TSX-Plus
System Manager's
Guide

s&h computer systems, inc.

TSX-Plus
System Manager's
Guide

s&h computer systems, inc.

TSX-Plus
System Manager's Guide

Third Edition
First Printing -- February, 1984

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

CONTENTS

Chapter 1

DISTRIBUTION KIT 1

Chapter 2

SYSTEM GENERATION 3
Assembling the TSGEN module 4
Setting parameters in TSGEN 5
General parameters 5
Device spooling parameters 15
Record locking parameters 16
Message communication parameters 17
Real-time program support parameters 18
Performance monitor parameters 19
Shared run-time systems 19
Time-sharing line definitions 20
Defining start-up files for detached jobs 26
Line definition example 27
Assembling the modified TSGEN module 28
Linking TSX-Plus 28
Starting TSX-Plus 28
Device handlers for TSX-Plus 31
Virtual memory handler (VM) 31
VTCOM/TRANSF support and XL handler 32
Building device handlers 33
Patching device handlers for use under TSX-Plus 34
Device handler restrictions 36
.TIMIO and .CTIMIO requests 37
Setting the memory allocation for system programs 37

Chapter 3

SYSTEM AND FILE ACCESS SECURITY 39
Start-up command files 39
Log-off command files 40
The RUN/LOCK switch 40
The ACCESS command 41
The SET MAXPRIORITY command 42
Operator privilege 43
Use of the LOGON facility 44

Chapter 4

ACCOUNT AUTHORIZATION PROGRAM 45
Command summary 46
Authorizing a project-programmer number 46
Deauthorizing accounts 48

Listing account status 48
Listing account usage statistics 49
Creating a charge information file 49
Resetting account usage statistics 50
Exiting from the account authorization program 50
AUTCVT program 51

Chapter 5

SYSTEM OVERVIEW 53
Memory organization 53
Physical layout of TSX-Plus 55
User memory 57
I/O mapping 57
Job scheduling 58
Job priorities 58
Execution states 59
Job scheduling algorithm 63
Job swapping 64
Real-time interrupt processing 64
Interrupt service routines 64
Interrupt completion routines 65

Chapter 6

SYSTEM TUNING 69
Memory utilization 69
System memory utilization 69
User program memory utilization 70
Job scheduling optimization 71
User program optimization 75
I/O optimization 76
I/O wait overlap with computation 76
Device spooling 78
Caching 78
Virtual memory handler (VM) 83

Chapter 7

SYSMON - DYNAMIC SYSTEM DISPLAY UTILITY 85
Creating and running SYSMON 85
SYSMON menu 86
System status display 87
Job execution status display 89
Terminal status display 91
Message queue display 92
User times display 93
CPU modes display 94

Directory cache display 95
Shared file data cache display 96
Data cache display 97
Exiting SYSMON 97

Appendix A

STARTUP ERROR MESSAGES 99

Appendix B

SYSTEM ERROR MESSAGES 103

Appendix C

DEVICE CSR AND VECTOR ADDRESS TABLE 105

Appendix D

DEVICE DRIVER SOURCE LANGUAGE PATCH FILES 107

Appendix E

SYSTEM SIZE CALCULATION 113
Size of system features 113
Device handler sizes 115

1. TSX-Plus DISTRIBUTION KIT

The TSX-Plus distribution package you have received should contain the following items:

1.* TSX-Plus Reference Manual which describes the features of TSX-Plus.

2.* TSX-Plus System Manager's Guide which provides information needed by the system administrator such as how to generate a system.

3.* TSX-Plus installation guide.

* Note that only new orders include these three manuals. They are not automatically included with updates.

4. TSX-Plus release notes

5. A magnetic medium (reversible RX01 diskette, RL01 or RL02 cartridge, or 1600 bpi magnetic tape) containing at least the following files:

AUTCVT.SAV	Program to convert accounting files to new (v4) format.
CCL.SAV	SAV file of CCL command processor.
DTSUB.MAC	Subroutines to perform record locking for DIBOL.
FILTIM.SAV	Program to obtain file creation time.
FTSUB.MAC	Subroutines to access RTSORT from FORTRAN.
LOGON.SAV	SAV file of TSX-Plus logon program.
SETSIZ.COM	Command file to set memory size of system programs.
SETSIZ.SAV	Program to store memory size info. into SAV files.
SYSMON.OBJ	Dynamic system status display program object module.
SYSODT.REL	Program used by system developers to debug TSX-Plus.
TSAUTH.SAV	TSX-Plus account management program.
TSGEN.MAC	Macro source file of TSX-Plus parameter module.
TSLNK3.COM	Command file used to link TSX-Plus under RT-11 3B.
TSODT.OBJ	Object file for TSX-Plus ODT debugging program.
TSODT.REL	Relocatable copy of TSODT debugging program.
TSXDB.SAV	Program used by system developers to debug TSX-Plus.
TSXLNK.COM	Command file used to link TSX-Plus.
TSXPM.SAV	TSX-Plus performance monitor reporting program.
TSXUCL.SAV	Program to process user-defined commands.

6. Device handler related files:

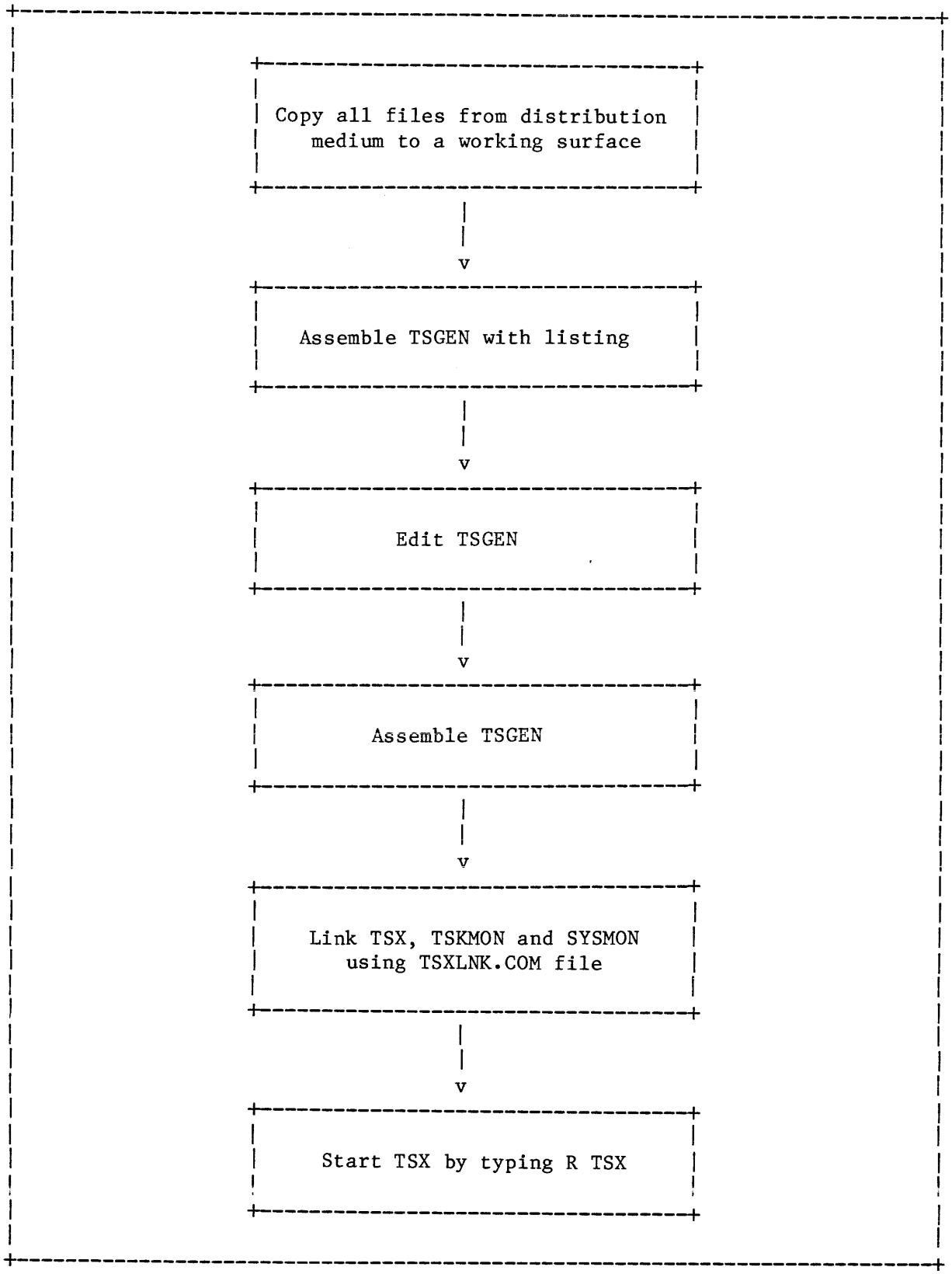
   a) The following device handlers, which, if necessary, have already been patched and are ready for use: CR, CT, DD, DL, DM, DP, DS, DT, DU, DX, DY, LP, LS, MM, MS, MT, NL, PC, RF, RK, VM, XL; all with the extension .TSX. Note that the VM provided is not the DEC VM handler.

b) The following source language patch files are provided for the rare situations in which it is absolutely necessary to rebuild the distributed DEC device handlers: 1) for RT-11 V5 sources --- DDTSX, DMTSX, DXTSX, DYTSX, FSMTSX, TJTSX, TMTSX, TSTSX; 2) for RT-11 V4 autopatch level E or later sources --- DDV4, DLV4, DMV4, DXV4, DYV4, FSMV4, TJV4, TMV4, TSV4. All files have the extension .SLP.

