 Parameter strings

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. In 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 "\" (backslash), and the backslash 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 six parameter strings may be specified when the command file is started. The total number of characters in the combined parameter strings may not exceed sixty (60).

Parameter strings are inserted into the command file during execution at locations specified by an up-arrow (^). The parameter to be inserted is designated by a digit (1 to 6) immediately following the "^". During execution of the command file, the *up-arrow-digit* sequence is replaced by the parameter string passed to the command file in the position specified by the digit. If a parameter string is called for that was not specified when the command file was invoked, the up-arrow-digit will be ignored and no characters will be inserted in their place. For example, consider the following command file named TEST.COM:

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

When this command file is invoked with the following command:

```
@TEST FORTRAN PROG
```

the command file is executed as if it contained the commands:

```
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 file 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.

When executing a command file from a logical disk caution should be used when issuing the DISMOUNT command. If a command file is executed from a logical disk which is DISMOUNTed from within the command file then the return path from a nested command file (including pseudo command files) called after the DISMOUNT will be lost, and the remainder of the outer command file will not be executed. In the following example, note how the commands in the outer command file which come after the inner command file are not executed:

```
.TYPE LDO:OUTER.COM
DISPLAY I am now in the outer command file
DISMOUNT LDO
@SY:INNER
SHOW ASSIGN
DATE

.TYPE SY:INNER.COM
DISPLAY I am now in the nested command file

.@LDO:OUTER
.DISPLAY I am now in the outer command file
I am now in the outer command file
.DISMOUNT LDO
.@SY:INNER
.DISPLAY I am now in the nested command file
I am now in the nested command file
```

The command file listing status (SET TT [NO]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 its execution is completed and control returns to the next outer level command file, the listing control is restored to the state in effect when the inner command file was called. This is not done for the outer-most command file, however, so the SET TT [NO]QUIET state in effect on completion of the outer level command file remains in effect when control is returned to the keyboard monitor.

4.3 Comments in command files

Comments may be included in command files following an exclamation point (!). Anything on a line in a command file following an exclamation point will be treated as comment. For example, in a command file containing the lines:

```
DISPLAY Watch out! Files may be destroyed.
PAUSE Hit RETURN to proceed   !Requires operator response
```

the resulting output would be:

```
Watch out
PAUSE Hit RETURN to proceed
>>
```

When an exclamation mark (!) is used in a command file line to be input as data to a program, then the exclamation mark and all text following it on the input line are not passed to the program.

4.4 Command file control characters

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 CTRL–C. The escape character may be represented by up-arrow, dollar sign (^$). The up-arrow character itself may be represented in the command file by preceding it with another up-arrow (^^).

Several character combinations act as special commands to influence the execution of the command file. These commands take effect at the point of their occurrence in the command file. The characters in these special sequences are not passed to programs. They are as follows:

Command Sequence	Command Function
^(	Stop listing command file. This has the same effect as SET TT QUIET, except that 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.
^!	Suppress all terminal output. Both the command file listing and any program terminal output are suppressed. The ^(and ^) command sequences restart program generated output.
^>	Accept all terminal input from the command file. Command file data is normally only passed to programs which request lines of input (using the EMT calls: .GTLIN, .CSISPC and .CSIGEN). RT-11 handles command file data in this manner. Programs called by a command file which request terminal input with the EMT calls .TTYIN or .READ normally still expect to get this from the terminal and cannot use command file data as input to these requests. The ^> command sequence causes *all* subsequent requests for terminal input 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. See also the description in the *TSX-Plus Programmer's Reference Manual* of the "N" and "O" program controlled terminal options that affect command file input. Note that field width limit is ignored when accepting input from a command file.
^<	Return to standard data mode. Subsequent command file data will only be passed to programs which do .GTLIN, .CSISPC or .CSIGEN EMTs (standard RT-11 mode).
^Z	CTRL-Z is translated to CTRL-C in command files.

4.5 PAUSE Command

The PAUSE command is provided to suspend the execution of a command file 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 executed, the PAUSE command is displayed at the terminal followed by ">>". Execution of the command file is then suspended until carriage return is typed. Typing a CTRL-C in response to the prompt aborts the command file. This can be overriden by installing the command file. See the *TSX-Plus System Manager's Guide* for information concerning the INSTALL attributes.

4.6 DISPLAY Command

The DISPLAY command causes a line of text to be displayed at the terminal. 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 to mark progress through command files running in *quiet* mode (not being listed).

Precautions should be taken when command files are executed on devices which may be SQUEEZEd. See the SQUEEZE command in the 3.

Chapter 5

Detached Jobs

The TSX–Plus Detached job facility is similar to subprocesses created by the process window facility (see Chapter 2): they both allow a time-sharing user to execute several jobs simultaneously. The major difference between detached jobs and subprocesses is that you may switch back and forth among subprocesses, but once a detached job is started it is dissociated from any terminal. 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. However, terminal logging may be enabled for detached jobs to accept their terminal output; see the SET LOG command.

The differences between subprocesses and detached jobs are summarized below.

- Terminal communication may be switched among several subprocesses. A detached job must receive all its terminal input from a command file and any terminal output it generates is discarded (unless terminal logging directs the output to a log file). If a detached job requests terminal input and is unable to obtain it from a command file, the job is terminated.
- Subprocesses are *owned* by a physical line. When the physical line logs off, the subprocesses also log off. Detached jobs are not associated with any physical line. Once started, any or all time-sharing users may log off without affecting detached jobs.
- Detached jobs may be started automatically when TSX–Plus is initiated. Subprocesses must be started by time-sharing users after TSX–Plus is running.
- 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. Subprocesses wait for more input.

There are three ways to start detached jobs:

- during system generation specify a command file to be started as a detached job whenever TSX–Plus is started
- use the keyboard DETACH command
- use the TSX–Plus EMT to start a detached job from a running program.

Detached jobs that are initiated automatically when the system is started run with ALL privilege. Detached jobs initiated by use of the DETACH command or the EMT inherit the characteristics of the (sub)process starting them.

5.1 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. The process which starts a detached job is known as the *parent* of the detached job.

5.1.1 Starting a detached job

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

```
DETACH file-spec [parameters]
```

where *file-spec* is the name of a command file to be executed as a detached job. The default device is DK: and the default extension is .COM. Use of the DETACH command requires DETACH privilege. *Parameters* is a string of parameters to be passed to the detached job command file just as if you were passing parameters to an ordinary command file. The ^*n* mechanism is used within the detached job command file to accept the parameters.

A detached job inherits characteristics (such as job name, device assignments, privileges, etc.) from its parent in the same way as subprocesses do, except that detached jobs inherit their characteristics from the (sub)process starting them, rather than always from the primary process as subprocesses do. See Chapter 2 for a list of the job characteristics which are inherited by subprocesses and detached jobs when they are started.

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. The SYSTAT command also indicates detached job numbers and flags detached jobs with "Det.". If *file-spec* is preceded by the symbol "@", then the first line of the command file will be used by the detached job, but any remaining commands in the file will be associated with the job currently connected to the terminal.

Examples

1. Start a command file PURGE.COM as a detached job.

   ```
   DETACH PURGE
   ```

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

   ```
   DETACH RK1:STATS.NEW
   ```

3. Start the command file PURGIT as a detached job and pass the string OLDFIL as a parameter.

   ```
   DETACH PURGIT OLDFIL
   ```

The system manager determines the number of slots available for detached jobs and who may use detached jobs. The job numbers assigned to detached jobs start after the slots reserved for primary time-sharing lines. Subprocesses are assigned line numbers above any slots reserved for detached jobs.

5.1.2 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. TSX–Plus displays the status (active or free) of the detached job.

Examples

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

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

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

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

5.1.3 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. Use of the DETACH/KILL command requires DETACH privilege. GROUP privilege is required to kill a detached job with a different programmer number than your own. WORLD privilege is required if the detached job has a different project number or if the detached job was started automatically by being specified as a system detached job during system generation. For example, to kill the detached job on line number 4:

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

Chapter 6

Device Spooling

6.1 The concept of device spooling

Device spooling is a technique that provides more efficient use of slow peripheral devices. Data sent to spooled devices is automatically redirected to a high speed disk file for temporary storage and passed on when the slow device is ready to accept it. This allows a program generating data for the spooled device to continue processing without being slowed down by the peripheral. TSX-Plus optionally provides automatic spooling to output devices such as printers, card punches and plotters. More than one device may be spooled.

When a program directs output to a spooled device, the output is diverted by TSX-Plus to a disk spool file. An entry is made in a spool file table indicating that 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 their output to be printed. Devices 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 to work properly with the TSX-Plus spooler. This can be done with the SET command. For example:

```
SET LP HANG
```

Since the SET command actually stores this information permanently in the disk copy of the device handler, it need only be issued once.

6.2 Directing output to spooled devices

Output is sent to a spooled device in exactly the same way as it would be directed to the 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
```

or

```
PRINT TEST.LST
```

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

Any number of users may simultaneously write to a spooled device without conflict. TSX-Plus separates the output from each user and prints it in an orderly fashion. A user may direct output through several I/O channels to the same or different spooled devices.

