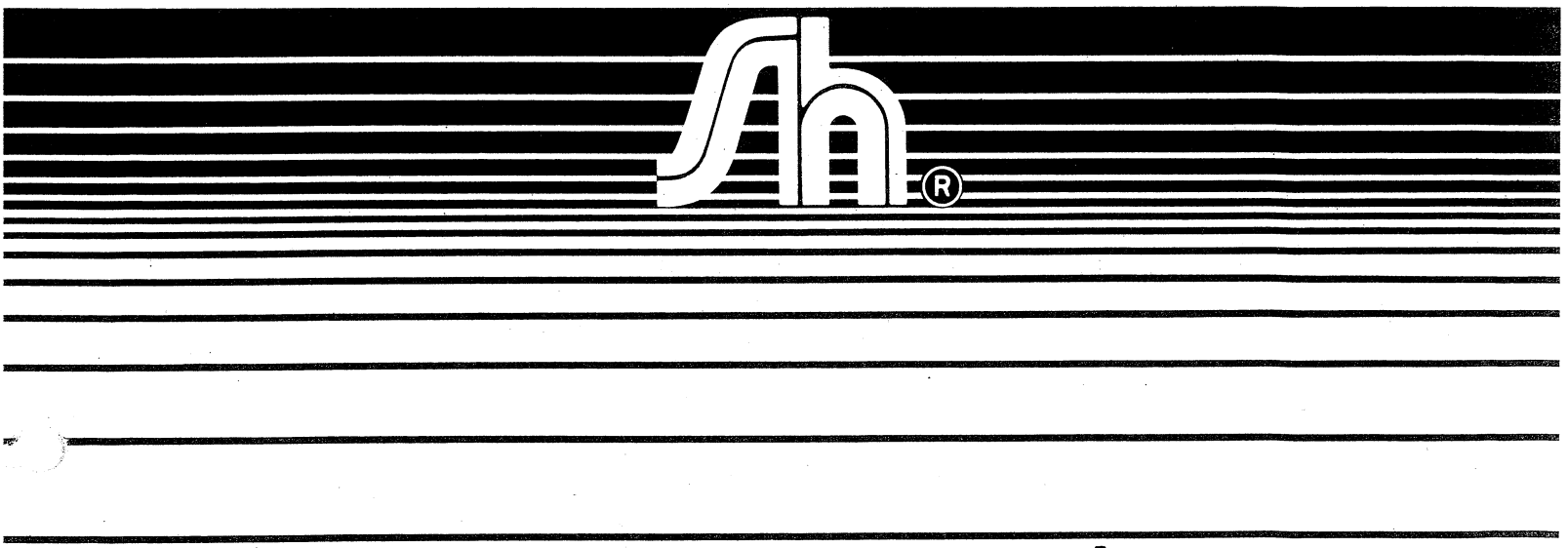
TSX-Plus
System Manager's
Guide



s&h computer systems, inc.

TSX-Plus
System Manager's
Guide

s&h computer systems, inc.

Fifth Edition
First Printing -- November, 1985

S&H Computer Systems, Inc.
1027 17th Avenue South
Nashville, Tennessee USA
37212-2299
615-327-3670

CONTENTS

Chapter 5

Chapter 6

Chapter 7

Chapter 8

Chapter 9

Appendix A

Appendix B

INTRODUCTION

The purpose of the TSX-Plus System Manager's Guide is to provide information necessary to manage the system resources for the TSX-Plus operating system. It is intended to provide more detailed information on the internal operation of TSX-Plus for people who are already familiar with the features provided. See the TSX-Plus Reference Manual for information on the features provided by TSX-Plus.

Chapter 1 - Privileges and Privilege Commands
TSX-Plus provides twenty seven privileges to be specified on a line-by-line or user-by-user basis. Privileges may be specified during program installation which temporarily authorized program execution without granting privilege to the user.

Chapter 2 - Account Authorization Program
An account authorization program may be used by the system manager to grant access to the system by authorizing accounts. The facility allows the system manager to group accounts by project/programmer number, assign privileges, and set the maximum job execution priority. Execution of a start-up command file can also be specified which may contain system and file access security restrictions.

Chapter 3 - System and File Access Security
The system manager can impose certain restrictions on system use and file access by selecting available security options. Terminal lines or authorized users can be locked to a program, limited in access to devices or files, or restricted to a maximum priority for program execution.

Chapter 4 - Device Handlers
The information in this chapter provides the system manager (and system programmers who wish to write special device handlers) an understanding of the extensions and restrictions imposed on device handlers in the TSX-Plus environment. Building and debugging device handlers is discussed.

Chapter 5 - Programming for Specific Device Handlers
Special device handlers are provided (or restricted) for use in the TSX-Plus environment. Information concerning the programming and use of these special device handlers (CL, DM, IB, and VM) is presented.

Chapter 6 - Terminal and CL I/O Processing
The information in this chapter provides the system manager with an understanding of the internal operation of the terminal and communication line (CL) handler. Modem control and the RS232 pin connection required for phone support are discussed.

Chapter 7 - System Operation
An understanding of the internal operation and organization of TSX-Plus provides the system manager with the the basic knowledge necessary for optimizing system performance. The system overview provides information concerning memory organization (with a detailed map of the TSX-Plus physical memory layout), I/O mapping, and execution scheduling (including a job scheduling flow diagram and algorithm).

Chapter 8 - System Tuning

With the basic knowledge of the organization and operation of TSX-Plus, the system manager can utilize various tools to better optimize the TSX-Plus execution environment. Suggestions concerning optimization for memory, I/O, and execution scheduling are provided.

Chapter 9 - SYSMON - Dynamic System Display Utility

The SYSMON utility displays information about system activities and resources. This utility can help the system manager gain more information about the specific environment to facilitate resource optimization.

Appendices

Appendix A describes the error messages which can be generated by TSX-Plus when it is started (R TSX). Appendix B describes the fatal system error messages generated when abnormal conditions occur during operation of TSX-Plus.

1. Privileges

Each job has associated with it a set of privilege flags that control which system services are available to the job. Privilege authorization is controlled by the system manager and may be allocated on a line-by-line basis or, if the LOGON facility is used, on an account-by-account basis. See the chapter describing the TSAUTH program for information about authorizing accounts with specific sets of privileges.

1.1 Privilege names

The following list of privileges shows the privilege keyword (e.g., ALLOCATE), the privilege word (1 or 2) which contains the privilege flag bit, and the bit position corresponding to the privilege flag within the word (0 to 15). The word and bit positions are provided for use with the TSX-Plus system service call (EMT) that is used to check and modify privileges. See chapter 7 of the TSX-Plus Reference Manual for information about this EMT.

ALLOCATE [1/14] (Allocate a device)
- o Perform ALLOCATE command or issue EMT to allocate a device.

BYPASS [1/9] (Bypass device/file access restrictions)
- o Allows access to .TSX and .SYS files.
- o Allows access to any device (that is not allocated by someone else).
- o Allows non-file-structured lookups.

DEBUG [1/13] (Use debugging facilities)
- o RUN/DEBUG a program.
- o SET CTRLD DEBUG.
- o SET EMT TRACE.

DETACH [1/12] (Affect detached jobs)
- o Start or kill detached jobs, commands or EMT's.

GETCXT [2/7] (Get file context from another job)
- o Use of TSX-Plus EMT to copy the file context from another job. A job with GETCXT privilege may gain access to the files of any other executing job.

GROUP [2/13] (Affect jobs with same project number)
- o Perform the following operations on any job which has the same project number as the job issuing the command:
  1. Issue KILL EMT.
  2. Issue KILL command.
  3. Change another job's priority (within range allowed to changer).
  4. Suspend or resume execution.

MEMMAP [1/10] (Access memory that may be significant to system, e.g., I/O page)
- o EMT's to peek/poke, BIS/BIC into I/O page.
- o .PEEK,.POKE requests outside of RMON (low memory or I/O page).
- o RUN/IOPAGE command.
- o EMT to map to I/O page or back to RMON.
- o EMT to map to physical memory.

MESSAGE [2/10] (Named message channels)
- o Use named message channels.

NFSREAD [1/2] (Non-file-structured read access)
- o Allow non-file-structured lookup on directory structured device with read-only access. (Needed to run PIP or DIR.)
- o Allows use of commands: COPY, CREATE, DELETE, DIRECTORY, PRINT, RENAME, TYPE (Some options to these commands may require NFSWRITE privilege.)

NFSWRITE [1/1] (Non-file-structured access)
- o Allow non-file-structured lookup on directory structured device and reads and writes.
- o Allows use of FORMAT, INIT, SQUEEZE, COPY/DEVICE, and other DUP functions.

OPER [1/8] (Operator)
- o Set system date and time (keyboard command and EMT's).
- o Perform any SET to device handler.
- o BOOT, $STOP, $SHUTDOWN commands.
- o SPOOL commands (ALIGN, LOCK, FORM, SKIP, BACK, SINGLE, MULT, HOLD, NOHOLD). OPER privilege is not required for SPOOL,STATUS and SPOOL,DELETE commands.

PSWAPM [1/7] (Change process swap mode)
- o EMT to lock job in memory or reenable swapping.
- o RUN/MEMLOCK command.

REALTIME [1/6] (Real-time EMT's)
- o .DEVICE request.
- o EMT to gain exclusive system control.
- o EMT to set processor priority level.
- o EMT to attach interrupts to service or completion routines.
- o EMT to release an interrupt connection.

RLOCK [2/9] (Shared file record locking)
- o Use shared file access control or record locking.

SAME [2/12] (Affect another job with same PPN)
- o Perform the following operations on any job which has the same project and programmer number as the job issuing the command:
  1. Issue KILL EMT.
  2. Issue KILL command.
  3. Change another job's priority (within range allowed to changer).
  4. Suspend or resume execution.

SEND [1/5] (Sending messages between jobs (not named message channels))
- o SEND command or EMT to send message to another line.

SETNAME [1/4] (Change user name or password)
- o SET PROCESS/NAME=name command.
- o EMT to change process name.
- o Change own password.

SETPRV [1/3] (Affect own privileges)
- o SET own privileges beyond those authorized.

SPFUN [1/11] (Use .SPFUN EMT for directory structued devices)
- o Allow .SPFUN (special function) operations to directory structured devices. (Needed to initialize some disks.)

SUBPROCESS [2/11] (Use subprocesses)
- o Ability to use subprocesses

SYSGBL [2/8] (Global regions)
- o Use named global regions. This privilege is also required to use terminal display windows since they create global regions.

SYSPRV [1/0] (System privilege - System management operations)
- o Access .SYS and .TSX files.
- o INSTALL ADD/DELETE command.
- o SHOW INSTALL command.
- o RESET command.
- o SET CACHE, IO ABORT, CORTIM, HIPRCT, INTIOC, NUMDC, QUANx, SYSPASSWORD.
- o Use the TSAUTH program under TSX-Plus.

TERMINAL [2/15] (Terminal and CL commands)
- o Perform any SET TT to line other than your own.
- o Make permanent SET to your own line.
- o Change line speed, parity, bits-per-character, SYSPASSWORD control.
- o Issue EMT to change line speed, parity, bits-per-character.
- o Perform any SET CL command.
- o Issue EMT to assign CL unit to a line.
- o Issue SET HOST command to cross connect a time-sharing line with a CL line.

UP1 [2/0] (User-defined privilege 1)

UP2 [2/1] (User-defined privilege 2)

UP3 [2/2] (User-defined privilege 3)

UP4 [2/3] (User-defined privilege 4)

WORLD [2/14] (Affect any job)

- o Perform the following operations on any job:
  1. Issue KILL EMT.
  2. Issue KILL command.
  3. Change another job's priority (within range allowed to changer).
  4. Suspend or resume execution.
- o Affect detached jobs started automatically by system initialization (jobs specified by DETACH command in TSGEN).

The four user-defined privileges, UP1, UP2, UP3, UP4, have no effect on system operation but are maintained by the system in the same fashion as the other privilege flags. User programs may check the setting of these privileges and base their operation on them.

1.2 Setting job privileges

At any time during the execution of a job, there are three sets of privilege flags associated with the job:

1. Authorized privileges.
2. Set privileges.
3. Current privileges.

The authorized privileges are those privileges for which the job is authorized. These are set by TSAUTH when an account is authorized, or by the SET PROC/PRIVILEGES/AUTHORIZED command.

The set privileges are those privileges which have been specified by use of the SET PROCESS command or the TSX-Plus EMT for setting privileges. Unless the job has SETPRV privilege, the set privileges will not exceed the authorized privileges.

The current privileges are the privileges currently in effect for the job. These privileges are selected as each program is started based on the set privileges and program-dependent privilege specifications for installed programs. A TSX-Plus EMT is available to allow running programs to change the current privileges.

The SET PROCESS keyboard command can be used to alter job privileges as well as other job characteristics. The form of this command is:

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

See the TSX-Plus Reference Manual for information about using the SET PROCESS command for changing process parameters other than the privilege flags.

The /PRIVILEGES qualifier 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 ALLOCATE privilege and removes the SUBPROCESS privilege:

```
SET PROCESS/PRIVILEGES=(ALLOCATE,NOSUBPROCESS)
```

In addition to the privilege keywords and their NO-complements, the following special privilege keywords may be specified:

```
ALL      -- All privileges
NONE     -- No privileges
STANDARD -- Standard privileges (ALLOCATE, DEBUG, DETACH, SPFUN, SEND,
            SETNAME, NFSREAD, NFSWRITE, SAME, SUBPROCESS, MESSAGE, RLOCK).
```

These special keywords may be used in conjunction with other keywords to select sets of privileges. When this is done the special keyword (ALL, NONE, or STANDARD) should be specified as the first privilege followed by additional privilege keywords. For example, the following command grants all privileges except REALTIME, SETPRV, and MEMMAP:

```
SET PROC/PRIV=(ALL,NOREALTIME,NOSETPRV,NOMEMMAP)
```

The following command grants the standard privileges plus the OPER privilege:

```
SET PROC/PRIV=(STANDARD,OPER)
```

The /AUTHORIZED qualifier can 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 standard privileges as the authorized, set, and current privileges for the job:

```
SET PROC/PRIV=STANDARD/AUTHORIZED
```

Note that this command affects the authorized privileges for the job only during the current job session.

The SET PROCESS command can always be used to remove a privilege from the authorized, set, and current privileges for the job. Privileges can only be granted for a job if the job is authorized for the privileges or if the job has SETPRV privilege which allows the job to override the authorized privileges and set any privilege.

The SHOW PRIVILEGE keyboard command can be used to display the authorized and current privileges for the job.

1.3 Initial job privileges

If the LOGON facility is used, a set of privileges may be specified for each account. See chapter 2 of this manual for information about authorizing accounts and setting privileges.

When a job is initiated it is authorized for all privileges. Privileges may be restricted by either of two methods: (1) the SET PROCESS/PRIV/AUTHORIZED command, or (2) the LOGON program. If the LOGON program is being used, a command of the form R LOGON should appear in the start-up command file for the line. When the LOGON program executes, it will set the privileges for the job based on privileges specified for the account by the TSAUTH program. If the LOGON program is not used, the job will have full privilege unless a SET PROC/PRIV/AUTHORIZED command is placed in the start-up command file.

A TSX-Plus EMT is available to allow running programs to determine the privilege sets for the job and to change privileges. See chapter 7 of the TSX-Plus Reference Manual for information about this EMT.

2. ACCOUNT AUTHORIZATION PROGRAM

TSAUTH, the TSX-Plus account authorization program, is used to authorize accounts for access to the system when the LOGON facility is used. It is also used to display the use accounting statistics that are collected by the LOGON facility.

A user must have SYSPRV privilege to run TSAUTH under TSX-Plus. However, TSAUTH may also be run directly under RT-11 without TSX-Plus. In a hostile environment it might be desirable to restrict access to the TSX-Plus distribution media and to keep the TSAUTH program on a removable medium rather than keeping it on the system disk. TSAUTH creates a file on SY named "ACCESS.TSX". Note that SYSPRV or BYPASS privilege is required to create or execute any file with the extension "TSX".

Whenever TSAUTH is started it checks to see if an account authorization file already exists. If not it prints the message:

```
Cannot open account authorization file "SY:ACCESS.TSX"
Do you want to initialize a new authorization file?
```

If you respond "YES" (or "Y") to this question it will ask you how many accounts you want to reserve room for in the file. Respond by entering the maximum number of accounts that you anticipate ever needing to have authorized at any one time. As old accounts are deauthorized, file space is recovered that can be used for new accounts. Note however that the only way to enlarge the ACCESS file is to delete it and build a new, larger one from scratch. Do not underestimate the potential number of accounts desired.

2.1 Account entry information

Each entry in the account authorization file contains a user name, a project-programmer number, a password, the name of a start-up command file, priority and privilege information, and resource usage statistics.

The user name is a string of up to 12 characters which is used when logging onto the system; it is also used with TSAUTH commands to identify an account. Each account must have a unique user name.

The project-programmer number is a pair of numbers. The first number is the project number, the second number is the programmer number. The project-programmer number is written with a comma separating the numbers. The project and programmer numbers must be integer values in the range 1 to 65534. Normally the user name is used to log on and identify and account and the project-programmer number has little significance (TSAUTH will automatically assign a project-programmer number if one is not specified). However, if a job has GROUP privilege, it can perform certain operations on other jobs with the same project number (such as aborting the job), hence the same project number should not be assigned to two jobs that have GROUP privilege unless this type of access is desired. TSAUTH allows a wildcard character ("*") to be specified for the project and/or programmer number with some of its commands, so it is is desirable to assign a common project or programmer number to accounts that will be managed as a group. Each account must have a unique project-programmer number combination.

A password is a string of 1 to 7 characters that is used as a security verification when logging onto the system. Accounts are not required to have passwords (although it is strongly recommended that they do). If an account does not have an assigned password, the user may log on by specifying only the user name for the account. The account password is normally assigned when the account is authorized and may be changed with the TSAUTH MODIFY command. If an account has SETNAME privilege, the user may change the password at the time that they log on. For security reasons, TSAUTH will not display account passwords. Hence, if a user forgets the password the correct procedure is to use the TSAUTH MODIFY command to assign a new password. Passwords do not need to be unique.

A start-up command file is an indirect command file that is executed when an account logs on. There are two types of start-up command file. One type of start-up command file is associated with each time-sharing line. This command file is executed whenever the line is initialized (i.e., at system startup time if the line is specified to be automatically started, or when carriage return is received for lines that are not automatically started). This line-dependent start-up command file may contain a "R LOGON" command to cause the logon program to be run when the line is initiated. The second type of start-up command file is associated with each account. This type of command file is executed when the account logs on. Thus the first type of start-up command file executes from the time the line is initiated up to and including the execution of the LOGON program. The second type of start-up command file is executed following the LOGON program and is determined by which account logs on. The line-dependent start-up command file is specified in TSGEN within the line definition block. The account-dependent start-up command file is specified with TSAUTH. The start-up command file specification may be up to 15 characters long and may contain a device name as well as a file name and extension. Accounts are not required to have associated start-up files and start-up file specifications need not be unique between accounts.

The maximum execution priority limits the execution priority that may be used by the account. This may be from 1 to 127. The default execution priority (and default maximum execution priority) is 50.

Account privileges determine which system services are available to the account. Privileges are specified as a set of privilege names which may be on (privilege granted) or off (privilege denied). Chapter 1 provides complete information about each privilege; a summary of privilege keyword names and functions is shown below:

Privilege	Function
ALLOCATE	Allocate a device
BYPASS	Bypass device/file access restrictions
DEBUG	Use system debugging facilities
DETACH	Start or kill detached jobs
GETCXT	Get file context from another job
GROUP	Affect other jobs with the same project number
MEMMAP	Access system memory (e.g., I/O page)
MESSAGE	Send and receive messages
NFSREAD	Perform non-file-structured reads
NFSWRITE	Perform non-file-structured writes
OPER	Operator functions (e.g., set time, spool control)
PSWAPM	Change process swap mode (i.e., lock in memory)
REALTIME	Real-time system services (e.g., interrupt connect)
RLOCK	Shared file record locking and access control
SAME	Affect another job with same project-programmer number
SEND	Send a message to another terminal
SETNAME	Change job name or password
SETPRV	Set privileges beyond those authorized
SPFUN	Allow .SPFUN operations to directory structured devices
SUBPROCESS	Use subprocesses (virtual lines)
SYSGBL	Use named global regions and display windows
SYSPRV	System manager functions (e.g., authorize accounts)
TERMINAL	Terminal and CL line control (e.g., set line speed)
UP1	User-defined privilege 1
UP2	User-defined privilege 2
UP3	User-defined privilege 3
UP4	User-defined privilege 4
WORLD	Affect any job

Most sites find it convenient to group accounts into three categories based on the type of privileges they require: (1) ordinary accounts which don't need any special privileges; (2) operator accounts used by people who run the computer but don't need to authorize accounts or tune the system; and (3) system manager accounts. The following sets of privileges are recommended for each of these account types:

Ordinary accounts: ALLOCATE, DEBUG, DETACH, SPFUN, SEND, SETNAME, NFSREAD, NFSWRITE, SAME, SUBPROCESS, MESSAGE, RLOCK.

Operator accounts: Same as ordinary accounts plus OPER, TERMINAL, WORLD.

System manager accounts: All privileges.

Ordinary and operator accounts must also be given SYSGBL privilege if the process windowing facility is to be used.

2.2 Command summary

The following commands may be used with the TSAUTH program:

Command	Function
AUTHORIZE	Authorize an account
MODIFY	Modify information associated with an account
KILL	Deauthorize an account
LIST	List information about accounts
USAGE	List account usage statistics
CHARGE	Create a charge file
RESET	Reset account usage statistics
EXIT	Exit from the TSAUTH program

TSAUTH commands begin with a keyword (AUTHORIZE, MODIFY, KILL, etc.) followed (in most cases) by the user name associated with the account. Commands may be continued by typing a minus sign ("-") as the last character on the line to be continued and then typing the continuation of the command on the next line.

2.3 Authorizing an account

The form of the command used to authorize a new account is:

AUTHORIZE username [/qualifiers...]

where "username" is the account user name which may be up to 12 characters long, and "qualifiers" include the following specifications:

/PPN=project,programmer specifies the project-programmer number to be associated with the account. The first number of the pair is the project number, the second number is the programmer number. If a PPN is specified, it must be unique. If the /PPN qualifier is not specified, TSAUTH will automatically assign a unique PPN for the account. It does this by assigning project and programmer numbers which are one larger than the current largest PPN of any authorized account. The automatically generated numbers are never smaller than 100. If you wish to specify either the project or the programmer number but allow TSAUTH to automatically assign the other number, you may specify an asterisk ("*") for the number which is to be assigned by TSAUTH and a numeric value for the number you wish to specify.

/PASSWORD=string specifies the password for the account. The password string may be up to 7 characters long. If the /PASSWORD qualifier is omitted, no password will be required for the account to log on. If the /PASSWORD qualifier is specified without an equal sign or string, TSAUTH prompts for the password and accepts it without echoing it to the terminal. This is useful it you want to prevent the password from being printed on a hardcopy terminal listing.

/START=file specifies the logon start-up file specification. The file specification may be up to 15 characters long. If the /START qualifier is omitted, no start-up command file will be executed when the account logs on.

/PRIORITY=value specifies the maximum authorized execution priority. If specified, the value must be in the range 1 to 127. If this qualifier is not specified, the maximum priority for the job will be set to 50.

/PRIVILEGE=privilege-list specifies the privileges which the job is to have. If the privilege list consists of more than one keyword, the keywords must be enclosed in parentheses and separated by commas. The parentheses may be omitted if a single privilege keyword is specified. The special privilege keywords ALL, NONE, and STANDARD may be used. The standard privileges are: ALLOCATE, DEBUG, DETACH, SPFUN, SEND, SETNAME, NFSREAD, NFSWRITE, SAME, SUBPROCESS, MESSAGE, RLOCK. Initially the privileges are set to STANDARD. The privileges specified in the privilege list modify the standard privileges. For example, the specification /PRIVILEGE=(NOSUBPROCESS,SYSPRV) authorizes the account with the standard privileges but without the SUBPROCESS privilege and with the SYSPRV privilege. The specification PRIVILEGE=(NONE,SPFUN,NFSREAD,NFSWRITE) cancels all of the standard privileges and grants only SPFUN, NFSREAD, and NFSWRITE. If the /PRIVILEGE qualifier is omitted, the standard privileges are assigned to the account.

Example:

The following account authorizes an account with user name "SALES"; project-programmer 20,34; password "ENGLAND"; maximum execution priority 60; standard privileges plus SYSGBL; start-up command file "DL0:STD.TSX":

```
AUTHORIZE SALES/PPN=20,34/PASSWORD=ENGLAND/PRIORITY=60-
    /PRIVILEGE=SYSGBL/START=DL0:STD.TSX
```

The following command authorizes an account with the user name "ACCT-MANAGER", password "SECRET", standard privileges plus SYSPRV, and a start-up command file named "DL0:MSTR.TSX". The default maximum execution priority (50) is used and TSAUTH automatically assigns a project-programmer number.

```
AUTHORIZE ACCT-MANAGER/PASSWORD=SECRET/PRIV=SYSPRV-
    /START=DL0:MSTR.TSX
```

If the AUTHORIZE command is used without specifying any qualifiers, TSAUTH will prompt for each qualifier. When operating in this mode, if you press return without entering anything in response to the PPN (project programmer number) prompt, TSAUTH automatically generates and assigns a project programmer number.

For example, the following operation authorizes an account named "OPERATOR"; project-programmer number 1,1; password "OP4376"; start-up command file "DL0:OPR.TSX"; and operator privilege.

```
AUTHORIZE OPERATOR
PPN:1,1
Password:OP4376  [this is not echoed]
Start-up file:DL0:OPR.TSX
Maximum execution priority:
Privileges:OPER,TERMINAL,WORLD
```

2.4 Modifying an account

The form of the command used to modify parameters for an account is:

MODIFY {username | project,programmer}/qualifiers...

Either a user name or a project programmer number may be specified with the MODIFY command. If a project programmer number is specified, a wildcard character ("*") may be specified for either or both of the numbers causing the modification to be applied to multiple accounts. For example, the following command grants SUBPROCESS privilege to all accounts with project number 43:

```
MODIFY 43,*/PRIV=SUBPROCESS
```

The qualifiers which may be used with the MODIFY command are: /PASSWORD, /PRIVILEGE, /PRIORITY, and /START. These qualifiers have the same form as specified with the AUTHORIZE command. The /PRIVILEGE qualifier adds or removes privileges relative to the privileges the account has at the time that the MODIFY command is executed, rather than adding or removing privileges from the standard set as is the case with the AUTHORIZE command. An account must be reauthorized to change either the user name or the project programmer number. The user of the account that is being modified, if logged on, must log off and back on before the modifications will take effect.

For example, the following command modifies the account with the user name MANAGER to add BYPASS privilege and change the password to SPIRIT:

```
MODIFY MANAGER/PRIV=BYPASS/PASSWORD=SPIRIT
```

As with the AUTHORIZE command, if the /PASSWORD qualifier is specified without an equal sign and string, TSAUTH will prompt for the password and accept it without echo.

2.5 Deauthorizing an account

The KILL command is used to deauthorize an account. The form of this command is:

KILL {username | project,programmer}

Either a user name or a project programmer number may be specified. A wildcard character ("*") may be substituted for either the project number, the programmer number, or both.

For example, the following command deauthorizes the account with the user name JONES:

```
KILL JONES
```

The following command deauthorizes all accounts with project number of 120:

```
KILL 120,*
```

2.6 Listing information about accounts

The LIST command is used to list information about accounts. The form of this command is:

```
LIST {username | project,programmer}
```

Either a user name or a project programmer number may be specified. A wildcard character ("*") may be substituted for either the project number, the programmer number, or both.

For example, the following command lists information about the account with user name OPERATOR:

```
LIST OPERATOR
```

The following command lists information about all accounts with project number 1:

```
LIST 1,*
```

2.7 Listing account usage statistics

The USAGE command is used to list the account usage statistics which consist of the number of sessions, the connect time, and the CPU time. The form of this command is:

```
USAGE {username | project,programmer}
```

Either a user name or a project programmer number may be specified. A wildcard character ("*") may be substituted for either the project number, the programmer number, or both.

2.8 Creating a charge information file

The CHARGE command causes TSAUTH to create a file of usage information. The file is named "DK:CHARGE.TSX"; it contains one record for each account; each record is terminated with a carriage return and line feed.

The format of a charge record is as follows:

Columns	Contents
1	(blank)
2 - 6	Project number
7	(blank)
8 - 12	Programmer number
13	(blank)
14 - 18	Number of logons
19	(blank)
20 - 24	Number of minutes of connect time
25	(blank)
26 - 33	CPU time used (0.1 second units)
34	(blank)
35 - 46	User-name (left justified and padded with blanks)
47	(carriage return)
48	(line feed)

2.9 Resetting account usage statistics

The RESET command resets the account usage statistics (number of sessions, connect time, and CPU time) to zero for all or a selected set of accounts. The form of the command is:

RESET {username | project,programmer}

Either a user name or a project programmer number may be specified. A wildcard character ("*") may be substituted for either the project number, the programmer number, or both.