7. The following object modules are used together with your edited and assembled TSGEN by the command file TSXLNK to build the executable programs TSX and TSKMON: TSX1, TSX2, TSTTY, TSEM2, TSPLAS, TSUSR, TSSPOL, TSLOCK, TSMSG, TSRTX, TSMIO, TSSLE, TSEXC2. TSXBND is used with TSLNK3 to build under RT-11 V3B. SYSMON is used to build the SYSMON utility.

The process of generating a TSX-Plus system is not long or difficult. If you understand what you are doing you can probably generate the system in 15 to 30 minutes. However, if you are not already familiar with TSX-Plus, before you begin the system generation process you should do two things. First, you should read the TSX-Plus Reference Manual. There are a number of features provided by TSX-Plus that are not available in standard RT-11 (deferred character echoing, virtual lines, and detached jobs, to name a few). It is necessary to understand the function of these features before you can perform a system generation. Secondly, you should determine the device status register and interrupt vector addresses of the communication equipment that will be used by TSX-Plus. Once you have done this you can proceed with the TSX-Plus system generation as described in Chapter 2 of this manual. If you would like a simplified, semi-automatic guide to system generation then use the TSX-Plus Installation Guide.

2. TSX-Plus SYSTEM GENERATION



The process of generating a TSX-Plus system tailored to the needs of a particular installation consists of 4 steps:

1. Assembling the TSGEN module with listing.

2. Editing parameters in the TSGEN module.

3. Assembling the TSGEN module.

4. Linking the TSX-Plus object modules to form the executable files TSX.SAV, TSKMON.SAV, and SYSMON.SAV.

2.1 Assembling the TSGEN module

The first step in building a TSX-Plus system is to use the MACRO assembler to assemble TSGEN.MAC with a listing. The command to do this is:

```
.MACRO/LIST TSGEN
```

The files SYSMAC.SML and MACRO.SAV must be present on the system device ("SY") during the assembly. These files are supplied by DEC with RT-11.

The TSGEN module of TSX-Plus is supplied in source form. TSGEN contains no executable code, but rather contains the definitions of parameters and tables that are used by TSX-Plus. In building a TSX-Plus system, the RT-11 KED, K52, TECO, or EDIT editor program is used to set appropriate values for parameters in TSGEN. This module is then assembled and linked with the other TSX-Plus object modules. Each of the parameters found in TSGEN is described below. Note that:

1. Numeric values are assumed to be octal unless the number is terminated with a decimal point.

2. When using editors that recognize them, the TSGEN module is divided into several "pages" by form-feed characters.

3. On each parameter needing a file name, the file name is specified as a RAD50 string in the format <DevFilnamExt> with no punctuation (i.e. no colons or periods) and with spaces where there is no character. For example SY:A.TSX would be specified as <SY A TSX>. Note that this does not hold true for the DETACH parameter.

2.2 Setting Parameters in TSGEN

Once a listing of TSGEN is available, save a copy of the original TSGEN.MAC file, then use an editor to set the appropriate parameter values for the system being generated. The beginning of the parameter section of TSGEN.MAC can be easily located by searching for a string of three equal signs.

2.2.1 General Parameters

Parameter	Meaning

SWDBLK This is the name of the file that will be used to hold programs swapped out of memory by TSX-Plus. The default name is "SY:TSXSWP.TSX". The first three characters of the file specification may be changed to direct the swap file to some other device. The size of the job swap file is determined by the number of time-sharing lines and the amount of memory each may use.

SPLBLK This is the name of the file that holds output directed to spooled devices. The file name must be supplied even if there are no spooled devices. The default file name is "SY:TSXSPL.TSX". Note that it is possible to place the swap and spool files on separate devices. The size of the spool file is determined by the SPOOL macro (see below).

RSFBLK This is the name of the PLAS (Program's Logical Address Space) region swap file. This file is used to store memory regions obtained by PLAS when they are swapped out of memory. The default name is "SY:TSXRSF.TSX". The first three characters of the file name may be changed to direct the PLAS region swap file to some other device. PLAS regions are used by programs that have virtual overlays or virtual arrays. The size of the PLAS region swap file is specified with the SEGBLK parameter (see below).

HIMEM This parameter is used to specify the maximum amount of memory that can be used by any job (exclusive of PLAS regions, such as virtual arrays and virtual overlays). The value is specified in terms of k-bytes. The maximum value that may be specified is 64 (Kb). The value of this parameter does not affect the size of the generated TSX-Plus system; however, it does affect the size of the TSX-Plus swap file whose size is approximately:

File size (blocks) = (Total lines) * (HIMEM+4) * 2

DFLMEM This parameter specifies the default memory size to be allocated to a job when it logs on. Specify the value as number of k-bytes. After a line is logged on, the "MEMORY" command may be used to alter the number of kilobytes of memory allocated to the job. The value for this parameter must not be greater than the value for the HIMEM parameter.

SEGBLK This parameter specifies the number of 512-byte blocks to allocate for the swap file that is used for extended memory PLAS (Program's Logical Address Space) regions. These regions are used by programs that have virtual overlays or virtual arrays. The name of the PLAS region swap file is specified with the RSFBLK parameter. If the system is generated as a non-swapping system (SWAPFL=0) then PLAS regions must all fit in memory, no region swap file is allocated or used but the SEGBLK parameter must be set to a non-zero value to cause code to support the PLAS facility to be loaded with the system. Note that this parameter specifies the total space in the PLAS swap file for all extended memory regions in use at any time by all jobs. For example, if a system is to support a maximum of 4 jobs each of which may use 50 Kb of PLAS regions, the total space required is 200 Kb (4 * 50) which requires 400 blocks (1 Kb = 2 blocks) in the region swap file. Actually the file should be allocated with more space than this, since free space in the file may become fragmented as regions are allocated and deallocated. Setting SEGBLK to 0 (zero) disables use of PLAS.

SWAPFL This parameter controls whether TSX-Plus is allowed to swap jobs to disk if insufficient memory is available to hold all active users. The normal case (SWAPFL=1) allows TSX-Plus to do job swapping. SWAPFL can be set to 0 (zero) in special situations such as when a small number of lines are being supported on a floppy disk based system that does not have room for a swap file. If SWAPFL is set to zero the following actions occur:

1. No disk swap file is created.
2. A line will not be allowed to log on if there is insufficient free memory space to support it.
3. Each job is allocated a memory size equal to DFLMEM (default job memory size).
4. Neither the MEMORY command nor EMTs to change the job size can be used.
5. Extended-memory PLAS regions can only be created if there is adequate contiguous free space in memory for them. No PLAS swap file is created.

BUSTYP This parameter defines the machine bus structure for TSX-Plus. There are two possible machine bus structures supported by TSX-Plus - the QBUS (LSI) and the UNIBUS. Select QBUS for 11/23, 11/23-Plus and 11/73, and UNIBUS for 11/24, 11/34a, 11/44 and 11/60.