6.3 Operation of the spooler

The spooling system consists of a memory resident spool file control table which contains information concerning the user's spool file and a single, system-managed spool data disk file which contains the user's intercepted output. Every time a user opens an output channel to a spooled device, a new entry is created in the spool file control table. Output records directed through separate channels to the same device have separate entries in the spool file control table. Placement in this table is governed by the processing of the user's open request for the spooled device.

The total number of spooled files that may be in existence is specified when TSX-Plus is generated. A spooled 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 processed by the spooled device. If a program opens a channel to a spooled device and the spool file control table already contains the maximum number of spooled files, the program is suspended until a slot becomes available in the spool file control table.

All output directed to spooled devices is stored in a common disk data file. The total file space that is available for spooled data is specified when the TSX-Plus system is generated. Space within this disk file is dynamically allocated as needed on a block by block basis. Each block contains 508 bytes of user's output data. Therefore, write requests smaller than 508 bytes will be combined into one spool data block. A write request larger than 508 bytes will be split into more than one spool data blocks. If the spool data file is totally filled, programs writing to the spooled device will be suspended until space becomes available.

Actual output to the spooled devices is processed on a sequential first-in/first-out basis. Since output is actually coming from the disk file, each I/O request to the device handler is the size of the spool data block (508 bytes). While spooled files are being printed, TSX-Plus maintains a list of the blocks most recently printed. The length of this list is specified by the number of backup blocks declared during system generation. When the list becomes full, the oldest block is deallocated and the new block is added. When the entire spooled file has been successfully written, the entire list is deallocated.

Read requests (.READ) and special function requests (.SPFUN) from a spooled device bypass the spooling system and connect directly with the device handler.

6.4 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 only used with some functions.

A second form of the SPOOL command is used to delete current or pending files in the spooler queue. The form of this command is:

```
SPOOL device,DELETE [id-number]
```

where the optional *id-number* is assigned by the system and can be determined from the SHOW QUEUE or SPOOL *device*,STAT commands. *Device* may refer to either the physical name of a spooled device (such as LP, LS, or CL2) or a logical name assigned to a spooled device. For example:

```
ASSIGN CL2 LP
```

The two following commands are then equivalent:

```
SPOOL CL2,STAT
SPOOL LP,STAT
```

A device specified without a unit number is equivalent to unit 0 (e.g., CL=CL0).

6.4.1 The FORM and LOCK Functions

The FORM and LOCK functions are used to specify the name of the currently mounted form. Use of the FORM and LOCK functions requires OPER privilege. 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 disables form mount request messages.

Examples

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

Note the difference between the FORM keyboard command 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 for subsequent spool files. The SPOOL FORM/LOCK commands are used by the TSX-Plus operator to tell the spooling system which form is currently mounted.

6.4.2 The ALIGN Function

The ALIGN function is used to print a form alignment file on a spooled device. Use of the ALIGN function requires OPER privilege. The name of the alignment file is specified in the *parameter* field of the command. The default file extension for form alignment files is .ALN.

Examples

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

6.4.3 The DELETE Function

The DELETE function is used to remove a file from the list of files to be printed by the spooler. Use of the DELETE function does not require any special privilege. This form of the SPOOL command is:

```
SPOOL device,DELETE [id-number]
```

If *id-number* is omitted, then the file currently being printed on the specified device is aborted. If the file is open, the file is marked to be deleted but is not actually deleted until it is closed and the spooler begins to process it. If the file is currently being printed when the delete command is issued, the spooler deletes blocks as they become available and the spooled device is tied up until the file deletion is completed.

When *id-number* is specified, that number is used to identify the file to be deleted. In this case, *device* may be the name of any spooled device and does not have to match the device on which the file is actually being printed.

A spooled file id-number may be determined from either the SHOW QUEUE command or the SPOOL *device*,STAT command. For example:

```
SHOW QUEUE
 ID   Dev  Job   File    Form   Blocks
----  ---  ---  ------  ------  ------
  37  LP *   1  PAYROL  STD      135
  39  LP     1  GL      STD       25
  41  LP     1  AR      STD       86
```

Examples

To delete the file PAYROL currently being printed on LP:

```
SPOOL LP,DELETE
```

To delete the file AR which has not yet started printing:

```
SPOOL LP,DELETE 41
```

6.4.4 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. Use of the SKIP function requires OPER privilege. Each block in the spool data 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
```

6.4.5 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. Use of the BACK function requires OPER privilege. The number of blocks involved is specified when TSX-Plus is generated. Each block in the spool data file contains 508. characters. This function is particularly useful for recovering from paper tears or remounts. The spooler will finish printing the current block before backing up.

Examples

```
SPOOL  LP,BACK
SPOOL  LS,BACK
```

6.4.6 The STAT Function

The STAT function is used to determine the status of a spooled device. Use of the STAT function does not require any special privilege. Information returned includes: the condition of the spooler (active, idle, or 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.

Example

```
SPOOL  LP,STAT
Spooler idle
Currently mounted form = STD
No files in queue
```

6.4.7 The SING and MULT Functions

The SING and MULT functions control how the spooler will handle multiple files queued for the same form. Use of the SING and MULT functions requires OPER privilege. 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
```

6.4.8 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.

Use of the HOLD and NOHOLD functions requires OPER privilege. The default mode is established when the TSX-Plus system is generated. A [NO]HOLD command remains in effect until another [NO]HOLD command is issued or the system is restarted.

Examples

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

6.5 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 a special form with the printing of a file that requires the form.

If the first character 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 on which the file should be 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 the spooler in any position other than the first character 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 (see Chapter 3). If no FORM command has been issued by the user, TSX–Plus uses the form name STD for the file.

Each time TSX–Plus selects a file to be printed on a spooled device, it first looks for a waiting file that requires the form currently mounted on the spooled device. If several such files are available, the oldest one is 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. The message appears on the operator's terminal as:

```
Mount form-name form on ddn
```

where *form-name* represents the form name, *dd* is the spooled device name, and *n* indicates the device unit number. The terminal to which the message is directed is the one declared as the operator's console when the TSX–Plus system was generated.

Once the form mount request message is sent, the spooler for the device requiring the new form is suspended. In order to restart the spooler the operator must issue 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 called for in the form mount request, but may mount any form desired, in which case TSX–Plus will search for a spool file that needs the form actually mounted.

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. After processing all files that need the currently mounted form, TSX–Plus checks for 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 spool files to be created that need the currently mounted form.

6.6 Form alignment procedure

When mounting a new form it is usually necessary 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 positioned, 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 a form-mount message and a SPOOL-FORM or SPOOL-LOCK command.

Chapter 7

TSX–Plus Restrictions

7.1 Programs Not Supported by TSX–Plus

Most programs which run under RT–11 will run under TSX–Plus without change.

The TSX–Plus program debugging facility (see the *TSX–Plus Programmer's Reference Manual*) should be used rather than ODT.

The FORMAT program may not be used under TSX–Plus.

The BATCH RT–11 facility is not supported by TSX–Plus.

The logical subset disk feature is provided internally to TSX–Plus and therefore does not use the LD pseudo-device handler.