2.10 Exiting from the TSAUTH program

The EXIT command (or control-C) is used to exit from the TSAUTH program. The form of the EXIT command is:

EXIT

2.11 Use of indirect files within commands

A portion of a command line may be drawn from an external file by typing @file at the position in the command where the contents of the file are to be inserted into the command line. The default extension for these files is TSX. For example, the following command uses parameters in a file named STUDNT as part of an AUTHORIZE command:

AUTHORIZE JONES @STUDNT

The following command draws the privilege list from a file named CLERK and in addition negates the SUBPROCESS privilege:

```
AUTHORIZE FRANK/PRIV=(@CLERK,NOSUBPROCESS)
```

More than one indirect file reference may occur within a command but the indirect files may not be nested.

3. SYSTEM AND FILE ACCESS SECURITY

TSX-Plus provides a number of security options that allow the site manager to control system resources to time-sharing users. The system manager can control who may log onto the system, which files or devices each user may access and can also lock users to particular programs. The following facilities can be used to control system access:

1. System password.
2. Start-up command files.
3. Log-off command files.
4. The RUN/LOCK switch.
5. The LOGON program with log-on command files.
6. The ACCESS command.
7. The SET MAXPRIORITY command.
8. Installed programs.

3.1 System password feature

A "system password" facility is available to provide additional security on dial-up lines in addition to the LOGON program. System password checking can be enabled on a line-by-line basis. If system password checking is enable for a line, the user is prompted with an exclamation point ("!") when the line is initiated (i.e., when carriage return is pressed). After the correct password is entered, the normal logon sequence begins which prints the system greeting message and runs the start-up command file (which typically runs the LOGON program). The idea is to force the user to provide a password before displaying the logon greeting which identifies the site and the nature of the system.

To enable system password checking, specify the system password using the SYSPS macro in TSGEN, and include the $SYSPS flag with the FLAGS macro within the line definition blocks for all lines that are to perform system password checking. The system password may be up to 20 characters long and may contain spaces. The same system password is used for all lines that perform password checking.

When a line that performs system password checking is initialized, the system does autobaud speed selection (if required) and then prints an exclamation point as the prompt for the system password. The password is not echoed as it is entered. Terminate the password by pressing carriage return. If the password is correctly entered, the system will print the standard TSX-Plus greeting message and perform the normal line initiation sequence. If an incorrect password is entered, no error message is printed but the exclamation point is redisplayed. If the password is entered incorrectly a second time,

the line is hung up. The password must be entered within the time limit specified by the OFFTIM sysgen parameter or the system will hang up.

A keyboard command of the form:

SET SYSPASSWORD password

can be used to change the password for the running system. SYSPRV privilege is required to use this command; the change only remains in effect until the system is rebooted. The TSXMOD program can be used to change the system password in the TSX.SAV file.

The system password can be displayed by use of the following command:

SHOW SYSPASSWORD

SYSPRV privilege is required to use this command.

System password checking may be enabled or disabled for individual lines by use of the following command:

SET TERMINAL n [NO]SYSPASSWORD

where n is the terminal line number. TERMINAL privilege is required to issue this command. The TSXMOD program can also be used to set or reset this flag for lines.

3.2 Start-up command files

In the system generation, a unique start-up command file which executes each time the line is initialized may be specified for each of the physical time-sharing lines. The command file name is defined using the CMDFIL macro within a line definition block in TSGEN. (See the TSX-Plus Installation Guide for information on the CMDFIL macro.) Different command file names may be specified for each line and any or all lines may be generated without these start-up command files.

When a line has an associated start-up command file, the command file is executed each time the line is initialized (e.g., when the user presses carriage return on an inactive line). Start-up command files are unique from other command files in that their execution cannot be aborted by typing control-C. This allows the system manager to place any desired commands in the start-up command file to be executed to completion regardless of the actions of a time-sharing user. However, if for some reason the command file abnormally terminates, the line may be granted full access to the system without proper initialization. This can be avoided by disabling command file aborts, except in the most serious circumstances, by setting the error abort level as the first command. This is expecially important for lines started with complex command files and for dial-up lines. For example:

```
SET ERROR NONE
     .
     .
     .
R/LOCK LOGON
OFF
```

This would prevent the line from accessing the system even if the LOGON program was not found.

A start-up command file may contain any keyboard command and can run one or more programs. Control-C resumes its normal function when the start-up command file is terminated or a program started by it requests input from the terminal. It is suggested that start-up command files be given the extension "TSX" to prevent their being tampered with by users who do not have system operator (SYSPRV) privilege or bypass access (BYPASS) restrictions (see below). If "TSX" is used as the file extension, it must be specified with the file name in the CMDFIL macro since the default extension is "COM". The default device is "SY:".

The listing of a start-up command file can be suppressed by placing the two character sequence "^(" at the front of the command file. See the chapter on Command Files in the TSX-Plus Reference Manual for more information on command file control characters.

Subprocesses (previously called virtual lines) are started by typing control-W followed by a digit which selects the subprocess. When a subprocess is initiated, it "inherits" a large part of the process context from the primary process but does not ordinarily execute any start-up command file. If you wish to have a start-up command file executed when a subprocess is started, place a command of the following form in the start-up command file for the primary process:

```
SET SUBPROCESS/FILE=file
```

where file is the file specification for the start-up command file that is to be executed when any subprocesses are initiated by the primary process. For example, the following command specifies that a file named "SY:VSTART.TSX" is to be executed each time a subprocess is started:

```
SET SUBPROCESS/FILE=SY:VSTART.TSX
```

The SET SUBPROCESS/FILE command is valid only within a start-up command file.

3.3 Log-off command files

It is possible to define a command file that is to be executed when a job logs off. To declare a log-off command file, place a command of the following form in the start-up command file for the job:

SET LOGOFF/FILE=name

where "name" is the file specification for the log-off command file. The SET LOGOFF command is valid only within the start-up command file for the job. The log-off command file is executed whenever the job logs off. Be careful with what you put in a log-off command file since the execution of a log-off command file cannot be aborted by typing control-C. The listing of a log-off command file can be suppressed by placing "^(" as the first two characters of the file.

3.4 The RUN/LOCK switch

The "R" and "RUN" commands accept a "/LOCK" switch that causes the program being run to be "locked" to the time-sharing line. A locked program executes in the normal fashion, and may chain to other programs (which are also locked). However, if a locked program exits or is aborted by typing control-C the line is automatically logged off. Note that one can prevent an ongoing program from being aborted by control-C by doing a .SCCA EMT or by defining control-C as and activation character using the TSX-Plus program controlled terminal options (see the TSX-Plus Reference Manual chapter on Terminal Control for information on defining activation characters).

In a situation in which a time-sharing line is to be automatically locked to a program when the line is started, simply build a start-up command file for the line and include as the last start-up entry in the file a "RUN/LOCK program" command.

3.5 Use of the LOGON facility

The TSX-Plus LOGON facility provides access security to the system by requiring users to enter a valid project-programmer number or user name and password before granting access to the system. In addition, the LOGON facility allows the system to grant different privileges to each user and provides system use accounting on a per user basis.

To use the LOGON facility the system manager must first use the account authorization program (see the chapter titled Account Authorization Program) to create an account authorization file. This file specifies the valid project-programmer numbers, user names, passwords, user start-up command file, and privileges. He must then generate a TSX-Plus system and specify a line-by-line start-up command file to be executed for each line that is to be forced to logon. The suggested name for this start-up command file is "SY:LOGON.TSX". This command file may contain any desired keyboard commands but should start by disabling error aborts, should lock the job to the LOGON program, and should end by logging the job off. In this fashion, the job will not be able to gain

access to the system even if the LOGON program is missing or some other command fails. For example:

```
SET ERROR NONE
     .
     .
     .
R/LOCK LOGON
OFF
```

This command causes the LOGON program to be started and "locked" to the line so that the user cannot run any other program until the logon has been successfully completed. Note that the logon program (LOGON.SAV) must be present on the system device. The OFF command will only be executed if the LOGON program cannot be run.

Note that for each job there may be two start-up command files: the first is specified with the CMDFIL macro in TSGEN and is associated with a physical time-sharing line; the second is associated with a particular user (account name, project-programmer number) and is invoked through the LOGON program and account authorization system.

To prevent listing the start-up command file, the character sequence "^(" may be placed at the beginning of the command file. Thus, the logon start-up file for a physical time-sharing line might contain:

```
^(SET ERROR NONE
R/LOCK LOGON
OFF
```

A SET LOGOFF command can be placed in the start-up command file to declare the name of a command file to be executed when the job logs off.

3.6 The ACCESS command

The ACCESS keyboard command is used to limit access to devices and files. The ACCESS command is valid only if executed as part of a start-up command file.

The form of the ACCESS command is:

```
ACCESS dev:file.ext/switch,dev:file.ext/switch,...
```

Up to twenty-five "dev:file.ext" expressions may be specified. Each logical subset disk mounted also counts toward the limit of entries in the access table.

If no ACCESS command is executed, the time-sharing user is allowed to access all devices and files on the system (with the exception of SYS and TSX extensions - see SYSPRV and BYPASS privileges). If an ACCESS command is executed, the user is restricted to accessing only the devices and files that are specified with the command.

The "dev:file.ext" expression has three items: the device name, the file name and the extension. The "*" (wildcard) character may be substituted for any or all of these three items. In this case the wildcard will allow access to any name that occurs in the wildcarded position. For example, "RK1:*.ABC" will allow access to any file on RK1 that has the extension "ABC". Consider the following ACCESS command:

```
ACCESS RK0:*.ABC,RK0:*.BAK,RK1:*.*,LP:
```

This allows access to any files on RK0 that have the extension "ABC" or "BAK"; it also allows access to all files on RK1 and LP. Note that the LP specification is needed if the user is to be allowed to access the line printer. Access privilege is needed to read, create, delete, or rename a file. A device can only be initialized (directory cleared) if full access to the device and non-file-structured write privilege (NFSWRITE) are granted.

The ACCESS facility works by matching the user-specified device, file and extension names with those that were specified on the ACCESS command. This matching is done after any ASSIGNS of logical to physical device names are carried out.

Since the utility programs PIP, DUP and DIR directly access device directories, they exhibit minor deviations from expected access protection behavior. If access is granted to any files on a device and non-file-structured read privilege (NFSREAD) is granted, then DIR will be able to obtain the device directory. In order for PIP and DUP to access an individual file, the job must have at least /READ access to the full device and non-file-structured read privilege (NFSREAD), even if access has been granted to the specific file of interest. These deviations affect the DIR, COPY, TYPE, and PRINT commands among others.

The "/READ" switch may be specified with a device-file name to restrict access to the device-file to be read-only. For example, the following command allows full access to RK1 but read-only access to RK0.

```
ACCESS RK1:,RK0:/READ
```

Remember that the common utility programs, such as PIP and DIR, are required by most users and consequently at least SY:*.SAV/READ access is usually desirable. Also, access to system library files (SY:SYSLIB.OBJ, SY:FORLIB.OBJ) and the system MACRO library file (SY:SYSMAC.SML) may be necessary for program development. Because of the limited number of ACCESS entries that may be made (25 for each job), it is not advisable to enumerate each specific file to which access is desired, but rather to cluster groups of files on the system disk or on logical subset disks. For example, the following ACCESS command could be used to grant full access to DL1 and limited access to the system disk:

```
ACCESS DL1:,SY:*.SAV/READ,SY:SYSLIB.OBJ/READ,SY:SYSMAC.SML/READ
```

The ACCESS and MOUNT commands can be used together to control access to logical subset disks. To control which logical disks are available to a user, specify the names of the files that contain the logical disks with the ACCESS command in the startup command file and then use MOUNT commands after the ACCESS command to associate logical disk units with the files. This will allow the user to access all files within the logical disk but will restrict access to other logical disks or files. For example, consider the following commands which could be placed in a startup command file:

```
ACCESS SY:/READ,DL0:CLASS1.DSK,DL0:CLASS2.DSK/READ
MOUNT LD1 DL0:CLASS1
MOUNT LD2 DL0:CLASS2
```

After executing this startup command file, the user will have read only access to all files on the system disk ("SY:"), read-write access to LD1 which is associated with the file DL0:CLASS1.DSK, and read-only access to LD2 which is associated with DL0:CLASS2.DSK. This will permit the user to initialize LD1 and create, edit, and delete files on LD1. The user may also create nested logical disks within LD1. Files on LD2 may be accessed for reading only.

3.7 The SET MAXPRIORITY command

TSX-Plus users can assign execution priority values to their jobs by use of the SET PRIORITY command and a TSX-Plus EMT. The maximum priority that a user is allowed can be controlled by use of either the TSAUTH program (in conjunction with the LOGON program), or the SET MAXPRIORITY command. Normally the TSAUTH program would be used to assigned maximum priorities if the LOGON facility is being used. The SET MAXPRIORITY command is intended primarily in situations where the LOGON facility is not being used but it is still desirable to limit the maximum authorized priority. In these cases the SET MAXPRIORITY command can be placed in the start-up command file for the line.

The form of the SET MAXPRIORITY command is:

SET MAXPRIORITY value

where "value" is in the range 0 to 127. The SET MAXPRIORITY command may only lower the maximum authorized priority value for the job, it may not increase it. Thus the system manager may restrict job priority by placing a SET MAXPRIORITY command in the start-up command file for a line.

3.8 INSTALL command

Programs may be "installed" in the system to allow special attributes and privileges to be invoked when the program is run. The form of the INSTALL command used to add a program to the install table is:

INSTALL ADD program[/attribute...][/PRIV=(privileges)]

Where program is the file specification for the program being installed, attribute is one or more of the attributes listed below, and privileges is a list of privilege keywords which specifies which privileges are to be temporarily granted or denied while the program is executing. If a program is already in the install table, the new specifications replace the existing ones.

The device name specified for the program (or DK by default) is translated to a physical device name at the time the INSTALL command is executed. Similarly the physical device name for a program being started is used when the install table is searched at program start-up time. Installed programs must reside on physical disks, and may not be located on logical disks.

The INSTALL command does not open the SAV file of the program being installed and no error will occur if installed programs are not actually present on the system.

The following programs are automatically installed when TSX-Plus is started: IND, KED, KEX, K52, LOGON, PATCH, SETUP, SYSMON, TECO, TRANSF, TSAUTH, VTCOM. These programs should be located on the disk from which the system was booted. Because of the installation of SYSMON, it is no longer required to have system privilege to execute SYSMON. The INSTALL command can be used to alter the standard installed programs.

One entry is reserved in the install table for each of the programs that is automatically installed by TSX-Plus plus the number of entries specified by the NUIP sysgen parameter.

The form of the INSTALL command used to remove a program from the install table is:

INSTALL DELETE program

where program is the file specification for the program whose entry is to be removed.

The SHOW INSTALL keyboard command may be used to list information about all installed programs.

SYSPRV privilege is required to use the INSTALL ADD/DELETE command or the SHOW INSTALL command.

The following attributes may be specified for installed programs:

Attribute	Meaning
BYPASN	(Bypass logical assignments) Bypass all logical device assignments. DK and SY will be directed to the system disk (disk from which RT-11 was booted).

HIGH	(High-efficiency mode) High-efficiency terminal mode is selected.
IOPAGE	(I/O page access) I/O page is mapped into PAR 7 of program virtual address space.
LOCK	(Lock program to job) Program is "locked" to job so that job is logged off if program exits.
MEMLOCK	(Lock program in memory) Program is locked in low memory while it is running.
NONINTERACTIVE	(non-interactive execution) Program is run with non-interactive execution priority.
NOWAIT	(Non-wait terminal I/O) Program is allowed to execute non-wait .TTYIN and .TTOUTR operations (bit 6 must also be set in the job status word to enable non-wait terminal I/O).
SINGLECHAR	(single-character activation) Program is enabled to perform single-character activation (bit 12 must also be set in the job status word).
TRANSPARENT	(transparent terminal output) Terminal output is processed in "transparent" mode where control characters such as the TSX-Plus "leadin" character are not interpreted by the system.

Attributes and privileges specified for installed programs take effect when the program is started (either by use of a R or RUN command or by chaining to the program) and remain in effect until the program exits or chains to another program at which time the program attributes are cleared and the current privileges are reset to the set privileges.

The /PRIVILEGE qualifier may be used to temporarily grant or deny any privilege including privileges for which the job is not authorized. For example, the following INSTALL command installs a program named CONTRL on DL1 which is to have PAR 7 mapped to the I/O page while the program is running. MEMMAP privilege is granted while the program is running but DEBUG privilege is denied to prevent a user from triggering a breakpoint while the program is running with access to the I/O page:

```
INSTALL ADD DL1:CONTRL/IOPAGE/PRIV=(MEMMAP,NODEBUG)
```

Note in this example, MEMMAP privilege is required to run this program since the /IOPAGE attribute was specified. If the MEMMAP privilege had not been specified when the program was installed, only those users who are authorized

for MEMMAP privilege would be able to run this program. By installing the program with MEMMAP privilege, users who are not authorized for MEMMAP privilege are temporarily granted the privilege to enable them to run the program.

4. DEVICE HANDLERS

TSX-Plus supports the following RT-11 device handlers: CT, CR, DD, DL, DM, DP, DS, DT, DU, DX, DY, LP, LS, MM, MS, MT, NL, PC, RF, and RK. In addition, the IEEE GPIB version 2.1 IB device driver is supported. The logical subset disk (LD) and single line editor (SL) are implemented in TSX-Plus as overlay regions and do not require device handlers. The virtual memory handler (VM.TSX) is proprietary and unique to TSX-Plus. TSX-Plus also supports the communication line (CL) device handler.

The following RT-11 device handlers are unsupported under TSX-Plus: BA (resident batch handler), EL (error logging pseudohandler), and PD (PDT-11/130/150 handler). Also the IBSRQ function of the GPIB IEEE IB handler is unsupported. In addition, TSX-Plus supports the functionality (but not the RT-11 device handler implementation) for logical subset disks (LD), the single line editor (SL), and the virtual memory device (VM).

4.1 Device handler extensions and restrictions

TSX-Plus requires device handlers which are written to support a memory management RT-11 XM environment. Error logging is not supported under TSX-Plus. See the RT-11 Software Support Manual for details on device handlers. Device handlers must follow the rules for RT-11 XM device handlers in order to function with TSX-Plus.

4.1.1 RT-11 version number checking

TSX-Plus will not install device handlers which were issued by Digital subsequent to the version of RT-11 under which TSX-Plus is being started. In other words, you must upgrade to the appropriate version of RT-11 in order to be able to use the newer device handlers with TSX-Plus. Specifically, the following device handlers minimally require the indicated version of RT-11:

Device	RT-11 version
DU	5.00
XL	5.01

In addition to the RT-11 version checking, TSX-Plus determines if the device is present before installing the handler. This requires the following: 1) the address of the device CSR register, as specified in location 176 of the device handler file, be found; and 2) the installation code starting at location 200 in block 0 of the device handler file, if present, executes and returns with the carry-bit clear.

4.1.2 I/O queue element extension

TSX-Plus requires and stores more information concerning each I/O request than does RT-11. To accomplish this, TSX-Plus uses an I/O queue entry which is 17 decimal words long. Each element in the I/O queue has the following format:

Name	Offset	Length	Description
Q.LINK	0	2	Link to next queue entry
Q.CSW	2	2	Address of CSW for channel making request
Q.BLKN	4	2	Physical block number of request
Q.FUNC	6	1	Special function code
Q.UNIT	7	1	Device unit number (bits 0 through 2)
Q.JNUM	7	1	Job number issuing request (bits 3 through 7)
Q.BUFF	10	2	User buffer address relative to Q.PAR
Q.WCNT	12	2	Word count (+ =Read, 0 =Seek, - =Write)
Q.COMP	14	2	Address of completion routine for request
Q.PAR	16	2	PAR relocation bias for buffer address
Q.PA5	20	2	Mapping value for kernel PAR 5
Q.UMRX	22	2	Address of UMR block assigned for I/O
Q.CHAN	24	2	User channel # associated with I/O request
Q.DEVX	26	1	Device index number
Q.FLAG	27	1	Device control flags
Q.JOB	30	1	Number of job that is making request
Q.UMVB	31	1	Unibus UMR base register number
Q.UMPB	32	2	Original value of Q.BUFF when I/O was initiated
Q.UMPP	34	2	Original value of Q.PAR when I/O was initiated
Q.PA6	36	2	Mapping value for kernel PAR 6
Q.UCSW	40	2	Virtual address of user's channel block
Q.ICSW	42	12	Copy of user's channel block

TSX-Plus stores the number of the job issuing an I/O request in the Q.JOB byte of the queue element. Normally RT-11 only uses bit positions 11 through 15 (Q.JNUM) of the fourth word of the queue element. For compatibility, TSX-Plus also stores the job number in these bits, however, for jobs numbers greater than 32 (requiring more than 5 bits for representation), all job bits in this fourth word are set. A job number of zero implies the I/O request was initiated from the operating system.

4.1.3 Device handlers use of PARs

Any handler that accesses the user's buffer directly by remapping kernel page address register (PAR) 1 must be altered to use kernel PAR 6. Addresses in the I/O queue entries are automatically adjusted to pass virtual addresses within the PAR 6 region (140000 to 157777). In addition, boundary checking must be altered to correspond to this virtual address region. Any handler using PAR 6 must first issue a .INTEN request.

Kernel page address register 5 is also available for use in device handlers which are not loaded as mapped handlers. If PAR 5 or PAR 6 is used within a handler in an interrupt service routine (after doing an .INTEN) they do not have to be saved since the .INTEN will do this; however, if they are used in a handler other than at interrupt or fork level (e.g., on I/O initiation) they must be saved and restored by the handler.

4.1.4 Extension for the LSI-11 bus

On the LSI-11 bus, only the DL, DM (with DILOG DQ215 and Emulex SC02C controllers), DU, and MS handlers are actually supported with full 22-bit DMA capability, and these devices must have controllers which also support 22-bit addressing and the controllers must be so configured in order to achieve actual 22-bit capability. In order to use any DMA device from a program located above 256K bytes in physical memory, the device and handler must be capable of and configured for 22-bit addressing or the device must be declared to use system I/O mapping in its TSGEN device definition. See the description of the DEVDEF macro (in the TSX-Plus Installation Guide) for more information on 22-bit addressing and system I/O mapping. If a DMA device or handler does not support or is not configured for 22-bit addressing and does not use system mapping, then attempts to use it will generally result in "Illegal or uninitialized directory" or "Device I/O error" error messages. Serial devices (such as LP, LS, and DX) which do not use direct memory access do not require 22-bit handlers or controllers or system I/O mapping.

4.2 Device handler programmed requests

TSX-Plus supports the standard code expansion for device handler programmed requests implemented in the RT-11 system macro library (SYSMAC.SML) which include .CTIMIO, .DRAST, .DRBEG, .DRBOT, .DRDEF, .DREND, .DRFIN, .DRSET, .DRVTB, .FORK, .INTEN, .MFPS, .MTPS, .SYNCH, and .TIMIO. See the RT-11 System Support Manual for details concerning the usage of these programmed requests. In some cases (as described below), TSX-Plus provides various extensions to these programmed requests.

4.2.1 .FORK requests

In order to understand the processing of fork requests by TSX-Plus, it is helpful to review the concept of interrupt priorities. The PDP-11 family of computers has 8 interrupt priority levels, numbered 0 through 7. The priority of an interrupt is selected by the device requesting the interrupt. The processor (CPU) remembers the current interrupt priority in the processor status word. An interrupt request is held in a pending state and is not allowed to interrupt the processor if the current interrupt priority is equal to or greater than the pending interrupt priority. Priority level 0 is the priority at which the processor runs when no interrupt is being serviced.

Fork processing under TSX-Plus implements a software based interrupt system which effectively operates at an interrupt priority level greater than hardware priority 0 and less than hardware priority 1. Like the hardware interrupt system, TSX-Plus fork requests have priority values that are specified by the software component that is making the fork request. Also like the hardware interrupt system a fork request may interrupt a currently executing fork request of lower priority but may not interrupt a currently executing fork request of equal or greater priority. The fork priority values range from 1 to 127 (decimal); the higher the numerical value, the higher the priority.

Conceptually, fork priorities correspond to interrupt priorities in the range 0.001 through 0.127. Thus, any hardware interrupt request (which has a priority in the range 1 through 7) can interrupt any fork request. Fork requests are queued in order by priority value. If two or more requests have the same priority value, they are queued in the order in which the requests were made.

The standard instructions generated by a .FORK request are:

```
JSR     R5,@$FKPTR
.WORD   fkblk-.
```

Where $FKPTR is a cell containing the address of the system fork routine and fkblk is the address of a four word fork block. Although TSX-Plus does not actually use the four word fork block specified by the request, it does consult the device handler's fork block when necessary to determine the status of the fork entry.

The standard fork call works under both RT-11 and TSX-Plus. When this form of fork call is used under TSX-Plus, the fork request is queued with a fork priority of 50 (decimal).

An alternative fork request call may be used under TSX-Plus to specify a priority for the fork. The form of this call is:

```
JSR     R5,@$FKPTR
.WORD   100000+priority
```

Where "priority" is a fork priority value in the range 1 to 127. The constant 100000 must be added to the priority value to produce a value which TSX-Plus can recognize as a priority rather than the standard RT-11 format above which points to the offset of a user fork block. The assumption is that a handler fork block is highly unlikely to occur at an offset greater than 100000 from its fork request.

The current TSX-Plus fork priority values are defined in the intial part of TSGEN as follows:

Symbol	Value	Description
FP$MAX	127.	Maximum legal fork priority
FP$RT	100.	Real-time interrupts
FP$CKT	70.	50/60 Hz clock interrupt processing
FP$CDI	60.	Terminal character input processing
FP$CDO	55.	Terminal character output processing
FP$DEF	50.	Default fork priority
FP$IOF	50.	I/O complete
FP$IOA	50.	I/O abort entry
FP$PIO	50.	PI output interrupt processing
FP$CK1	30.	0.1 second clock processing
FP$IOS	12.	I/O initiation
FP$MOV	10.	Move data to/from cache buffer

Special care must be used to prevent contentions when coding internally queued handlers which must manipulate the handler's queue link pointers (CQE and LQE).

4.2.2 .SYNCH and completion requests

TSX-Plus groups completion requests into three categories. Normal completion requests (such as I/O completion, .MRKT, etc.) are assigned the lowest class number one. Completion routines scheduled as a result of real-time interrupt are assigned class number two. Completion routines resulting from a device handler which issues a .SYNCH request are of class number three. A completion request of a higher class may interrupt a currently executing completion routine of a lower class. Thus, a .MRKT completion routine can be interrupted by a real-time completion routine which can be interrupted by a .SYNCH completion routine. Completion routines cannot be interrupted by completion routines of the same or lower class but are queued serially by class number.

4.2.3 .TIMIO and .CTIMIO requests

Under TSX-Plus it is not necessary for a handler to go to fork level before issuing .TIMIO and .CTIMIO requests. If a job number is placed in the timer control block used with a .TIMIO request, the handler will be synchronized with the specified job number when the timeout routine is entered. If a zero job number is specified in the timer control block, the handler timeout routine will be running at fork level but not synchronized with any job if an I/O timeout occurs. See the RT-11 Software Support Manual for more information on the .TIMIO and .CTIMIO programmed requests.

4.3 Generating device handlers for use under TSX-Plus

TSX-Plus generally uses standard RT-11 XM device handlers, however, some handlers supplied with RT-11 require minor modifications to function correctly with TSX-Plus. The necessary handler modifications have already been applied and are included in the dd.TSX handlers supplied with TSX-Plus.

If you ordinarily need to make no modifications to the handlers supplied by Digital on your system, then you may use the handlers provided with the TSX-Plus distribution. Most common changes can be accommodated through device SET options. However, if you need to change the handlers supplied with RT-11,

you may need to apply some patches before using them. See the TSX-Plus Installation Guide for information concerning patching device handlers for use with TSX-Plus.

4.3.1 Building device handlers

When building device handlers, it is necessary to set certain switches before assembly which control conditional code exclusion and inclusion. TSX-Plus requires memory management and optionally allows device timeout. However, it does not support error logging, therefore, error logging should be excluded when the handlers are built.

The conditional file used to build the device handlers supplied on the TSX-Plus distribution media is named TSXCND.MAC (also present on the distribution media). It contains the following conditionals:

```
TSX$P   = 1             ;TSX-Plus support
BF      = 1             ;No SJ support
MMG$T   = 1             ;Memory management support
TIM$IT  = 1             ;Device time out
ERL$G   = 0             ;No Error log support

DXT$O   = 0             ;No Second RX11 controller support
DX$CSR  = 177170        ;Status register for first RX11
DX$VEC  = 264           ;Vector for first RX11
DY$DD   = 0             ;No RX02 double density only
DYT$O   = 0             ;No Second RX02 controller
DY$CSR  = 177170        ;Status register for first RX02
DY$VEC  = 264           ;Vector for first RX02
DDT$O   = 0             ;No Second DECtape II controller
DD$CSR  = 176500        ;Status register for first DECtape II
DD$VEC  = 300           ;Vector for first DECtape II
$RFNUM  = 1.            ;Number of RF11 platters
RJS0$3  = 1             ;RJS disk is RJS03
DL$UN   = 4.            ;Number of RL01/RL02 units
DU$POR  = 1             ;Number of MSCP ports
RP0$3   = 1             ;RP11 disk is RP03
MT$FSM  = 1             ;TM11 file-structured support
MT$UN   = 2.            ;Number of TM11 units
MM$FSM  = 1             ;TJU16 file-structured support
MM$UN   = 2.            ;Number of TJU16 units
MS$FSM  = 1             ;TS11 file structured support
MS$UN   = 1.            ;Number of TS11 units
MS$CSR  = 172522        ;Status register of first TS11
MS$VEC  = 224           ;Vector of first TS11
XL$CSR  = 176500        ;XL CSR
XL$VEC  = 300           ;XL Vector
LS$PC   = 0             ;No LS PC300 support
LS$CSR  = 176500        ;Serial LP CSR
LS$VEC  = 300           ;Serial LP CSR
```

Note that setting a conditional parameter to zero (0) disables the option and setting it to one (1) enables the option. Since device timeout support is always available with TSX-Plus, but is optionally supported by handlers, TIM$IT may be either 0 or 1. Other parameters may be included to specify device characteristics. Refer to Appendix C of the RT-11 System Generation Guide for an entire list, default value, and description of device conditionals. These parameters are not required and will use a default value if left unspecified, except TSX$P and MMG$T which must both be set to 1 for TSX-Plus.