EXTMCH This parameter, when set equal to 1, enables 22-bit addressing for the 11/23-PLUS, 11/73, 11/24, and 11/44 model CPU's. This should be used when the machine has more than 256 Kb of memory installed. This feature requires the use of 22-bit extended memory mapping for the QBUS or the UNIBUS. If your system (CPU, backplane, and memory) does not support 22-bit addressing, then set this parameter to 0 (zero). It is possible with TSX-Plus to use Q-BUS DMA device controllers which only support 18-bit addressing, however because this is supported by transferring the data through an intermediate system buffer this type of I/O suffers some speed degradation. For this reason it is strongly recommended that on Q-BUS systems the system device controller and handler support 22-bit addressing if the system is to be used with more than 256 Kb of memory (EXTMCH = 1). See the description of the MAPIO modifier to the DEVDEF macro for further information on system mapping of I/O transfers. The only Q-BUS DMA device handlers supported by Digital for 22-bit addressing are DL, DU and MS.

MEMSIZ This parameter controls the maximum memory available for TSX-Plus system use. The value is the memory upper limit size specification expressed in number of k-bytes. Memory above this upper limit will not be used by the operating system. If the MEMSIZ parameter is set to 0 (zero), TSX-Plus will use all available memory on the machine. To disable the use of extended memory, set MEMSIZ to 248 or less (but greater than zero). On machines with a large amount of memory, it is convenient to set an upper limit on the amount of memory to by used by TSX-Plus so that the virtual memory handler (VM) may use the remainder as a RAM based pseudo disk device. This is especially useful for compiler intermediate files. See the section on the VM handler for more information on use of VM with TSX-Plus.

INIABT This parameter controls the action taken by TSX-Plus when certain errors are detected during system initialization. If INIABT is set to zero, TSX-Plus ignores the error and continues running. If INIABT is set to one, TSX-Plus aborts the initialization and prints an error message. The following initialization errors are controlled by the INIABT flag:

1. A device that was specified in TSGEN is not installed in RT-11 or does not have a TSX handler on the system disk.
2. A time sharing line that was generated into TSX-Plus is not installed on the machine.
3. A shared run-time system file could not be found during startup.

IOABT This parameter controls the action taken by TSX-Plus when a job terminates execution. If IOABT is set to zero, TSX-Plus will wait for all outstanding I/O pending for the job to complete before the job is actually terminated. If IOABT is set to one, TSX-Plus will call the handler abort entry point for all outstanding I/O pending for the job. This parameter is usually set to zero. The SET IO command may be used to dynamically alter this parameter during system operation. See the TSX-Plus Reference Manual for more information on the SET IO command.

U$CL This parameter controls whether or not support for user-defined keyboard commands is included in the system. If U$CL is non-zero, TSX-Plus calls on the TSXUCL program to process user-defined commands. If U$CL is zero, user defined commands are not supported by the system. Note that if U$CL is set non-zero, the TSXUCL.SAV file should be on the system disk when TSX-Plus is started.

UCLMNC This parameter sets the maximum number of user-defined commands that may be declared by each job. It also determines the size of the file used to store these definitions (SY:TSXUCL.TSX). The size of this file, in blocks, is approximately:

```
File size (blocks) = UCLMNC * (Total lines) / 5
```

UCLORD This parameter specifies the default order in which the TSX-Plus command interpreter checks for user-defined commands. The UCLORD parameter should be equated to one of the symbolic names FIRST, MIDDLE, LAST, or NONE. The SET UCL command may be used to change the order of command interpretation for a job. See the description of keyboard command interpretation in the TSX-Plus Reference Manual for a full discussion of command processing order.

LDSYS This parameter controls whether the standard system support for logical disks is to be included in the system. If LDSYS=1, system support is provided for logical disks; if LDSYS=0, system support is not provided for logical disks. Normally logical disk support should be included; however system support for logical disks may be excluded if a specialized LD handler providing custom logical disk support is being used rather than the standard system support.

SLEDIT This parameter controls whether support for the Single Line Editor facility is included in the system. If SLEDIT=1, the single line editor is included in the generated system; if SLEDIT=0, the single line editor is not included in the system. Use of the single line editor for a given time-sharing job is controlled by use of the SET SL command; however the SLEDIT parameter must be set to 1 (one) if the single line editor facility is to be made available to any lines.

QUANO This parameter specifies the time-slice value used to schedule jobs with user-specified priorities equal to or greater than the PRIHI parameter. High priority jobs that have the same priority are scheduled on a round-robin basis using QUANO as the time-slice value. If QUANO is set to 0 (zero), high-priority jobs are not time-sliced. Specify the value of QUANO in 0.1 second units. The SET QUANO keyboard command may be used to dynamically alter this parameter during system operation. See the TSX-Plus Reference Manual for more information about the SET QUANO command.

QUAN1 This parameter specifies the length of time a job will run in an interactive state after receiving input from the terminal. Specify the value in 0.1 second units. A job is classified as "interactive" and given a priority boost each time it receives input from the terminal. If the job uses up more than QUAN1 units of CPU time before it receives more input from the terminal, the job is classified as "non-interactive" and runs at normal priority. The SET QUAN1 keyboard command may be used to dynamically alter this parameter during system operation. See the TSX-Plus Reference Manual for more information on the SET QUAN1 command.

QUAN1A This parameter specifies the length of time a non-interactive job will run in a high-priority state after being restarted from a wait state. Increasing the value of this parameter tends to give priority to I/O active jobs and allow them to dominate over other jobs. The SET QUAN1A keyboard command may be used to dynamically alter this parameter during system operation. See the TSX-Plus Reference Manual for more information on the SET QUAN1A command.

QUAN1B This parameter specifies the execution time-slice value for round-robin scheduling of interactive jobs. Specify the value in 0.1 second units. The SET QUAN1B keyboard command may be used to dynamically alter this parameter during system operation. See the TSX-Plus Reference Manual for more information on the SET QUAN1B command.

QUAN1C This parameter specifies the length of time a job will execute in the highest priority interactive state after receiving an activation character. Specify the value in 0.1 second units. The SET QUAN1C keyboard command may be used to dynamically alter this parameter during system operation. See the TSX-Plus Reference Manual for more information on the SET QUAN1C command.

QUAN2 This is the time-slice given to compute-bound jobs. A compute-bound job is allowed to run this long if there are no high-priority tasks that need service. Specify the value in 0.1 second units. The SET QUAN2 keyboard command may be used to dynamically alter this parameter during system operation. See the TSX-Plus Reference Manual for more information on the SET QUAN2 command.

QUAN3 This is the time-slice used for round-robin scheduling of jobs with user-assigned priority values less than or equal to the PRILOW parameter. Specify the value in 0.1 second units. The SET QUAN3 keyboard command may be used to dynamically alter this parameter during system operation. See the TSX-Plus Reference Manual for more information on the SET QUAN3 command.

INTIOC This parameter controls the scheduling of interactive jobs which also do non-terminal I/O. An interactive job which exceeds this number of I/O operations before receiving another activation character will be rescheduled as a non-interactive job. This parameter should be large enough to keep jobs that are operator intensive in an interactive state. The SET INTIOC keyboard command may be used to dynamically alter this parameter during system operation. See the TSX-Plus Reference Manual for more information on the SET INTIOC command.

HIPRCT This parameter controls the scheduling of non-interactive jobs which do non-terminal I/O. On completion of non-terminal I/O, jobs are usually scheduled into a high-priority state. However, a job which exceeds this number of I/O operations will be rescheduled in the normal priority compute-bound state. The SET HIPRCT keyboard command may be used to dynamically alter this parameter during system operation. See the TSX-Plus Reference Manual for more information on the SET HIPRCT command.

CORTIM Each time a job is swapped into memory from disk a timer is started for that job. The job is not eligible to be swapped out of memory until CORTIM units of time have elapsed. However, a job becomes immediately eligible to be swapped if it goes into any wait state other than non-terminal I/O, regardless of the value of CORTIM. Specify the CORTIM parameter value in 0.1 second units. The SET CORTIM keyboard command may be used to dynamically alter this parameter during system operation. See the TSX-Plus Reference Manual for more information on the SET CORTIM command.

PRILOW This parameter specifies the highest user-specified job priority that is part of the fixed-low-priority group. Jobs with priorities less than or equal to PRILOW are considered low priority jobs and execute at fixed priorities below normal time-sharing jobs. The value of PRILOW must be in the range 0 to 126, and must be less than PRIHI.

PRIHI This parameter specifies the lowest user-specified job priority that is part of the fixed-high-priority group. Jobs with priorities greater than or equal to PRIHI are considered high priority jobs and take precedence over normal time-sharing jobs. Priorities in the fixed-high-priority group are normally reserved for real-time jobs and should never be assigned to normal time-sharing jobs. The value of PRIHI must be in the range 1 to 127, and must be greater than the value specified for PRILOW.