The single line editor feature is provided as an optional system overlay region with TSX–Plus and does not use the SL pseudo-device handler.

The VM handler supplied with TSX–Plus was written specially for use with TSX–Plus and is not the same as the DEC VM handler.

The QUEUE package (QUEUE and QUEMAN) is not supported by TSX–Plus.

The RESORC utility is not supported.

7.2 Special program suggestions

FILEX must be locked below 248 Kb in memory in order to work correctly on RX02 (DY) diskettes. A simple way to do this (although not guaranteed to force it below 248 Kb) is to install FILEX with the MEMLOCK attribute:

```
INSTALL ADD SY:FILEX.SAV/MEMLOCK
```

Typical symptoms are input and output errors or empty directories on interchange diskettes known to contain files. The errors will appear random because they only happen if I/O mapping occurs when the job is positioned above the 248 Kb boundary.

The reason is that FILEX uses special function read and writes to absolute tracks and sectors in which the word-count indicates the absolute track number and not the actual transfer size. Thus, for MAPIO operations, TSX-Plus has to provide the correct transfer size internally. It assumes a sector size of 128. words for DY and of 64. words for DX. However, interchange format diskettes are single density, meaning that the actual sector size is 64. words in either type drive. Therefore, MAPIO will return too much information in the case of absolute track and sector reads on a single density disk in RX02 drives. This

apparently corrupts code or data inside FILEX causing the failures. It will corrupt the diskette on absolute track and sector writes.

The communication program VTCOM supplied with RT-11 is used for connecting a terminal on one computer system as a virtual terminal on another computer system and for file transfers between two systems. The XM version of VTCOM (VTCOM.SAV, not VTCOM.REL) must be used with TSX-Plus, and this requires that TSX-Plus be generated to include PLAS (XM) support. This is controlled by the TSGEN parameter SEGBLK. If no other use is planned for PLAS regions, then a SEGBLK size of 100. is more than adequate to support VTCOM. Under RT-11, VTCOM uses the XL device handler for the low-level interface protocol. Under TSX-Plus, the CL facility replaces the XL handler and provides more flexibility in use of communication lines. CL units may either be designated as dedicated CL lines and used only as I/O devices, or may be used to *take over* inactive time-sharing lines. Either type of CL unit may be used with VTCOM. However, whichever CL unit is attached to the remote computer (possibly through a modem) must be assigned the device name XL (XC on the Professional 300 Series) so that VTCOM can access it. The designated CL unit should also be set NOLFOUT. To avoid interference between two persons attempting to access the same CL unit at the same time (and thereby causing great confusion) the designated CL unit should also be allocated for exclusive use. The following example command file indicates how a time-sharing line may be *taken over* and used with the VTCOM program:

```
SET CLO LINE=6      !Take over time-sharing line
ASSIGN CLO XL       !Assign for VTCOM
ALLOCATE XL         !Prevent multi-user collisions
SET XL NOLFOUT      !Inhibit line feed transmission
R VTCOM             !Part of RT-11 5.01 distribution
...                 !Communicate with remote system
DEALLOCATE XL       !Let others use CLO
DEASSIGN XL         !Clean up assignment
SET CLO LINE=0      !Re-enable time-sharing line
```

Appendix A

FILTIM Program

In addition to the date of creation of a file, TSX–Plus also stores file creation times in device directories. At the time a file is closed, the current time of day is automatically stored in the sixth word of the directory entry for that file. Under RT–11, this word is unused for permanent files and contains the job and channel number for tentative file entries. In order to represent the time as a positive 16-bit value, the time is converted to an integer representing the number of three-second intervals since midnight. For example, if a file were closed at 11:13:22, then the sixth word of the permanent directory entry for that file would contain 13467 (32233 octal).

11 hr	×	60 min/hr	×	60 sec/min	=	39600 sec				
		13 min	×	60 sec/min	=	780 sec				
				22 sec	=	22 sec				
						40402 sec	÷	3	=	13467 3-sec units

An EMT to obtain file directory information, including file creation times is provided by TSX–Plus. An EMT is also provided to set the file creation time value into file directory entries. See the *TSX–Plus Programmer's Reference Manual* for information on use of these EMTs.

The DIR utility provided with RT–11 does not interpret file creation time information as set by TSX–Plus, so a utility program (FILTIM) is provided with TSX–Plus to obtain this information. The FILTIM program should be copied from the distribution medium to the system device SY:. It may then be run either explicitly (R FILTIM) or implicitly as a system program (FILTIM). Although FILTIM does not accept wildcard file specifications, it will accept up to 6 file specifications. The default device is DK: and the default extension is MAC. A device specification also applies to subsequent file specifications which do not explicitly include a device. Files created under RT–11 or before this feature was implemented in TSX–Plus will have a creation time of 00:00:00.

Example

```
.FILTIM CKACT,TPRMAN.TXT,MAN:CH13.DPS,APPC.DPS,RKO:FILTIM
DK:CKACT.MAC        3P  31-May-83  00:00:00    184
DK:TPRMAN.TXT      835   17-Jul-83  22:15:33  14226
MAN:CH13.DPS        35   18-Jul-83  08:31:42   3570
MAN:APPC.DPS         5   18-Jul-83  15:54:48   3605
RKO:FILTIM.MAC      23   18-Jul-83  11:24:57   4240
```

Note that the default device DK is used for CKACT and is carried over to the next file specification, and that the default file type is .MAC. The logical device MAN is used for the next two file specifications until the device specification, RK0. And again, the default extension for RK0:FILTIM is .MAC.

WARNING: Due to the method used by PIP for copy operations, file creation times are not preserved during copy operations, although the date is handled correctly. When copying a file with PIP, the destination file will acquire the time the copy is made as its creation time. The EMT to set file times, described in the *TSX-Plus Programmer's Reference Manual*, may be used to set the correct time in the new file.

Appendix B

Logical Subset Disks

Logical subset disks provide a method of logically partitioning a large disk into smaller units which can themselves be treated as directory structured devices. This is done by creating a (relatively large) file on a physical device and then creating a device directory and files within that file. The resulting pseudo device which is created within the file on the mother device is called a logical subset disk. Logical subset disks may also be nested. That is, one logical subset disk may be contained within another logical subset disk. This nesting may continue up to seven levels of logical subset disks within other logical subset disks. Logical subset disks may be initialized, squeezed, and have files created, opened, closed and deleted. They may also be assigned logical device names (e.g., OUT:, DK:, ABC:). Several keyboard commands are used to manipulate logical subset disks: DISMOUNT, MOUNT, SET LD and SHOW SUBSETS.

Support for logical subset disks is built into the kernel of TSX-Plus. Therefore, it is not necessary to use the LD device handler and logical subset disks may be used under TSX-Plus with either version 4 or version 5 RT-11 utilities.

Logical subset disks are given the device names LD0 through LD7. Each user may use all eight logical subset disks at any time. That is, if one user has mounted LD0 then any other user may also use LD0 simultaneously and they need not (and usually will not) refer to the same file containing the logical subset disk. If logical subset disks are nested, then the device numbers must increase with the level of nesting. For example, LD0 may contain a file to be mounted as LD1. However, a file contained within LD1 may never be mounted as LD0.

The typical sequence of operations when using logical subset disks is to:

1. create a file on a physical disk device
2. mount that file as a logical subset disk
3. initialize the new logical subset disk
4. proceed to use it as any other disk device (except that it cannot be physically handled independently of the surrounding real disk).

Subsequent uses of the logical disk only require that the disk be remounted. The file need not be recreated, nor the logical subset disk reinitialized.

The following example shows a typical sequence of commands which might be used to initiate use of a new logical subset disk.

```
CREATE DL1:MYDISK.DSK/ALLOCATE:500.
MOUNT LD0: DL1:MYDISK
INITIALIZE/NOQUERY LD0:
...
DISMOUNT LD0:
```

In order to use this new logical disk at a later time, it would only be necessary to issue the MOUNT command:

```
MOUNT LDO: DL1:MYDISK DK:
```

The default (and recommended) file type for files intended to contain logical subset disks is .DSK.