Before building device handlers, the appropriate patches (provided as .SLP files on the distribution medium) must be applied. See the TSX-Plus Installation Guide for information on applying the patch files. The distribution medium contains the .SLP files for all device handlers which require modification for use with TSX-Plus. Most handlers may be built by the following commands:

```
MACRO  TSXCND+dd/OBJ
LINK/EXE:SY:dd.TSX  dd
```

where "dd" represents the two character device name.

Only the file structured magtape handlers require different commands. They may be built by using the following commands:

```
MACRO  TSXCND+FSM/OBJ
MACRO  TSXCND+td/OBJ
LINK/EXE:SY:dd.TSX  td,FSM
```

where "td" represents the tape device source module name (TJ, TS, or TM) and "dd" represents the corresponding magtape device name (MM, MS, or MT). Notice that the LINK command automatically appends the "TSX" file extension. Since TSX-Plus uses handlers with the extension "TSX", the handlers must be linked with that extension rather than with the extension "SYS". This allows the TSX-Plus handlers to coexist on the same system disk with standard RT-11 handlers without conflict. Handlers for all devices included in your TSGEN DEVDEF list, including the system disk, must be on the system disk when TSX-Plus is started.

4.3.2 Defining device handler attributes

For each device to be available to the system an entry must be made in TSGEN using the DEVDEF macro. (Note that CL, LD, TT, and SL have other generation parameters and must never be included in a DEVDEF declaration.) This entry requires optional parameters which specify the characteristics of the device handler. Based on these characteristics, TSX-Plus can determine any special operating considerations. The standard device drivers distributed with TSX-Plus have predetermined flag settings known by the TSX-Plus start-up program. Therefore, it is not necessary to specify flag options when using the device handlers distributed with TSX-Plus. In cases where non-standard handlers are installed, it is necessary to choose the correct device attributes to insure correct operation.

The nine optional device parameters control the following operation:

DMA — Device performs Direct Memory Access (DMA).
In UNIBUS systems with more than 256 K bytes of memory, TSX-Plus allocates and controls UMRs (Unibus Mapping Registers) to perform I/O requests for a DMA device.

MAPIO — Perform I/O mapping.
In QBUS systems with more than 256 K bytes of memory, TSX-Plus must buffer the I/O request into an 18-bit addressable memory region and move the information into the user's area when the users job is above an 18-bit memory address. Requests are not buffered if the job is below the 18-bit low memory region (256 K bytes).

EVNBUF — Require even byte buffer address for I/O transfers.
Some device controllers (DMA devices) and device handlers (VM) which implement a word transfer (rather than byte), require the buffer address to begin on a even byte address (word aligned). In these cases, odd byte addresses may cause I/O failure or fatal system errors which could halt system execution. TSX-Plus will check the buffer address to insure that the transfer is word aligned. If the I/O request does not begin on a word boundary, a user error will be returned from the EMT request.

NOCACHE — Do not use generalized data cache for this device.
For certain devices, it is desirable to disable generalized data cache. For example, since the VM handler uses memory as a device, it would be wasteful of machine resources to also allow it to utilize generalized data cache. This would not only result in displacement of information contained within the cache but would also have the additional overhead of a useless memory to memory transfer.

NOMOUNT — Do not allow mounts for this device.
If this option is specified, the physical device cannot be mounted and therefore will not use directory caching.

REQALC — Require device allocation before use.
If this option is specified, access to the device units is only allowed to users who have allocated the device by use of the ALLOCATE command.

MAPH — Load the device handler into a mapped handler region.
TSX-Plus will place device handlers within an extended memory region, reducing the size of the low memory kernel region (restricted to 40 K bytes). Handlers which are placed in extended memory are known as "mapped" handlers. TSX-Plus communicates with mapped device handlers by mapping PAR5 to the handler's extended memory base address. As device handlers are loaded, the interrupt entry point is intercepted and directed to a low memory address which will map to the handler then enter the handler's interrupt entry code.

Handlers may be mapped under the following conditions:

1. Since only one PAR register is used to access the device handler it must not be larger than 8 K bytes.

2. Since handlers are accessed by kernel PAR 5, the handler must not use kernel PAR 5.

3. Since only two device interrupt vectors per handler are redirected, the handler may not connect to more than two device interrupt vectors. In addition, since the redirection is performed during initialization, the handler may not dynamically connect to interrupt vectors.

4. When the device handler contains an internal buffer used for DMA access, it must calculate the correct physical address taking into account it's own mapped address. It must also declare the HANBUF option which will not allow it to be mapped on extended UNIBUS configuration or when MAPIO is also specified. See the HANBUF option for more information concerning this restriction.

NOMAPH — Always load the handler into the low memory 40k byte region.
Some device handlers are not eligible for mapping into extended memory regions and TSX-Plus will place them in the low memory kernel region. The NOMAPH option will take precedence over the MAPH option if both are specified.

HANBUF — Handler contains an internal I/O buffer used for DMA transfers.
Handlers with internal DMA buffers require special coding to be used as a mapped device handler. In addition, when TSX-Plus is evaluating the system definitions and device characteristics for loading device handlers, it will never map a handler which uses an internal buffer if the handler also requires mapped I/O transfers in QBUS systems with more than 256 K bytes of memory (MAPIO) or if the handler resides in a UNIBUS system with more than 256 K bytes of memory.

The following subroutine illustrates how a handler can translate the virtual address of an internal I/O buffer into the correct physical address. This subroutine will function correctly under TSX-Plus whether or not the handler is mapped:

```
;  Calculate a 22-bit buffer address from the
;  virtual address of a buffer which is contained
;  within the handler.
;  This is necessary if the handler is mapped.
;
;  Input:
;     OLDBA - Virtual address of the internal buffer
;     EXTADR - Zero
;
;  Output:
;      OLDBA - Low order (16-bits) of the physical address
;      EXTADR - High order (6-bits) of the physical address
;      All registers are preserved
;
MAPADR: CMP     OLDBA,#120000   ;Handler loaded into a mapped region?
        BLO     10$             ;Br if handler is below mapped region
        MOV     R0,-(SP)        ;Save registers
        MOV     R1,-(SP)        ;
        MOV     @#172352,R1     ;Get the handler PAR5 relocation value
        CLR     R0              ;Clear high order address cell
        ASHC    #6,R0           ;Convert to physical memory address
        BIC     #160000,OLDBA   ;Isolate offset (clear virtual PAR #)
        ADD     R1,OLDBA        ;Add the phys. relocation to low order
        ADC     R0              ;Add carry to high order
        ASH     #4,R0           ;Shift high order to bits 4 thru 9
        MOV     R0,EXTADR       ;Store high order address
        MOV     (SP)+,R1        ;Restore registers
        MOV     (SP)+,R0        ;
10$:    RETURN                  ;Return
```

4.4 Debugging a device handler

A special version of the ODT debugging program is provided with TSX-Plus for use in debugging user written device handlers. In order to perform this type of debugging the following two files must be available on the system disk when TSX-Plus is started: TSXDB.SAV and SYSODT.REL.

In order to start TSX-Plus under control of the system debugging program, type

```
R TSXDB
```

This is analogous to the R TSX command that is normally used to start TSX-Plus but has the effect of loading the system debugging program into memory with TSX-Plus and transferring control to it before TSX-Plus performs its initialization.

The system debugging program requires approximately 2,900 bytes in the 40K byte low-memory portion of TSX-Plus. If your generated system is so large that there is insufficient space to start it with the debugging program, regenerate the system with all handlers removed except the ones that are needed for execution and reduce the number of time-sharing lines.

When you start the system by typing R TSXDB, it responds by printing an asterisk indicating the debugger is in control and waiting for a command. The debugger only accepts commands from the machine's console terminal. The commands take the same form as for standard ODT except some of the lesser used commands (such as searching through memory) have been removed to save space.

On entry to the debugger, register R1 ($1) contains the address of an instruction that is executed each time control-R is typed. If you set a breakpoint at this location you can trigger a breakpoint whenever you wish, after the system is started, by typing control-R at any active terminal.

Once the control-R breakpoint is set, start the system by typing ";G". After the system has started and you have logged in, determine the address of the base of the handler by use of the SHOW DEVICE keyboard command. The handler being debugged should be specified as non-mapped (i.e., do not load into extended memory) when the system is generated. As a result of this, the "P. base" (physical memory) base address will be zero, and the "V. base" address will be the actual address of the base of the handler. This is the address of the first cell in the handler header (block 1 of the handler) which contains the device vector address. Note, this address is the base of the handler not the handler entry point; the entry point is at base address + 12 (octal).

Once the base address of the handler is known, trigger a breakpoint by typing control-R, set a debugger relocation register at the base of the handler, set breakpoints within the handler, and proceed by typing ";P".

The following sequence of commands illustrates the process of starting TSX-Plus with the system debugger. The values shown for addresses within the system will vary depending on the options included when the system is generated.

```
.R TSXDB                ! Start system with debugger
ODT  V04.00             ! Debugger indicates it is in control
*$1/020504              ! Get address of control-R breakpoint
*020504;1B              ! Set a breakpoint at control-R
*;G                     ! Start the system
```

In the case where you are debugging a handler which fails before the system is started (such as the handler for the system device), the procedure for debugging is somewhat more complex. You still start the system under debugger control by typing "R TSXDB", but instead of relying on the control-R breakpoint (which requires that the system get started) set a breakpoint at the location whose address corresponds to the symbol INIJMP which can be found in the TSEXEC section of the system LINK map. The instruction at INIJMP is executed after

the portion of the system initialization that loads handlers but before the system handler is used. Once you have set a breakpoint at INIJMP, start execution by typing ";G".

Once the breakpoint at INIJMP is triggered, you must determine the base address of the handler by examining system tables. The table PNAME (in the TSGEN portion of the map) contains a one word entry for each device; the entry is the RAD50 name of the device. Examine each entry in this table, using the "X" debugger command to convert the octal value to RAD50, until you locate the entry for the handler to be debugged. The HANENT table (also in the TSGEN section of the map) is a table that is parallel to PNAME and contains the addresses of the handler entry points. Examine the entry in HANENT that has the same offset as the device name in PNAME. This is the entry point of the handler. Subtract 12 (octal) to obtain the address of the base of the handler. Breakpoints can then be set in the handler and execution continued by typing ";P".

4.5 Internally queued device handlers

Handlers that perform internal queuing must take special precautions to avoid possible contention from prioritized fork request processing. Handler initialiation code is procesing at fork priority 12 while all other code (interrupt, abort, timout) is running at fork priority 50. See the previous section in this chapter discussing fork prioritization. If necessary, a fork request of priority 50 may be issued from the handler's initiation section in order to serialize processing within the device handler. In certain circumstances, it may be desirable to alter FP$IOS (the I/O initiation priority) in TSGEN to be 50.

Handlers that perform internal queuing must set the error flag in the Channel Status Word (CSW) if, for any reason, they abort or fail to complete pending I/O requests. This is particularly important when aborting a pending I/O operation which is requested by the generalized data cache facility.

5. PROGRAMMING FOR SPECIAL DEVICE HANDLERS

5.1 Special TSX-Plus device handlers

Several device handlers are uniquely integrated into the TSX-Plus environment. The communication line handler (CL), the logical subset disk handler (LD), the terminal line handler (TT), and the single line editor (SL) are provided as integrated system features and do not require device declarations (DEVDEF). The professional interface handler (PI) is provided with PRO/TSX-Plus and when used is defined as a shared run-time system and not as a device handler. The TSX-Plus virtual memory handler (VM) utilizes knowledge of the TSX-Plus environment to determine the usable memory space available. VM is the only special TSX-Plus device handler which requires a device declaration (DEVDEF) in order to be used.

5.1.1 Communication line handler (CL)

The CL handler allows Input/Output operations to be performed to serial communication lines connected to DL11, DLV11, DZ11, DZV11, DH11, and DHV11 communication controllers. With the CL handler it is possible to have some lines on a multiplexer used as TSX-Plus time-sharing lines, and other lines on the same multiplexer used to drive I/O devices such as printers, plotters, and modems. It is also possible to use a line as a time-sharing line some of the time and as a communications line at other times. Some of the important features of the CL handler are summarized below:

1. Up to 16 communication lines may be controlled through the CL handler. The first 8 CL units are named CL0 through CL7, the second set of 8 units are named C10 through C17. The lines may be connected to any type of communication controller that is supported by TSX-Plus and may share the same multiplexer controllers as TSX-Plus time-sharing lines.

2. Lines may be dedicated as communication lines or may be switched between time-sharing lines and communication lines.

3. Internal queueing is used within the handler to allow concurrent input/output operations to be performed on all of the lines.

4. The CL handler allows both input (read) and output (write) operations. Full duplex (simultaneous) read and write operations may take place on each line.

5. The communication lines may be used with the TSX-Plus spooling system to allow spooled output to devices on communication lines.

6. The CL handler responds to XON/XOFF (control-Q/control-S) control characters to stop and start its transmission and will generate XON/XOFF characters to control the speed of a device transmitting to a CL line.

7. A "binary mode" is available for CL lines to allow full 8-bit, transparent I/O to devices.

8. Modem control is supported. Ring and carrier detect signals may be monitored and data terminal ready (DTR) can be controlled by a program or SET command.

9. The CL handler is implemented as a system virtual overlay, minimizing the amount of code and data that is required in the unmapped portion of the system.

10. The CL handler can be used as a replacement for the LS, XL, and XC handlers (the XL and XC handlers are used with the RT-11 VTCOM program).

11. A terminal can be "cross connected" to a CL line by use of the SET HOST command so that characters typed at the terminal are sent directly out the CL line and characters received on the CL line are displayed at the terminal.

Once a system has been generated with communication lines, the lines may be accessed as normal devices using the names CL0, CL1, C10, C17, etc. If the CL handler is used to drive the system printer, it is convenient to use an assign command to assign the logical name LP to the corresponding CL device.

The device name "CL" is functionally equivalent to "CL0". "C1" is equivalent to "C10". Attempts to use a CL unit which is not currently associated with a line will return an error status just as if the CL device was not recognized by the system.

5.1.1.1 I/O operations: The .READ/.READC/.READW and .WRITE/.WRITC/.WRITW EMT's may be used to perform standard read/write operations to the CL lines. The CL handler allows full duplex input/output operation, which means that read and write operations may be simultaneously active on a CL line.

When a .READ[C/W] EMT is used to read from a CL line, the operation is complete when the requested number of words have been accepted or a control-Z character is received.

The input character silo is used to store characters received from the line. This buffer prevents characters from being lost during the interval when one read EMT is completed and another is issued to the CL handler.

The CL handler responds to received XON (control-Q) and XOFF (control-S) characters, starting and suspending transmission to synchronize its character flow with the device connected to the line.

5.1.1.2 Control character processing: Processing of certain control characters through CL units depends on the individual character, the settings of the CL unit and the particular operation in progress.

The following table illustrates the handling of special input characters to a CL unit. Control characters not listed are treated as normal characters on input.

Char	Octal Value	Input handling
NUL	0	Discarded unless in binary input mode (BININ).
LF	12	Discarded unless in LFIN mode (always input by .SPFUN 203).
CR	15	Always input. Also terminates read for .SPFUN 260.
XON	21	Re-enables transmission to CL unit, except in BININ mode when it is input as a normal character.
XOFF	23	Halts transmission to CL unit, except in BININ mode when it is input as a normal character.
^Z	32	Sets end-of-file flag and terminates read (except for .SPFUN 203 by which ^Z is treated as a normal character).

The following table illustrates the handling of special output characters to a CL unit. In NOCTRL mode control characters not listed are not sent. In BINOUT mode, all special control character output processing is bypassed. In addition, in BINOUT mode automatic XOFF transmission when the input silo becomes full is disabled.

Char	Octal Value	Output handling
NUL	0	Never sent except in BINOUT mode.
TAB	11	Expanded to spaces in NOTAB mode. In TAB mode and width not set to 0, TAB is discarded if it would exceed the set width, just as normal characters are.
LF	12	Discarded in NOLFOUT mode if preceding character was a carriage return.
FF	14	In NOFORM mode, FF is expanded to enough line feeds to advance to the top of the next page.
CR	15	Discarded in NOCR mode.

5.1.1.3 .SPFUN operations: The following special function codes (.SPFUN EMT's) are recognized by the CL handler. The special functions apply to the specific CL unit to which the channel was opened.

Code	Function
201	Clear handler
202	Control break transmission
203	Read with byte count
204	Get handler status
205	Terminate I/O
250	Set option flags
251	Clear option flags
252	Set page length
253	Set skip lines
254	Set page width
255	Get modem status
256	Set line speed
257	Abort pending I/O
260	Read a line of input
261	Get number of input characters pending
262	Get number of output characters pending
263	Write with byte count
264	Set ENDPAGE and ENDSTRING parameters

Function 201 -- Clear handler. This function clears the internal handler flag that says an XOFF (control-S) character has been received and transmits an XON (control-Q) to the CL device.

Function 202 -- Control break transmission. This function starts or stops the transmission of a break signal. The word count specified with the .SPFUN controls whether transmission of a break signal is started or stopped. If the word count is non-zero, break transmission is started; break transmission continues until another .SPFUN is done with function code 202 and a word count of zero.

Function 203 -- Read with byte count. This function performs a read operation but the "word count" value specifies a byte count instead. This function does not complete until at least one byte is read. However, if a byte count greater than one is specified, bytes are moved from the input ring buffer until either the specified byte count is satisfied or the input ring buffer is emptied. If fewer than the requested number of bytes are available, the remainder of the buffer is filled with nulls. The control-Z character does not signal end of file for this type of read -- control-Z is read as an ordinary character.

Function 204 -- Get handler status. A status code is stored into the first word of the buffer specified with this function. The meaning of the flag bits is as specified below:

Bit 0: 1 ==> XOFF has been sent to stop transmission.
Bit 1: 1 ==> XOFF has been received from the remote device.
Bit 2: 1 ==> Carrier has been detected

Function 205 -- Terminate I/O to the line. This function "turns off" a communication line. The input ring buffer is emptied (its contents are discarded) and a flag is set causing any other characters received from the line to be discarded. Data Terminal Ready (DTR) status is dropped. The line will be turned on again whenever another I/O operation is performed to it.

Functions 250 and 251 -- Control option flags. These special functions are used to set and clear handler option flags. Function 250 sets the specified flag bits, function 251 clears the specified flag bits. The flag bits correspond to handler SET options. If the option flag is cleared (0), this corresponds to the NOoption setting. The bit positions of the options are shown in the following table. The option flags are contained in a one word buffer for the .SPFUN. For a detailed description of the SET commands, see the TSX-Plus Reference Manual.

Bit	Mask	Option	Function summary
0	000001	FORM	Send form feed characters
1	000002	TAB	Send tab characters
2	000004	LC	Send lower case characters
3	000010	LFOUT	Send line feed characters
4	000020	LFIN	Accept line feed characters
5	000040	FORM0	Send form feed on block 0 write
6	000100	BINOUT	Send binary output characters
7	000200	BININ	Accept binary input characters
8	000400	CR	Send carriage return characters
9	001000	CTRL	Send control characters
10	002000	DTR	Raise Data Terminal Ready (DTR)
11	004000	EIGHTBIT	Accept and send 8 bit characters

Function 252 -- Set page length. This function performs the operation of the SET CL LENGTH=n command. The .SPFUN must have a one word buffer containing the number of lines per page.

Function 253 -- Set skip lines. This function performs the operation of the SET CL SKIP=n command. The .SPFUN must have a one word buffer containing the number of lines to skip at the bottom of the page.

Function 254 -- Set page width. This function performs the operation of the SET CL WIDTH=n command. The .SPFUN must have a one word buffer containing the line width.

Function 255 -- Get modem status. This function is used to check on the status of a modem connected to a CL line. The modem status is returned into the first word of the buffer specified with the .SPFUN. The flag bits returned are described below:

Bit	Meaning when set
0	Ring indication
1	Carrier is detected
2	Data Terminal Ready is asserted

Function 256 -- Set line speed, character length and parity control. This function is used to set the transmit/receive speed for a CL line. This .SPFUN requires a one word buffer containing a value which has the following form (in binary):

OPLxSSSS

The low-order 4 bits ("SSSS") specify the speed. The following baud rates are represented by the indicated speed codes (speed code values are shown in decimal): 50=0, 75=1, 110=2, 134.5=3, 150=4, 300=5, 600=6, 1200=7, 1800=8, 2000=9, 2400=10, 3600=11, 4800=12, 7200=13, 9600=14, 19200=15. Bit 5 ("L") specifies the character length. If this bit is 0, the character length is 8

bits; if this bit is 1, the character length is 7 bits. Bit 6 ("P") specifies parity control selection. If this bit is 0, parity is disabled and bit 7 is ignored; if this bit is 1, parity generation and checking is enabled. Bit 7 ("O") selects even or odd parity and is only meaningful if bit 6 is 1 (enable parity). If bit 7 is 0, even parity is selected; if bit 7 is 1, odd parity is selected. Note that if only the speed value is specified, with all other bits zero, 8 bit characters with no parity are selected.

This .SPFUN can only be used for lines connected to hardware controllers that support programmable baud rates such as DLV11E, DZ11, DZV11, DH11, DHV11, and the Professional printer and communication ports. A baud rate of 19200 is not supported by some hardware controllers including the DEC DZ11 controller (although it actually works with most DEC DZ11 controllers). The DH11 does not support baud rates of 2000, 3600, or 7200; and the DHV11 does not support baud rates of 3600 or 7200.

Function 257 -- Abort pending I/O. Abort all pending read and write operations issued by the job executing the .SPFUN on the CL unit.

Function 260 -- Read a line of input. This special function reads a line of input terminated by a carriage return character. The "word count" value specified with this special function is interpreted as a byte count. The read terminates when any of the following conditions is met:

1. A carriage return character is received. The carriage return is stored in the buffer and the remainder of the buffer is null filled.

2. The buffer is filled before a carriage return is received.

3. A control-Z is received.

Function 261 -- Determine number of input characters pending. One word is returned into the user buffer, containing the number of characters available to be read from the CL unit associated with the specified channel. This function can be used to test for pending input prior to issuing a read (.READx or .SPFUN) on the channel. If no characters are pending, attempts to read from the channel will not complete until the word (or byte) count is fulfilled. If you do not wish to have a read request pending until the word (or byte) count is fulfilled, then first determine the number of characters pending in the input buffer. If there are none, then do not issue the read.

Function 262 -- Determine the number of output characters pending. One word is returned into the user buffer, containing the number of characters in the output buffer which have not yet been transmitted.

Function 263 -- Write with byte count. This special function is used to write a block of characters to a CL line with the length of the block specified by a byte count rather than the word count used with the .WRITE EMT. This is useful in situations where an odd number of bytes must be written and null characters cannot be used to pad out the last word. The .SPFUN follows the standard form

except a byte count is specified for the fifth parameter (which is normally used to specify a word count).

Function 264 -- Set ENDPAGE and ENDSTRING parameters. This special function is used to allow a running program to specify the number of form-feed characters (ENDPAGE) and a seven character string (ENDSTRING) which will be appended to the end of each output file. The buffer address must point to a word aligned storage area of which the first word contains the number of form-feed characters. The second and subsequent words contain the string in ASCIZ form to append to the end; any characters beyond the first seven are ignored.

5.1.1.4 Redirecting CL and time-sharing lines: A system service call (EMT) is available to allow a program to assign a CL unit to a particular line. The form of the EMT is:

```
EMT     375
```

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

```
.BYTE   0,155
.WORD   CL_unit
.WORD   line_number
```

where "CL_unit" is the CL unit number, and "line_number" is the number of a TSX-Plus time-sharing line or dedicated CL line. If the specified line number is 0 (zero), the CL unit is disassociated from any line. TERMINAL privilege is required to use this EMT.

If an error is detected, the C-flag is set on return and the following error codes are returned:

Code	Meaning
1	User issuing the EMT does not have TERMINAL privilege
2	An invalid CL unit number was specified
3	An invalid line number was specified
4	The specified line is already assigned to a CL unit
5	A time-sharing user is logged onto the specified line
6	The specified CL unit is currently busy

CL units specified using the CLDEF macro in TSGEN are initially connected to dedicated CL lines. Note that although these lines are dedicated for use by CL, the CL units which are initially assigned to these lines may be reassigned to other lines. The unallocated CL units declared by use of the CLXTRA parameter in TSGEN are initially not associated with any line. The SET CLn LINE=n and SET HOST/PORT=CLn keyboard command or the system service call (EMT) can be used to assign any CL unit to any free time-sharing line or free dedicated CL line. Thus it is possible to use a line as a TSX-Plus time-sharing line during certain portions of the day and then assign a CL unit to the line and use it to drive a modem or other device during other portions of

the day. Dedicated CL lines use less memory space than time-sharing lines but may only be accessed as CL units. See the TSX-Plus Reference Manual for a full description of the SET CL and SET HOST command. TERMINAL privilege is required to use these SET commands.

The following example commands illustrate how CL unit 1 can be assigned to time-sharing line 2. The logical name "LP" is then assigned to CL1 so that the PRINT command will direct output through CL1. CL1 can be declared to be a spooled device in TSGEN:

```
.SET CL1 LINE=2
.ASSIGN CL1 LP
```

The SHOW CL and SHOW TERMINALS keyboard commands can be used to display information about which CL units are associated with which lines. The SHOW CL command also indicates if a CL unit is spooled and lists the options which are set for the unit. See the TSX-Plus Reference Manual for information concerning the SHOW command.

5.1.1.5 VTCOM/TRANSF support and CL handler: The RT-11 VTCOM/TRANSF file transfer programs may be used to communicate and transfer files between RT-11 and/or TSX-Plus systems.

When VTCOM is used to communicate with another system, the system where the user is located and running VTCOM is known as the "local" system whereas the remote system to which communication is taking place is known as the "host" system. TSX-Plus may be used either as the local system, the host system, or both.

The user at the local system runs the VTCOM program to initiate communication with the host system. The VTCOM program uses the CL handler to connect to a communications line. The CL handler must be set up to drive a DL11, DLV11, DZ11, DZV11, DH11, DHV11, or Professional printer or communication port that is connected either directly or through a modem to the host system.

When TSX-Plus is used as the local system, the IOABT sysgen parameter must be set to 1 to enable handler abort entry code.

If the CL handler is used with the VTCOM program, it is necessary to assign the logical name XL (or XC if on the Professional) to the CL (or C1) unit controlling the communications line. It is also a good idea to allocate the device so that conflicts with other users will not occur. The NOLFOUT option should be specified for a CL line used with VTCOM. For example, the following commands would be appropriate to direct VTCOM to use line CL0:

```
.SET CL0 NOLFOUT
.ASSIGN CL0 XL (or XC)
.ALLOCATE XL (or XC)
```

When TSX-Plus is used as the host system, the connection from the local system may be made through any TSX-Plus time-sharing line on the host system.

5.1.1.6 Terminal/Communication line cross connection: It is possible to cross connect a time-sharing line with a CL (communication line) line in such a fashion that all characters received from the time-sharing line are transmitted directly to the CL line and all characters received from the CL line are transmitted directly to the time-sharing line. This is useful to allow a time-sharing line on one TSX-Plus system to be used as a terminal on another system connected through a CL line.

This function is similar to using VTCOM to communicate through a CL line but has the advantage that there is much less overhead because the cross connection is made at a low level within TSX-Plus such that characters do not have to be passed to a running application program. Of course the internal cross-connection feature does not provide the file transfer capabilities of VTCOM.

The keyboard command used to establish a cross connection has the form:

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

where "ddn" is the name of a CL or Cl device to which your terminal is to be cross connected. For example, the following commands would connect CL unit 1 with terminal line 4 at 9600 baud and then cross connect the current terminal with the CL unit:

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

TERMINAL privilege is required to use the SET HOST command. Once the cross connection is established, characters typed at your terminal are transmitted to the CL line.

5.1.2 RK06/RK07 handler (DM)
The DM device handler .SPFUN function codes 376 and 377 attempt to return a status code into the first word of a user buffer which is one word longer than the actual transfer size. This is incompatible with system I/O mapping. It appears that the only system utility program which issues these functions is DUP (SQUEEZE and INITIALIZE commands). If it is necessary to use the MAPIO option with the DM device handler, it is recommended that both INITIALIZE and SQUEEZE commands for DM units be issued only under RT-11.

The DM handler supports 22-bit (as well as 18-bit) QBUS I/O with the DILOG DQ215 and the Emulex SC02C controllers.

5.1.3 IEEE GPIB handler (IB)

The normal IB supplied subroutines attempt to open the IB device on decimal channel numbers 16, 17, 18, and 19. TSX-Plus normally allocates 20 (decimal) channels and allows these IB subroutines to execute without changes. Non-standard configurations where multiple devices may be used and more than 20 (decimal) channels are required are not supported.