PRIDEF This parameter specifies the default job priority that will be assigned to jobs. The SET PRIORITY keyboard command may be used to dynamically set job priority, and priority may also be set from within a job by use of an EMT. The value of PRIDEF must be in the range 0 to 127. Normally, PRIDEF should be greater than PRILOW and less than PRIHI.

PRIVIR When a job switches to a virtual line, the job execution priority of the disconnected line is reduced by this amount. This automatic priority reduction does not apply to jobs with priority values less than or equal to PRILOW or greater than or equal to PRIHI. Also, jobs with priorities in the normal time-sharing range, between PRILOW and PRIHI, will never have their priority reduced to less than (PRILOW+1). See Chapter 5 for more information on priority and job scheduling.

MAXSEC This parameter is used to specify the maximum number of virtual lines that a single user may own at any given time.

MAXFIL Maximum file size (number of blocks) that will be returned in response to a .ENTER programmed request that specifies a file size of 0 blocks. This parameter does not limit the space that will be allocated to .ENTER requests that specify a size. Rather, it only affects .ENTER requests that specify a file size of 0. If a value of 0 (zero) is specified for MAXFIL, no limit is placed on the size of a file created with a specified size of 0.

CACHE This parameter controls the number of 512 byte data blocks allocated in extended memory for use by the generalized data caching facility. Data caching is a technique for improving system performance by keeping in memory a "cache" of the most recently accessed blocks of data. Use of the generalized data cache is not recommended for systems with less than 256Kb of memory. If generalized data caching is not wanted, set CACHE=0. If generalized data caching is wanted, set CACHE to the number of 512 byte blocks to be allocated for the cache.

In selecting this parameter, consideration must be given to the tradeoff between the improvement to system performance to be gained by data caching versus the decrease in total free memory space for jobs which may cause increased job swapping. While data cache buffers are not included in the low memory area, they do remove space from that available to user jobs. Generally it is recommended that CACHE be set to zero if less than 256Kb of memory is installed on the system or if the system is primarily bound by CPU utilization rather than I/O throughput. If data caching is used at all, it is recommended that CACHE be set to at least 50.

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 CACHE keyboard command to vary the number of buffers used while observing the effect on system performance.

MAXCSH The MAXCSH and NMFCSH parameters relate to the cache of file directory entries maintained by TSX-Plus. This cache is used to reduce the number of disk accesses required to do .LOOKUPs on frequently accessed files. The system disk directory is always cached. Other devices are only cached if they are introduced to the system by use of the "MOUNT" command. File directory caching can have a dramatic affect on the speed of .LOOKUPs of commonly used files. It does not affect the time taken to do .ENTER, .DELETE and .RENAME requests. The MAXCSH parameter is used to specify the maximum number of device units whose directories may be cached. Note: the value of MAXCSH must be large enough to include all mounted logical disks as well as mounted physical devices.

NMFCSH This parameter specifies the maximum number of file entries that can be held in the file directory cache. This number is the total number of file entries that will be cached for all users on the system (the cache is common to all users).

TIMOUT This parameter is only used for lines connected to dial-up telephone equipment. It is the time between the reception of the ring signal and the user's logging on that will be allowed before the telephone connection will be dropped. It also specifies the time until the job is terminated after carrier is lost if the job has not logged off. Specify in 0.5 second units.

TSLICH This is the "lead-in" character that tells TSX-Plus that the following character, which is being output by the program, is to be interpreted by TSX-Plus as a special command (for example, defining a new activation character). See Chapter 6 of the TSX-Plus Reference Manual for more information on program controlled terminal options. The default value for this parameter is 35 (29 decimal).

VLSWCH This is the character used to signal a request to switch to a virtual line. See Chapter 3 of the TSX-Plus Reference Manual for a discussion of the use of virtual lines. The default value is 27, control-W.

MXSPAC TSX-Plus allows running programs to dynamically define activation characters. (An activation character is a character which completes a terminal input field, such as carriage return.) MXSPAC specifies the maximum number of user defined activation characters that each line may define.

EDITOR This parameter specifies the default system editor. The SET EDIT keyboard command can be used to select a different default editor for a job. The allowable editors are EDIT, TECO, KED and K52.

WILDFL This flag sets the system default for implicit or explicit wildcards in file names. The SET WILDCARDS command can be used to alter this setting for a line. Specify 0 (zero) for explicit wildcards or 1 for implicit wildcards. See the RT-11 System User's Guide for further discussion of explicit and implicit wildcards.

DEVDEF The DEVDEF macro must be used to define the names and characteristics of all devices which are to be available to TSX-Plus users. The form of the DEVDEF macro is:

```
DEVDEF <dd>,[NON]DMA[,MAPIO]
```

The first parameter defines the 2 character device name. The device name must be enclosed in angle brackets ("<" and ">").

The second parameter defines whether the device is direct memory access (DMA) or not direct memory access (NONDMA). DEC standard direct memory access (DMA) devices include DL, DM, DP, DS, DT, DU, DY, MM, MS, MT, RF, and RK. Typical non-DMA devices include CR, CT, DD, DX, LP, LS, NL, PC, VM, and XL.

The third parameter (MAPIO) is optional and is used only for Q-BUS DMA devices which only have 18-bit controllers or handlers (such as RX02 -- DY) but are being used in an otherwise 22-bit environment. MAPIO should not be specified for any NONDMA devices nor for any DMA device for which the handler and controller actually support 22-bit addressing. MAPIO should likewise never be specified for any device in an 18-bit environment (EXTMCH = 0; 256 Kb or less memory; 11/23 without modified backplane). Older LSI-11/23 systems can be usually be upgraded to support 22-bit memory by changing to the H9275-A backplane. In addition, 18-bit DMA controllers should be replaced with 22-bit controllers whenever possible if more than 256 Kb of memory will be used. MAPIO should never be specified for any device in a UNIBUS system. If MAPIO is specified for any device, approximately 1Kb of code is added to the mapped portion of the system.

All device definitions must take place in the device definition block. This block starts with the DEVBEG macro and ends with the DEVEND macro. Each device is specified using the DEVDEF macro which must be placed between the DEVBEG and DEVEND macros.

The devices TT, LD, and SL do not require device definitions and should not be included in the DEVDEF table. These devices are an integral part of TSX-Plus and do not require separate device handlers. See the section on device handlers for TSX-Plus for more information. Note that a maximum of 15 devices can be installed in TSX-Plus as distributed, including TT and LD. SL is not implemented as a pseudo-device in TSX-Plus. By convention, the system device (device from which RT-11 was booted and TSX-Plus is run) should be the first device definition.

For example:

```
DEVBEG              ;Beginning of device definitions
DEVDEF  <DL>,DMA
DEVDEF  <RK>,DMA,MAPIO
DEVDEF  <DY>,DMA,MAPIO
DEVDEF  <DX>,NONDMA
DEVDEF  <MS>,DMA
DEVDEF  <LP>,NONDMA
DEVDEF  <NL>,NONDMA
DEVDEF  <VM>,NONDMA
DEVEND              ;End of device definitions
```

MIONBF This parameter specifies the number of system I/O buffers to be allocated for I/O mapping. I/O mapping is used for devices with 18-bit controllers being used with 22-bit Q-bus systems. The MAPIO parameter to the DEVDEF macro specifies which devices require I/O mapping. One buffer should be allocated for each device which requires I/O mapping and which will be in use simultaneously with other devices which also require system I/O mapping. For example, if both RK and DY need system I/O mapping, but both devices will never be in use at the same time, then 1 buffer would be adequate. If however, both devices are likely to be in use at the same time, then 2 buffers should be allocated. These buffers are shared by the system and all user jobs that are doing I/O to devices needing system I/O mapping. If a transfer is requested to a device which requires system I/O mapping and a buffer is not available, the transfer will be delayed until a buffer becomes available.

MIOBSZ This parameter specifies the size of the buffers used for system I/O mapping. The value specifies the number of 512 byte areas to allocate for each buffer. The larger this parameter is, the faster system mapped I/O transfers will occur. The maximum value for this parameter is 15. Because directory operations are performed in 1024 byte chunks, if system I/O mapping is selected at all, the minimum recommended number for MIOBSZ is 2. It is strongly recommended that the system device not require I/O mapping. However, if the system device does require I/O mapping, MIOBSZ should be set to 15, the maximum value.

2.2.2 Device Spooling Parameters

Device spooling is an optional feature of TSX-Plus. If any spooled devices are wanted, give appropriate values to the parameters of the SPOOL macro. If spooling is not wanted, specify 0 (zero) as the first parameter to the SPOOL macro; the other parameters will be ignored. See Chapter 5 of the TSX-Plus Reference Manual for more information on device spooling.