When logical subset disks are mounted, information about them can be obtained with the SHOW SUBSETS command. For example:

```
SHOW SUBSETS
LDO --> DL1:MYDISK.DSK[500]
```

To remove a logical subset disk from use, use the dismount command. For example:

```
ASSIGN DL1 DK
DISMOUNT LDO:
```

Note that the disk files containing logical subset disks are automatically marked as protected files. In order to delete them, they must first be unprotected.

It is possible with logical subset disks to create some unusual situations and conflicts. For example, it is possible to mount a logical subset disk and then unprotect and delete the file which contained it. This condition would display "(not available)" with the SHOW SUBSETS command. In addition, the SET LD CLEAN command can be used to force the system to compare, verify and correct the information in its internal logical subset disk tables if possible. Obviously, the system will not go back and recreate deleted files. An implicit SET LD CLEAN is done each time the DUP utility is called (e.g., with the INITIALIZE and SQUEEZE commands).

For more information on the details of the commands used with logical subset disks, see the descriptions in Chapter 3 of the following commands: DISMOUNT, MOUNT, SET LD, and SHOW SUBSETS.

Appendix C

System Startup Errors

The following error messages can be displayed during the startup of TSX-Plus. All are fatal error messages and once reported, abort running TSX-Plus. All messages are in the format:

```
?TSX-F-error message displayed here
```

For more information concerning these errors and system generation parameters to alter to correct these problems, consult the *TSX-Plus Installation Guide*

Cannot find device handler file: dd
: The device handler file, *SY:dd.TSX* (where *dd* represents a two character device driver name), could not be found. A device handler file must exist for each device listed in TSGEN using the DEVDEF declaration. Note: this error message only occurs if the INIABT sysgen parameter is set to 1. If INIABT is zero, the system ignores device declarations for which there is no TSX-Plus handler. Check the devices specified by the DEVDEF declarations in TSGEN, make any necessary corrections, and generate a new system. Verify that the handler file is present on the system disk and if it is not, place a copy on the system device. Standard device drivers supported by TSX-Plus are supplied on the distribution media. If this is a standard device driver, the file may be copied from the distribution media to the system device. If the device handler is not provided on the distribution media, a device handler file must be built to run with TSX-Plus. See the *TSX-Plus System Manager's Guide* for information concerning the building of device handlers for TSX-Plus.

Cannot find "SY:CCL.SAV" file
: The file SY:CCL.SAV cannot be found. This file was provided on the distribution media and should be copied to the system device.

Cannot locate "SY:SYSODT.REL" file
: The file SY:SYSODT.REL cannot be located when attempting to run with the system debugger. This file was provided on the distribution media and should be copied to the system device when attempting to install TSX-Plus with the system debugger.

Cannot find "SY:TSKMON.SAV" file
: or

Cannot locate "SY:TSX.SAV"
: The file SY:TSKMON.SAV or SY:TSX.SAV cannot be found. These files are created during the system generation process. If TSX-Plus was not generated on the system device, copy the missing file from the device used to build TSX-Plus to the system device. If the file does not exist on the generation device, the system generation was not successful. See the *TSX-Plus Installation Guide* for information concerning system generations. These files are provided on the distribution media for PRO/TSX-Plus, which does not require system generation; copy these files to the system disk.

Cannot open PLAS region swap file
Number of contiguous blocks needed = nnnnnn
The PLAS (Program Logical Address Space) swap file defined in TSGEN cannot be created because the specified device does not have enough contiguous unused blocks. The number of contiguous blocks required is represented by the number *nnnnnn*. Remove any unnecessary files from the disk assigned to contain the PLAS swap file and consolidate the unused blocks until enough contiguous free space is available. See the SQUEEZE command in the *RT-11 User's Guide*.

Cannot open program swap file
Number of contiguous blocks needed = nnnnnn
The program swap file defined in TSGEN cannot be created because the specified device does not have enough contiguous unused blocks. The number of contiguous blocks required is represented by the number *nnnnnn*. Remove any unnecessary files from the disk assigned to contain the swap file and consolidate the unused blocks until enough contiguous free space is available. See the SQUEEZE command in the *RT-11 User's Guide*.

Cannot open shared run-time file: dev:file.ext
The shared run-time file named *dev:file.ext* specified in TSGEN could not be opened. Check the run-time file names defined in TSGEN, make any necessary corrections, and generate a new system. Verify that the file is present on the device specified and if it is not, place a copy on that device.

Cannot open spooled device: dd
The spooled device named *dd* cannot be opened by TSX-Plus. Check the device names specified in the SPOOL declaration in TSGEN, make any necessary corrections, and generate a new system. Verify that the device handler file *SY:dd.SYS* is present on the system device and if it is not, place a copy on that device and install the device handler in RT-11 See the INSTALL command in the *RT-11 User's Guide*.

Cannot open SY:TSXIND.TSX file. # blocks needed = nnnnnn
The IND storage file named SY:INDTMP.TSX cannot be created because the system device does not have enough contiguous unused blocks. The number of contiguous blocks required is specified by the number *nnnnnn*. Remove any unnecessary files from the system disk and consolidate the unused blocks until enough contiguous free space is available. See the SQUEEZE command in the *RT-11 User's Guide*.

Cannot open TSXUCL data file. # blocks needed = nnnnnn
The UCL (User Command Language) data file named SY:TSXUCL.TSX cannot be created because the system disk does not have enough contiguous unused blocks. The number of contiguous blocks required is specified by the number *nnnnnn*. Remove any unnecessary files from the system disk and consolidate the unused blocks until enough contiguous free space is available. See the SQUEEZE command in the *RT-11 User's Guide*.

Computer line time clock (50 or 60 Hz) is not working
Each PDP-11 computer has a *line time clock* which interrupts the system at a frequence based on the local power system (50 or 60 Hz). The line time clock must be running for the proper operation of TSX-Plus. If you receive this error message, confirm that the line time clock is not running by using the TIME command while running under RT-11, then contact your DEC field service engineer or take other corrective action to get the clock working.

Error on read of SYSODT rel file
An input error occurred when reading the system debugger file named SY:SYSODT.REL into memory. Check the system disk and the hardware involved.

Error reading device handler file: dd
An input error occurred when reading the device handler file *SY:dd.TSX* into memory. Check the system disk and the hardware involved.

Error reading "SY:TSX.SAV"

An input error occurred when reading the memory resident overlays from the file SY:TSX.SAV into memory. Check the system disk and the hardware involved.

Generated TSX system is too large

[Reduce size fo TSGEN by n bytes]

The TSX–Plus system is generated too large to load and run. Although this may occur when there is not enough total physical memory, it usually implies that the unmapped portion of TSX–Plus exceeds 40 Kb. When the second line is displayed concerning the reduction amount of TSGEN, this means that TSX–Plus was too large to run because TSGEN exceeds 16 Kb by *n* bytes Remove any unnecessary features, decrease excessive parameters, and generate a smaller system. See the *TSX–Plus Installation Guide* for information on the effect of optional features on the size of TSX–Plus.

Handler for SY device was not loaded

The handler for the system device was not defined using a DEVDEF declaration in TSGEN. Specify the system device handler with a DEVDEF declaration in TSGEN and generate a new system.

Handler not generated with extended memory support: dd

The device handler file named *SY:dd.TSX* (where *dd* represents the two character device driver name) was not generated with the required memory management support. Verify that the device handler was written to include support for memory management and review the *TSX–Plus System Manager's Guide* for building device drivers for TSX–Plus.

Insufficient disk space for spool file

The spool file defined in TSGEN cannot be created because the specified disk does not have enough contiguous unused blocks. The number of required contiguous blocks was defined using the SPOOL declaration in TSGEN. Either decrease the number of blocks required; or remove any unnecessary files from the disk assigned to contain the spool file and consolidate the unused blocks until enough contiguous free space is available. See the SQUEEZE command in the *RT–11 User's Guide*.

Insufficient memory space for data cache

Not enough memory was available to load TSX–Plus and allocate the specified number of data cache blocks defined by the CACHE parameter in TSGEN. Decrease the number of data cache blocks specified or install more memory.