A change is also necessary to the IB device handler to alter the mapping register used from PAR1 to PAR6. See the section concerning device handlers use of PARs discussed earlier in the chapter on Device Handlers. See the Patching and Building TSX-Plus Device Handlers chapter in the TSX-Plus Installation Guide for information on how build an IB handler which will function with TSX-Plus.

The IB subroutine "IBSRQ" is implemented in the DEC IB handler as a subroutine call from the handler directly to the user code region. Since TSX-Plus does not load any user job in the same map region as the operating system, the call will execute part of the operating system usually resulting in a fatal system error or halt. Therefore, the "IBSRQ" call is unsupported in TSX-Plus.

5.1.4 Virtual memory handler (VM)

The virtual memory handler (VM) allows memory which is not allocated for use by the operating system to be used as a RAM based pseudo-disk device. VM may not be used to contain either the swap or spool system files due to the nature of system completion routine nesting. When VM is used as a spool or swap device, unpredictable operation may occur resulting in fatal system errors and system halts.

The VM handler uses the memory space above the top of memory used by TSX-Plus. TSX-Plus can be limited to using less than all installed memory by specifying the TSGEN MEMSIZ parameter. (See the TSX-Plus Installation Manual for details on the MEMSIZ setting.) Since a memory access is quite a bit faster than a disk access, VM can be use for greater speed in locating and reading files which are frequently accessed.

Since most machines will lose the contents of memory during a power outage, VM should be restricted to read-only, scratch, or executable files. It may be used to speed the execution of heavily overlaid programs or store temporary intermediate sort or work files.

After TSX-Plus is started, VM must be initialized before it can be used. Since VM is implemented as a block structured device, and each block contains 512 bytes, the number of blocks available to VM will be two times the number of K bytes allocated. The directory does require some storage and therefore the number of blocks reported after initialization will be slightly smaller than this total. For instance, in a system which contains 512K bytes total physical memory and with MEMSIZ=256., VM will have 256K bytes available. After initialization, a directory of VM will then show slightly less than 512 blocks.

VM will normally calculate the correct base address to use to be just above the last address used by TSX-Plus. You may increase this base address. The format of the SET command used to adjust the base address used by VM is:

```
SET VM BASE=nnnnnn
```

where "nnnnnn" represents bits 6 through 22 of the base memory address (in octal) which VM is allowed to use. However, if you specify a base address below the top address of TSX-Plus, VM will dynamically adjust this base address back above the top of TSX-Plus. For example, if you wish to set the base address of VM to start after the first 512K bytes, then "nnnnnn" should be 20000 since the memory address is 2000000 (octal). Any time a new base address is defined, VM should be initialized.

VM will normally calculate the correct top address to use to be at the absolute top of physical memory. You may decrease this top address. The format of the SET command used to adjust the top address used by VM is:

```
SET VM TOP=nnnnnn
```

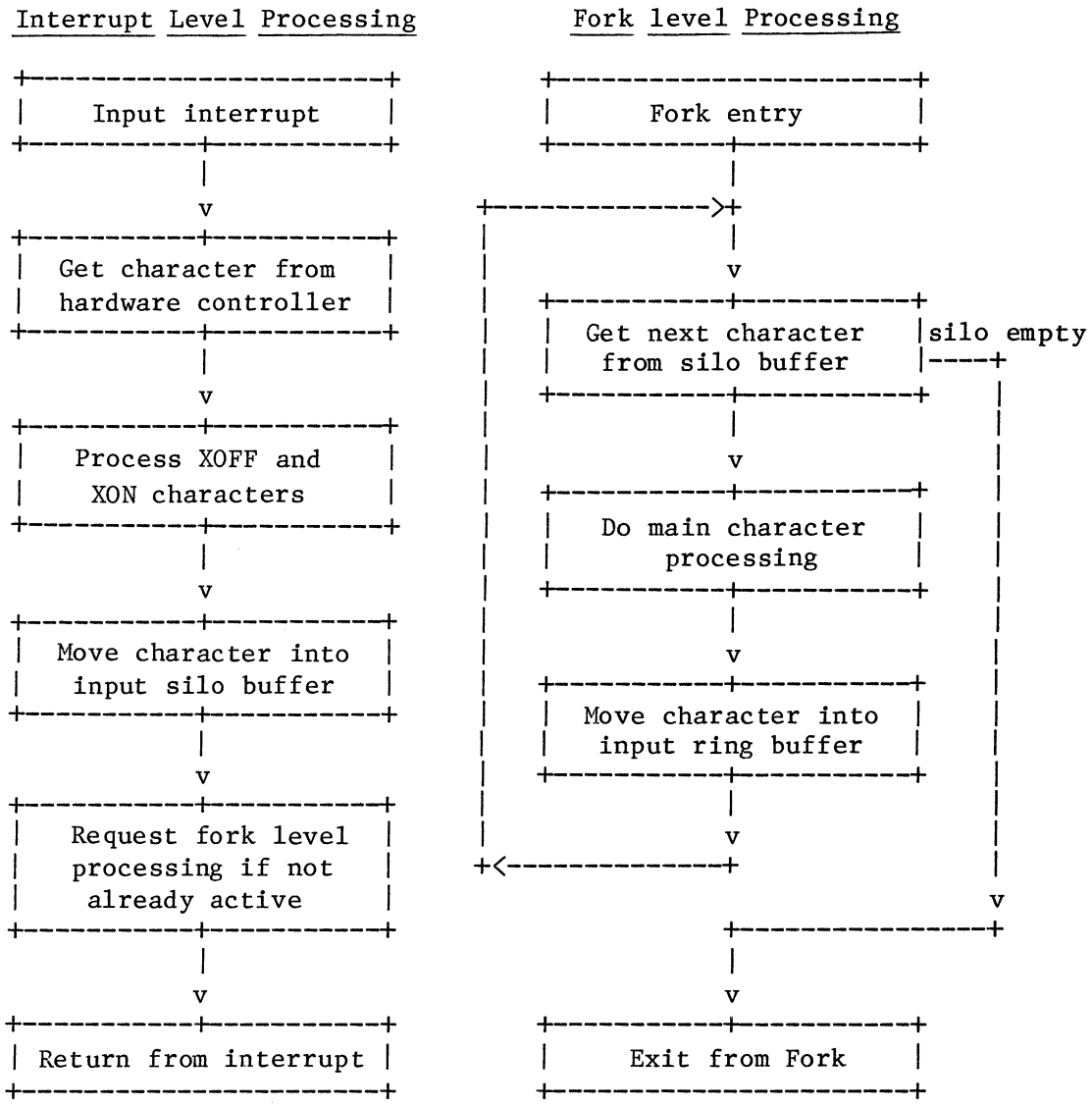
where "nnnnnn" represents bits 6 through 22 of the top memory address (in octal) which VM is allowed to use. For example, if you wish to set the top address of VM to end at 1280K bytes, then "nnnnnn" should be 50000 since the memory address is 5000000 (octal). Any time a new top address is defined, VM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

6. TERMINAL AND CL INPUT/OUTPUT PROCESSING

6.1 Terminal input character processing

To achieve maximum efficiency, TSX-Plus divides the terminal character input processing into three sections. The first section performs minimal character processing and runs at device interrupt priority level. The second section performs more lengthy character processing and runs at fork level. The third level passes characters to requesting programs and runs at program level. This system was chosen to minimize both system overhead and the time spent running at interrupt level. The interrupt and fork level character processing routines are illustrated in the following diagram:

Terminal Input Character Processing



6.1.1 Interrupt level input character processing

In order to minimize system overhead and the length of time running at interrupt level, only a small amount of character processing takes place in the input character interrupt routine. When an input interrupt occurs, the input interrupt routine is entered at device interrupt priority level. The received character is checked to see if it is an XOFF (control-S). If so, a flag is set causing character output on the line to be suspended. If the terminal controller is a DMA device such as a DH-11 or DHV-11 the current DMA transfer is aborted and information is saved to allow it to be restarted later. Next the character is checked to see if it is an XON (control-Q). If so, the output-suspension flag is cleared for the line and transmission to the line is restarted.

If the received character is other than XOFF or XON, it is stored into a holding buffer known as the input character "silo". It is called a silo because it functions as a first-in-first-out holding buffer. There is a separate character silo for each line. The default silo size is set by the TSGEN parameter NCSILO and the size of a silo for a particular line may be controlled by the SILO macro within a TSGEN line definition block. The maximum silo size allocated is 255; specification of silo sizes greater than this are reduced to be 255. If the number of free character positions available in the silo is reduced to a value equal to the TSGEN parameter NCXOFF (or specified by the SILO macro), an XOFF character is transmitted. If the silo overflows, the input character is discarded. When an XOFF character is transmitted due to the silo becoming nearly full, a flag is set which causes an XON character to be sent when the number of characters remaining in the silo decreases to a value equal to the TSGEN parameter NCXON (or specified by the SILO macro).

The process of getting a character from the hardware controller, checking it, and storing it in the input silo, is repeated until all pending characters have been accepted from devices such as DZ(V)-11 and DH(V)-11 which have hardware silo buffers.

After all pending characters have been processed and moved to the input silo, a check is made to see if fork level input character processing is active due to a previous input interrupt. If fork level processing is not active, a flag is set saying fork level processing is active and then a .FORK is done and fork level input character processing takes place. Once the fork level processing routine has been entered, the interrupt priority level is set to 0 (zero) and further input interrupts can occur. If additional interrupts occur while the fork level routine is running, they move characters into the input silos but do not reenter the fork level processing routine until it has finished processing all of the characters in the input silos.

The most critical parameter related to the input silo buffers is the one which controls how nearly full the silo is allowed to become before an XOFF character is transmitted. This parameter must be large enough to allow time for the XOFF character to be transmitted by the TSX-Plus system, and received and processed by the remote system. The minimum acceptable value depends on the speed of transmission and the responsiveness of the remote system. The recommended value is 12.

The silo parameter that controls when an XON character is transmitted is not critical. The recommended value is 4.

The total silo buffer size should be large enough to allow some room between the XOFF and XON points. If the character input rate is slow (for example from a terminal being used by a typist) the buffer size can be as small as a few characters plus the XOFF and XON parameter values. However, if the input is being received from another computer which can send high speed bursts of characters, then the buffer size should be increased to avoid rapid transmission of XOFF/XON character pairs. The ideal size of the silo buffer for this type of application is equal to the received packet size plus the XOFF cutoff parameter value. The silo buffers occupy space in the 40Kb low memory portion of TSX-Plus so they should not be made excessively large.

6.1.2 Fork level input character processing

The work performed on each character in the fork level routine is much more extensive than that done at interrupt level. The major tasks performed in the fork level routine are:

1. Processing of control characters such as control-C, control-W, control-U, control-R, and delete.

2. Checking for activation characters (such as carriage return) and restarting a program waiting for terminal input when one is received.

3. Echoing characters (including updating the window contents if process windowing is turned on).

4. Checking for field width activation and field width limits.

5. Storing the character into the appropriate input ring buffer.

In addition to the input silo buffer, each line has an "input ring buffer" which is used by the fork level routine to hold characters until they are accepted by the program. The default size of the input ring buffer is set by the DINSPC sysgen parameter but it may be controlled on a line-by-line basis by use of the BUFSIZ macro in TSGEN. If the number of free character positions in the input ring buffer is reduced to 8, a flag is set for the line preventing further characters from being moved out of the silo buffer. Thus characters will accumulate in the silo buffer until it is nearly full at which time an XOFF character is transmitted.

The fork level routine executes until all pending characters in silo buffers for all lines have been processed.

6.1.3 Program level input character processing

The third level of character processing takes place at program execution level. The primary purpose of this routine is to move characters from the input ring buffer to the user program as they are requested by system service calls such as .TTYIN, .GTLIN, and .READ. If the single line editor (SL) facility is active, its processing is performed within this routine. This routine is also responsible for suspending the execution of a job which requests terminal input when no more activation characters have been received.

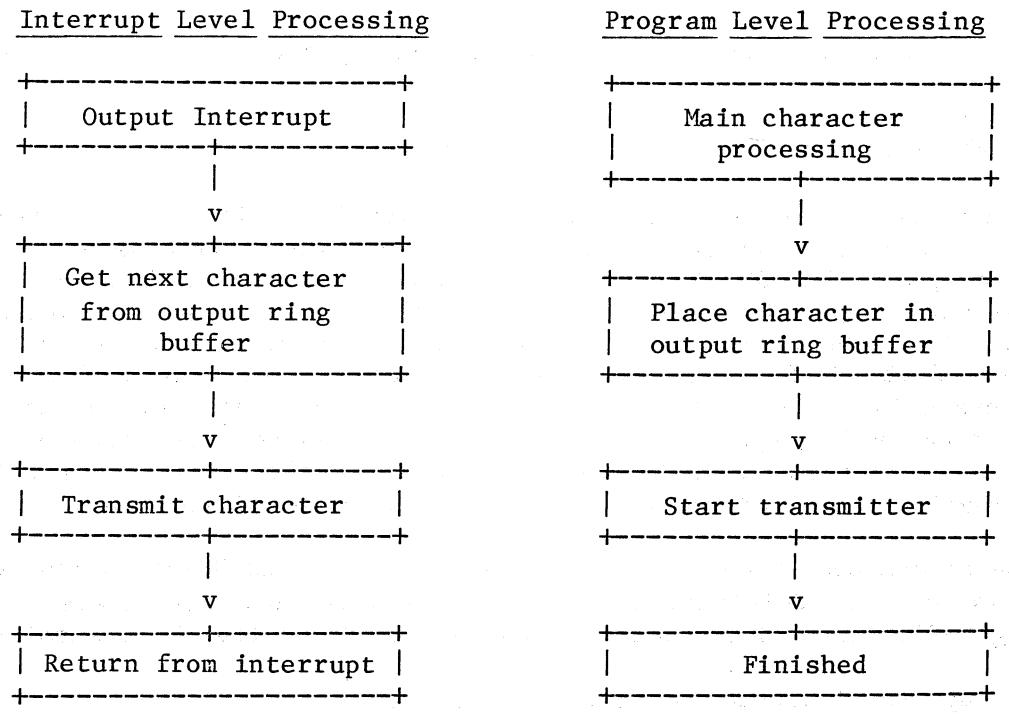
6.2 CL input character processing

Input character processing for CL lines is organized in a fashion similar to that used for time-sharing lines. The same input interrupt routine is used for time-sharing lines and CL lines and the same silo buffers are used to store characters as they are received. However, different routines are called to perform fork level processing. Fork level processing for CL lines is simpler than for time-sharing lines since there are fewer significant control characters. Also, no input ring buffers are used for CL lines. Instead, the fork level routine moves characters from the silo buffer into the data buffer specified for the .READ issued to the CL device.

6.3 Terminal output character processing

As with terminal input character processing, terminal output character processing is separated into levels. However, there are only two levels for character output processing: program level and interrupt level. The diagram below illustrates the output character processing routines:

Terminal Output Character Processing



6.3.1 Program level output character processing

Almost all of the processing done on characters being transmitted is performed at program level. The major character processing operations are summarized below:

1. Check for TSX-Plus terminal control program operations such as defining a new activation character, setting field width, etc. (Lead-in character followed by function code.)

2. Optionally perform terminal logging.

3. Optionally expand tabs into spaces.

4. Optionally convert form feed characters into 8 line feeds.

5. If process windowing is turned on, update the contents of the current window.

6. Place character in output ring buffer.

As each character is processed by the program level routine, it is checked to see if it is a control character which requires special processing, such as tab, form feed, and the TSX-Plus lead-in character. Characters are then stored into the output ring buffer for the line. The default size of output ring buffers is set by the DOTSPC sysgen parameter or may be set for an individual line by use of the BUFSIZ macro in TSGEN. If the output ring buffer becomes full, the execution of the job is suspended until the number of characters in the output ring buffer equals the OTRASZ sysgen parameter, at which point the execution of the job is resumed. As characters are placed in the output ring buffer, a routine is called to try to start transmission to the line.

In the case of communication controllers such as DH-11 and DHV-11 which support DMA transmission, an additional routine is called to move characters from the output ring buffer into linear buffers that are used for the DMA transmission.

6.3.2 Interrupt level output character processing

Since most of the character processing is done at program level, the output interrupt level routine is very simple. If the output ring buffer is not empty, the interrupt routine removes the next character from the output ring buffer and transmits that character. It also checks an output-suspended flag which is set when an XOFF character is received.

6.4 CL output character processing

Output character processing for CL lines is similar to that for time-sharing lines. The primary difference is that a separate output ring buffer is used for CL lines than for time-sharing lines. The default size for the CL output ring buffers may be set by use of the CLORSZ sysgen parameter or by use of the BUFSIZ macro in TSGEN. The recommended size for output ring buffers is ((3*baud_rate)/1000+3). For example, the recommended ring buffer size for a 9600 baud line is ((3*9600)/1000+3) or 32.

6.5 Terminal and modem protocols

TSX-Plus provides full support for dial-up lines connected to modems. A line is declared to be a dial-up line by use of the "$PHONE" flag in the line definition block in TSGEN. The SET TT n PHONE keyboard command may also be used. A line which is declared to be a phone line may also be used with a directly connected terminal.

When experiencing difficulties with terminals or modems, it is important to keep in mind that several levels of hardware and software are involved. These include at least: terminal hardware, terminal firmware, cable(s) from terminal to interface card, interface card hardware, interface card firmware, computer hardware, and system software.

6.5.1 Interface cards

Under RT-11 the console (operator's terminal) must be connected through a DL(V)11 type interface. A serial printer using the LS handler must also use a DL(V)11 type interface. TSX-Plus supports several types of asynchronous interface protocols to which you may attach terminals, modems, serial printers, or other serial devices. These include cards such as: DL11 and DLV11, which may support one or more lines and modem control, depending on the version; DZ11 and DZV11 multiplexers, which support 4 to 8 lines, depending on version, and include modem control; DHV11 multiplexers which support 8 lines and include modem control; and DH11 multiplexers which support 16 lines with optional modem control. The DH11 and DHV11 multiplexers utilize DMA (direct memory access) to improve terminal output efficiency. Note also that the programming protocols for DH11 and DHV11 multiplexers are distinctly different - they must be declared correctly during system generation.

Non-DEC vendors also supply interface cards which implement most of these protocols, but they may vary in implementation of such features as number of lines supported or optional modem control. The type of interface selected will vary according to which features you need, number and type of peripherals to be connected and expense. The DL11 and DLV11 type cards are in most cases not programmable for features such as speed and parity control, instead being jumper or switch selectable on the interface card when it is installed. In addition, they utilize separate interrupt vectors for each individual line.

When configuring a system with many terminals, it is difficult to configure the cards without conflicting with other devices. Remember that no two devices can share the same interrupt vectors or CSR addresses. Interface card selection is also influenced by the relative power consumption of the boards. For example the 4-line DLV11J only draws 1 A from the 5V supply, whereas the 8-line DHV11 draws 4.3 A at 5V (as listed in the Digital Networks and Communications Buyer's Guide). The consequences of inadequate or marginal power can lead to errors which are extremely difficult to reproduce and to diagnose.

The DZ11, DZV11, DH11 and DHV11 type interfaces permit software selection of features such as speed and parity control. They support multiple terminals through the same vector and CSR. This enhances the flexibility of configu-

ration and use when adding or changing peripherals. However, most serial device handlers such as LS or specially written handlers support only the DL11 and DLV11 type interfaces.

TSX-Plus includes a special device handler, CL, which supports up to 16 lines as serial I/O devices. The CL handler can use any mixture of the interface types listed above and may be used to connect serial printers through multiplexers. The CL handler can replace the standard LS and XL (and XC) device handlers for most purposes. But if you have special requirements like filler characters or flow control other than the XON/XOFF protocol, you will probably need to use a special device handler and a DL11 or DLV11 type interface card. If you wish to attach a modem, you should also have an interface card which supports modem control.

6.5.2 Wiring

The connection from the interface card to the peripheral device is most commonly made according to the EIA RS-232-C standard with DB-25 connectors. The interface cards themselves usually terminate in DB-25P (male) connectors, and are considered as DTE (Data Terminal Equipment) devices. (DLV11J cards have 2X5 pin AMP connectors, but are commonly converted with short "pigtail" cables to a DB-25P connector.) Most terminals are also DTE devices and have DB-25P connectors. For DEC interface cards, only a few of the 25 pins in these connectors are significant. The pins significant to TSX-Plus are:

1	Protective ground
2	Transmitted data
3	Received data
7	Signal ground
8	Received line signal detector (Carrier detect)
20	Data terminal ready
22	Ring indicator

When using DEC interface cards and DEC peripherals, protective ground (pin 1) is usually connected internally to signal ground (pin 7) so there is no need to wire it through the cable. Since both the interface cards and peripherals are DTE equipment, there is a slight conflict in definition of the sender and receiver, resulting in a conflict on pins 2 and 3. When two DTE devices are connected through modems (modems are DCE devices, Data Communications Equipment), the modems translate the signals and reverse the sense of send and receive between the two devices so that what one device sends is considered received data on the other device. However, when the interface card is directly connected through a simple cable to the peripheral, the send and receive lines need to be reversed so that transmitted data from one end is correctly converted to received data at the other end. This type of connection is usually called a "null modem". DCE devices like modems usually use DB-25S connectors, so a null modem should also terminate in DB-25S (female) connectors. In the simplest case of connecting a terminal directly through a null modem cable to the interface card (or panel connector routed to it), only pins 2, 3, and 7 need be connected and 2 and 3 should be reversed at the two ends. This would look diagrammatically something like:

```
Interface      Cable                      Cable
  Card           End          Cable         End          Terminal
(DB-25P)      (DB-25S)                   (DB-25S)       (DB-25P)
   2 ->       >- 2 --------\/--------- 2 -<             <- 2
   3 ->       >- 3 --------/\--------- 3 -<             <- 3
   7 ->       >- 7 -------------------- 7 -<             <- 7
```

When actually attaching to a modem or modem-like device such as a line-driver, the DCE device performs the translation of transmitted data to received data and pins 2 and 3 should not be reversed. If you are connecting to a device which uses modem control (declared to TSX-Plus as a phone line), then pins 8, 20 and 22 should also be wired. An extension cable with 6 wires is then usually used to connect the interface card or terminal to the modem. This typically looks like:

```
Interface                                                          (((
 Card or       Cable                      Cable                 ((((
 Terminal       End          Cable         End          Modem  (((((  Phone
(DB-25P)      (DB-25S)                   (DB-25P)      (DB-25S)  ((((
   2 ->       >- 2 -------------------- 2 ->          >- 2          (((
   3 ->       >- 3 -------------------- 3 ->          >- 3
   7 ->       >- 7 -------------------- 7 ->          >- 7
   8 ->       >- 8 -------------------- 8 ->          >- 8
  20 ->       >-20 --------------------20 ->          >-20
  22 ->       >-22 --------------------22 ->          >-22
```

Other devices, such as non-DEC terminals or printers, can sometimes have additional wiring requirements. For example, we worked a long time to correct a buffer overrun condition on a NEC printer attached to a DHV11 using the CL handler. There appeared to be some problem with XON/XOFF flow control which caused CL to send characters when the printer was not ready to accept them, while the same printer worked correctly with an IBM PC. We finally identified the additional requirement of the NEC printer to receive a signal on pin 6 (Data Set Ready) before it could transmit. This prevented it from sending XOFF to the system, resulting in the buffer overruns. This signal was present with the interface and cable used with the IBM PC. The problem was finally remedied by simply shorting together pins 6 and 20 at the printer end of the cable. Since CL always raises DTR (Data Terminal Ready, pin 20) for each I/O operation, this kept pin 6 high for the NEC, allowing it to transmit an XOFF when its buffer fills. The object lesson is to fully understand the requirements of your hardware when dealing with unusual equipment.

6.5.3 Communication parameters

Several parameters control the data transmission format. For DL11 and DLV11 type cards, the character formats are selected by jumpers during installation of the cards. Baud rates are also preset by jumper, except on the DLV11E for which programmable baud rates are available, but not character length or parity. On DZ11, DZV11, DH11 and DHV11 type cards, TSX-Plus allows you to dynamically select: baud rate within the range supported by the card; parity control of even, odd or none; and 7 or 8 data bits. The most common format for

terminals is 9600 baud with 8 data bits and no parity. For hardcopy terminals such as an LA120, the common settings are 1200 baud with 8 data bits and no parity control.

6.5.3.1 Character frames: The character format can be conceptualized as a stream of high and low voltage pulses (representing bits) clustered into groups, called frames, which represent individual characters. All the devices supported as terminals or CL units by TSX-Plus are asynchronous, which means that a new frame is signalled by a start bit and ended with a stop bit, and that the separation between frames (characters) may be variable. The baud rate controls the duration of voltage pulses within each frame; the higher the baud rate, the less time between individual data bits.

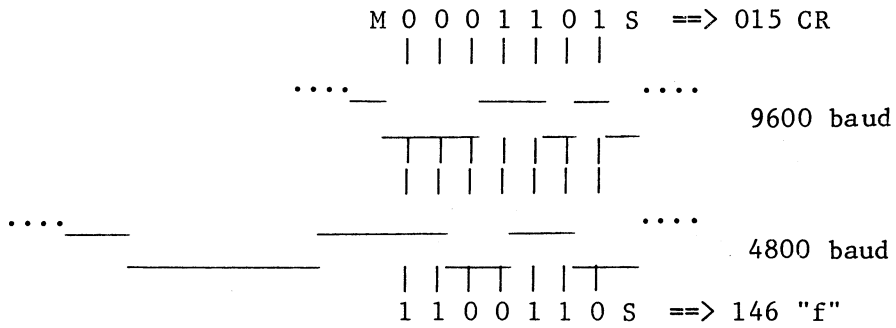
A new frame is always signalled by one start bit. This is followed by either 7 or 8 data bits representing the character, then an optional parity bit and at least one stop bit. The number of data bits, parity control and number of stop bits are selected by jumpers on DL type interface cards. Multiplexer lines may be set to either 7 or 8 data bits during system generation (default is 8), or during operation with the SET TT BITS command. Parity may be set to EVEN, ODD or NONE on multiplexer lines either during system generation (default is NONE), or during operation with the SET TT PARITY command. TSX-Plus always defaults to 1 stop bit on multiplexer cards. The number of stop bits is not usually critical since it effectively controls only the minimum separation between frames.

6.5.3.2 Parity: Parity control works by counting the number of data bits in the frame which are a logical 1. If parity is turned off, then no parity bit is transmitted or it is ignored on receipt. If parity is set EVEN, then the parity bit will be turned on when the number of logical 1 data bits in the character is odd and will be turned off when the number of logical 1 data bits is already even, so that the cumulative count of data and parity bits is even. The inverse operation is performed for odd parity. TSX-Plus does not support other parity protocols such as: MARK - parity bit always on; or SPACE - parity bit always off.

6.5.3.3 Automatic baud rate determination: TSX-Plus permits time sharing lines to be generated with automatic baud rate detection. That is, the speed of the terminal does not have to be preselected during system generation, but rather can be determined by the system by analyzing the first one or two carriage returns received from the terminal when it first logs on. This is done by initially setting the inactive line speed to 9600 baud. Then, when the first carriage return character is typed to activate the line, the character received by the interface may or may not appear to be a carriage return. If the terminal is set to 9600 baud, then the character should match and the system leaves the baud rate set at 9600. If the terminal is not transmitting at 9600 baud, then the bit pattern seen by the interface card will not match a carriage return. The character apparently received by the interface card is compared with a table of values which would be received if the terminal were transmitting at some other speed. If a match is found, then the line is set to the corresponding baud rate. If no match is found, then the speed for that line is

set to 600 baud and another carriage return is required. The process is repeated to match the apparent character with a table of expected values for a lower range of baud rates. If no match is found with this lower set of baud rates, then the speed is reset to 9600 baud and characters are ignored for a few seconds; the process then repeats until one of the allowed baud rates is selected.

This process can be demonstrated by looking at the input bit pattern for an ASCII CR (carriage return) character at 9600 and 4800 baud. The diagram below shows how the signal voltage levels might look on an oscilloscope trace for a single character frame (ASCII CR) at the two rates. Think of the signal as flowing in from the left and moving out to the right. The inactive line between character frames is indicated by the dots, usually at the MARK level (M). The start of the character frame is signalled by a start bit (usually a SPACE level (S)), followed by high or low levels appropriate for the individual bits of the character being transmitted, least significant bits first. The vertical lines indicate the approximate time points at which a device monitoring the line at 9600 baud would measure the voltage in both cases. Note that the character that would appear to be received if the 4800 baud CR were sampled at the expected timing for a 9600 baud line would be an octal 146 ("f") instead of 015 for a CR. Other baud rates would generate other apparent characters, and by matching the current baud rate with a table of these apparent characters the actual external baud rate can be surmised.



6.5.3.4 Break keys: Pressing the break key on a terminal usually generates a "long space" condition. When received, this results in a framing error because the stop bit does not occur within the expected time interval after the start bit. If the time-sharing line is set for a fixed baud rate or speed has already been selected on an "autobaud" line, then the (NUL) character is flagged with a framing error and passed through to normal character input handling. If the line is set for autobaud, and speed has not yet been selected, then it is handled as described in the section on autobaud. What happens at the computer on receipt of a break also depends on the type of interface card receiving the character. On multiplexer cards, receipt of a break character simply sets the framing error flag for the line. On DL type cards, installation options determine the result of incoming break signals; when the framing error bit is set it may cause a reboot or cause the processor to halt. On DLV11-J cards, these options only apply to channel 3, and should only be enabled when connected to the console terminal. The only effect of

framing errors on terminal input to TSX-Plus (except as noted for autobaud detection) is to bypass masking for 7-bit characters and skipping the tests for XON/XOFF receipt. The (erroneous) character is otherwise simply stored in the terminal input buffer.