SPOOL The SPOOL macro is used to define information about devices that are to be spooled by TSX-Plus. The form of the SPOOL macro is:

```
SPOOL  ndev,nfile,nbuf,nblocks,<dev...>,hold,nback
```

The meanings of these parameters are:

ndev The number of devices that are to be spooled by TSX-Plus. Specify 0 (zero) if there are none.

nfile The number of spooled files that may be open to all users. A spooled file entry is required for each file that is being printed, waiting to be printed, or is in the process of being generated by a running program for printing on a spooled device.

nbuf The number of 512 byte buffers that are to be used by the spooling system. If two buffers are available for each active device, the I/O will be "double-buffered" to achieve maximum speed. If fewer buffers are available than active devices, the devices will operate in bursts and share the buffers. Space for these buffers is allocated in the mapped portion of TSX-Plus.

nblocks The number of disk blocks to be allocated within the spool disk file. All spooled files share this space in the common disk spool file. If the file fills up, running programs are suspended until space becomes available as blocks are printed and released.

<dev> The names of those devices that are to be spooled. Specify exactly three characters per name. The spooled devices must be non-file structured output devices such as line printers, card punches, or plotters. Note that spooled devices must also be specified in the DEVDEF (device definition) list.

hold Specify 1 for the this parameter if the default mode for the spooler is to be 'HOLD' (see SPOOL command description in the TSX-Plus Reference Manual). Specify 0 (zero) for 'NOHOLD' mode. In HOLD mode, spooled output will not be processed until the spool file is completely created and the I/O channel associated with the file is closed. In NOHOLD mode, a spool

file may begin to be copied to the spooled device while the spool file is being created. This mode may be changed dynamically with the SPOOL <dev>,[NO]HOLD command. This mode may also be controlled from within programs on an individual file basis with an EMT request.

nback This parameter specifies the number of spool blocks that TSX-Plus will back up in response to the SPOOL <dev>,BACK command (see SPOOL command description in the TSX-Plus Reference Manual, Chapter 5).

Example:

The following SPOOL macro declares that there is 1 spooled device; there may be up to 10 active spooled files; two 512 byte buffers are to be used for spooling I/O; the spool file is to be 500 blocks large; the spooled device is "LP"; default mode is HOLD; and the SPOOL BACK command is to backup 10 blocks.

```
SPOOL 1,10.,2,500.,<LP >,1,10.
```

2.2.3 Record Locking Parameters

If the shared file record locking and data caching feature of TSX-Plus is wanted, the three parameters MAXSF, MAXSFC, and MXLBLK must be given appropriate values. If the shared file record locking and data caching facility is not wanted, set MAXSF, MAXSFC, and MXLBLK to 0 (zero).

MAXSF MAXSF specifies how many shared files may be open simultaneously. Note that several users accessing the same shared file count as one open shared file.

MAXSFC Maximum number of I/O channels that all users may simultaneously have open to shared files. Note that this is the total number of channels for all users not for each user.

MXLBLK Maximum number of file blocks that may be simultaneously held locked by any channel. A file block contains 512 characters.

NUMDC Number of 512-byte data blocks to be allocated for shared file data caching. There are two data caching facilities in TSX-Plus: a generalized data caching facility that caches blocks from all files, and a shared-file data caching facility that only caches blocks from files declared to be "shared" to the system. Both data caching facilities should not be used at the same time. The generalized data caching facility is controlled by the CACHE parameter (see above); the shared-file data caching facility is controlled by the NUMDC parameter.

The shared file data caching facility provides data caching only for files which have been declared as shared files (regardless of access

and protection category). Data caching causes the most active blocks for shared files to be held in memory cache buffers. This eliminates all disk I/O when these blocks are read. Data caching is particularly effective for COBOL-Plus ISAM files. The NUMDC parameter controls the number of 512-byte cache buffers that are allocated for data caching. If NUMDC is set to 0 (zero) shared file data caching is not done (but generalized data caching will be done if CACHE is non-zero).

In selecting this parameter, consideration must be given to the tradeoff between the improvement to system performance to be gained by data caching versus the decrease in total free memory space for jobs which may cause increased job swapping. While data cache buffers are not included in the low memory area, they do remove space from that available to user jobs. Generally it is recommended that NUMDC be set to zero if less than 192Kb of memory is installed on the system or if shared files are not accessed heavily. If data caching is used at all, it is recommended that NUMDC be set to at least 5. 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.

2.2.4 Message Communication Parameters

If the message communication feature is not wanted, the three parameters MAXMC, MSCHRS and MAXMSG should be set to zero. If the message communication feature is wanted, assign appropriate values to the three parameters.

MAXMC Maximum number of message communication channels that may be simultaneously active. A message channel is active if any messages are pending on it or if any users are waiting for messages to come through it.

MSCHRS Maximum length of messages; specify in bytes.

MAXMSG Maximum number of messages that may be simultaneously held in message queues for all channels. Note, this is the maximum number of messages that can be queued on all channels, not each channel.

2.2.5 Real-time Program Support Parameters

TSX-Plus provides a real-time program support facility that allows multiple real-time programs to be run concurrently with normal time-sharing operations. Note: real-time program support must be included in the system if the SYSMON system monitor display program is to be used.

The basic functions provided by this facility are summarized below.

1. The ability to map the I/O page into the user's virtual memory region so that device status and control registers may be directly accessed by the program.

2. The ability to connect device interrupt vectors to program interrupt service routines. System service support is restricted with this method, but it is quite fast.

3. The ability to connect device interrupt vectors to program completion routines. These real-time completion routines run at user-selectable real-time priority levels that preempt execution of normal time-sharing jobs.

4. The ability for a program to lock itself in memory so that rapid interrupt response can be assured.

5. The ability for a program to dynamically set its execution priority.

6. The ability for a program to suspend its execution until an interrupt occurs.

7. The ability to convert a virtual address within the job's region to a physical address for DMA I/O control.

8. The ability to map a virtual address region to a physical address region.

9. The ability for a program to declare a list of addresses of device control registers to be reset when the program exits or aborts (.DEVICE EMT).

The RTVECT parameter controls whether or not the TSX-Plus real-time program support facility will be included in the generated system. If real-time program support is not wanted, set the RTVECT parameter to 0 (zero). If real-time program support is wanted, set RTVECT to the number of real-time interrupt vectors that will be used by all real-time programs. If some of the real-time support features are wanted, but no interrupt vectors are necessary, then set RTVECT to 1.

If the SYSMON system monitor display program is to be used to monitor system performance, real-time support must be included in the system and RTVECT must be at least 1.

See Chapter 11 in the TSX-Plus Reference Manual for more information on real-time support.

2.2.6 Performance Monitor Parameter

TSX-Plus includes a performance monitor facility that allows you to monitor the execution of an application program running under TSX-Plus and produce a histogram showing the amount of time spent in various regions of the program.

There is one parameter in TSGEN that is associated with the performance monitor feature. This parameter, PMSIZE, specifies the number of bytes of memory to set aside for use in accumulating histogram values during a performance analysis run. Memory space equal to the size specified with PMSIZE is allocated in a mapped data region of TSX-Plus for use by the performance analysis facility. If you do not intend to use the performance analysis feature, set PMSIZE to 0 (zero) to avoid using any memory space for this feature. The maximum value that may be given to PMSIZE is 8192. See Chapter 13 of the TSX-Plus Reference Manual for information on use of the performance monitor feature.

2.2.7 Shared Run-time Systems

TSX-Plus supports shared run-time systems. These are reentrant programs or common data buffers that can be shared by multiple users. The RTDEF macro is used to declare shared run-time systems. The form of this macro is:

```
RTDEF   <program-name>,r-flag,skip-count
```

where "program-name" is the 12 character name of the file containing the run-time system. This must be specified in the form <DevFilnamExt>, that is, three characters for the device name, six characters for the file name and three characters for the extension. "r-flag" is either "R" if user programs are to have read-only access the run-time system, or "RW" if read-write access is to be granted. Most run-time systems will use read-only access. Read-write access is primarily useful when the shared run-time facility is being used to provide common data areas being accessed and updated by multiple jobs. The "skip-count" parameter is the number of blocks to be skipped over at the front of the run-time system file when loading it into memory.