Insufficient memory to load all mapped system regions

Not enough memory was available to load memory resident mapped system regions in TSX–Plus. Review the *TSX–Plus Software Product Description* to determine if you have sufficient memory. Review the system generation parameters in TSGEN, remove any unnecessary features, decrease excessive parameters and generate a smaller system or install more memory.

Insufficient memory to load all shared run-time systems

There is not enough memory to load all the shared run-time systems specified in TSGEN. Review the system generation parameters in TSGEN, remove any unnecessary features, decrease excessive parameters (remove some or all of the shared run-time systems) and generate a smaller system or install more memory.

Insufficient total physical memory for generated system

The TSX–Plus system generated with the specified features was too large for the physical memory available. Review the system generation parameters in TSGEN, remove any unnecessary features and decrease excessive parameters, and generate a smaller system or install more memory.

Invalid interrupt vector address for T/S line:

Line # = nn

The line numbered *nn* has an invalid vector address. Time-sharing lines may only be vectored in the range 60 to 477, and may not use a vector assigned to any other device. Lines are numbered sequentially in increasing order by LINDEF specification in TSGEN. Correct the vector address for the indicated line and generate a new system.

Invalid RT–11 version for device handler dd

A device handler named *dd* (where *dd* represents a two character device driver name) required a later version of RT–11 than the one installed on the system device. During initialization, TSX–Plus analyzes each device defined by the DEVDEF specification in TSGEN and determines if the present RT–11 version will support the inclusion of that handler. Update RT–11 to the correct version which supports the device specified.

Invalid status register address for T/S line:

Line # = nn

The line numbered *nn* has an invalid status register. When the specified CSR is addressed by the CPU, the location does not respond. Lines are numbered sequentially in increasing order by LINDEF specification in TSGEN. Correct the status register for the indicated line and generate a new system.

Mapped handler is larger than 8 Kb:dd

A device handler named *dd* (where*dd* represents a two character device driver name) was specified in the DEVDEF declarations in TSGEN to be mapped in high memory (with the MAPH attribute). The handler may not be loaded as a high memory handler because it is larger than 8K bytes.

System is not equipped with extended memory management hardware

TSX–Plus could not find extended memory management hardware. Set the TSGEN parameter EXTMCH to zero (0) and generate a new system or install extended memory management hardware.

System is not equipped with memory management hardware

TSX–Plus will not run on the current hardware configuration because it requires memory management support. Check the *TSX–Plus Software Product Description* to determine if any other required hardware is necessary and install the necessary hardware.

TSX generation did not include device dd

One of the files defined in TSGEN included a device handler named *dd* (where *dd* represents the two character device driver name) that was not specified in the handler declaration section DEVDEF in TSGEN. Check all file specification and DEVDEF declarations in TSGEN, make any necessary corrections, and generate a new system.

TSX is already running

TSX–Plus is currently running and therefore cannot be started again.

Appendix D

Fatal System Errors

The error messages detailed below are reported when fatal system errors occur and once displayed on the system console the operating system will halt. All of the system error messages have identical formats which display the following information:

```
<BELL>?TSX-F-Fatal system error at nnnnnn
EEE-Error message displayed here (see below)
Arg. value = nnnnnn
(Additional information)
SP at time of crash = nnnnnn
```

The additional information provided will indicate the identity of the system region mapped when the error occurred. This additional information will take one of three formats.

The following additional information is displayed when the mapped system region points to a mapped device handler:

```
Device name: xxx
```

The following additional information is displayed when the mapped system region points to a memory resident code overlay region:

```
Seg. value = nnnnnn
Overlay: xxx
```

The following additional information is displayed when the mapped system region does not point to either a mapped device handler or a memory resident code overlay region:

```
PAR5 value = nnnnnn
```

When the crash dump facility has been included in teh system generation, important information concerning the status of the system will be printed at the designated device and then the system will halt. See the *TSX-Plus Installation Guide*for more information concerning the crash dump facility.

DTL-Demonstration system time limit reached
: The time limit has expired for the demonstration system. This time limit is generally thirty minutes. TSX-Plus may be started again.

FRK-No free FORK blocks
: A system routine issued a FORK request when the FORK queue was full. One or more devices is repeatedly interrupting faster than can be processed by the system. Try increasing the value of the NUMFRK parameter in TSGEN.

JM0-Jump occurred to location 0
The jump instruction occurred to location 0. If the TSX–Plus crash dump facility is enabled, information concerning the current state of the system will be output to the chosen crash dump device.

KRE-KMON read error
An input error occurred when attempting to read the file SY:TSKMON.SAV into memory. This indicates a hard error was detected and reported by the system device handler.

KTP-Kernel mode trap
A trap through vector 4, 10, or 250 occurred in kernel mode while at interrupt level. The argument value indicates the address at which the trap occurred. If the trap address was 120000 or higher, the additional information specifies the overlay code section which was mapped when the error occurred. Enable the system crash dump feature by setting the SYSDMP parameter to one (1) and assigning an appropriate value to the DMPTCR parameter in TSGEN.

LMF-Job lock mem failure
A system failure occurred when no memory was available in which to lock a job that had previously requested memory.

MIO-Need to increase value of MIONWB sysgen parameter
The system attempted to perform an I/O operation to a device that requires I/O mapping and there were no free system I/O mapping buffers or wait queue elements. You must increase either the MIONBF parameter which will allocate more I/O mapping buffers or the MIONWB parameter that increases the number of wait queue elements.

MPR-Memory parity error
A trap occurred through vector 114 indicating a hardware memory parity error was detected.

NQE-Ran out of free I/O queue elements
An attempt was made to queue a system I/O request and no I/O queue elements were available. Try increasing the NUMIOQ and NUMSYQ parameters in TSGEN.

NSP-No free swap command packets
No free swap command packets were available to queue job swap request. Try increasing the NSCP parameter in TSGEN.

PFT-Power-fail trap
A trap occurred through vector 24 indicating a hardware power failure.

RIT-Trap in real-time interrupt service routine
A trap in a real-time interrupt service routine causes a system halt because interrupt service routines run at fork level. A trap in a real-time interrupt completion routine does not cause a system halt.

SFO-Job swap file overflow
Too few job swap file slots were declared in TSGEN and there was no room in which to place a job which required swapping. Regenerate the system and increase the value of the SWPSLT sysgen parameter, or set SWPSLT to zero (0) in which case the system will allocate a swap file slot for each job.

SIE-Swap file I/O error
An input or output error occurred either reading or writing into the program swap file. This indicates a hardware malfunction on the program swapping disk.

SJN-Job # 0 at STOP
A job number of zero was detected during a request to stop the current job and execute SY:TSKMON.SAV. User job numbers must be greater than zero.

SOF-Stack overflow
One of the three system stacks has overflowed. The argument value indicates which stack has overflowed.

SSE-PLAS region swap file I/O error
A hardware I/O error was detected while reading or writing a PLAS region to the swap file. The device used for the PLAS swap file is specified in TSGEN with the RSFBLK parameter.

UEI-Interrupt occurred at unexpected location
An interrupt occurred through a vector that was not attached to a terminal or CL line, a device handler, or a real-time completion routine. The parameter UXIFLG in TSGEN controls the handling of unexpected interrupts. When UXIFLG is set to zero, unexpected interrupts are ignored and never reported. When UXIFLG is set to one, the above error message is reported along with an argument value which identifies the vector location of the unexpected interrupt.

Appendix E

User Error Messages

Several different categories of errors can generate messages, ranging from errors which are fatal to the operating system to something as simple as an incorrect file specification. Errors are identified by the name of the program which recognized the error and one or more lines of descriptive information. Errors identified by utility programs are identified by the name of the utility (e.g., ?PIP-F, ?DUP-F, ?KED-F) and may or may not abort the program. For more information about utility program error messages, consult the appropriate RT–11 manual (*RT–11 System Message Manual*, *RT–11 System Utilities Manual*, *RT–11 Keypad Editor User's Guide*, etc.). Errors which are fatal to the TSX–Plus operating system (identified as ?TSX-F) report error conditions for which time-sharing users are not usually responsible and usually cannot correct. TSX fatal errors should be reported to the system manager who should consult the *TSX–Plus System Manager's Guide* for more information. Errors which are not fatal to the operating system but which indicate a user mode error are reported as monitor errors. This Appendix describes TSX–Plus monitor error messages.