6.5.3.5 7 or 8 data bits

There are two separate levels of data bit control. One level controls the multiplexer hardware (number of data bits on DL type interfaces is selected by jumpers at installation) and determines whether the multiplexer will transmit 7 or 8 data bits. This hardware control is selected by the SPEED macro in TSGEN or dynamically with the SET TT BITS command.

In addition, software masking of characters may also be done, regardless of whether the hardware is set for 7 or 8 data bits. This is the process of using only some of the characters which may actually be presented by the hardware. When 7 bit mode is selected, TSX-Plus clears any incoming eighth bit on terminal input before passing the character on to the program and masks output characters to the low 7 bits before sending them to the terminal interface card. This software masking is selected with the $8BIT option to the FLAGS macro in TSGEN or by the SET TT [NO]EIGHTBIT command during execution. Note that hi-efficiency terminal mode and CL binary mode bypass software 7-bit masking.

6.5.4 Flow control (XON/XOFF)

You may use almost any standard asynchronous terminal with TSX-Plus. However, the only flow control method supported is XON/XOFF (DC1/DC3, Control-Q/Control-S). Filler characters or other protocols such as ETX/ACK are not supported. Using the XON/XOFF protocol, if the terminal can't handle characters sent to it as fast as the computer can transmit them, then it sends to the computer an XOFF character, which causes the system to stop transmitting until it sees an XON character. Then, when the terminal empties its internal buffer and is ready to accept more characters, it sends an XON character to the system which in turn resumes transmission of characters to the terminal. If your terminal does not understand XON/XOFF, then buffer overruns are likely except at very low baud rates. Similarly, if the terminal transmits characters faster than the system can handle them, the system will send an XOFF to tell the terminal to stop transmitting, and will send an XON when it is ready to receive more characters.

In some special circumstances, it may be desirable to disable XON/XOFF processing by the system, such as when it is necessary for a program to read one of those two special characters from the terminal or when special circumstances dictate that the program handle its own flow control. The $PAGE option to the FLAGS macro in TSGEN and the SET TT [NO]PAGE command control whether TSX-Plus uses XON/XOFF flow control. The choice is between XON/XOFF control or no control. Disabling system XON/XOFF control causes TSX-Plus to treat XON and XOFF as ordinary control characters. The system does not intercept them, but rather passes them through to the terminal input buffer. This allows user-written programs to handle special cases.

6.5.5 Terminals

TSX-Plus cannot create features which your terminal does not have; for example, lower case characters are meaningless when using an ASR33 teletype. Other terminals may or may not understand such characters as TAB or FORM FEED. These, however, can be simulated to some extent by TSX-Plus. You may use the SET TT NOFORM and SET TT NOTAB commands to simulate them when necessary.

The SYSMON program, which is currently the only screen oriented utility supplied with TSX-Plus, can use several terminal types (VT200, VT100, VT52, ADM3A and HAZELTINE).

The RT-11 KED editor is used with VT100 type terminals or VT200 type in VT100 mode. VT52 terminals should use the similar K52 program. Lower case must be enabled to use either of these programs since the numeric keypad in alternate keypad mode generates escape sequences with lower case characters. If the system translates these to upper case, then the editing programs will not correctly recognize some keypad commands. When the terminal type is known to TSX-Plus, it automatically selects the correct program, according to the terminal type, when either KED or K52 is invoked through the EDIT command. This means that if you use the commands:

```
SET TT VT52
SET EDIT KED
EDIT TEST.DAT
```

Then, the program which is actually run by the system will be K52 to correspond to the terminal type, rather than KED as requested. If on the other hand, you explicitly RUN KED, then you will actually execute KED. The system interprets the EDIT command and selects the appropriate program, but does not override explicit program selection.

VT200 series terminals may be used either in either VT100 or VT200 mode. When in VT100 mode, the terminal setup should be VT100 with 7-bit controls. When in VT200 mode, the terminal setup should use 8 data bits and no parity. Some VT200 keys behave differently in VT100 and VT200 modes, most notably the F11, F12, and F13 keys. In VT100 mode, these keys are:

```
F11     ESC (escape)
F12     BS  (back space)
F13     LF  (line feed)
```

However, in VT200 mode, the top row function keys F6 through F20 all generate four character special ANSI escape sequences. Most programs written for use with VT100 or VT52 terminals will not recognize these sequences. Using the TRMTYP VT200 macro in TSGEN or the SET TT VT200 command also automatically sets the $8BIT flag; unless the terminal itself is also set to 8 data bits and no parity, it will appear dead. The $8BIT flag is not automatically reset by the SET TT VT100 command.

6.5.6 Modems

There are many kinds of modems available today, ranging from those with only the most basic capability to convert between digital signals and acoustic tones to very sophisticated, "intelligent" devices. Most modern modems at least have the features of direct connection to the telephone system and automatic answering capability. We will not specifically discuss acoustic coupler type modems as they require a lot of manual intervention.

If TSX-Plus is to handle a line as a modem, then it must be identified as a phone line either by the $PHONE option to the FLAGS macro in TSGEN or with the SET TT PHONE command during system execution. If TSX-Plus is to handle the modem, then the line must also be connected to an interface card which supports modem control (most multiplexers do, a DLV11J does not), the modem must be configured to a "standard" auto-answer configuration, and the modem and cable must support the following signals:

Pin	Signal name
2	Transmitted data
3	Received data
7	Signal ground
8	Carrier detect
20	Data terminal ready
22	Ring indicator

The normal sequence of events for a dial-in phone line is:

1. The phone rings (possibly several times)
2. TSX-Plus detects the ring signal, raises the data terminal ready line, and starts the OFFTIM and TIMOUT timer
3. When DTR is raised, the modem should answer the phone and establish a connection with the remote modem, raising the carrier detect signal when successful
4. When carrier detect is present, TSX-Plus cancels the TIMOUT timer
5. A carriage return is typed at the remote terminal
6. TSX-Plus transmits the greeting message, performs line initialization, executes the start-up command file(s) (should include running LOGON), and cancels the OFFTIM timer
7. Normal time-sharing session operations
   .
   .
   .
8. The remote terminal logs off
9. TSX-Plus starts the OFFTIM timer
10. The remote modem hangs up
11. The local modem drops carrier detect
12. TSX-Plus starts the TIMOUT timer
13. When either the OFFTIM or TIMOUT timer elapses, TSX-Plus lowers DTR
14. The local modem hangs up the phone

In this normal sequence of events, any time the carrier detect signal is lost, TSX-Plus starts the TIMOUT timer. If the carrier detect signal is re-established, then TSX-Plus cancels the TIMOUT timer. If the carrier detect signal remains lost for the period specified by TIMOUT, then TSX-Plus logs off the line and lowers DTR which should cause the modem to hang up the phone. This takes care of the case in which the operator of the remote terminal hangs up the phone without logging off. The converse case, in which the operator of the remote terminal logs off but does not hang up the phone, is handled by the OFFTIM timer. If the line remains logged off, but connected (carrier detect signal present) longer than the period specified by OFFTIM, then the system lowers DTR and the modem should hang up the phone. The OFFTIM value also deals with the case in which someone dials in and is unable to successfully log in within the specified period. Some trial and error adjustment is usually necessary for the TIMOUT and OFFTIM values to account for the timing variability of the particular interface card, modem and phone system at each installation.

Even if a line has been declared to TSX-Plus as a phone line, when it first logs on (first carriage return received on an inactive line), the system will treat it as a true phone line only if the carrier detect signal is asserted; if not, then TSX-Plus treats it as a local line and does not perform any of the modem control functions described above. This is usually the case for an "intelligent" modem, which answers the phone for itself without regard to the DTR signal, and does not assert carrier detect to the system or is connected through an interface which does not support modem control.

If it is necessary to connect a modem to an interface card which does not support modem control (e.g. DLV11J), then it is necessary to either manually establish the connection or use an "intelligent" modem which itself takes care of monitoring the signal quality (carrier) and answering and hanging up the phone. Lines connected in this manner may be generated either with or without the $PHONE flag; TSX-Plus will treat them as local lines in either case and will not provide any form of modem control, including establishing or monitoring the TIMOUT and OFFTIM telephone security timers.

When using "intelligent" modems, it is important to keep them quiet except for normal operations. Some modems offer an option of providing information messages concerning their operation. This can cause futile loops when used with TSX-Plus. Consider, for example, the following cycle:

1. An incoming call is recognized by the modem which transmits the message "RING" preceded and followed by carriage return/line feed combinations.

2. TSX-Plus sees the first carriage return, activates the line, displays the greeting message, sees no carrier (since the connection has probably not yet been established) and assumes it is a local line.

3. The modem either then attempts to respond to the greeting message as a command or TSX-Plus attempts to accept "RING" as a command, which it normally is not, or if LOGON is running, attempts to use "RING" as a logon user name, followed by whatever comes in next as the password. This normally does not match the correct password.

4. Synchronization almost always fails between the informative messages from the modem and the greeting message. This results in the modem continually sending "ERROR", "RING", "CONNECT" or other messages to TSX-Plus, and TSX-Plus continually sends the greeting message, fails to log the line on, and disconnects the line.

The outcome is that the line repeatedly initiates and runs the LOGON program, but never successfully logs on. This informative message mode is good when the modem is connected directly to a terminal, but wreaks havoc with normal modem operations by TSX-Plus. When using a D.C.Hayes Smartmodem (or compatible modem), it is normal to enable messages when using it as a dial-out device, but the default switches should be set to disable messages on power-up (on a Hayes Smartmodem 1200 for example, switch 3 should be UP) and the modem should always be reset to the quiet mode before returning the line from a CL "dial-out" operation to use as a time-sharing line. The following example dial-out session shows how to take over a time-sharing line, connect to it as a virtual terminal with VTCOM, issue Hayes style modem commands, execute a time-sharing session on the host system, and then return everything to the original state. Text after an exclamation point and everything in parentheses are comments. Especially note the modem command to set it to quiet mode (AT Q1), which is essential for subsequent use of the modem as a "dial-in" time-sharing line.

```
.SET CL1 LINE=8,SPEED=1200,NOLFOUT  !TAKE OVER T-S LINE
.SET SL OFF                         !TURN OFF SINGLE LINE EDITOR
.ASSIGN CL1 XL                      !ASSIGN FOR VTCOM
.ALLOCATE XL                        !EXCLUSIVE USE
.R VTCOM                            !CONNECT TT TO XL
AT Q0                (TURN ON MODEM INFORMATION MESSAGES)
                     (CR/LF FROM MODEM)
OK                   (MODEM MESSAGE)
AT DT 19005551212    (MODEM COMMAND DIAL THIS NUMBER)
                     (CR/LF FROM MODEM)
CONNECT              (MODEM MESSAGE)
^G                (SINGLE BELL CHARACTER)

?VTCOM-I-Connection established



^G                (SINGLE BELL CHARACTER)
```

```
              Exemplary VAX/VMS System

Username: USERNAME

Password: PASSWORD

        Welcome to VAX/VMS version V4.1
    Last interactive login on Wednesday,  1-MAY-1985 12:52

     .
     .              (TIME-SHARING OPERATIONS ON HOST)
     .
$ LOG/BRIEF

USERNAME logged out at  1-MAY-1985 12:56:21.43

?VTCOM-I-Connection lost
^G^G^G^G          (FOUR BELL CHARACTERS)
                      (CR/LF FROM MODEM)
NO CARRIER            (MODEM MESSAGE)
AT H                  (MODEM COMMAND HANG UP PHONE)
                      (CR/LF FROM MODEM)
OK                    (MODEM MESSAGE)
AT Q1                 (MODEM COMMAND QUIET ***IMPORTANT***)
^P                    (VTCOM COMMAND TO COMMAND MODE)
TT::VTCOM> EXIT       (VTCOM COMMAND EXIT)
.SET SL ON                          !TURN SINGLE LINE EDITOR BACK ON
.SET SL KED                         !SET SL KED MODE
.DEALLOCATE XL                      !NON-EXCLUSIVE USE
.DEASSIGN XL                        !CLEAN UP XL ASSIGN
.SET CL1 LINE=0                     !RETURN CL LINE BACK TO T-S USE
```

Although most "intelligent" modems support the de facto Hayes "standard", not all do, and many of those that do, are not completely compatible. They must be configured either to conform to the modem control method supported by TSX-Plus or to handle the telephone themselves without modem control signals from the system.

A note on using VTCOM: When running VTCOM, the command character to return VTCOM from terminal mode to command mode is a control-P (^P). (Note the use of ^P in the example above.) This is a minor conflict with the console break character on some computer models (e.g. PDP-11/44). When using VTCOM from the console terminal on such computers, you must set the computer itself into the "local disable" mode. Otherwise, hardware ODT will intercept the ^P character and begin ODT execution on the console rather than pass the ^P through to the VTCOM program. This is not a problem on terminals other than the console.

6.5.7 TSX-Plus system generation of terminal lines

The configuration of time-sharing lines for TSX-Plus is done by including certain declarations near the end of the user-modifiable section of TSGEN.MAC. Macro operators are provided to handle most of the work. Remember that information on any line to the right of a semicolon (";") is ignored by the MACRO assembler. In other words, lines which begin with a ";" are treated as comments. The assembler also treats numbers as octal values unless the radix is specifically changed (we do not ever do that in TSGEN) or the number is followed by a decimal point ("."). Except for vector and address values, we recommend that you use a decimal point for most numbers unless you are in the habit of thinking in base 8 (octal). This applies in particular in the time-sharing line definition section in the DZDEF, DHDEF, DHVDEF and LINDEF macros. By convention, vectors and CSR addresses should always be specified as octal values. Referring to them in any other way will only lead to confusion and error. Port numbers on a multiplexer should be specified as decimal values (e.g. 0. thru 15.).

6.5.7.1 TBLDEF: The TBLDEF macro declares the number of primary time-sharing lines (physical terminals) which will be attached to the system, the number of subprocesses (virtual jobs which are secondary to some primary line - use the same terminal), the number of detached jobs (jobs which run independently of any terminal), and the number of dedicated CL lines (lines which will only be used as I/O devices and never used as a terminal). The number of primary lines (first TBLDEF parameter) MUST correspond to the number of LINDEF macros which follow. The maximum number of jobs possible in a given system is determined by the total of the first three parameters (primary+detached+subprocess). Jobs are identified during system operation based on these numbers. Primary jobs are numbered from 1 to the number of primary lines. Detached jobs (both active and unused) are numbered next, followed by subprocesses. The SYSTAT (or WHO or SHOW JOBS) command identifies jobs according to these numbers. The SHOW TERM command only displays primary lines. The parameters to TBLDEF should be decimal numbers. For example:

```
TBLDEF    11.,2.,3.,0.
```

declares eleven primary time-sharing lines which will be numbered 1 thru 11, two virtual job slots which will be numbered 15 and 16, and three detached job slots which will be numbered 12 thru 14. No dedicated CL lines are reserved. Dedicated CL lines are generally used for printers, plotters and similar devices which are read and written by jobs on other lines and do not need to act as terminals. If these devices are attached to DL(V)11 type interface cards, then they might well be serviced by an LS handler. If they must be attached through multiplexer ports (DZ, DH or DHV) then they must be used through the CL system. Note that the CL pseudo-device is NEVER declared in the device definition section of TSGEN. (Neither TT nor LD is declared with a DEVDEF either.)

6.5.7.2 Multiplexer definitions (DZDEF, DHDEF and DHVDEF): These are used to define DZ(V)11 and DH11 and DHV11 multiplexer devices. These macros accept vector and CSR addresses which are octal numbers. (The old MUXDEF macro is equivalent to DZDEF.) Some vendors supply multiplexers which are compatible with either DH or DHV protocols (which are significantly different from each other). You must select the correct multiplexer definition macro according to the protocol option enabled with the hardware.

6.5.7.3 LINDEF: On DL(V)11 interfaces, this macro requires the port vector and CSR address. Within multiplexer definition blocks, this macro requires only the port number. Multiplexer ports are numbered from 0 to the number of ports supported by the device, up to a maximum limit for each device. Per vector and CSR pair: the DZV11 supports 4 ports; the DZ11 supports 8 ports; the DHV11 supports 8 ports; and the DH11 supports 16 ports. Other vendors may permit one multiplexer to support more than these numbers of ports, but TSX-Plus constrains the multiplexer definitions to certain upper limits. That is, you may specify multiplexer port numbers to the LINDEF macro within the following limits (remember to either use decimal points or think octal):

Multiplexer Definition	Port Numbers
DZDEF	0. - 7.
DHDEF	0. - 15.
DHVDEF	0. - 7.
DHUDEF	0. - 15.

Remember that the total number of LINDEF macros (both DL(V)11 type and within multiplexer definition blocks) must add up to the number of primary lines declared with the TBLDEF macro (first parameter).

Also remember when specifying vectors and addresses for DL(V)11s and multiplexers, that you must not have a device handler which uses the same vector or address. In fact, it is a general rule that no two device interfaces (whether they be time-sharing lines, printers, plotters, modems, disk drives, or any other device) can share the same vector and CSR address. This is a common mistake when a DL(V)11 port is to be used with VTCOM. VTCOM uses the XL device which communicates through a serial port. It is acceptable to use XL with VTCOM under TSX-Plus; you should then include a device definition (DEVDEF) for XL, but then no time-sharing line may also use that port. Note also that XL can only be used on a DL(V)11 type port. If a port is to be used sometimes as a time-sharing line and sometimes as a communications line with VTCOM, then

1. declare it as a normal time-sharing line (possibly with the $PHONE flag);
2. do NOT include the XL device handler;
3. include at least 1 CLXTRA line;
4. do NOT declare a device definition for CL;
5. do NOT declare the port as a dedicated CL line (CLDEF).

Several other macros may be used within a line definition block, such as FLAGS, SPEED, and TRMTYP. These and others are described in the TSX-Plus Installation Guide. (Note that most terminals use XON/XOFF control and should have the $PAGE flag set either in their unique FLAGS macro or in NRMFLG.)

6.5.7.4 LINEND and MUXEND: For each LINDEF, there must be a corresponding LINEND. For each DZDEF, DHDEF, or DHVDEF, there must be a corresponding MUXEND. Failure to observe these rules will probably result in an error during assembly of TSGEN.MAC. Failure to match the number of LINDEFs with the number of primary lines declared as the first parameter to TBLDEF may result in an "Invalid status register address for T/S line:" error message during start-up of TSX-Plus.

6.5.8 Troubleshooting

When communication trouble occurs with terminals or modems, several approaches can provide insight to the origin of the problem. Don't overlook the obvious: Is it plugged in? It is not uncommon to suffer much exasperation only to discover that a cable is not securely connected.

6.5.8.1 A note about hardware addresses: Device addresses are always specified in TSGEN as 16-bit virtual addresses. By convention, they are always specified as octal values. Due to the virtual system address mapping, device virtual addresses will always be in the range 160000 to 177776. This 8Kb range of virtual addresses is referred to as the I/O page. The actual physical addresses which encompass the I/O page depend on your computer. If your machine can only address 64Kb of memory (16-bit addressing), then the I/O page will be in physical addresses 160000-177777. (However, you cannot use TSX-Plus on a machine that only supports 16-bit addressing.) If your computer supports 18-bit addressing, then the physical addresses of the I/O page are from 776000 to 777777. The PDP-11/34 and LSI-11/23 (not PDP-11/23-PLUS) are typical 18-bit machines. Newer machines such as the PDP-11/23-PLUS, LSI-11/73, PDP-11/24 (with PAX option), PDP-11/44, and PDP-11/84 support 22-bit addressing. The physical addresses of the I/O page on 22-bit machines are in the range 17776000-17777777. When using ODT to address locations in the I/O page you must specify physical addresses as appropriate for your hardware.

6.5.9 Unknown configurations: One of the most common errors when interfacing terminals is not knowing the correct CSR and vector addresses at which the card was configured when it was installed. A second common error is to use the same CSR and/or vector for two different pieces of equipment, e.g. both a terminal and a printer. This mistake is most frequently made on DL(V)11 type interfaces, since they are commonly used for both serial printers and terminals.

Determining the correct CSR and vector is best done at the time of installation. The configuration information should then be recorded where it can be conveniently found for later reference. In fact, a blank form is provided as an appendix to the TSX-Plus Installation Guide for recording device interface information.

If the addresses were not recorded during installation or have since been lost, the second best method of determining the configuration information is removing the card and comparing the jumpers or switches with the installation and configuration instructions.

Finally, if you think you know the CSR but are not sure, or if you are willing to do some hunting by trial and error, then you can use the hardware ODT on most machines to test the address.

6.5.9.1 Using ODT: On Q-bus machines, examining an address in the I/O page will return a value only if some installed hardware responds to that address, otherwise it will return with a question mark. For DL type cards, you can deposit values into the XBUF (transmitter data buffer). If everything is wired correctly, the value will be transmitted to the terminal and should display on its screen. Depending on your machine, you may need to use either 18-bit or 22-bit addresses to access the I/O page. This method can be tested with the console terminal interface on almost all machines.

The logical structure for control of and I/O to a DL type interface card (including the console terminal) is four words located in the I/O page for each device. These are:

RCSR (Receiver Control & Status Register)	Base address
RBUF (Receiver data BUFfer)	Base + 2
XCSR (Transmitter Control & Status Register)	Base + 4
XBUF (Transmitter data BUFfer)	Base + 6

The address at which the device is configured is the base address. This is also the address of the RCSR for the device and the octal value of this address always ends with a 0. That is, the base address is always a multiple of 10 (octal). Some common base addresses for DL type cards are: 177560 (console terminal), 176500 (default serial printer), 176510, 176520, 176530, 175610. In order to determine whether a card is configured at a given address, first try to examine the base address. If it responds (gives a value rather than a "?"), then try to send a character through the XBUF, where XBUF equals RCSR plus 6, to the terminal attached to it. If the terminal displays the character put in the XBUF, then that is the correct base address for that interface card or port. If not, then

1. the terminal is attached to a different interface card or port;
2. the terminal is not correctly wired to the interface card;
3. the baud rate or parity is incorrect;
4. the port is defective;

5. the terminal is defective.

If a garbled character shows up on the terminal, then the baud rate or parity is probably mismatched between the terminal and the interface card.

For example on a PDP-11/23-PLUS (with 18-bit ODT):

```
*777560/000100          (SEE IF CONSOLE RCSR RESPONDS)
*777566/000000 101      (TRANSMIT CAPITOL "A" TO CONSOLE)
*777566/000000 7        (TRANSMIT "BELL" TO CONSOLE)
*777570/                (TEST NON-EXISTENT HARDWARE)
?
*
```

On a MicroPDP-11/73 (with 22-bit ODT):

```
*17777560/000100        (SEE IF CONSOLE RCSR RESPONDS)
*17777566/000000 101    (TRANSMIT CAPITOL "A" TO CONSOLE)
*17777566/000000 7      (TRANSMIT "BELL" TO CONSOLE)
*17777570/              (TEST NON-EXISTENT HARDWARE)
?
*
```

The ODT interface looks slightly different on some other machines, such as old PDP-11/34's on which you may have to operate through the front panel. Once you demonstrate that you can use the ODT to read and transmit to the console, you can use the same technique with different addresses to test the presence of the interface card you are interested in and try to send characters through the XBUF if it is a DL type card.

If your problem is with a multiplexer interface card (DZ, DZV, DH or DHV), then you can test the presence of the device's base address, but it is usually not worth the effort to try to manipulate the registers to transmit a character to a specific port. This is much more easily done with a program designed to use or test the specific type of multiplexer port. (TSX-Plus is generally quite effective at exercising these type of multiplexers.)

Determination of the vector at which a card is configured is not as simple as testing the base address (CSR). If the vector is not known from installation or from examination of the card, then you can generate a line definition block (LINDEF) for the device with a guess at the vector (but do not use a vector known to be used by another device). If you are lucky, then your guess will be correct and a terminal attached to the port will behave correctly. If not, then sending a character from the terminal will cause the following error:

```
?TSX-F-Fatal system error
UEI-Interrupt occurred at unexpected location
```

(Be sure that the TSGEN parameter UXIFLG is set to 1, or TSX will ignore unexpected interrupts.) The argument value which follows the error message is the vector selected in the card's hardware. Go back and correct your LINDEF

and regenerate TSX-Plus. This method only works if the cards RCSR interrupt bit is enabled. TSX-Plus does this for you if the CSR is correctly specified in the LINDEF, otherwise you will have to go back and set it yourself using hardware ODT. Input interrupts are usually enabled by loading a 100 into the board's CSR address. To enable input interrupts, put a 100 in the CSR (same as base address on a DL or DZ type card). For example:

```
*776510/000000 100      (ENABLE INTERRUPTS ON DLV11J CHANNEL 1)
                        (TOGGLE HALT SWITCH TO "RUN")
*P                      (ODT PROCEED COMMAND)
                        (TYPE A CHARACTER AT TERMINAL ON CHANNEL 1)
?TSX-F-Fatal system error at 23626
UEI-Unexpected interrupt
Arg. value = 310
```

The argument value will be the vector through which the device interrupted.

When dealing with multi-port interface cards, always be sure your terminal is actually plugged in to the port number you want. Some cards have the ports numbered left to right, some are numbered the opposite direction. Some cards have the ports brought out in two groups of four so you have both left/right and two banks to choose from. If the ports are not correctly or clearly marked, then trial and error may be necessary to determine the actual port numbering. For multi-port interface cards, it sometimes can be simplest to generate your TSX-Plus system to include all the ports on that interface card in your line definition blocks, then plug a terminal into one of the ports, hit return, log on, and use the SHOW TERMINALS command to see to which port it is attached. Note that multiplexer port numbers for the LINDEF macro in TSGEN may be from 0 to 3 for a DZV11, 0 to 7 for a DZ11, 0 to 7 for a DHV11, and 0 to 15 for a DH11. If you use a DZV11 with 4 ports, include a DZDEF in TSGEN, and specify a LINDEF 4, then you will never get any errors, but neither will you ever be able to use the line since there is no #4 port on a DZV11.

6.5.9.2 Replace the device: When problems arise with a particular device, try replacing it with a different device to see if it makes any difference. If it does, then determine what different characteristics that device has and correct the problem. For example, when a dial-out modem connected to a CL line does not seem to respond, attach a terminal to the line in place of the modem. Can you send data through the CL line to the terminal? (Be sure the terminal baud rate is set correctly. You will also probably need a null modem cable.)

6.5.9.3 Break-out boxes: One of the most helpful and inexpensive tools available to solve wiring and cable problems is a device called a "break-out box" or "signal activity monitor". These are devices with DB-25 connectors and indicator lights which monitor some or all of the RS-232-C signals. Many modem problems are easily solved in a few seconds when you have the ability to monitor the signals passed between the modem and the interface card, easily justifying the cost of some type of RS-232-C signal monitor. These devices are particularly useful for watching the ring, DTR, and carrier detect signals when interfacing a modem, for checking cable continuity (Is it broken?), and for

determining when a null modem cable is necessary. They are available from most computer supplies companies through a wide range of price and capability. Most needs can be satisfied with the simplest devices in the $40 to $100 range Examples are: Micro Peeper from Global ($60); Line Status Indicator from Jade ($40); Easy-Cabler from Misco ($100); or Quick Test from Black Box ($50).

Assuming the signal monitor is attached to a terminal or printer as the test device, then typing or printing text from the computer to the peripheral should modulate the received data line (pin 3) and typing characters at the terminal should modulate the transmitted data line (pin 2). If transmissions from both the computer and the terminal cause the same indicator to light, then you probably need a null-modem cable to reverse pins 2 and 3. Some test devices even incorporate a normal vs. null modem switch to instantly verify the need for inverting the wiring. If TSX-Plus refuses to treat an incoming modem line as a "phone" line, see if the carrier detect signal (pin 8) is raised when the line logs on. If not, TSX-Plus will treat it as a local line and neither manipulate the DTR signal nor use the TIMOUT and OFFTIM timers. If TSX-Plus never answers an incoming call, see if the modem passes the ring signal (pin 22) through to the interface card. (Remember, the card must also support modem control.)

6.5.10 References

Many sources of information are available to aid in understanding data communications. However, most references either present much more than you ever wanted to know or not enough to be of any use. The following are among the more usable:

Technical Reference section of the Black Box catalog.

RS-232 Made Easy, Martin D. Seyer (1984) Prentice-Hall.

Technical Aspects of Data Communications, John E. McNamara (1978) Digital Press.

7. SYSTEM OPERATION

This chapter presents an overview of the TSX-Plus system organization and operation. It is intended to provide background information for users who want to know more about the internal system organization and operation.

7.1 Memory organization

Memory is organized into two major divisions: memory used by the operating system and memory available for user programs. The memory required by the operating system is permanently allocated and contains both code regions and data structures reserved for its exclusive use. In contrast, the content of user memory changes frequently as different jobs are swapped in and out of memory. Associated with each job, the system maintains a 6K byte job context region. Job swapping only occurs when a user job needs service and there is not enough contiguous free memory to load it and its job context region. Job swapping may be disabled entirely as a system generation option. In this case, a new job can only be started when sufficient user memory is already available.