Run-time system files are normally SAV files. However any type of file could potentially be used. TSX-Plus simply reads it into memory (without interpreting its contents) and maps portions of it into the job space as requested by EMT's. Shared run-time systems are loaded into memory below the mapped system overlay regions. See Chapter 12 of the TSX-Plus Reference Manual for information about using a shared run-time system.

Examples of shared run-time declarations:

```
RTDEF   <SY CBR050SHR>,R,1.          ;COBOL-Plus runtime
RTDEF   <SY DBLSHRRTS>,R,1.          ;DBL runtime
RTDEF   <RK2COMDT1SAV>,RW,0.
```

2.2.8 Timesharing Line Definitions

DINSPC This parameter specifies the default number of characters that will be reserved for the input ring buffer for each line. This value is used for all virtual lines and for actual lines that do not have any other value specified. It must be large enough to hold an entire line of input plus any characters that are typed ahead.

DOTSPC This parameter specifies the default number of characters that will be reserved for the output ring buffer for each line. This value is used for all virtual lines and for actual lines that do not specify any other value. A running program will be suspended when its output ring buffer is filled.

OTRASZ A job's execution is suspended and the job may be swapped out of memory when that job's character output buffer is filled. As the output buffer is emptied the job is reactivated when the number of characters remaining in the buffer equals OTRASZ. The idea is to get the job running again before all of the available output is exhausted.

Each line that is to be used as a TSX-Plus timesharing line must be declared in TSGEN. The total number of timesharing lines is first declared by setting the proper values as arguments to the TBLDEF macro.

The TBLDEF macro has three arguments:

1. The number of real (physical) timesharing lines.

2. The number of virtual timesharing lines.

3. The number of job slots to allocate for the execution of detached jobs.

See Chapter 4 of the TSX-Plus Reference Manual for more information on virtual lines and detached jobs.

TSX-Plus will support up to 31 total time-sharing lines, including virtual lines and detached jobs. However, we do not recommend that you enable that many unless you actually plan to use all of them. The memory that is used by these lines is limited, and is shared with many other facilities. Therefore, you should not define more lines than would be useful for your system. Performance with a large number of lines generated is highly dependent on configuration, specifically: CPU speed, I/O devices, and types of applications (programs) being run.

Refer to the example at the end of this section and to the examples in the supplied TSGEN module as you read the following explanation.

The actual line definitions follow the invocation of the TBLDEF macro. Each line definition is specified by creating a Line Definition Block (LDB). There must be exactly as many LDB's as there are physical lines. Virtual lines and detached jobs are not described by LDB's.

A Line Definition Block begins by calling the LINDEF macro and ends by calling the LINEND macro. Each LDB must have matching calls to LINDEF and LINEND. Other optional macros may be called between LINDEF and LINEND to specify parameters for the line.

TSX-Plus supports lines connected to DL11 and DLV11 serial communication cards and lines connected to DZ11 and DZV11 multiplexors. TSX-Plus will support a mixture of DL11 and DZ11 dial-up and direct connect lines. Unless otherwise stated, there is no distinction between DL11 and DLV11 support, nor between DZ11 and DZV11 support.

When generating a TSX-Plus system, each real line must be declared with a line definition block (LDB) that begins with a LINDEF macro call and ends with a LINEND macro call. This is true for both DL11 and DZ11 lines.

The line definition blocks for lines connected to a DZ11 multiplexor are enclosed within a Multiplexor Definition Block (MDB). An MDB begins with a MUXDEF macro call, contains the LDB's for all lines connected to the multiplexor, and ends with a MUXEND macro call. The MUXDEF macro requires two parameters: the first is the address of the multiplexor receiver interrupt vector and the second is the address of the multiplexor Control and Status Register (CSR). If there are no DZ11 lines, do not use the MUXDEF or MUXEND macros.

The Line Definition Block macros require different parameters depending on whether they describe DL11 type lines or DZ11 multiplexor lines. In the case of DL11 lines, two parameters are required by the LINDEF macro. The first is the memory address of the input (receiver) interrupt vector for the line. The second argument is the address of the receiver status register for the line.

A LINDEF macro for a DZ11 multiplexor line requires only a single argument; it is the number of the line on the DZ11 multiplexor. Note that DEC DZ11 lines are numbered 0 to 7 and DZV11 lines are numbered from 0 to 3. TSX-Plus will support up to four DZ11 multiplexors (but not more than 31 total lines, including virtual lines and detached jobs). If there is more than one DZ11, each must be defined using a separate multiplexor definition block.

Note that different model DL11 cards have different ranges of addresses for the status register. DL11-A and B cards generally start at 176500, while DL11-C, D, and E cards start at 175610. The addresses increase by 10 (octal) per line. Note that 16-bit device addresses are specified in TSGEN. The receiver interrupt locations for DL11 cards normally start at 300 and increase by 10 (octal) per line. The receiver status register and interrupt vector addresses for all devices are normally written on a card that is attached to the top cover of the CPU drawer for Unibus machines and somewhere in the cabinet for Q-bus machines. If you cannot locate the status register and interrupt vector addresses, default addresses may be found in the PDP-11 Programmer's Reference Card or the processor handbook for your machine. If you still are having problems, contact the person who installed your machine. The most common problem in getting started with TSX-Plus is specifying incorrect addresses for the communication cards.

The LINDEF macro also accepts a third (second for DZ11 lines) optional parameter. One terminal may be declared to be the operator's console that receives system control messages such as requests for special form mounts if spooling is used. The terminal to be the operator's console is signified by specifying "OPER" as the third argument. Only one terminal may be declared to be the operator's console.

Optional macros may be invoked between the LINDEF and LINEND calls to set parameters for a line. The available macros are listed below.

FLAGS This macro is used to set a variety of control flags for the line. The form of the FLAGS macro is:

```
FLAGS   flag_values
```

The single argument to FLAGS must be the logical sum of those flags that are to be set for the line. When more than one flag is specified, the names of the flags should be joined together with exclamation marks ("!") which is the MACRO assembler syntax symbol for the logical OR operation.

The valid flags are listed below.

Flag	Meaning when set
$SCOPE	Terminal is a CRT type terminal and DELETE is to echo as backspace-space-backspace.
$ECHO	Echo characters to the terminal.
$TAPE	If this flag is set the line will be placed in "TAPE" mode. This mode of operation is useful if the line is receiving input from a paper-tape reader, cassette tape, floppy disk or other devices that respond to X-ON/X-OFF control characters to start and stop transmission. TAPE mode can also be controlled by use

of the SET TT [NO]TAPE keyboard command and the "W" and "X" program controlled terminal option functions. Setting a line to TAPE mode has three effects:

1. An X-OFF (CTRL-S) is sent by TSX-Plus whenever the line input buffer fills to the point that there are only 10 remaining free character positions.

2. An X-ON character (CTRL-Q) is sent by TSX-Plus when a program is about to enter an input wait state and an X-OFF has previously been sent.

3. Line-feed characters are ignored -- this is done so that each line of input can be terminated by a carriage-return line-feed pair.

$START If this flag is set the line will be automatically initiated when TSX-Plus is started. If the flag is not set, the line will not be initiated until carriage-return or control-C is pressed at the terminal.

$NODET If this flag is set the line is prevented from using the DETACH keyboard command which controls detached jobs.

$TAB Do not simulate tabs by inserting spaces. Use with terminals whose hardware responds to tab characters, such as VT100 terminals.

$FORM Do not simulate form feed by inserting line feed characters. Use with terminals whose hardware responds to form feed characters, such as LA120 terminals.

$PAGE Allows CTRL-S to suspend output and CTRL-Q to restart output. If $PAGE is not set, CTRL-S and CTRL-Q are not interpreted by TSX-Plus and are passed directly to the user's program. $PAGE is usually selected.

$LC Enables lower case input from the terminal. Note that bit 14 of the job status word must also be set to enable lower case input.

$NOVLN If this flag is specified, the line will not be allowed to use the TSX-Plus virtual line facilities. That is, the line will always be connected to its primary line.

$DEFER If this flag is set, "deferred" character echoing will be enabled. If the flag is not set, "immediate" character echoing will be used. See the description of the DEFER option to the SET TT command in the TSX-Plus Reference Manual for an explanation of deferred character echoing. It is recommended that deferred echoing mode be used.

$QTSET If this flag is set the line will be initialized as if a "SET TT QUIET" command had been executed. This prevents the listing of command files.

$PRIV If this flag is set the line will be authorized for "operator privilege". See the section on operator privilege in Chapter 3 of this manual for an explanation of this privilege.