All user mode fatal monitor error messages have the syntax:

```
?KMON-F-message
```

or

```
?MON-F-message
```

where the ?KMON-F form is used if the error occurs while using the keyboard monitor and the ?MON-F form is used for errors which occur within a user program.

Warning messages have the form:

```
?KMON-W-message
```

In some cases, the message is only informational rather than indicating an error. In these cases, the message is of the form:

```
?KMON-I-message
```

Error messages are consistent with RT–11 usage insofar as possible. Monitor error messages unique to TSX–Plus are described below. Consult the *RT–11 System Message Manual* for monitor errors not described here.

A line number must be specified with the SEND command

This error message is printed if line number is not specified with the SEND command. See the description of the SEND command for more information about options related to the line number.

ALLOCATE privilege is required for this command
ALLOCATE privilege is required to be able to allocate or deallocate devices.

Ambiguous command
Too few characters of a command keyword have been entered to allow the command to be uniquely identified. For example, this error would result if CO were entered as a command since CO could be an abbreviation for either COPY or COMPILE.

Ambiguous option
Not enough characters were specified to distinguish between options.

ASSIGN table full
Maximum number of logical assignments (25 per user) exceeded. SHOW ASSIGNS and DEASSIGN unnecessary logical assignments.

Attempt to increase MAXPRIORITY above current value
Maximum job priority may be restricted by the system manager. It may be reduced, but it may not be increased during a time-sharing session.

Cannot change privileges for another job
The SET PROCESS command cannot be used to change the privileges for any job other than your own.

Cannot change name of another job
The SET PROCESS command cannot be used to change the name of any job other than your own.

Cannot find spool file with specified ID
There is no spool file in the queue with an ID number that matches the ID specified with the SPOOL command.

Cannot find SY:CCL.SAV to process CCL command
The CCL.SAV program has been deleted from the system disk. This program is required to process some keyboard commands. Reboot the system and copy CCL.SAV from your TSX–Plus distribution disk to the system disk while you are running under RT–11.

Cannot find SY:TSODT.REL file
The relocatable copy of the TSX–Plus ODT debugging program was not found on the system device. Copy the file TSODT.REL from the distribution medium to SY.

Cannot find SY:TSXUCL.SAV
A command was attempted to be interpreted as a user-defined command, but the TSXUCL program could not be found. Copy TSXUCL.SAV from the distribution to SY.

Cannot open alignment file
The alignment file specified in a SPOOL align command cannot be opened.

Cannot open logoff command file
A logoff command file which was specified during job start-up could not be found while logging the job off.

Cannot open spool device
The spooler cannot access the spool device. Verify the device name used in the SPOOL macro during TSX–Plus generation and check for correct device assignments.

Cannot open system accounting file
The file SY:ACCESS.TSX cannot be found during logon or logoff. This file is created by the system manager.

Can't do this to device containing TSX system files
An operation (e.g., INITIALIZE or SQUEEZE)has been attempted to a device containing one of the following system files: TSXSWP.TSX, TSXRSF.TSX, TSXSPL.TSX, TSKMON.SAV, CCL.SAV, IND.SAV, or TSXIND.TSX.

Character length must be 7 or 8 bits
The value specified for the number of bits in a character with the BITS=*value* qualifier on a SET TERMINAL or SET CL command must be 7 or 8.

Closing log file
If a device on which a terminal output log file is open is initialized or squeezed, the log file is first closed.

Command file nesting too deep
Maximum depth of command file nesting exceeded. The depth of command file nesting allowed depends on the number and length of parameter strings. Three levels are possible even with long strings and as many as seven levels are possible with no parameters. See Chapter 4 for more information on command files.

Command file not found
The specified command file was not found. See Chapter 3 for an explanation of command interpretation including default devices, and Chapter 4 for more information on command files.

Command file parameter string too long
Total number of characters in parameter string exceeds the maximum allowed length of 60 characters. See Chapter 4.

Command string too complicated
The command string expansion is too large for the internal CCL buffer. Reduce the complexity of the command string. The SET CCL TEST command may be used to examine the expanded command string generated by high level commands.

Command syntax error
The command entered was recognized as a system command but the form of the command (i.e., its syntax) does not match the expected form. Check the description of the command.

Date unknown
The current date is not known. The date should be set by the operator by use of the DATE keyboard command.

DEBUG privilege is required to use debugging facilities
DEBUG privilege must be granted in order to be able to use any of the system debugging facilities.

DETACH privilege is required to use detached jobs
DETACH privilege must be granted in order to be able to use the detached job facility.

Device full
There is insufficient free space available to open a file.

Device is allocated to another user
An attempt has been made to access a device that is currently allocated to another user by use of the ALLOCATE command.

Device is allocated to job nn
The device you are trying to access is allocated to job *nn*. The device must be deallocated before you may access it.

Device is being used by job nn
Job *nn* has a channel open to a device that you are trying to ALLOCATE, INITIALIZE, SQUEEZE, or SET. These commands may only be used when no one else is accessing the device.

Device is mounted by another job
The INITIALIZE and SQUEEZE commands are invalid if the device is MOUNTed by any other user. This minimizes the opportunity for data corruption when other users may be writing to a device. See the INITIALIZE and SQUEEZE commands for motivation.

Device is still mounted by other users
This informational message is displayed when a DISMOUNT request is issued for a device which is also MOUNTed by other users. See the INITIALIZE and SQUEEZE commands for motivation.

Device must be allocated before use
The system manager has specified that the device you are attempting to access must be allocated to a job by use of the ALLOCATE command before it can be accessed by the job. Use the ALLOCATE command to allocate the device to your job.

Device name is required
A device name must be specified with the INITIALIZE, SQUEEZE, and FORMAT commands.

Device or file is access restricted
User does not have access privilege to requested device or file. The system manager may restrict user access to individual devices and files.

Directory I/O error
A device directory failed internal consistency checks. Devices must have valid RT–11 format directories for use with TSX–Plus. This error may also occur when attempting to use 22-bit addressing with DMA devices when the handler or hardware does not support 22-bit addressing.

Directory overflow
All empty file entries in the directory for a device have been used up. The maximum number of file entries may be specified when you initialize a device. See the description of the INITIALIZE command for information about specifying the number of file entries in the directory.

Error creating window to access key region
An error occurred while creating a virtual memory (PLAS) region window to access user-defined key information. Check your sysgen and make sure PLAS support is included (SEGBLK parameter must be non-zero).

Error on write to spool file
The prefix to this message is ?TSX-W-. An output error occurred when writing data into the spool file. This is caused by an error in the hardware performance of a write to the device on which the spool file resides. Check the spool file for bad blocks. Should the problem continue to occur, move the position of the spool file (if possible to another device) and run hardware diagnostics. Following the receipt of this message, printed output may not be accurate or complete.

File not found
The specified file was not found on the specified device.

Floating point trap
A floating point operation trap occurred and job had not done a .SFPA operation to specify a floating point trap processing routine.

Handler active—Can't update running copy
An attempt was made to perform a SET command on a device handler which was in use. If the handler is idle when the SET is issued, the change will be made, otherwise, the running copy of the handler is not altered. Reissue the command when the handler is not active.

Handler not installed. Set applied to handler disk file only.
A SET command has been issued to a device handler that exists on the disk (SY:xx.TSX file) but which is not installed in the running TSX–Plus system. The SET has been applied to the disk file copy of the handler.

ID qualifier required with SUSPEND or RESUME
You must use the /ID qualifier on the SET PROCESS command to specify which job is to be suspended or resumed.

Illegal use of wildcards
Wildcards may not be specified for (part of) this command.

IND already active
The IND program cannot be run from within a command file which is being executed under control of the IND program.

IND is not available
The IND.SAV program was not on the system disk during TSX-Plus start-up. IND is provided with RT-11 version 5 and later.

Install table full
An attempt has been made to install a program but there is no free space in the install table. Resysgen TSX-Plus and increase the value of the NUIP parameter, or delete some program from the install table.

Installed program cannot be on a logical disk
When the INSTALL command is used to add a program to the installed-program table, the physical device name where the program file is located is entered into the table. It is not legal for an installed program to be located on a logical (subset) disk.