7.1.1 System memory mapping

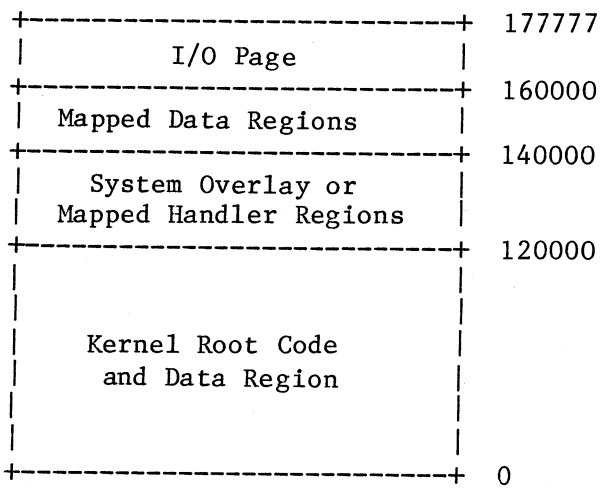
The operating system is divided into four distinct regions: kernel root, system overlays, mapped data, and the I/O page. The kernel root is mapped using kernel PARs (page address registers) 0 through 4. Because of this, the kernel root code region is restricted to a maximum of 40K bytes. (Each PAR maps 8K bytes.) The I/O page is mapped through kernel PAR 7.

Each system overlay code region and mapped device handler is accessed through kernel PAR 5 and is therefore restricted in size to a maximum of 8K bytes. Only one memory resident overlay or handler code region may be mapped at a time.

Each mapped data region is an individual storage area mapped through kernel PAR 6. Because of this, each data region is restricted in size to a maximum of 8K bytes. Only one data region may be accessed at a time.

The following diagram illustrates the virtual address organization of TSX-Plus during execution.

Virtual Memory in the TSX-Plus Kernel



7.1.1.1 Kernel root: The kernel root contains: device handler vectors (located from zero to octal 500); the memory resident overlay handler and tables necessary for interfacing to overlay code sections; data tables allocated in TSGEN; executive code including the job execution scheduler; I/O related processing code; clock and terminal interrupt entry code; startup initialization code; generalized data cache code (optional); and shared file record locking code (optional). To conserve space, TSX-Plus re-uses the memory containing the startup initialization code by loading the optional generalized data cache code and the shared file record locking code over the initialization code once the initialization is complete. Data structures which do not require initialization are also allocated over the initialization code. If additional space is necessary, the top of TSX-Plus is extended. Unmapped device handlers are loaded above these data structures. The size of the entire kernel root region described here (including unmapped device handlers) must not exceed 40K bytes.

7.1.1.2 System overlay and mapped handler regions: There are currently 21 memory resident overlay code regions. They are separated logically by function. Since only one overlay code region may be mapped at a time this functional separation reduces the number of calls to the overlay handler. Thirteen of the overlay code regions are optional and will only be loaded if the feature is selected in TSGEN. The functions performed by the overlay code regions are:

1. Terminal input and output operations
2. Programmed EMT requests code
3. Directory manipulation requests and directory cache buffers
4. Miscellaneous executive functions such as clock processing and fatal error processing
5. * Job swapping
6. * Program logical address space requests (PLAS)
7. * Device spooling with buffers
8. * Shared file record locking
9. * Message communication
10. * Real-time service requests
11. * Mapped I/O servicing
12. * Single line editor
13. * Communication line (CL) handler
14. * User program debugger
15. * Process windowing
16. * System crash dump generation
17. * Generalized data cache

* Denotes optional overlays that are only loaded into memory if the corresponding feature is selected during system generation.

The number of mapped handlers will depend on the device declarations (DEVDEF) in TSGEN and the corresponding attributes declared or imposed by the initialization routine for each device handler.

7.1.1.3 Mapped data regions: The mapped data regions allocated during startup contain the memory map table; the terminal input and output character buffers, and the following optional buffers: shared data cache buffers, mapped I/O buffers, performance monitor buffers, generalized data cache buffers, shared file record locking tables.

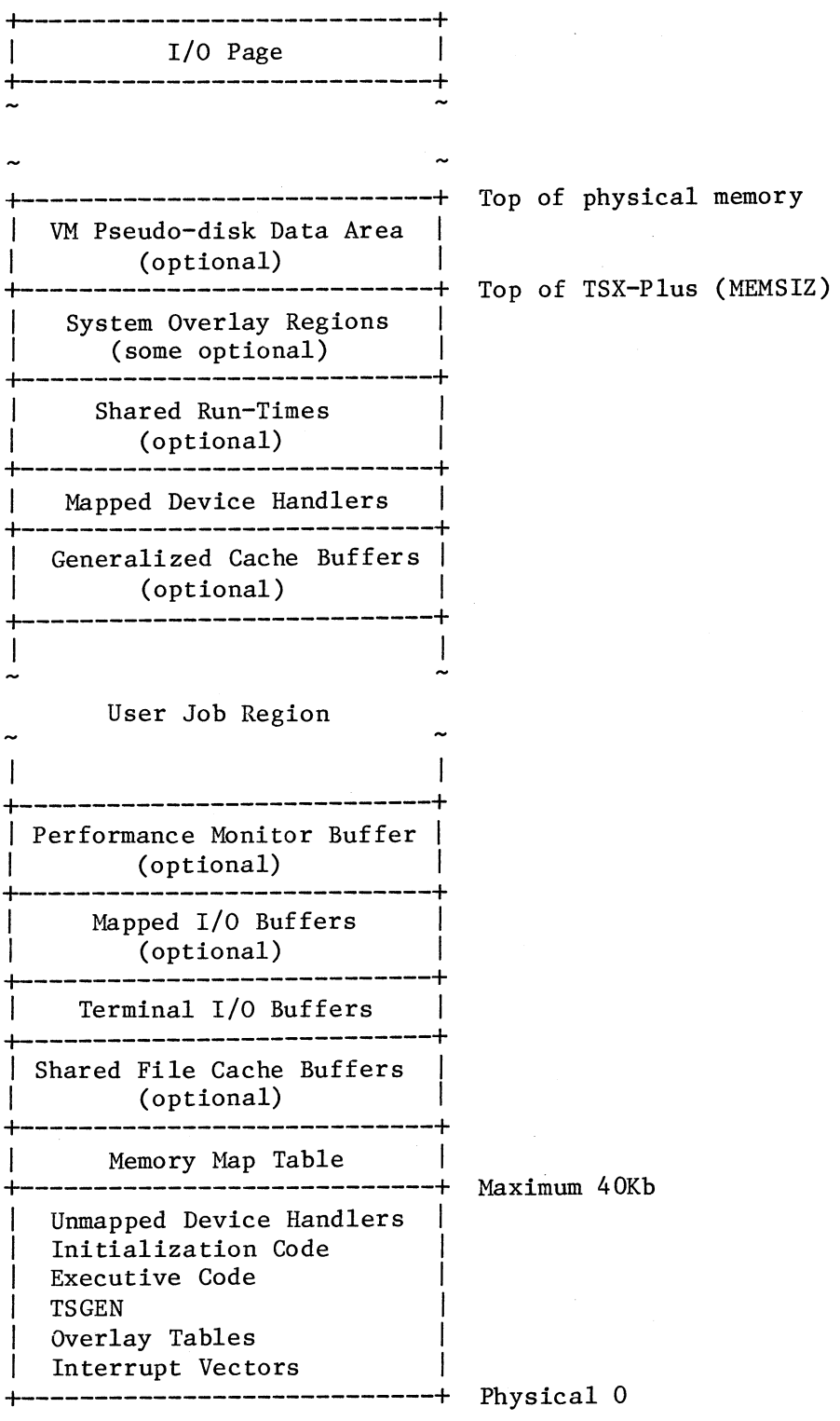
7.1.1.4 Shared run-time systems: In addition to the system regions described, a reserved memory region, also pre-allocated by the system, contains user-defined shared run-time systems such as those provided with COBOL-Plus and DBL.

7.1.2 Physical layout of TSX-Plus

The kernel root begins at physical memory address zero. Its size is variable, depending on options selected during system generation, and may extend up to 40K bytes. All of the mapped data regions are allocated directly above the kernel root with the exception of the generalized data cache buffers which are allocated directly below the system overlay regions and any optional shared run-times. The system overlay regions are allocated at the top of physical memory, or at the top selected by the MEMSIZ parameter if not all physical memory is to be used by the system. For example, some portion of memory may be reserved for use by a memory based disk emulator such as VM. Shared run-time systems, if any, are loaded directly below the system overlay regions as are mapped device handlers. Data buffers used by the generalized data caching facility are allocated below any mapped device handlers. Finally, all the

physical memory between the two memory areas allocated by the operating system is available for time-sharing users. The following diagram depicts the physical memory allocation of TSX-Plus during execution:

Physical Memory Use by TSX-Plus



7.1.3 User memory

The user's job region, sandwiched between memory used for the operating system, is allocated dynamically, placing each user's job in the first available free memory area large enough to contain it. In a swapping system, each job can potentially be positioned anywhere within the region. A 4K byte job context region is appended immediately below each job image, allowing the job and its context region to be swapped together.

The virtual address space of each job is intrinsically limited to 64K bytes by the PDP-11 architecture, although the job may remap itself by use of real-time or shared run-time EMTs. In addition, each job may request and be granted more physical space by use of PLAS requests. These extended memory regions may be used for virtual overlays or virtual arrays and need not be contiguous with the job's base image. When an extended job is swapped, the PLAS regions are swapped into a disk file separate from the base image.

7.2 I/O mapping

I/O mapping is a facility which allows DMA devices with 18-bit controllers or device handlers to be used with Q-bus systems with 22-bit address space.

The original LSI Q-bus used with 11/23 systems had 18 address lines allowing I/O transfers to take place within 248K bytes of memory. Device controllers developed during this period supported 18 address bits. With the introduction of the 11/23-Plus processor, four additional address bits were added to the Q-bus bringing the total to 22 address bits which allowed I/O transfers to take place to 4M bytes of memory. Unfortunately, many sites still have older device controllers that only support 18 bits and, in fact, DEC still does not build a Q-bus DY (RX02) controller that supports 22 bit DMA transfers. The 18-bit controllers will operate satisfactorily with 22-bit Q-bus systems provided that the I/O transfer is always within the lower 248K bytes of memory. This could cause problems with TSX-Plus since jobs may be located anywhere in physical memory and I/O transfers are normally done directly to buffers located in the job region.

The I/O mapping facility causes the system to "map" I/O transfers through system buffers that are always located in the lower 248K bytes of memory. This facility may be specified selectively for those DMA devices that only have 18-bit controllers or device handlers. The "MAPIO" option for the DEVDEF macro is used to indicate that I/O mapping should be done for a device. Devices which support 22-bit addressing and have device handlers which will execute 22-bit DMA transfers do not need system buffering and can operate normally. See the TSX-Plus Installation Guide for information pertaining to device handlers which support 22-bit DMA.

When I/O mapping is selected for a device, TSX-Plus examines each I/O operation directed to the device and if the buffer is outside of the lower 248K bytes it moves the data from the user's buffer to/from a system buffer located in the lower 248Kb memory area and performs the actual data transfer from the system buffer to/from the I/O device. This allows 18-bit devices to be accessed by

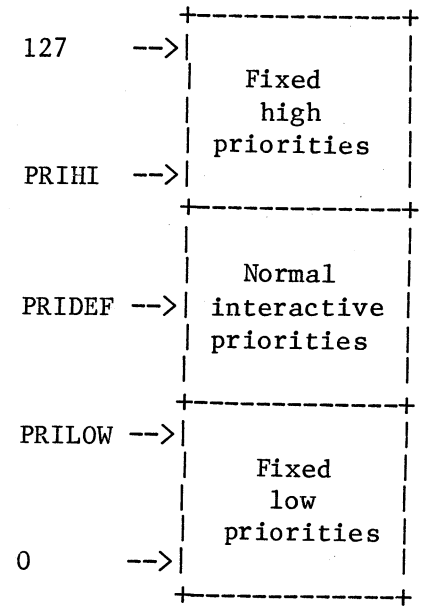
all time-sharing jobs regardless of their location in physical memory. However, it introduces a significant speed penalty since the data must be moved between the system buffer and the buffer in the job space. A further speed penalty is introduced in cases in which the amount of data being transferred is larger than the system buffer. In this case, an I/O operation which would normally be accomplished as a single transfer will be broken down into a series of smaller transfers. When a large operation is broken down into a series of smaller operations time is lost waiting for the device to reposition itself for the start of the next operation. This speed penalty can be minimized by allocating a large enough system buffer to accommodate most I/O transfers as a single operation. The generalized data caching facility can also significantly overcome the speed penalty since data read from the cache does not have to be mapped.

7.3 Job scheduling

TSX-Plus uses a unique Adaptive Scheduling Algorithm (tm) (patent pending) which schedules jobs for execution based on two factors: (1) the value of a user-assigned job priority that may range from 0 to 127; and (2) the execution state of the job.

7.3.1 Job priorities

Job priority values are arranged in three groups: the fixed-low-priority group consists of priority values from 0 up to the value specified by the PRILOW sysgen parameter; the fixed-high-priority group ranges from the value specified for the PRIHI sysgen parameter up to 127; the middle priority group ranges from (PRILOW+1) to (PRIHI-1). The following diagram illustrates the priority groups:



7.3.1.1 Fixed priority jobs: Job scheduling is performed differently for jobs in the fixed-high-priority and fixed-low-priority groups than for jobs with normal interactive priorities. Jobs with priorities in the fixed-low-priority group (0 to PRILOW) and the fixed-high-priority group (PRIHI to 127) execute at fixed priority values. That is, the priority absolutely controls the scheduling of the job for execution relative to other jobs. The job state does not influence the execution scheduling except as to whether the job is in a ready-to-run state or a wait state. A job with a fixed priority is allowed to execute as long as it wishes until a higher priority job becomes active. Jobs having identical fixed priorities are scheduled on a round-robin basis at rates determined by the QUAN0 and QUAN3 parameters.

The fixed-high-priority group is intended for use by real-time programs. See the chapter on real-time program support in the TSX-Plus Reference Manual. The fixed-low-priority group is intended for use by very low priority background tasks. Normal time-sharing jobs should not be assigned priorities in either of the fixed priority groups.

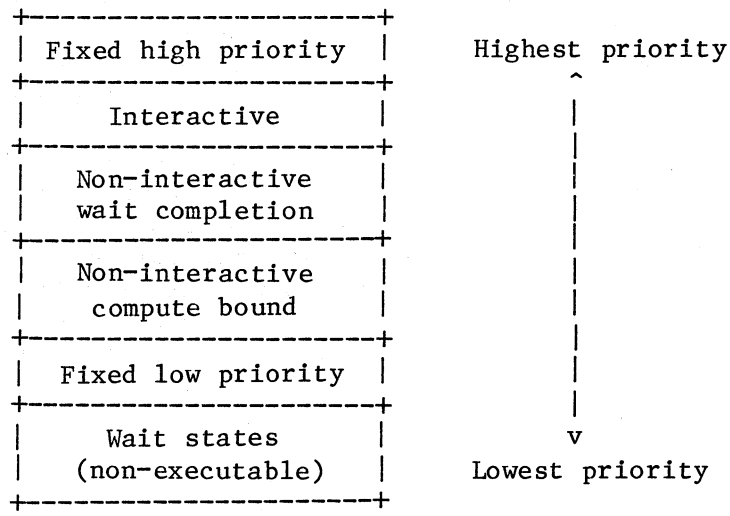
7.3.1.2 Normal priority jobs: The middle group of priorities from (PRILOW+1) to (PRIHI-1) are intended to be used by normal, interactive, time-sharing jobs. Jobs with these assigned priorities are scheduled in a more sophisticated manner than the fixed-priority jobs. In addition to the assigned priority, external events such as terminal input completion, I/O completion, and timer quantum expiration play a role in determining the effective scheduling priority. For these jobs the job state is the primary factor in determining execution scheduling and the user-assigned job priority only influences the scheduling of jobs in the same state.

For most situations, the best strategy is to assign a single priority in the middle of the interactive job priority group to all interactive jobs and reserve the fixed priority groups for real-time or very low priority jobs. The default job priority is specified by the PRIDEF sysgen parameter.

When a job with a normal priority switches to a subprocess, the priority of the disconnected process is reduced by the amount specified by the PRIVIR sysgen parameter. This causes jobs that are not connected to terminals to execute at a lower priority than jobs that are. This priority reduction does not apply to jobs with priorities in the fixed-high-priority group or the fixed-low-priority group. Priority reduction is also constrained so that the priority will never be reduced below the value of (PRILOW+1).

7.3.2 Execution states

TSX-Plus assigns each job a "state" based on actions taken by the job, and external events such as I/O interrupts and timed interval expirations. These states can be grouped into six categories as illustrated by the following diagram:



7.3.2.1 Wait states: Currently, there are seventeen states to identify jobs waiting for events or resources. These jobs are in non-executable states. When a particular event occurs or resource becomes available, the jobs waiting for these events or resources are readily identified by their wait state and are scheduled for execution.

7.3.2.2 Executable states: There are 10 executable job states which can be grouped into five categories: (1) fixed-high-priority; (2) interactive; (3) non-interactive wait completion; (4) non-interactive compute bound; and (5) fixed-low-priority. Jobs that have user-assigned priorities greater than or equal to PRIHI are always in either a wait state or in the fixed-high-priority state. They are never assigned one of the other executable states. Similarly, jobs with user-assigned priorities less than or equal to PRILOW are always in either a wait state or the fixed-low-priority state. Jobs with priorities

between (PRILOW+1) and (PRIHI-1) are in one of the states: interactive, non-interactive wait completion, non-interactive compute bound, or wait.

The job scheduler gives preference to interactive jobs to provide rapid terminal response. Each time a job accepts a character from the terminal the job is classified as "interactive" and the following actions are taken:

1. The job is placed in the highest priority state within the interactive state group.

2. A system timer is started for the job.

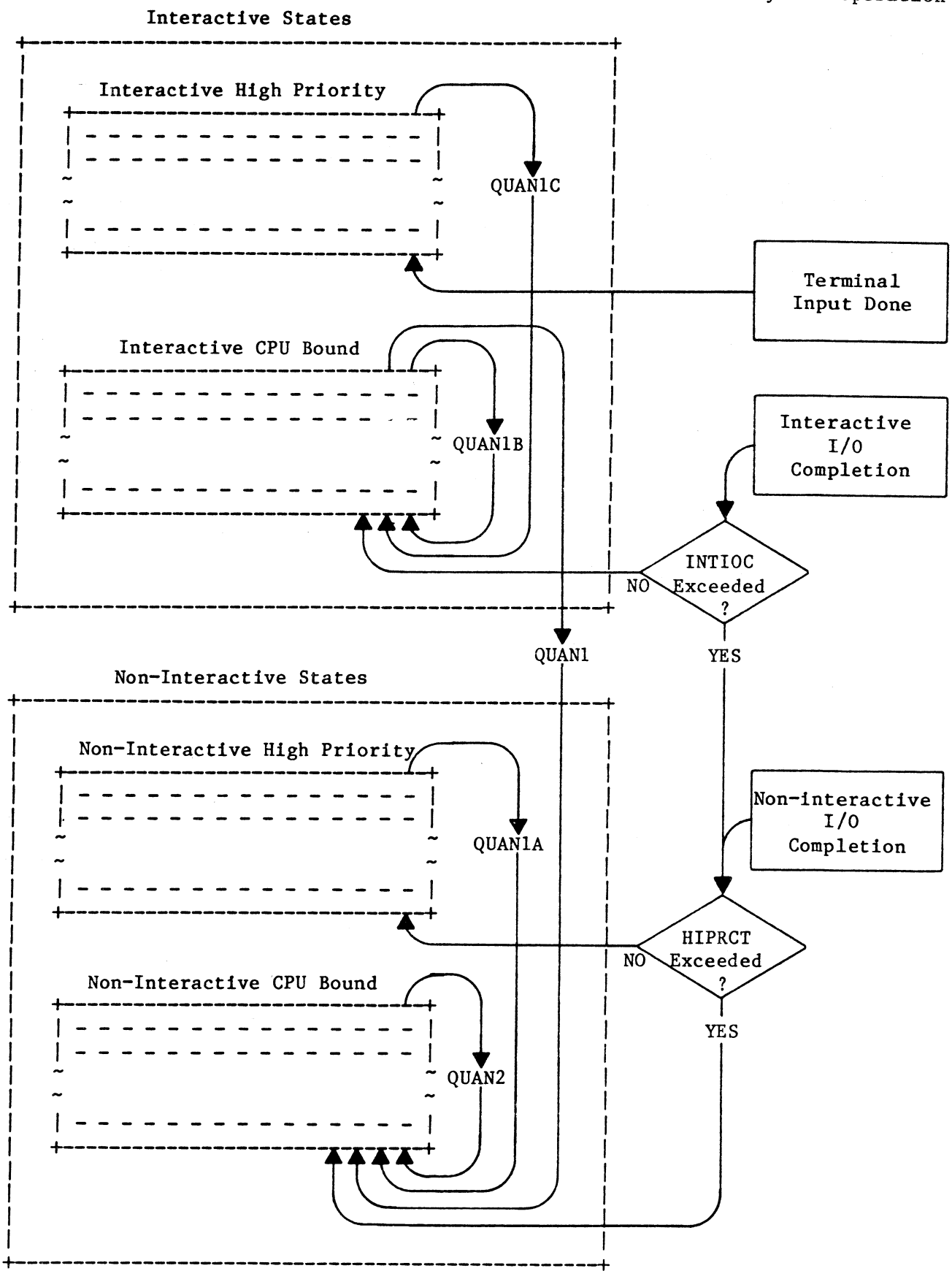
3. The I/O count for the job is set to zero.

The job remains in the highest priority interactive state until it either has executed for QUAN1C units of time or performs an I/O operation. At that time, the job is rescheduled into the next lower execution state in the interactive group (interactive-CPU). On return from an I/O operation (during which the job was probably in an I/O wait state) an interactive job is placed in the interactive-CPU state. Interactive jobs which accumulate a total of QUAN1 units of time or which perform more than INTIOC I/O operations are reclassified as non-interactive and placed in the non-interactive compute bound state.

Non-interactive jobs normally execute in the non-interactive compute bound state. Whenever a non-interactive job waits on a resource (such as an I/O operation), the job is placed in a wait state. On completion of the wait condition, the job is placed in a non-interactive wait completion state for a short period of time. The wait completion state has a higher priority than the normal non-interactive compute state but lower priority than any of the interactive states. The job remains in the wait completion state until it reenters a wait state or executes for QUAN1A units of time at which point it is placed back in the non-interactive compute bound state.

The only way that a non-interactive job can move back into one of the interactive states is by receiving input from the terminal.

The diagram on the following page illustrates how time-slice parameters and external events affect job state transitions.


Interactive States
Interactive High Priority
QUAN1C
Terminal Input Done
Interactive CPU Bound
QUAN1B
Interactive I/O Completion
INTIOC Exceeded ?
NO
YES
QUAN1
Non-Interactive States
Non-Interactive High Priority
QUAN1A
Non-interactive I/O Completion
HIPRCT Exceeded ?
NO
YES
Non-Interactive CPU Bound
QUAN2

7.3.3 Job scheduling algorithm

The job scheduler selects which job to run based on the job states and user-assigned job priorities. The scheduling priority of a job is determined primarily by the priority of the job state and secondarily by the user-assigned priority. In the case of equal state and priority, jobs are scheduled on a first queued - first executed basis. Fixed-high-priority jobs and fixed-low-priority jobs are scheduled solely on the basis of the user-assigned priority value.

The scheduler selects the job to be executed according to the following steps:

1. Select the job in the highest priority state that has the highest user-assigned priority.

2. If this job is not in memory, bypass it and search the job queue in order of decreasing state priority and, within a state, decreasing user-assigned priority looking for a job that is in an executable state and in memory. If there are no jobs in memory in an executable state, then no job is executed until some job enters an executable state or an executable job is swapped into memory.

3. Run the job until: a) the job enters a wait state; b) the allotted time-slice expires; or c) a higher priority job becomes executable.

   a) If the job enters a wait state, remove it from its current queue position and place it in the appropriate wait state queue.

   b) If the allotted time-slice has expired, remove the job from its current queue position and reposition it in the queue based on (1) the priority of its state, and (2) the value of the user-assigned priority. The job is placed behind any other jobs that have the same state and priority. (Note: the quantum expiration may cause the job state to change to a lower-priority state.)

   c) If an external event interrupts an executing job before it either enters a wait state or its time-slice expires, then leave the job in its current state and queue position, but execute the higher priority job. When the interrupted job is resumed, continue its time-slice with the unused remainder of its previous time-slice parameters.

7.4 Job swapping

The role of the job swapper is to keep in memory the highest priority jobs that are in an executable state.

A job swap is initiated when the following three conditions occur together:

1) A job is in an executable state and is swapped out of memory.

2) There is insufficient contiguous free memory space available for the job.

3) There is a job in memory with a lower priority executable state than the out-of-memory job.

Note that the wait states have a lower priority than any executable state. When a job swap becomes necessary, the job with the highest priority executable state that is out of memory is selected to be brought into memory. The lowest priority job that is in memory is swapped out of memory to make room for the job being brought in. If this outswap does not yield adequate free memory space, the next lowest priority job is outswapped and the process is repeated until enough space is made available for the selected job to be brought into memory.

The job scheduler attempts to overlap job swapping time with the execution of jobs that are in memory. The "Swapping-I/O" statistic produced by the SYSTAT command indicates the percentage of time that some job swapping was taking place; the "Swap-wait" statistic indicates the percentage of time that no executable job was in memory and swapping was taking place.

A system parameter (CORTIM) is used to keep executable jobs in memory for a reasonable minimum length of time. As long as the job remains executable, it is not eligible to be swapped out of memory until CORTIM units of clock time have elapsed. If the job enters a wait state (other than waiting for non-terminal I/O completion), then it becomes immediately eligible for swapping.

Jobs are temporarily locked in memory by the system during non-terminal I/O until released by the device handler in order to make the data transfer into the correct job area. If the job has exceeded its time-slice parameter and the job swapper wants to swap it out of memory, then the system "holds" its I/O by preventing it from starting any new I/O operations. Jobs may also lock themselves in memory by using real-time EMT requests.

7.5 Real-time interrupt processing

The real-time interrupt handling facility has two subdivisions: real-time interrupt service routines and real-time interrupt completion routines.

7.5.1 Real-time interrupt service routines

Real-time interrupt service routines provide rapid interrupt response by running at fork level in user mode, but do not maintain the full context of the job. They can execute without requiring a job scheduling cycle, but only a very limited subset of system service calls can be used from within an interrupt service routine. Since real-time interrupt service routines execute at fork level, they are intermixed with system interrupt service fork routines and are queued and executed in order of occurrence. These real-time interrupt service routines will execute prior to any other job regardless of the associated job's priority.

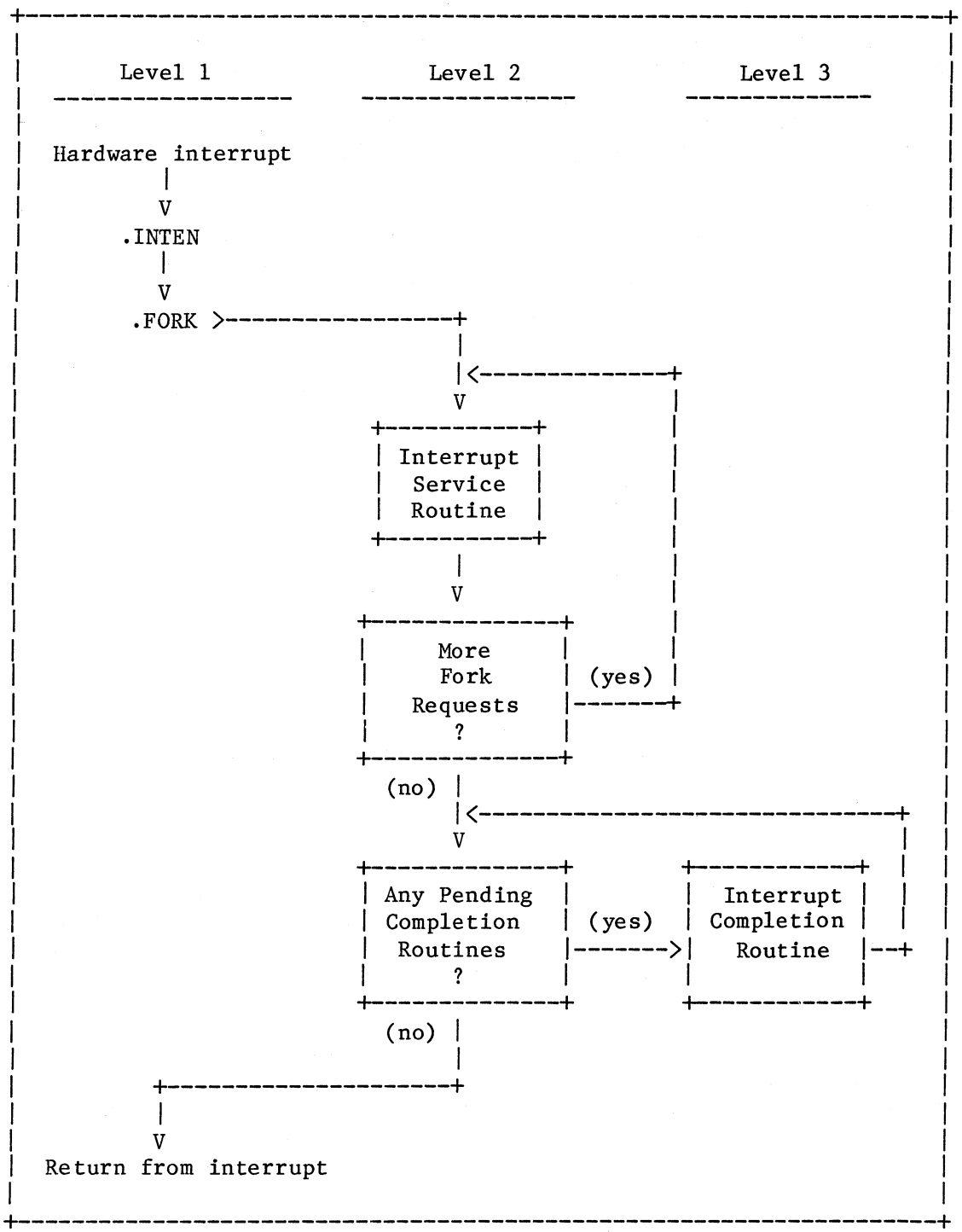
7.5.2 Real-time interrupt completion routines

Real-time interrupt completion routines run with the full context of the job and require a job scheduling cycle before execution. This mechanism does not provide as rapid response as interrupt service routines, but allows normal access to programmed requests from within the completion routine. Each real-time interrupt completion routine has a real-time software priority which is used by the scheduler to compute the execution priority of the real-time completion routine. The priority of the completion routine is calculated by adding the priority specified with the EMT that connects the interrupt to the completion routine to the PRIHI system generation parameter; this priority is constrained by the maximum allowed (127). If a real-time completion routine enters a wait state, then when it resumes execution it returns to the same priority as prior to the wait condition.