$PHONE This flag should be set if the line is connected to a dial-up telephone modem. If this flag is set, TSX-Plus will perform modem control such as answering the phone when the ring signal occurs and hanging up when carrier is lost. It is important that this flag not be specified for a line unless that line is actually connected to a line with hardware modem control facilities (DL11-E, DZ-11, etc.)

NRMFLG The NRMFLG parameter, located just before the timesharing line definition section of TSGEN, is used to define the flags which are common to all time-sharing lines. If all lines require the same set of flags, simply set NRMFLG to this combination and do not use the FLAGS macro.

Note that the FLAGS macro sets the default line characteristics when each line is started and that the "SET" keyboard command may be used to alter flag settings for a line. Thus, the command "SET TT LC" would enable lower case input, and "SET TT NOSCOPE" would say that the terminal is not a CRT device.

LINPRM The LINPRM macro is used to define parameters for DZ11 and DZV11 mux lines. It must be used only within LDB's for DZ11 lines. The form of the LINPRM macro is:

```
LINPRM  speed,parity,stopbits
```

The LINPRM macro requires three arguments. The first argument is a code that specifies the line's operating baud rate. The following codes may be used:

Speed code	Baud rate	Speed code	Baud rate
0	50	10	1800
1	75	11	2000
2	110	12	2400
3	134.5	13	3600
4	150	14	4800
5	300	15	7200
6	600	16	9600
7	1200		

The second parameter specifies whether even (0) or odd (1) parity is to be generated for transmitted characters. Parity is ignored on received characters.

The third parameter specifies the number of stop bits to be sent with each transmitted character. The value may be 1 or 2. Two stop bits should be used for 110 baud mechanical teletype terminals. One stop bit should be used for all other types of terminals. If a DZ11 line definition block does not contain a LINPRM macro, the last set of parameters defined for an earlier line will be used. The first DZ11 LDB must contain a LINPRM macro.

TRMTYP The TRMTYP macro is used to declare what type of terminal will be used with the line. The form of the TRMTYP macro is:

```
TRMTYP  terminal
```

The terminal types DIABLO and QUME are treated as equivalent and no longer support the ETX/ACK protocol which was available with earlier versions of TSX-Plus. Note that newer terminals which use the X-ON/X-OFF (DC1/DC3, CTRL-Q/CTRL-S) protocol are acceptable. See the SET TT command in Chapter 2 of the TSX-Plus Reference Manual for more information on the meaning of each terminal type. The valid choices are listed below:

Name	Terminal type
VT100	DEC VT100 terminal
VT52	DEC VT52 terminal
LA36	DEC LA36 terminal
LA120	DEC LA120 terminal
HAZEL	Hazeltine brand terminals
ADM3A	Lear Siegler ADM3A terminal
DIABLO	Diablo brand terminals (with X-ON/X-OFF protocol)
QUME	Qume brand terminals (with X-ON/X-OFF protocol) (Diablo and Qume are treated as equivalent)

BUFSIZ The BUFSIZ macro is used to specify the number of characters to reserve for the line's input character ring buffer (argument 1) and the output character ring buffer (argument 2). The form of the macro is:

```
BUFSIZ  inputsize,outputsize
```

If a BUFSIZ macro is not used in a Line Definition Block, the default sizes as specified for DINSPC and DOTSPC will be used.

CMDFIL The CMDFIL macro is used to specify the name of a start-up command file to be executed when the line is initialized. The form of the macro is:

```
CMDFIL  dev:file.ext
```

This macro has one argument that is the name of the command file (dev:file.ext). This argument must be included in order to use the TSX-Plus LOGON facility.

This ends the description of macros that can be used within Line Definition Blocks.

2.2.9 Defining Start-up Files for Detached Jobs

The DETACH macro may be used to specify the names of start-up command files to be initiated as detached jobs when the TSX-Plus system is started. The use of the DETACH macro should follow the last line definition block. The form of the DETACH macro is:

```
DETACH  dev:file.ext
```

The DETACH macro requires one argument which is the name of the command file to be initiated as a detached job. The physical device name must be included in the command file specification.

There may be one use of the DETACH macro for each detached job slot specified with the TBLDEF macro. If there are more detached job slots defined than there are invocations of the DETACH macro, the excess job slots are left idle when the system is started and detached jobs may be started on these lines by use of the DETACH keyboard command.

2.2.10 Line Definition Example

The following example shows the definition of two DL11 lines and three DZ11 lines. Two virtual lines are also declared (LDB's are not used for virtual lines). Note that the exclamation mark is used to perform the logical OR (sum) operation when combining flags.

```
NRMFLG  =  $ECHO!$PAGE!$DEFER!$LC

    TBLDEF  5.,2.,1.        ;5 real, 2 virtual, 1 detached

; Define DL11 lines
; Define DL11 line #1
    LINDEF  300,175610  ;DL11-E line
    TRMTYP  VT100
    FLAGS   NRMFLG!$SCOPE
    BUFSIZ  120.,300.
    CMDFIL  SY:START.COM
    LINEND
; Define DL11 line #2
    LINDEF  60,177560,OPER  ;console terminal
    TRMTYP  VT100
    FLAGS   NRMFLG!$PRIV!$START
    LINEND

; Define DZ11 lines
    MUXDEF  310,177620  ;Start of MUX definition block
; Define DZ11 line # 0
    LINDEF  0       ;MUX line #0
    LINPRM  7,1,1       ;1200 baud, odd, 1 stop bit
    FLAGS   NRMFLG!$SCOPE
    CMDFIL  SY:LINE1.TSX
    LINEND
; Define DZ11 line # 1
    LINDEF  1       ;MUX line #1
    LINPRM  5,1,1       ;300 baud, odd, 1 stop bit
    TRMTYP  LA36
    CMDFIL  SY:LOGON.TSX
    LINEND
; Define DZ11 line # 6
    LINDEF  6       ;MUX line #6    (use same LINPRM as #1)
    LINEND
; End of MUX lines
    MUXEND          ;End of MUX definition block
; Define start-up command files for detached job slots
    DETACH  SY:INITDJ.TSX
;End of line definition example
```

2.3 Assembling the modified TSGEN module

Once the TSGEN module has been modified to contain the desired parameter settings, it must be assembled using MACRO. The command to do this is:

```
.MACRO TSGEN
```

If you want to get a listing of the modified TSGEN (a good idea), use the command:

```
.MACRO/LIST TSGEN
```

No errors should occur during the assembly.

2.4 Linking TSX-Plus

The final stage of building TSX-Plus is to link the component parts together. A command file to do this is provided on the TSX-Plus distribution disk with the name "TSXLNK.COM". The system command to execute this command file is

```
.@TSXLNK
```

If you are linking under RT-11 version 3B you should use the command file "TSLNK3.COM". These command files create three SAV files: TSX.SAV, TSKMON.SAV, and SYSMON.SAV. The TSX.SAV and TSKMON.SAV files must be on the system disk (SY:) before TSX-Plus can be started. The SYSMON.SAV file is only needed if the SYSMON dynamic system display utility program is to be used (see Chapter 7). Note that both TSX-Plus and TSKMON must be rebuilt if any parameters are changed in TSGEN.

Warning: Do not relink TSKMON onto the system device or copy it there while running under TSX-Plus. The position of the TSKMON file on the system disk must not change while TSX-Plus is running.

2.5 Starting TSX-Plus

Before starting TSX-Plus, you should check the following items to make sure the system is set up correctly:

1. The TSX.SAV, TSKMON.SAV, and CCL.SAV files must be on the system disk (SY:). CCL.SAV is provided on the TSX-Plus distribution disk; TSX.SAV and TSKMON.SAV are built using the TSXLNK.COM command file as part of the system generation process.

2. The TSX-Plus version of device handlers (with the extension ".TSX") must be on the system disk for each device declared in TSGEN.

3. The RT-11 version of the device handlers for all devices to be used by TSX-Plus must be on the system disk and the devices must be installed (but need not be loaded) in the running version of RT-11.

4. If the TSX-Plus logon facility is being used, the LOGON.SAV and ACCESS.TSX files must be on the system disk. (The ACCESS.TSX file is created by the TSAUTH account authorization program.) If the account authorization file was created with a version of TSAUTH prior to the one supplied with version 4.0, use the AUTCVT program to convert the format of the account authorization file.

5. If user-defined commands are allowed, the TSXUCL program must be on the system disk.

6. Any startup command files associated with time-sharing lines (such as LOGON.TSX or LINE1.TSX) must be on the system disk.