Invalid address as EMT argument
An address specified in a monitor call was odd or was not within the job's address space. The value returned as the abort location is the address of the instruction or the EMT that caused the error.

Invalid boot device
An invalid device was specified with the BOOT command.

Invalid channel
An invalid channel number was used with an EMT.

Invalid command. Expected numeric value missing
A numeric parameter value is missing from a command that requires it.

Invalid command. Expected octal value missing
A numeric parameter (expressed in octal) is missing from a command that requires it.

Invalid date
An invalid value was specified as a date.

Invalid device
An invalid or unrecognizable device name was specified.

Invalid EMT
An invalid or unsupported function code was specified in an EMT argument block.

Invalid EMT argument
An invalid value was specified in an EMT argument block. Frequently this is caused by specifying an odd address where an even address is required—such as for the address of a completion routine.

Invalid file name
An invalid file name was specified. Check for typing errors, names too long, and correct file specification format.

Invalid key name
A DEFINE/KEY command has been issued to specify a new key definition but the name of the key (e.g., KP0, PF1, etc.) is not recognized.

Invalid line number
An invalid line number was specified for a KILL or DETACH command.

Invalid logical disk name
Logical subset disks must be mounted as LD0 through LD7.

Invalid multiple value on option
More than one parameter was specified for an option which only accepts one.

Invalid option
An invalid or unrecognized option keyword has been specified with a command.

Invalid or uninitialized directory
A disk device must contain a valid RT-11 format directory to be accessed by any operation other than INITIALIZE. Logical subset disks must be initialized before first use. This error may also occur when attempting to use 22-bit addressing with a DMA device and either the device handler or hardware does not support 22-bit addressing.

Invalid or unrecognizable device
A device name was specified which is not recognized by the system. This could happen if a logical device name was used without assigning the logical name to a physical device.

Invalid printer type
The type of the printer specified with the TYPE qualifier for the SET PRINTWINDOW command is not recognized. See the description of the SET PRINTWINDOW command for a list of valid device type names.

Invalid SAV file
A file specified with an R or RUN command failed internal consistency checks. The file may be damaged. An incorrect file type may have been specified—the default type is .SAV.

Invalid speed value
An invalid speed has been specified with a command used to set a line's transmit/receive speed. The valid speed values are: 50, 75, 110, 134.5, 150, 300, 600, 1200, 1800, 2000, 2400, 3600, 4800, 7200, 9600, and 19200.

Invalid start address for program
The start address for the program was outside of the program's memory bounds or on an odd address. A MACRO program may have an incorrect starting address specified with the .END directive. The file may be damaged. An incorrect file type may have been specified—the default is SAV.

Invalid string specification
A character string parameter which was specified with some command qualifier was not properly enclosed in quotation marks. Either quotation marks (") or apostrophies (') may be used to enclose character string parameters but the same character must be used to start and end the value.

Invalid time
An invalid time value was specified with the TIME command.

Invalid to specify /NOECHO with /NOTERMINATE
You may not specify both the /NOECHO and /NOTERMINATE qualifiers together.

Invalid unit number
An out of range unit number was specified with a SET CL*n* command. The valid unit numbers range from zero (0) up to the highest CL unit specified when the system was generated.

Kernel mode trap within TSX-Plus
A kernel mode trap error usually indicates the operating system image has been corrupted due to hardware or software failure. Record the full error message and report the error to the system manager.

This error can also be generated by attempting to access a non-existent memory address with the real-time EMTs to access the I/O page.

Line is active
The specified line or job is still active.

Line is already being used as a CL unit
An attempt has been made to assign a CL unit to a line that already has another CL unit assigned to it. You may disassociate a CL unit from a line by assigning the CL unit to another line or to line zero (0) which disassociates it from any line.

Line is gagged
Messages cannot be sent to lines which have SET TT GAG unless those lines are waiting in TSKMON for a command.

Line is in use by a time-sharing user
An attempt has been made to assign a CL unit to a line on which a time-sharing user is logged in. The time-sharing user must log off before you can assign the CL unit to the line.

Line not logged on
The specified line number does not currently have a logged on user or job.

Line too long
The expanded CCL command was too long. Commands are expanded into the chain data area, locations 500–777. See the SET CCL TEST command.

Log file overflow
An error was reported while writing to the terminal output log file. This is most commonly an attempt to write past end-of-file. Make more room on the output device, specify a file size with the SET LOG command, or use the SET LOG NOWRITE option to reduce the volume of log file output.

Logical Disk support was not generated into system
Attempts to MOUNT a logical subset disk are not valid unless support for logical subset disks was selected when generating TSX–Plus.

Logical disks must be nested in order of increasing unit #
When a file residing within a logical subset disk is to be mounted as another logical subset disk, it must be assigned a higher number than the outer level logical subset disk. For example, the following command is invalid: MOUNT LD1 LD3:MYDISK; whereas the following is acceptable: MOUNT LD3 LD1:MYDISK.

Max allowed number of devices already mounted
The number of device directories which may be cached is defined during system generation. DISMOUNT devices on which caching is no longer needed or consult the system manager.

Max windows already in use
The maximum number of process windows that can be supported by the system are already in use. Regenerate the system and increase the value of the MAXWIN sysgen parameter, or delete some existing windows.

Maximum authorized priority is nn
You have requested that your execution priority be set to a value greater than you are authorized for.

Maximum allowed number of devices already allocated
The table that is used to keep track of allocated devices is full. Resysgen TSX–Plus and increase the value of the MAXALC parameter.

Maximum legal value is nnnn
A numeric parameter has been specified which exceeds the maximum allowed value.

Maximum number of keys already defined
The maximum number of keys that each user can defined is controlled by the KEYMAX sysgen parameter. Resysgen TSX-Plus and increase the value of this parameter. The maximum allowed value for KEYMAX is 60.

MEMMAP privilege is required to run this program
MEMMAP privilege is required to run the program because it accesses a restricted area of memory (usually the I/O page). Either the user who wishes to run the program must be given MEMMAP privilege or the program must be installed with MEMMAP privilege.

Memory size change not allowed in non-swapping TSX system
Jobs may not change size dynamically unless job-swapping was enabled during system generation. Consult the system manager.

Missing equal sign
The equal sign was missing on a SET device command.

NFSWRITE privilege is required for this command
NFSWRITE privilege must be granted to you in order to be able to use this command.

No defined operator's console
No operator console was defined during system generation. Form mount requests and messages using the OPERATOR command are sent to the operator console.

No free detached job lines
The maximum number of detached job lines is specified during system generation. Either wait for a job to terminate, stop one by use of the DETACH/KILL command, or ask the system manager to increase the number of detached lines.

No user-defined keys are allowed
The system was generated with the KEYMAX parameter set to zero (0). This disallows any user key definitions. If you wish to define keys, resysgen TSX-Plus and set KEYMAX to a non-zero value.

Not enough memory to run program
A file was too large (or the size specified in the SAV image was too large) to fit in the space allocated to the job. See the *TSX-Plus Programmer's Reference Manual* for more information on program size specifications. Use overlay segments or chain requests to reduce the program size. Use the MEMORY command to request more memory for your job. Note that you cannot use the MEMORY command in a non-swapping environment; see the system manager.

OPER privilege is required to perform this operation
The user must be grated OPER privilege in order to use the command that failed.

Overlay error
An I/O error has occurred while reading a program overlay segment.

Password may not exceed 20 characters
The system password specified with the SET SYSPASSWORD command must not exceed 20 characters in length.

Performance monitor is in use by user number nn
The performance monitor facility is already in use. Only one job may use it at a time.

Port device must be a CL or C1 unit
The device specified with the SET HOST/PORT=*device* command must be a CL or C1 unit such as CL2 or C13.

Printer type is ambiguous
The printer type name specified is abbreviated too shortly so that it is ambiguous.

Priority value must be in the range ...
Job priority may not be set less than 0, nor higher than the maximum allowed priority.

Program debugger was not included when system was generated
A request was made to run a program with the TSX–Plus program debugging facility (i.e., RUN/DEBUG program) but the debugging facility was not included when the system was generated.