A real-time interrupt completion routine may also have a software priority of zero, in which case its execution priority depends on the execution priority of the job. If the execution priority of the job is greater than or equal to PRIHI, then the real-time interrupt completion priority is calculated to be PRIHI and the real-time completion routine is treated the same as above. If the priority of the job is less than PRIHI, then the real-time completion priority is scheduled as a time-shared job in a non-interactive wait completion state.

The following diagram illustrates the processing of an interrupt and shows the relationship between interrupt service routines and real-time interrupt completion routines:

Interrupt Processing



This diagram shows that there are three "levels" of interrupt processing. Level 1 is entered when a hardware interrupt occurs. In this level the processor (hardware) priority is set to 7 which causes other interrupt requests to be temporarily blocked. After some brief interrupt entry processing, the system performs a .FORK operation which queues up a request for processing at fork level and then drops the processor priority to 0. At this time another hardware interrupt can occur, in which case the cycle will be repeated and another request for fork level processing will be placed on the queue.

Level 2 processing is also known as "fork level" processing. This level of interrupt processing services requests that were placed on a queue by the .FORK operation. Hardware interrupts are enabled during this processing and if any other interrupts occur their service requests are placed at the end of the fork request queue. Interrupt service requests are processed serially in the order that the interrupts occurred. Only a limited set of system service calls can be used from service routines running at fork level. One of the valid EMT's is a request to queue a user completion routine for subsequent processing.

Level 3 processing occurs in "job state". That is, the TSX-Plus job execution scheduler selects the highest priority job or completion routine and passes execution to it. During level 3 processing, interrupts are enabled and job execution may be interrupted to process fork level interrupt service routines.

7.6 Window Print Operation

TSX-Plus provides a facility known as Process Windowing (tm) which, if enabled for a process, causes the system to monitor all characters sent to the terminal and maintain an in-memory image of what is currently presented on the terminal screen. This allows the system to restore the terminal display when switching between subprocesses. Process windowing also provides a "print window" function which allows a user to request the current window contents be printed by typing a control character.

The WINPRT program must be executing to process a print-window request. (The WINPRT.SAV program is supplied with the TSX-Plus distribution). This program is ordinarily run as a detached job. A command file named WINPRT.TSX is provided which may be specified with a DETACH command in TSGEN to cause WINPRT to be started as a detached job when TSX-Plus is initiated.

Each time a print-window request is made, a completion routine is executed in WINPRT. This completion routine accesses the current window contents for the job issuing the print request and saves this information in a memory area that is part of the WINPRT job. If multiple print requests occur rapidly, WINPRT captures the window data and queues the print requests for processing in the order in which they occurred.

The number of print requests which can be queued by WINPRT is determined by the amount of memory space available to the WINPRT job. The space required for the program itself and one print request is 11Kb. Each additional print request

requires an additional 6.4Kb. Thus by using a MEMORY command in the command file that starts the WINPRT detached job or by use of the SETSIZ program, you can control the amount of memory used by WINPRT and the number of requests it can queue.

WINPRT must execute with GETCXT privilege. It must also have access to all of the devices to which print window requests will be directed.

If you have problems getting the print window function working, check the following things:

1. Is windowing turned on (SET WINDOW)?

2. Are you using the correct control character to request the window to be printed? Control-B is the default character but it may have been changed by use of the PWCH sysgen parameter.

3. Is the WINPRT program running?

4. Does the WINPRT job have GETCXT privilege and can it access the print device you specified with the SET PRINTWINDOW/DEVICE=device command?

If you still have problems, try running WINPRT from a time-sharing line rather than as a detached job. It will print error messages if any error occurs.

8. SYSTEM TUNING

Since every computer site is unique, there is no single optimum set of parameters for TSX-Plus system generation. Performance depends on both the system configuration and its actual use. It is necessary to analyze the hardware available as well as the type of application programs which are most commonly run. Together with a knowledge of those programs' characteristics and a basic understanding of the performance features of the operating system, decisions can be made to improve system performance. System tuning is an on-going process which becomes more apparent with increased experience.

Since the basic function of the operating system is to execute user programs, the most important tool in tuning system performance is knowledge of the features used by these jobs and the resources available to them.

Above all, have reasonable expectations; do not expect a LSI-11/23 with slow disk devices to perform like a LSI-11/73 or PDP-11/44 with high-speed disks.

Three distinct system concepts interact with each other to affect system performance tuning: memory utilization, job execution scheduling, and I/O optimization.

8.1 Memory utilization

With the drastic reduction in memory price that has taken place in the past few years, and the availability of models of the PDP-11 family such as the 11/23-Plus, 11/24, 11/44, and 11/73 which can address up to 4M bytes of memory, there is a tremendous disparity between the sizes of systems running TSX-Plus. Fortunately, TSX-Plus has the flexibility to run well in small systems and also take full advantage of large memory systems.

The tuning of TSX-Plus is quite a bit different depending on the amount of memory available. From the point of view of system tuning and operation, a "small" system is one which has inadequate memory to simultaneously accommodate all of the time-sharing jobs that are routinely active. A "large" system is one which has more than enough memory to accommodate all active jobs. A "medium" size system is one which has enough memory to accommodate most active jobs and which has some, but not heavy, job swapping. It must be realized that the most careful tuning of a small system will not yield the performance improvement that could be gained from upgrading the system by adding more memory.

In tuning a small system, the primary consideration is to minimize job swapping. This, in turn, reduces to a problem of minimizing the size of the operating system and the amount of space used by frequently run applications.

8.1.1 System memory utilization

The memory utilization of the operating system is discussed and illustrated in the chapter titled System Internals. The components over which you have some control are:

1. Optional features such as the single line editor, generalized data caching, PLAS support, real-time support, etc.

2. The number of device handlers.

3. Space allocated for job tables; this depends primarily on the number of lines generated into the system (real, subprocess, and detached) and, to a lesser extent, on the size of the terminal character buffers allocated for each line.

4. The use of shared run-time systems that allow multiple users to access a common run-time system.

Since the operating system is permanently resident, keep it to a minimum size. Do not include unused device handlers or unused time-sharing line definitions. Do not include more subprocesses or detached jobs than will be actually used. Do not include optional features which will not be used (e.g. PLAS, performance monitoring, real-time support). The TSX-Plus Installation Guide provides specific information about the amount of memory space used by each optional feature and each additional time-sharing line. Features such as the single line editor, PLAS support, and the generalized data cache are not recommended for small systems.

Include a shared run-time if it will be used regularly by more than two jobs. But, remember that shared run-times are permanently resident and are wasting system space when not in use.

Specify reasonable values for system parameters such as terminal I/O and spool buffers. Some experimentation may be necessary to determine what buffer sizes are necessary to achieve satisfactory performance for the job mix in your situation. Balance the use of adjustable system features with the knowledge that excessive job swapping may be caused by overly large system parameter selections.

8.1.2 User program memory utilization

The memory partition allocated to a job under TSX-Plus is dynamic and may change size from time to time. The key to user memory optimization is to set the partition size to the smallest size possible for each program that is run. The amount of memory that is allocated to the job partition can be controlled through three techniques:

1. The MEMORY keyboard command.

2. A TSX-Plus system service call (EMT).

3. The SETSIZ program that can store into a SAV file a value indicating how much memory to allocate for the program when it is run.

The .SETTOP EMT does not change the amount of memory allocated to a job partition. If the partition size is to be changed while a program is executing, a TSX-Plus specific EMT must be used.

The first step in optimizing program memory utilization is to determine how much memory space is actually needed by each application program. This is most easily done by using the MEMORY keyboard command to set the partition size and then attempting to run the application program. By varying the size specified with the MEMORY command you should be able to determine the minimum amount of memory which can be allocated for each program.

Note that most programs either execute or don't depending on whether there is adequate memory available; however, some programs such as the COBOL-Plus run-time system may execute more slowly if there is restricted memory space. Hence, you should not only determine the minimum amount of memory required to run the program but should also note the effect of restricted memory space on the performance of the program.

Once the minimum memory size has been determined for a program, the SETSIZ program can be used to store a value into the SAV file for the program that automatically sets the partition size each time the program is run. A command file named SETSIZ.COM is provided with the TSX-Plus distribution to set appropriate sizes for system utility programs such as PIP, DIR, KED, etc. Note that the SETSIZ.COM file needs to be executed only once - when the TSX-Plus system is installed.

The default partition size (as specified by the DFLMEM sysgen parameter) should be set to a reasonable value.

The SHOW MEMORY command can be used to determine the distribution of memory resources between the system and users.

8.2 Job scheduling optimization

Eight time-slice and two I/O count parameters are used to control job scheduling. The eight time-slice parameters are QUAN0, QUAN1, QUAN1A, QUAN1B, QUAN1C, QUAN2, QUAN3, and CORTIM. The two I/O count parameters are INTIOC and HIPRCT. These parameters are assigned initial values during system generation. Their values can be changed dynamically during the operation of the system by use of a command of the form:

SET parameter value

where "parameter" is the name of one of the ten parameters. Values for the time-slice parameters are specified in 0.1 second units. Operator command privilege is required to change the value of a system parameter. Note that system tuning parameters (QUAN0, QUAN1, QUAN1A, QUAN1B, QUAN1C, QUAN2, QUAN3, CORTIM, INTIOC and HIPRCT) are global to all users and may not be set on a line-by-line basis.

The SET SIGNAL command can be used to monitor the job state transitions and is very useful for selecting values for job scheduling parameters. The form of the SET SIGNAL command is:

SET SIGNAL [NO]parameter

where "parameter" is one of the following system parameters: QUAN0, QUAN1, QUAN1A, QUAN1B, QUAN1C, QUAN2, QUAN3, INTIOC, or HIPRCT.

When signaling has been set for a system parameter, the bell will be rung at the terminal of the job which set the signal each time a job state transition occurs because the job has reached the specified parameter value. This allows the system manager to observe how often the job changes state based on different parameter values. The SET SIGNAL command operates on a line-by-line basis and affects only the line that issued the command.

Signaling may be turned on for any combination of parameters, but each parameter must be specified by a separate SET SIGNAL command. Signaling for an individual parameter may be turned off by specifying "NO" in front of the parameter name. All parameter signaling may be turned off by use of the following command:

SET SIGNAL OFF

When a job receives an activation character from the terminal it is classified as "interactive" and placed in the highest priority state within the interactive state group. The job remains in this state until QUAN1C units of time have passed at which time the job is reclassified into a lower priority state that is still within the interactive job state group. Jobs in this group are scheduled on a round-robin basis every QUAN1B units of time.

If a job performs more than INTIOC I/O operations or exceeds QUAN1 units of time before it receives another activation character from the terminal, it is classified as non-interactive and is placed in the non-interactive compute bound state. Jobs in this state are scheduled on a round-robin basis every QUAN2 units of time. Whenever a non-interactive job waits on a resource (such as an I/O operation), the job is placed in a wait state. On completion of the wait condition, the job is placed in a non-interactive wait completion state which has a higher priority than the compute bound state but a lower priority than the interactive states. The job is allowed to run in the completion state for QUAN1A units of time after which is is placed back in the non-interactive compute bound state.

In selecting values for these parameters, the following guidelines should be considered: It is highly desirable that interactive jobs such as data entry applications and editing programs be classified as interactive through each terminal interaction. Thus, QUAN1 should be set large enough so that the total CPU time used by the application program during one interaction can be completed. Note that if a job performs I/O operations the CPU time counter is suspended (time is not counted while a job is in a wait state) and restarted

(but not reinitialized) when the I/O operation completes. Also, the INTIOC parameter should be set to a value large enough to allow all I/O operations required during a single interactive transaction to be completed.

It is much better to select values for QUAN1 and INTIOC that are too large rather than too small. If the values are too large they will allow long running (non-interactive) programs to be scheduled as interactive slightly longer than necessary. If they are too small, interactive jobs will be reclassified as non-interactive (and given a lower priority) while they are executing an interactive transaction.

The QUAN1 and INTIOC system parameters are two of the most critical scheduling parameters. Jobs are classified as interactive from the time that a character is received from the terminal until QUAN1 units of CPU time are used or INTIOC I/O operations have been performed. The following procedure can be used to select optimum values for these parameters:

1. Issue the following keyboard commands:

   ```
   SET SIGNAL QUAN1
   SET SIGNAL INTIOC
   ```

2. Set INTIOC to a large value by use of the following keyboard command:

   ```
   SET INTIOC 1000
   ```

3. Run an application program whose execution is to be optimized.

4. From a separate terminal, vary the value of QUAN1 by use of the keyboard set command:

   SET QUAN1 value

   For each trial value of QUAN1, enter several transactions to the application program and see if the bell rings at the terminal running the application program. If the bell rings, increase the value of QUAN1 and try again. The optimum value of QUAN1 is slightly larger (add 1 to 5) than the smallest value found which is large enough so that the bell does not ring while processing a transaction.

5. Repeat the process for INTIOC by setting QUAN1 to a large value (e.g., 1000) and varying INTIOC starting with a reasonable value such as 30.

6. Try several values of INTIOC until the smallest value is found which is large enough to keep the bell from ringing while processing a single transaction. The optimum value for INTIOC is slightly larger than this (i.e., add 2 to 10).

7. After the appropriate value for QUAN1 and INTIOC have been determined, the system default values for these parameters may be set by modifying TSGEN and regenerating TSX-Plus.

Note: When performing this type of optimization, choose the most frequent and important type of transactions for the test. Don't worry about longer and less frequent operations such as chaining between separate programs. The performance measurements should be carried out with a variety of application programs. Then use the largest values of QUAN1 and INTIOC found for the various applications as the standard system values. Note that system parameters (QUAN0, QUAN1, QUAN1A, QUAN1B, QUAN1C, QUAN2, QUAN3, CORTIM, INTIOC and HIPRCT) are global to all users and may not be set on a line-by-line basis.

The QUAN1C parameter controls the length of time after each terminal input activation that a job remains at the highest priority interactive state before being dropped down to a lower priority interactive state. The ideal value for QUAN1C is just large enough to allow programs, such as KED, which do single character processing to complete the processing of a single character at the highest priority state. It is not desirable to set QUAN1C large enough to encompass longer editing operations such as cutting and pasting, or moving to the top or bottom of a file.

To select the optimum value of QUAN1C, use the SET SIGNAL QUAN1C command and find that value of QUAN1C which is as small as possible but which does not cause the bell to ring while performing normal text entry to the editor.

The QUAN1B parameter controls the round-robin scheduling of interactive jobs within the same state. Its value is usually not critical but should be in the same range as QUAN1C (typically 1 to 4).

The QUAN2 parameter controls round-robin scheduling of non-interactive, compute bound jobs. In medium to large systems where most programs reside in memory, the value of QUAN2 is not critical and should be set to a reasonably small value in the range 2 to 5. In small systems, the value of QUAN2 should be set large enough to reduce job swapping that could take place when multiple compute bound programs are running. The recommended value for small systems is in the range 10 to 30.

Each time a non-interactive job completes an I/O operation, or finishes waiting on some other resource, the job is given a priority boost. The job remains in the high priority state until either (1) it goes into a wait state again, such as waiting on another I/O operation; or (2) it has executed for QUAN1A units of time, at which time it is rescheduled in the non-interactive compute bound state. The idea is to give the job a chance to start another I/O operation without having to wait its normal turn for service. This allows I/O intensive jobs to keep their I/O active even if there are multiple compute bound jobs also running.

Jobs with the same user-assigned priority in the fixed-high-priority group are scheduled in a round-robin fashion based on the QUAN0 system parameter. If QUAN0 is set to 0 (zero), no round-robin scheduling is done for high-priority jobs. Jobs with the same priority in the fixed-low-priority group are scheduled in a round-robin fashion based on the QUAN3 system parameter. Note that this round-robin scheduling of fixed-priority jobs only pertains to jobs that have the same assigned priority value. A job with a higher fixed priority is never time-sliced with a job with a lower priority.

The CORTIM system parameter controls how long a job is held in memory after being swapped in from disk. Each time a job is swapped into memory, a timer is started for the job. The job is not eligible to be swapped out of memory until either:

1. The job begins executing and enters a wait state (other than non-terminal I/O).

2. CORTIM units of time have elapsed.

Note that a job is never swapped out of memory just because a certain time interval has elapsed. There must be a higher priority job in an executable state out of memory to force a lower-priority job to be swapped. The CORTIM parameter serves as a "throttle" to control the job swapping rate. Increasing the value of CORTIM decreases the job swapping rate but slows the interactive response time. Jobs with user-assigned priorities equal to or greater than PRIHI override the CORTIM parameter and may force outswapping of lower priority jobs regardless of the length of time they have been in memory.

8.3 User program optimization

The TSX-Plus performance monitor feature allows the execution of some application program to be monitored and a histogram produced showing the percentage of time spent in various regions of the program.

The use of single character input activation should be minimized because of the frequency with which this places programs in the high-priority terminal input complete state. The use of no-wait character input may degrade system performance even more since this can place the program in a high-priority terminal input completed state without having received an input character. If at all possible, terminal input should be buffered and completed with a specific activation character (this is normally a carriage return although other activation characters may be defined).

During buffered input, the job is suspended and may even be swapped to disk to allow other jobs to execute. High efficiency terminal mode can be used to reduce the system overhead by eliminating much of the special character processing associated with terminal I/O.

8.4 I/O optimization

TSX-Plus uses three basic techniques to improve system I/O efficiency: (1) overlapping of job execution with I/O wait; (2) device data caching; and (3) device spooling. It is not obvious, but true, that memory size is one of the key factors in optimizing I/O with TSX-Plus.

8.4.1 I/O wait overlap with computation

One of the benefits of a multi-user operating system like TSX-Plus is that system resource utilization is improved by allowing multiple users to be accessing different system resources concurrently.

Whenever one job enters a wait state, waiting for a resource such as an I/O device to transfer data, the TSX-Plus job scheduler looks for another job that is ready to run. The second job might initiate an I/O operation on a different device or might compute and utilize the CPU while the first job is waiting on the I/O operation to complete. Thus, in an ideal situation, the CPU could be utilized 100% of the time as could all of the I/O devices. Generally, 100% utilization of all resources is neither possible nor desirable but the overall system utilization is typically much higher than for a single user system.

The SYSTAT command provides statistics that indicate the degree of overlap that occurred between job execution and I/O. The "User I/O" statistic is the percent of time that some I/O was being performed; the "I/O wait" statistic is the percent of time that the system is idle because there is no executable job and some I/O is taking place. If 100% I/O overlap took place, the "I/O wait" value would be 0 (zero) because there would always be some job to run whenever I/O was active. You can demonstrate this by running a small "loop" program that will execute continuously while other jobs perform I/O. The RESET command can be used to reset SYSTAT statistic values.

In attempting to optimize overall system utilization, the first factor to consider is the number of programs that can fit in memory. Naturally the more programs that are in memory and ready to run, the better the system utilization will be. Also remember that job swapping has multiple negative effects on system utilization: the job being swapped into or out of memory cannot be executed but the memory space is tied up during the swap and cannot be used by any other job; the I/O device to which job swapping is being done is tied up by the swapping and may block I/O operations by the jobs that are in memory and want to run.

The QUAN1A and HIPRCT parameters affect the amount of overlap that occurs between compute-bound and I/O-bound jobs. A non-interactive job is given a priority boost each time it completes an I/O operation. This is done to increase the amount of overlap that occurs between compute-bound and I/O-bound jobs.

For example, consider a system that has two continuously executing compute bound jobs and one I/O bound job. If the job priority was not boosted on I/O completion, the following cycle would occur:

1. Initiate an I/O operation.

2. Place the I/O job in a wait state, waiting for the I/O operation to complete.

3. Alternately execute the two compute bound jobs while the I/O is taking place.

4. When the I/O completes, place the I/O bound job at the tail of the compute-bound queue.

In step 4, the I/O job is placed at the tail of the compute bound queue which means that it will have to wait until both compute bound jobs have used up their time slices before it is allowed to execute and initiate another I/O operation.

Instead of this, the TSX-Plus job scheduler handles the situation as follows:

1. Initiate an I/O operation.

2. Place the I/O job in a wait state, waiting for the I/O operation to complete.

3. Alternately execute the two compute bound jobs while the I/O is taking place.

4. When the I/O operation completes, place the I/O job in a higher priority state which causes it to interrupt the execution of the current compute bound job.

5. The I/O bound job executes for a short period of time and initiates another I/O operation.

6. Put the I/O bound job back in the I/O wait state.

7. Resume execution of the interrupted compute bound job.

The effect is that the I/O job is able to keep the I/O device busy by "stealing" time from the compute bound jobs when each I/O operation completes. However, if there are several I/O intensive jobs they may tend to steal so much time from the compute bound jobs that the compute bound jobs receive little or no time. The HIPRCT parameter is used to control this. After HIPRCT consecutive priority boosts, the I/O job is scheduled at the tail of the compute bound state queue, which means that it will not be executed until all other jobs in the compute bound queue have executed for their full time slice.

If HIPRCT is set to 0 (zero), jobs are never given a priority boost on I/O completion. The recommended value is in the range 5 to 50. The SET SIGNAL HIPRCT command can be used to monitor how often the HIPRCT parameter cuts off a priority boost.

QUAN1A should be set to a small value which is just long enough for I/O intensive jobs to perform completion processing for one I/O operation and initiate another I/O operation. For example, a data base application might have to follow a linked list through an index file to find a selected record. The QUAN1A parameter should be set large enough to allow the program time to locate the forward link in each index block and initiate the I/O operation to read the next block. The SET SIGNAL QUAN1A command can be used to monitor the effect of varying the value of the QUAN1A parameter. The recommended value for QUAN1A is in the range 1 to 4.

8.4.2 Device spooling

Spooling is a technique which intercepts output to slow devices, like printers, directs the output to a disk file and then services the printer as it becomes ready for more data. This mechanism is transparent to the user job and returns the job to an active status more quickly than if the job actually had to wait for the slow device to complete the transfer.

When the operating system services an I/O queue request, it temporarily locks the job into its current memory position so that the data transfer can be correctly fulfilled according to the information in the I/O queue element. When the output is sent to a slow device, this would prevent job swapping until the last data was accepted by the handler and the transfer request satisfied. If several users need access to the system, this could seriously degrade apparent system performance to those users waiting to be activated. However, when a slow device is spooled, then the output is redirected to the system spool file and the transfer completes at the faster rate of disk I/O, returning control to the job and permitting it to be swapped if necessary. In addition, TSX-Plus will always attempt to double buffer the spooled output request if two or more buffers have been defined.

8.4.3 Caching

Caching is a technique for improving system performance by keeping in memory a "cache" of the most recently accessed blocks of data. Each time a read operation is performed a check is made to see if the requested data block(s) are in the cache. If so, the data is copied from the cache buffer to the receiving program buffer and no actual device I/O is done. Write operations update the data in the cache as well as writing to the I/O device.

Caching speeds up read operations so that they are performed at the speed that the CPU can move data around in memory rather than the speed of an I/O device. Write operations are slightly slowed down by caching since updating of the cache must be done as well as writing of the data to the I/O device.

TSX-Plus offers three distinct types of information caching: directory caching, generalized data caching, and shared file data caching.

8.4.3.1 Directory caching: When a program opens an existing file on a disk, it is necessary to determine the location of the file by consulting the file directory on the disk. This results in one or more disk I/O operations each time a file is opened. In order to speed this process, TSX-Plus contains a memory resident cache which contains directory information for a selectable number of files. If one or more jobs open the same file several times, then the ability to locate that file's directory information in the directory cache can eliminate many I/O requests and significantly improve system performance.

The system device directory is always cached; directory caching for other devices can be enabled by use of the "MOUNT" keyboard command. See the TSX-Plus Reference Manual for a detailed description of the MOUNT command.

TSX-Plus manages the entries in the directory cache by retaining those most recently used. When no space is available in the cache buffer to add a new directory entry, the least recently accessed entry is discarded and replaced with the new entry. File operations which change the disk directory information (such as .ENTER, .DELETE and .RENAME) are always "written through" the cache, changing both the directory cache entry and the disk directory. This eliminates the speed advantage on these types of operations, but reduces the chances of data corruption.

It is very important to remember to DISMOUNT a disk when changing removable packs on that device. The DISMOUNT command clears all entries from the directory cache for the device. If this is not done the new pack may be corrupted by use of the (incorrect) directory information maintained in the cache for the previous disk pack. The SHOW MOUNTS command identifies which devices are currently eligible for directory caching. Note that all jobs which have MOUNTed a device must either DISMOUNT it or log off before the device's directory entries are cleared from the cache.

8.4.3.2 Shared file data caching: Shared file data caching maintains memory resident copies of data blocks from files which have specifically been declared to use data caching. After a file is opened in the normal manner, a special system service call must be issued to declare that file eligible for data caching. (Data caching is requested by using the TSX-Plus EMT to request shared access to the file, regardless of the protection level selected.)

When a request is issued to read data from that file, a check is made to see if the requested block(s) are currently in the data cache. If the data is in the cache the data is moved from the cache to the user's program with no disk I/O. Data blocks are maintained in the cache according to frequency of use. When the data cache is full, the least active block is replaced whenever a new block is read. This replacement algorithm is highly efficient for files with indexed organization, like COBOL-Plus ISAM files. As with directory caching, the data cache is always written through. That is, if the information in a block in the cache is changed, then the disk copy of that block is also updated. If shared file data caching is used at all, it is recommended that at least 8 blocks be allocated for the cache. If a large area is available for a data cache, it is recommended that the generalized data caching facility (described below) be used instead of shared file data caching.

The number of blocks allocated in memory for the shared file data cache is controlled by the NUMDC parameter in TSGEN. One way to determine the best value for this parameter is to generate a system with a large number of cache buffers and then use the SET NUMDC keyboard command to vary the number of buffers used while observing the effect on system performance. The SYSMON program can be used to display statistics about shared file data caching operation.

8.4.3.3 Generalized data caching: Generalized data caching maintains memory resident copies of data blocks from devices which are mounted using the keyboard "MOUNT" command. Each time a read operation is performed, the memory resident cache of data blocks is searched to see if the block(s) requested are already contained in one of the data cache memory buffers. If the block is in the memory cache, it is moved directly from the cache buffer without performing any disk I/O. If the block(s) are not within the data cache, they are read into the least recently used data cache buffer(s) and then moved to the requesting job. Write operations update the memory cache as well as writing to the device, thus eliminating the possibility of data loss or corruption.

Unlike shared file data caching, generalized data caching applies to all files that are on mounted devices. This means that SAV files for commonly executed programs such as PIP, KED, TSKMON, and application programs will benefit from the cache as well as program overlay segments, and application data files.

To enable generalized data caching, assign a non-zero value to the CACHE parameter in TSGEN. This causes the data caching code to be included in the generated system and controls the number of blocks of memory allocated for data caching buffers. If data caching is not wanted, set the CACHE parameter to 0 (zero).

A SET command is available to dynamically alter the number of blocks of data held in the data cache. The form of this command is:

SET CACHE value

This command does not alter the amount of space allocated for the data cache (that is directly controlled by the CACHE sysgen parameter), but can be used to cause the system to use less than the full cache area. Operator command privilege is required to use the SET CACHE command. The primary use of this command is to allow the system manager to experiment with different cache sizes to determine the effect on system performance. Once an optimum cache size has been determined, the TSGEN parameter CACHE can be set to this value and the system regenerated. The SHOW CACHE keyboard command can be used to display the current number of blocks currently being used in the data cache.

The effectiveness of the data caching facility increases with the number of blocks allocated for the data cache. In systems with large amounts of memory it is reasonable to allocate several hundred blocks to the data cache. However it is not wise to allocate so much memory space to the data cache that job swapping is significantly increased due to limited memory space for time-sharing users.

The amount of improvement due to data caching also depends on the ratio of the processor (CPU) speed to the speed of the I/O device being cached. The effects of data caching are most pronounced when a fast processor is running with a slow I/O device. Data caching is not recommended for systems which are primarily bound by CPU utilization rather than I/O throughput.

Data caching can have a dramatic effect on the execution of overlayed programs if the cache is large enough to hold the overlay segments. FORTRAN and COBOL-Plus compilation times are typically reduced by 20% to 40% by data caching.

The following table shows typical cache "hit" rates as a function of the cache size (in blocks) for various language processors performing assemblies or compilations:

Cache size versus percent of blocks read from cache while performing assemblies and compilations

Cache Size	MACRO	FORTRAN	F77	COBOL-Plus	DBL	Pascal-2
20	2%	0%	23%	11%	5%	0%
35	3	1	23	21	9	0
50	4	1	23	82	10	5
75	14	2	24	83	25	8
100	36	2	24	84	45	9
150	48	4	27	84	55	90
175	49	51	33	87	84	90
200	50	87	33	87	84	90
250	66	90	34	87	84	90
275	92	92	35	88	84	91
300	92	93	87	88	84	92
400	92	94	94	95	84	92
500	92	97	94	98	84	93

The single job (non-XM) versions of F77 and Pascal-2 were used in making these measurements.

The following statistics for cache hit rates were measured while running a COBOL-Plus program performing 5000 random reads on an indexed organization (ISAM) file containing 44000 records with a 16 byte key.

Cache Size Versus Hit Rate For Reads From ISAM File

Cache Size	Cache Hit Rates for Random Reads
5	24 %
10	32
15	38
20	46
25	50
30	55
40	60
50	64
60	65
70	67
80	70
90	71
100	72
200	79
300	82
400	83
500	84
1000	85

These statistics were gathered by generating a TSX-Plus system with a 1000 block data cache and then using SYSMON to measure the cache hit rate while varying the effective cache size by use of the "SET CACHE nnn" command. It is recommended that a similar procedure be carried out to determine the optimum cache size for a given application program.

The shared-file data caching facility should be used instead of the generalized data caching facility in the following cases:

1. If the primary goal is to speed up application programs which make heavy use of shared files, and the memory space which can be devoted to data caching is limited (less than 50 blocks), then the shared-file data caching facility is more effective than the generalized data caching facility.

2. If the size of the unmapped portion of the TSX-Plus system is such that code for the generalized data caching facility cannot be added. Note that the shared-file data caching facility does not add any code to the unmapped portion of the system.

If the generalized data caching facility can be used, it is recommended that the shared-file caching facility not be used (it is redundant) and the NUMDC sysgen parameter be set to 0 (zero).

8.4.4 Virtual memory handler (VM)

The virtual memory handler (VM) allows memory which is not allocated for use by the operating system to be used as a RAM based pseudo-disk device. Since a memory access is quite a bit faster than a disk access, VM can be use for greater speed in locating and reading files which are frequently accessed.

Since most machines will lose the contents of memory during a power outage, VM should be restricted to read-only, scratch, or executable files. It may be used to speed the execution of heavily overlaid programs or store temporary intermediate sort or work files.

VM is similar to data caching and the following considerations may help you to decide which is best suited to your application:

1. Data is "written through" the cache to the I/O device that is being cached. Since there is no I/O device associated with the VM handler, no I/O takes place on write operations. This means that is faster to write to VM than to a cached I/O device. This could make VM considerably faster for a "scratch" file that has as many blocks written to it as read.

2. Data written to VM is volatile and will be lost when the system is shut down or halts due to hardware or software malfunction.

3. The amount of space in VM is fixed at sysgen time and an attempt to use more space will result in a no-free-space error return. The number of blocks allocated for caching affects the performance of the cache but not the capacity. As long as there is available space on the I/O device, it is accessible through the cache.

4. Caching is automatic and transparent to application programs. VM requires that program and data files be copied to VM and that application programs open files on VM.

5. Data placed in VM is held there until it is deleted or the system is restarted. Data in the cache is dynamic and may be replaced by data accessed more recently by other jobs. Therefore the speed of access to data in VM is guaranteed whereas the speed of accessing data through the cache depends on whether the data is currently in the cache.

9. SYSMON - DYNAMIC SYSTEM DISPLAY UTILITY

SYSMON is a dynamic interactive utility program used to display information about system activities at a VT2xx, VT1xx, VT52, ADM3a, TVI-912, or HAZELTINE type terminal. It is used to assist the system manager in optimizing system resource use and judging the effects of system tuning operations. SYSMON can also be used by the general user to obtain information regarding system load and resource usage. It currently provides dynamic screen displays of CPU and I/O usage; job status; terminal status; message channel status; user time bar chart; CPU time bar chart; directory cache contents; data cache usage; and CL device status.

SYSMON obtains much of its information from tables within TSX-Plus, and thus requires the use of the TSX-Plus EMT to map to physical memory. Therefore, either the user must have the MEMMAP privilege, or this must be provided for SYSMON in the program installation tables. SYSMON is, by default, installed with the MEMMAP privilege, providing that it is run from the system disk. If SYSMON is run without the MEMMAP privilege, the user will receive the message:

```
?SYSMON-F-MEMMAP privilege required to run SYSMON.
```

SYSMON requires the user to have either OPER or SYSPRV privilege to use the message channel display screen. This is done to prevent unauthorized users from discovering file names and other sensitive information that might be passed in a message channel (to RTSORT, for example). If a user attempts to run this display without the required privilege, he will receive the message:

```
?SYSMON-E-You must have SYSPRV or OPER privilege to examine message
channels.
```

To run SYSMON on a HAZELTINE or TVI-912 type terminal, execute the TSX-Plus terminal command

```
SET TERM HAZELTINE
```

before running SYSMON. Similarly, set the terminal type to ADM before running SYSMON on an ADM3a type terminal. Note that if the terminal type is not one of the above terminal types, SYSMON will query the terminal to determine if it is a VT52, VT100, or VT200 class terminal. If the terminal is none of these, SYSMON will refuse to run with the message:

```
?SYSMON-F-Invalid terminal type for SYSMON.
```

9.1 Creating and running SYSMON

SYSMON is automatically created by the command file which links TSX-Plus. Because it depends on global information from TSX-Plus, it must be relinked whenever TSX-Plus is changed. If TSX-Plus is not linked on the system disk, copy the file SYSMON.SAV from the link output device to SY:. Once this is done, you can run SYSMON by typing:

```
R SYSMON
```

If you do not choose to put SYSMON on the system disk, you must use the RUN command with the full device/file specification. SYSMON must be installed with the MEMMAP privilege (the default case) if non-privileged users are to run it. Note that many of the displays shown here are slightly narrower than SYSMON produces; this is done to allow the examples to fit on the printed page.

9.2 SYSMON menu

```
+-------------------------------------------------------------------------+
|                                                                         |
|                        TSX-Plus SYSMON Utility                          |
|                        18-Nov-85      14:14:06                          |
|                                                                         |
|                                                                         |
|                                                                         |
|  Enter selection :                    Sample time :                     |
|  (RETURN to exit)                     (RETURN defaults to 5 seconds)    |
|                                                                         |
|                                                                         |
|                                                                         |
|  1. System status                      6. CPU modes                     |
|                                                                         |
|  2. Process execution status           7. Directory cache               |
|                                                                         |
|  3. Terminal status                    8. Shared file cache             |
|                                                                         |
|  4. Message channels                   9. Generalized data cache        |
|                                                                         |
|  5. User times                        10. CL device status              |
|                                                                         |
+-------------------------------------------------------------------------+
```

Once you have started SYSMON, you will be prompted from this menu for a display number and the sample rate. The minimum sample time is one second; you may set the sample time as high as you wish. Be aware that on slower systems (for example, 11/23 based systems) using a small sample time can have a detrimental effect on system response in general. Once you are in a display, press RETURN to return to the menu.

9.3 System status display

```
                        TSX-Plus SYSMON Utility
                        18-Nov-85      14:14:06

                      *****  System Status  *****


  Total Uptime      00:35:41.0      Cur      Total        System Parameters

  User Time         00:08:11.6     94.1%     22.9%          QUAN0  =     2
                                                            QUAN1  =    20
  I/O Wait Time     00:01:24.8      4.8%      3.9%          QUAN1A =     2
                                                            QUAN1B =     2
  Swap Wait Time    00:00:00.6      0.0%      0.0%          QUAN1C =     1
                                                            QUAN2  =    10
  Idle Time         00:26:04.0      0.0%     73.0%          QUAN3  =    20
                                                            INTIOC =    30
  User I/O Time     00:02:13.2 *    6.7%      6.2%          HIPRCT =    40
                                                            CORTIM =     2
  Swap I/O Time     00:00:00.6 *    0.0%      0.0%          IOABT  =     1

  * - Time is overlapped
```

The system status display provides information on how time is being used in the system and current settings for the dynamically modifiable scheduling parameters.

Three columns of information are presented for the system time usage display. The first is the total time spent in a given activity since the system was booted. (The RESET keyboard command also clears the time counters as if the system had been booted.) The second column is the percentage of total time spent in that activity during the last sample period. The final column is the percentage of time spent in that activity since the system was booted or the keyboard RESET command was last issued.

Seven rows of information are presented. The first is Total Uptime; the amount of time since the system was booted. The User Time is the time used in computation, that is, actual CPU activity. The I/O Wait time is the amount of time user processes spend waiting on information from various I/O devices. The Swap Wait time is the time the system spends waiting for a process swap to complete. The Idle Time is the amount of time the system spends idle. The User I/O time is the amount of time user processes spend performing I/O. Finally, the Swap I/O time is the amount of time the operating system spends swapping user processes in and out of memory. Note that the percentages will

not always add up to 100 percent. TSX-Plus overlaps I/O and execution time, so I/O time might be building up on one process as another process is executing. Also, program execution and particularly, the actual sample rate, are dependent on current system load and number of real-time interrupts taking place. As an example, the display may be interrupted while computing I/O wait time, during which the times for the subsequent display items may change.

Job scheduling parameters are displayed down the right side of the screen. The values of these parameters can be dynamically changed during system operation by use of the keyboard SET command.

9.4 Process execution status display

```
                        TSX-Plus SYSMON Utility
                        18-Nov-85      12:36:29

                  *****  Process Execution Status  *****

 Job Line   Pri  Program  User Name     Size  Pos  Run  State
 --- ----   ---  -------  ---------     ----  ---  ---  -----
  1  1(0)    50  KED      TSX            34   176       Wait-TT input
  3  3(0)    50           Craig          37   Swp       Wait-TT input
  7  7(0)    50           TSX            37   429       Wait-TT input
 10 10(0)    40           Support        37   213       Wait-TT input
 16  Det.    50  RTSORT   SYSTEM         62   Swp       Wait-.SPND/.RSUM
 20 10(1)    50  SYSMON   Support        28   357   I   TT input done
 21  1(1)    40  MACRO    TSX            62   284   C   CPU bound process
```

For each process on the system, the process execution status display provides:

1. Job Number - the TSX-Plus process number assigned to that particular process, subprocess, or detached process.

2. Line - This entry tells the process number for the primary process and the subprocess index of this process for that primary line. If the subprocess index is 0, than the process is a primary process. If Det. appears, then the process is detached, and as such belongs to no line.

3. Priority - the current execution priority for this process.

4. Program being run - the name of the program file being executed.

5. User - the name of the user who owns the process. If there is no user name associated with the process (detached process or no name assigned to an account), then the process's project, programmer number will be displayed.

6. Memory size - the amount of memory that the process is using - expressed in Kb (1024 bytes).

7. Memory position - the start of the process's region in memory, expressed in Kb from the beginning of physical memory. Swp indicates that the process is swapped out of memory.

8. Running state - All processes that are not in a wait state are classified as either interactive or CPU-bound, depending on the nature of the work being done.

9. Current execution state - What the process is currently doing. The execution states are:

State Displayed	Meaning
Real time state	Executing high priority realtime process
TT input - sing char act	Input character just received while in single character activation mode.
TT input done	Activation character just received
TT output buffer empty	Terminal output buffer empty
Interactive compute	Interactive priority process executing
Timed wait completion	Finished .TWAIT or .MRKT request
TT output buffer low	Ready for program to continue output
I/O completion	I/O transfer completed
CPU bound	Normal process executing
Low Priority Computation	Low priority process executing
Wait-I/O queue element	Waiting on I/O queue element
Wait-Mapped I/O Buffer	Waiting for buffer to be available
Wait-Cache Control Block	Locating device cache control block
Wait-Proc Context Block	Waiting for access to the Process CB
Wait-USR data access	Waiting on access to USR data base
Wait-I/O completion	Waiting for I/O operation to complete
Wait-TT output buf full	Terminal output buffer full
Wait-locked block	Shared file block needed by program lock
Wait-system message buf	Waiting for free system buffer
Wait-spool file space	Print spool file currently full
Wait-TT input	Waiting for activation character
Wait-SPD access	Waiting for access to device data base
Wait-spool entry	Waiting for spool file control block
Wait-message	Waiting for a message
Wait-.SPND/.RSUM	Waiting for .RSUM request after .SPND
Wait-timed interval	Waiting for interval to finish
Wait-memory expansion	Waiting for memory expansion to complete

9.5 Terminal status display

```
                          TSX-Plus SYSMON Utility
                          18-Nov-85      12:37:08

                       *****  Terminal Status  *****

 Process number =  1  (DL)
 TT line name   = Console
 TT type        = VT100, logged on
 SL status      = Enabled, KED, NOTTYIN, SUBSTITUTE, Active
 SET TT options = DEFER ECHO NO8BIT NOFORM NOFORM0 NOGAG LC PAGE NOPHONE
                  NOQUIET SCOPE NOSINGLE NOSYSPASSWORD TAB NOTAPE WAIT

 Speed . . . . . . . . . . 9600        Command file input  . . . disabled
 Parity  . . . . . . . . . none        Echo LF after CR  . . . . enabled
 Bits per character  . . . 8           Escape-letter activation  disabled
 XOFF sent . . . . . . . . no          High-efficiency mode  . . disabled
 XOFF received . . . . . . no          Non-wait .TTYIN . . . . . disabled
 Rubout filler character .  " "         Single char activation  . disabled
 Field width activate  . . 0           Transparent output  . . . disabled
 Field limit activate  . . 0           Subprocesses  . . . . . . enabled
 UCL setting . . . . . . . middle      Default editor  . . . . . KED
 Activation characters . . none
```

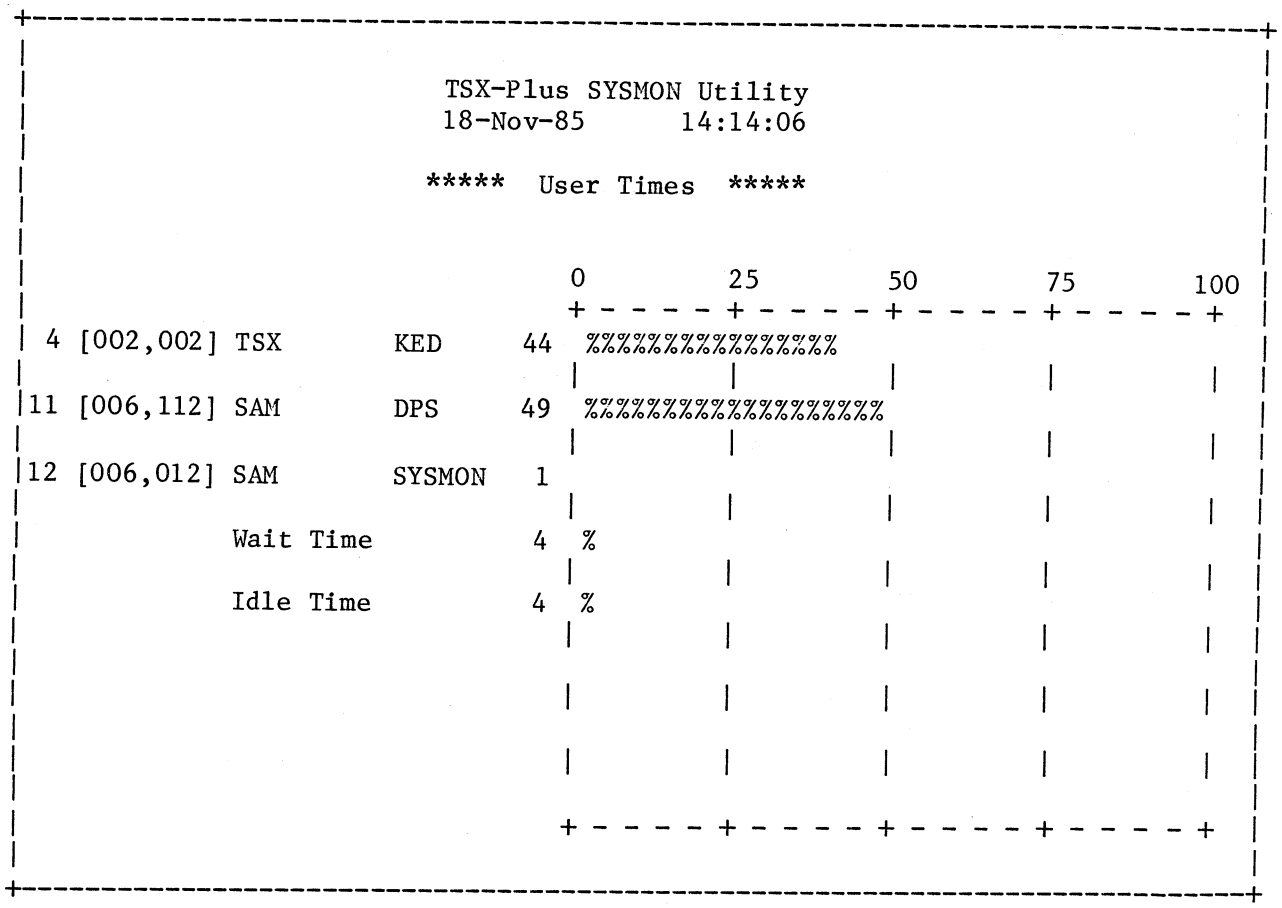
The terminal status display shows the parameters that are currently set on a given terminal line, various terminal characteristics, the terminal's owner, the type of process the terminal is connected to, and if an ASCII DC3 (control-S) has been transmitted or received. This display will prompt you first for a process number - this can be obtained from the TSX-Plus SYSTAT command or the process execution status display. If you simply type return at this point, you will see the information on your own process. If you enter 0, you will get a cycling display through all of the processes. For information on the parameters displayed, see the TSX-Plus Reference Manual.

9.6 Message channel display

```
                          TSX-Plus SYSMON Utility
                          18-Nov-85      14:55:13

                       *****  Message Channels  *****

 Job   Channel   Message
 ---   -------   -------
 15    RTSORT    SYS:SLSANL.DDF/IT:DRA/SI:250/EOF:134/KEY:CA2.2:DA58.2:DA60.3:
                 CA222.2:CA224.2:CA16.10:DA91.2:DA87.4
 15    RTSORT    SYS:GHGARG.DDF/IT:DRA/SI:250/EOF:134/KEY:CA222.2:CA224.2:DA87
                 .4:CA16.10
 13    TSTMSG    [Message request - completion routine]
  3    TGTPRG    [Message request - process waiting]




      MAXMC =   5       MSCHRS = 200      MAXMSG =   6      MAXMRB =  10
```

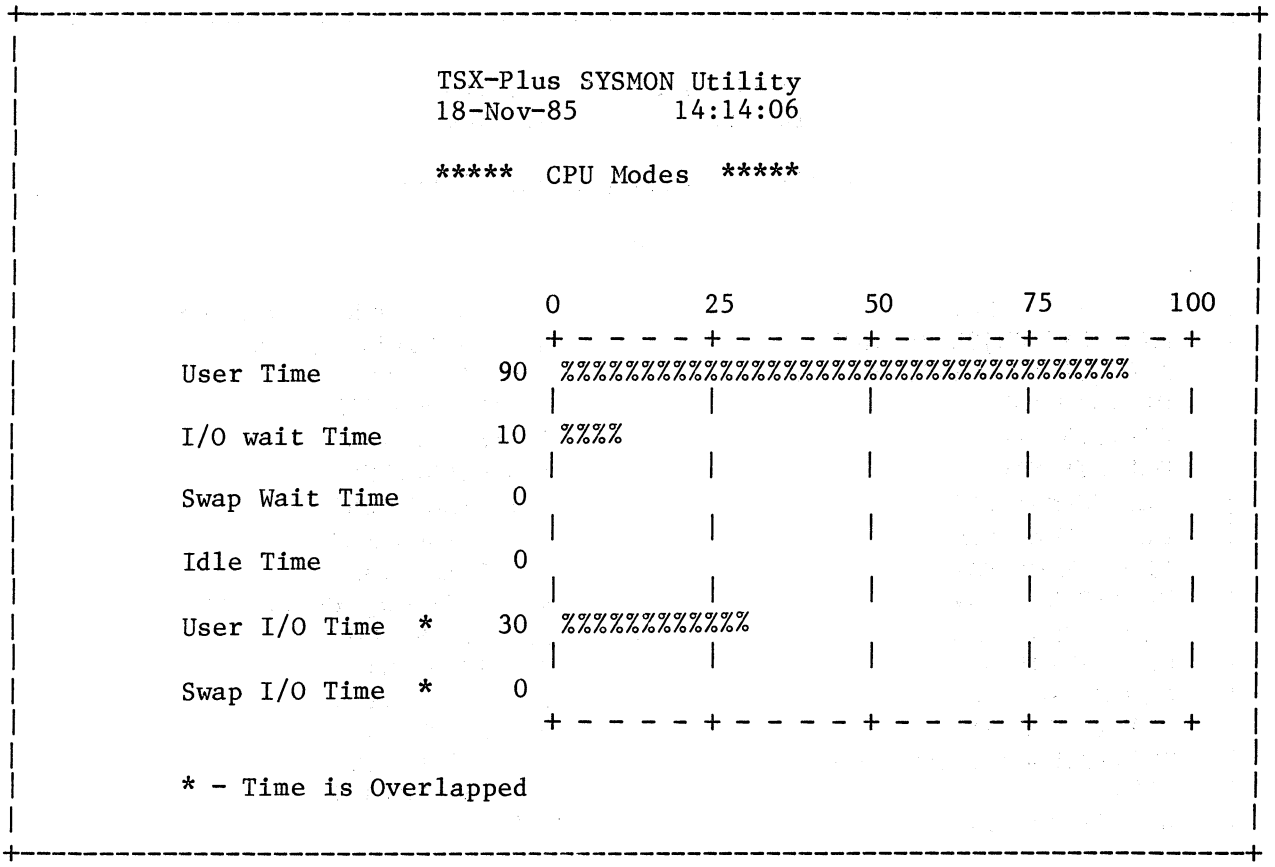
The message queue display shows waiting messages and their respective channel names. This information is useful to verify correct message channel usage, and to debug programs using message channels. In this example, a sort command string is sent on the RTSORT message channel without RTSORT running to verify that the proper information is being sent (the message will remain in the channel until read). Similarly, you can verify that a process is waiting on the proper channel, or that a completion request is active. For more information on message channels see the TSX-Plus Reference Manual.

9.7 User times display



The user times display shows, for each process that used at least one percent of the CPU time during the last sample period: the process number, project-programmer number, user name, program running, percentage of time used by that process during the last sample period, and a bar graph depiction of that percentage. This display is useful for determining how CPU time is being used, and the relationship between process time, wait time (which includes both I/O wait and swap wait time), and idle time.

9.8 CPU modes display



The CPU modes display shows the same information about sample period time usage that is shown in the system status display, however, it is shown here in a bar chart format.

9.9 Directory cache display

```
                        TSX-Plus SYSMON Utility
                        18-Nov-85      14:14:06

                     *****  Directory Cache  *****

                                  Page 1

   DL0:TSKMON.SAV                        RK0:[UTIL.WORKIT]OUT.DSK
   RK0:[SYSMON]SYSMON.SAV                RK0:[UTIL]WORKIT.DSK
   RK0:[UTIL..OUT]SUPER.TXT              RK0:UTIL.DSK
   DL0:KED.SAV                           RK0:SYSMON.DSK
   DL0:DIR.SAV                           RK0:SYSMGR.DSK
   DL0:PIP.SAV                           RK0:COBOL.DSK
   DL1:TSKMN1.OBJ                        DL1:TSKMN3.OBJ
   DL1:TSKMN2.MAC                        DL1:TSKMN3.MAC
   DL0:TSXUCL.DAT                        DL1:TSKMN3.BAK
   DL0:TSXUC..SAV                        RK0:[SYSMON]SYSMON.COM
   DL0:SYSMAC.SML                        DL0:STDASN.COM
   DL1:TSKMN1.MAC                        DL0:SU05.TSX
   DL0:MACRO.SAV                         DL0:ACCESS.TSX
   DL1:TSKMON.BAK                        DL0:LOGON.SAV
   RK0:[SYSMGR]SYSMON.NEW                DL0:SU05B.TSX
   DL0:DEF.COM                           DL1:TSKMN2.BAK
   DL0:SYSMON.SAV                        DL0:DUP.SAV
```

The directory cache display shows what file entries are in the directory cache. This information can be used to determine what files are being used frequently and to determine the best size for the directory cache. Logical disks are shown in square brackets; if the file in the entry is nested more than two logical disks deep, each omitted intervening logical disk is denoted with an extra period.

9.10 Shared file data cache display

```
                    TSX-Plus SYSMON Utility
                    18-Nov-85      14:14:06

              *****  Shared File Data Cache  *****


                                          Cur        Total

          Reads from shared files         240         6818

          Reads from data cache           196         4924

          Percent reads from cache         81 %         72 %

          Writes to shared files           30         3328

          Writes through cache             28         3216

          Free shared file channels                     25

          Blocks in cache (NUMDC)                        0
```

The shared file data cache display shows information on utilization of the TSX-Plus file locking facility. The NUMDC parameter controls the number of buffers used for shared file data caching. The I/O counters can be reset using the TSX-Plus monitor RESET command. This may be necessary, as the I/O counts are stored in TSX-Plus as sixteen bit (single word) integers; these tend to overflow after a large amount of processing. The information presented here can be useful in tuning the data cache. This is done by setting NUMDC to various values and observing the speed of the I/O and the percentage of I/O being done out of the data cache. Also, other effects can be noted, such as the effect of two programs doing shared file I/O, and the speed of I/O on various devices.

9.11 Data cache display

```
                        TSX-Plus SYSMON Utility
                        18-Nov-85      14:14:06

                  *****  Generalized Data Cache   *****


                                                Cur        Total

            Reads from mounted devices           57         7968

            Blocks read from mounted devices    292        55629

            Blocks read from cache              292        48468

            Percent blocks read from cache      100 %        87 %

            Writes to mounted devices           101         4556

            Blocks written to mounted devices   103         8051

            Blocks updated in cache             103        15212

            Data cache size (CACHE)                         1000
```

The data cache display shows information on utilization of the TSX-Plus generalized data caching facility. The information presented here can be useful in tuning the data cache. This is done by using the SET CACHE command to enable various cache sizes, and observing the effect on the cache hit ratio. The intent is to allow as much memory for the data cache as may be effectively used, but to also leave as much free memory as possible for user processes to minimize process swapping. In this context, swapping activity includes both swapping of processes in and out of memory, and the moving of processes around in memory to enable all processes to get the memory they need when process sizes are changing. Note that both the SET CACHE command and the RESET command will reset the cache counters.

9.12 CL device display

```
                          TSX-Plus SYSMON Utility
                          18-Nov-85      12:38:02

                       *****  CL Device Status  *****

  CL Unit Number    =  0    (Not spooled)
  Assigned to line  =  4    VAX link    (DH- 2)
  In use by         = 20    Support

  CL SET Options    = CR DTR NO8BIT NOFORM NOFORMO LC TAB CTRL LFIN NOLFOUT
                      NOBININ NOBINOUT

  Line Speed . . . . . . . .  9600        XOFF sent . . . . . . . . .   no

  Parity . . . . . . . . . .  none        XOFF received . . . . . . .   no

  Bits per character . . .       8        Read pending . . . . . . .    yes

  Line Width . . . . . . . .     0        Write pending . . . . . . .   no

  Page Length . . . . . . . .   66        Input ring buffer used  .     0

  Skip at bottom of page  .      0        Break being sent  . . . .     no
```

The CL device status display shows the CL device settings, which process has the device allocated or actually in use, and various device status values. To obtain a cycling display of all the CL devices, type return when prompted for the device number.

9.13 Exiting SYSMON

To exit SYSMON, type RETURN to leave the current display, and type RETURN to the display number prompt. This will cause SYSMON to exit, and will clear the screen.

Appendix A - Startup Error Messages

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

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 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 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
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 (see the SQUEEZE command in the RT-11 User's Guide) until enough contiguous free space is available.

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 (see the SQUEEZE command in the RT-11 User's Guide) until enough contiguous free space is available.

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:INDTMP.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 (see the SQUEEZE command in the RT-11 User's Guide) until enough contiguous free space is available.

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 (see the SQUEEZE command in the RT-11 User's Guide) until enough contiguous free space is available.

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
: 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 40Kb. 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 (see the SQUEEZE command in the RT-11 User's Guide) until enough contiguous free space is available.

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 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.

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 B - SYSTEM ERROR MESSAGES

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
```

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.

JMO-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 1 (one) 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 0 (zero) 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.