Program not installed
The program specified with the INSTALL command is not currently in the install table. Use the SHOW INSTALL command to display the names of the programs that are currently installed.

Prompt string too long
Prompt strings may not be longer than 8 characters.

PSWAPM privilege is required to lock programs in memory
PSWAPM privilege is required to lock a program in memory while it is running.

Save file I/O error
An I/O error occurred while reading the SAV file. Check that the disk is still on line. It is also possible that the SAV file or the directory is damaged.

SEND privilege is required for this command
SEND privilege must be granted to you in order to be able to send messages to other jobs.

SETNAM privilege is required for this operation
The user must be granted SETNAM privilege to use the command that failed.

SL was not included at system generation
The SET SL ON command may not be issued unless support for the single line editor was selected when generating TSX–Plus.

SL width is fixed at 80
Under TSX–Plus, the single line editor command buffer length is fixed at 80 characters and may not be changed with the SET SL *width=nn* command.

Spool file is full
The spool file is currently full. Additional space will become free as currently queued files are printed. If you wish to increase the spool file size, regenerate TSX–Plus and increase the value specified with the SPOOL macro.

Spooler in single file mode
The spooler has been set so that it will pause between each file that is printed. See the description of the SPOOL command for information about setting this mode.

Spooler waiting for form mount
A message to mount a new form has been printed at the operators console and the spooler is waiting for a form to be mounted before it will proceed with printing. See the description of the SPOOL command for information about specifying form names.

Stack overflow
The stack allocated within the program (usually below location 1000) has overflowed. Take the appropriate action based on the nature of the program to increase its stack space.

String too long
A character string parameter specified with the command exceeds the maximum legal length. See the description of the command to determine the maximum legal length.

SYSGBL privilege is required to use windows
In order to use process windows you must have SYSGBL privilege in order to create a named global memory region in which the window image is stored.

SYSPRV privilege is required to perform this operation
The user must be granted SYSPRV privilege in order to use the command that failed.

System was not generated to support performance monitoring
Performance monitoring is an optional feature that must be included during system generation. See the system manager.

Table overflow
Too many devices/files have been specified in the ACCESS command. See the system manager.

TERMINAL privilege is required to perform this operation
The user must be granted TERMINAL privilege in order to be allowed to use the command that failed.

Terminal type must be set to VT100, VT200 or VT52
The SET SL ON command may only be issued if the current terminal type is VT100, VT200 or VT52. Other terminal types are not supported by the single line editor.

This CL unit is currently busy
An attempt has been made to assign a CL unit to a line and the CL unit is currently busy processing an I/O operation.

This CL unit is not currently assigned to a line
A SET CL command has been issued to set some option for a CL unit and the CL unit is not assigned to a communications line.

This command may interfere with other users
The prefix to this message is ?CCL-W-. This warning is issued by CCL on INITIALIZE and SQUEEZE commands to warn you that other users using that device might be dismayed by what you are about to do. See the warning with the SQUEEZE command.

This command only legal in startup command file
The ACCESS and SET LOGOFF commands are only valid in startup command files.

This operation not legal with SY (system) device
The device which was booted when starting TSX–Plus may not be initialized or squeezed while running TSX–Plus. If it is necessary to initialize or squeeze this device, do so while running RT–11.

Too many completion routines
You have attempted to connect too many completion routines to interrupt vectors. The number of interrupt vectors that can be connected to completion routines is determined during system generation. See the system manager.

Too many files
The command interpreter uses standard command string format: up to 6 input files and 3 output files. Reduce the number of file specifications accordingly.

Too many parameters to command file
TSX–Plus command files accept a maximum of six parameters. See Chapter 4.

Trap to 4, Abort location = nnnnnn
An execution trap has occurred to location 4. This trap is caused by either attempting to address an odd location by an instruction that requires an even address, or attempting to access an invalid memory location (i.e., outside the valid range of addresses for your job).

Trap to 10, Abort location = nnnnnn
A reserved or unimplemented instruction has been executed. This can be caused by executing a floating point instruction on a machine that does not have a floating point unit.

Trap to 14, Abort location = nnnnnn
Invalid breakpoint trap executed. The abort location provided is the address of the instruction following the trap. In order to use breakpoint, the program must store the PC and PS for the breakpoint service routine in locations 14 and 16.

Trap to 20, Abort location = nnnnnn
Invalid IOT instruction executed. The abort location provided is the address of the instruction following the trap. In order to use an IOT instruction, a program must store the PC and PS of the IOT service routine in locations 20 and 22.

Trap to 24, Abort location = nnnn
A hardware power failure trap has occurred. This indicates that either the system power is being interrupted or a hardware error exists that is causing the system to think a power interruption has occurred.

Trap to 34, Abort location = nnnnnn
Invalid TRAP instruction executed. The abort location provided is the address of the instruction following the trap. In order to use a TRAP instruction, a program must store the PC and PS of the TRAP service routine in locations 34 and 36.

TSGEN was modified without relinking TSKMON
Whenever a new TSX–Plus system generation is done, *both* TSX and TSKMON must be relinked. See the system manager.

Unable to allocate memory for virtual overlays
The system failed to successfully allocate extended memory regions when trying to run a program with virtual overlays. Unlock jobs from memory or increase the PLAS region swap file size.

Unable to attach to region
The name of a memory region specified with the REMOVE command does not correspond to any existing global or local memory region for the job.

Unable to create memory region for window
An error has occurred while attempting to create a global memory region to hold window information. Make sure memory region (PLAS) support is generated into the system (SEGBLK parameter must be non-zero). Use a SHOW REGIONS command to see how many regions have already been created. This error can also occur if there is insufficient contiguous physical memory space available to create the region.

Unable to create memory region for key definitions
An error has occurred while attempting to create a local named memory region to hold user key definitions. Try increasing the value of the SEGBLK sysgen parameter.

Unable to eliminate region
An error has occurred while attempting to delete a memory region. Use the SHOW REGIONS command to make sure the region is not being used by another job.

Unable to open file
The file specified with the command that failed could not be opened. Either it does not exist, or it is on a different device than specified, or you do not have the proper authorization to access the file.

Unable to open log file
The system reported an error when attempting to create a file for terminal output logging. Check the file specification, verify that there is room and that the output device is not write protected.

Unimplemented option
A command option has been specified which is not currently implemented by TSX-Plus.

Unrecognizable command
TSX-Plus could not find a system command, a user-defined command, or a command file with that name. See Chapter 3 for more information on command interpretation.

User command interface program not available
The system could not locate the file specified with the command SET KMON UCI[=*filnam*]. Check the file specification. The default file is SY:UKMON.SAV.

USR called from completion routine
You cannot call system service routines that use TSUSR from a completion routine. This includes: .LOOKUP, .ENTER, .RENAME, .DELETE, .CLOSE, .DSTATUS, .SFDAT, and .FPROT.

USR err # *n*
All of the following USR errors result from consistency checks performed in closing a tentative file (creating a permanent file). These are usually indicative of a strange hardware condition, such as exchanging or squeezing a disk while a file is open.

USR err # 1
TSUSR can't find tentative file entry for specified file on close.

USR err # 2
File length in channel block is not equal to length in file entry.

USR err # 3
Highest block number written is greater than file length.

USR err # 4
Empty file entry doesn't follow tentative file entry.

USR err # 5
Tentative file entry status was lost during close operation. This is usually an indication of hardware failure.

Value required for option
A command was issued that required a value for an option, but no value was specified. Reissue the command correctly.

Window support not available
The system was generated with the MAXWIN sysgen parameter set to zero (0). Regenerate the system and assign a non-zero value to MAXWIN if you wish to use windows.

You are not authorized for that privilege
You have attempted to grant yourself a privilege that you were not authorized to use. SETPRV privilege is required to allocate privileges beyond those that are authorized.

You are not authorized to access that job
GROUP privilege is required to affect a job that has the same project number but a different programmer number than your own. WORLD privilege is required to affect a job that has a different project number. WORLD privilege is also required to affect system detached jobs.

You are not authorized to write to that device or file
The system manager may restrict access to individual devices or files. See the system manager.

Index