7. TSX-Plus must be started under RT11SJ (single job); an attempt to start it under the foreground-background (FB) or extended-memory (XM) versions of RT-11 will result in a "?KMON-F-Insufficient memory" error message. This message may also occur if RT-11SJ has been sysgenned to include multi-terminal support or other features which make it too large to be able to start TSX-Plus, or if the USR is set NOSWAP or if too many device handlers are loaded. If your RT-11 has been sysgenned to include multi-terminal support, copy and boot the distributed versions of RT11SJ or RT11BL, then run TSX-Plus.

Once you have determined that all of these conditions are met, you can start TSX-Plus by typing

```
R TSX
```

After this is typed, time-sharing lines that were generated to be automatically starting ($START flag) should start up and print the TSX-Plus greeting message. Other lines will be initiated when carriage return or control-C is pressed at the terminal or when the phone rings for dial-up lines.

During its initialization TSX-Plus performs a test to make sure the physical lines defined in TSGEN actually exist. It does this by trying to access the receiver status register for each line. If a trap occurs, TSX-Plus displays the message:

```
?TSX-F-Invalid status register address for T/S line: xxxxxx
Line # = nn
```

If this occurs you must edit in the correct line address in the TSGEN module. TSX-Plus does not check the interrupt vector addresses for the lines, so if the system dies when a line is started, check to see if its interrupt vector address is correctly specified.

When TSX-Plus is running, the system line frequency clock must be operating at all times. This is true even if only a single job is being run under TSX-Plus.

If TSX-Plus does not start properly, carefully review the parameter settings in TSGEN. Check especially the values provided for DL11 interrupt vectors and receiver status registers. Different models of DL11 cards use different addresses.

In the case where RT-11 runs successfully on a system but TSX-Plus does not, look carefully at the memory installed on the machine above 56Kb. If it is not functional or improperly configured, TSX-Plus will not run. If you are using non-DEC peripherals, check with the peripheral vendor to make sure the device can support extended memory addressing. Refer to Appendix A for information on error messages received during TSX-Plus startup and Appendix B for information on error messages received during TSX-Plus operation.

2.6 Device Handlers for TSX-Plus

The names of TSX-Plus device handlers take the form "dd.TSX". Device handlers for most common devices are included on the TSX-Plus distribution media. If you received TSX-Plus on a floppy disk, be sure to copy both sides of the reversible disk to your working surface. Most device handlers (e.g. DL.TSX, DY.TSX, etc.) are located on the reverse side of the diskette.

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. In which case, move on to the next section. Do not unnecessarily recreate device handlers. 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. An example of a change that requires regenerating a device handler is adding a second controller (vector and CSR) to the MT handler. TSX-Plus generally uses standard RT-11 XM device handlers, however, the DD, DM, DX, DY, MM, MS, and MT handlers as supplied with RT-11 require minor modifications to function correctly with TSX-Plus. The RT-11 version 4 DL handler also requires modification. The necessary handler modifications are supplied as SLP files with TSX-Plus. These SLP files have already been applied and are included in the dd.TSX handlers supplied with TSX-Plus. The VM.TSX handler is proprietary and unique to TSX-Plus.

2.6.1 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.

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. The MEMSIZ parameter accepts a numeric argument defining the number of K-bytes for TSX-Plus to use. For instance, the statement:

```
MEMSIZ = 256.
```

would restrict the TSX-Plus operating system to using only the first 256 K-bytes of memory. (Note: The decimal point is required, otherwise the number is interpreted as an octal value.)

System Generation

A device definition entry for VM must be made in TSGEN, if VM is to be used. The correct VM device declaration is:

```
DEVDEF    <VM>,NONDMA
```

In order to use VM with TSX-Plus, the VM handler must be installed in RT-11 before running TSX-Plus. Users who do not have an RT-11 VM.SYS handler can create one by copying NL.SYS to VM.SYS. This VM.SYS, although not functional, may be installed in RT-11 so that TSX-Plus can load and use VM.TSX.

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 512 K-bytes total physical memory and with MEMSIZ=256., VM will have 256 K-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 512 K-bytes, then nnnnnn should be 20000 since the base memory address is 2000000 (octal). Any time a new base address is defined, VM should be initialized.

2.6.2 VTCOM/TRANSF Support and XL Handler

The VTCOM/TRANSF file transfer programs may be used to communicate and transfer files between RT-11 and TSX-Plus systems or between two 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 XL handler (or XC on the PRO-350) to connect to a communications line. The XL handler must be set up to drive a DL11 or DLV11 communications port that is connected either directly or through a modem to the host system. Note that this port must not also be declared as a time-sharing line.

When TSX-Plus is used as the local system, a modified version of the XL handler (supplied with this release) must be used. XL must be specified as a device to TSX-Plus by use of a standard TSX-Plus device definition in TSGEN of the form:

```
DEVDEF  <XL>,NONDMA
```

An XL handler must also be installed in RT-11. If VTCOM is to be used only with TSX-Plus, the NL (null) handler can be copied to a file named XL.SYS and this dummy handler installed in RT-11.

The address of the interrupt vector and CSR register for the DL11 line to be controlled by XL must be set before XL can be used with the system. This can be done using the following commands which must be executed under RT-11 before TSX-Plus is started:

```
RENAME/SYS XL.SYS XL.TMP
RENAME/SYS XL.TSX XL.SYS
SET XL VECTOR=xxx
SET XL CSR=xxxxxx
RENAME/SYS XL.SYS XL.TSX
RENAME/SYS XL.TMP XL.SYS
```

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

The VTCOM program references the XL device by use of the logical name XC. So an assign of the following form must be used before running VTCOM:

```
ASSIGN XL 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.

2.6.3 Building device handlers

When building device drivers, 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 not be specified when the handlers are built.

An easy method of building device drivers for TSX-Plus is to create a TSX-Plus conditional file. Using the editor, create a file "TSXCND.MAC" with the following conditionals:

```
MMG$T   = 1 ;enable memory management
ERL$G   = 0 ;disable error logging
TIM$IT  = 1 ;optionally enable timeout
```

Note that setting a conditional parameter to zero (0) disables the option and setting it to one (1) enables the option. Since device timeout is optionally supported, TIM$IT may be either 0 or 1. Other parameters may be included to specify device characteristics. For instance, the following conditionals may be specified for RL01/RL02 support:

```
DL$UN   = 2      ;define 2 units for the DL handler
DL$CSR  = 176400 ;define the CSR address for the DL handler
DL$VEC  = 164    ;define the vector for the DL handler
```

The following conditionals might be specified for file structured MT support:

```
MT$FSM  = 1      ;enable file structured MT support
MT$UN   = 1      ;define 1 unit for the MT handler
MT$CSR  = 172520 ;define the CSR address for the MT handler
MT$VEC  = 224    ;define the vector for the MT handler
```

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 MMG$T which must be set to 1 for TSX-Plus.

2.6.4 Patching device handlers for use under TSX-Plus

Handlers which do not require patches for use with TSX-Plus are: CT, CR, DL, DP, DS, DT, DU, LP, LS, NL, PC, RF, and RK. Some device drivers (DD, DM, DX, DY, MM, MS, and MT) require minor modifications to execute properly with TSX-Plus. (The RT-11 version 4 DL handler also requires modification if used.) When using the file structured magtape device drivers, the file structured module (FSM) as well as the device specific drivers (TJ, TM, and TS) must be patched. The dd.TSX device handlers provided with TSX-Plus have already been patched using the SLP files provided on the distribution and listed in Appendix D. These patched handlers have been assembled with the appropriate conditionals (ERL$G=0; MMG$T=1; TIM$IT=1) and linked to create the dd.TSX files provided with TSX-Plus. Device handlers do not need to be rebuilt unless you require some modification which can not be made via a device SET option.

If it is absolutely necessary to rebuild a device handler, then, to apply the patches, either create the SLP files (from Appendix D) using a familiar editor or copy the SLP files from the TSX-Plus distribution media. The names of the SLP files are "ddTSX.SLP" where "dd" represents the two character source file name (such as DMTSX.SLP) for application to RT-11 version 5 sources. The SLP files for RT-11 V4 autopatch level E are named "ddV4.SLP". Copy the RT-11 handler source, named dd.MAC, from the RT-11 distribution media to dd.OLD on a scratch working pack (e.g., COPY DL1:DM.MAC DL0:DM.OLD). Apply the patch using the following command (do not use the /C:nnnnnn switch with RT-11 version 4):

```
SLP dd.MAC=dd.OLD,ddTSX.SLP/C:nnnnnn
```

where "dd" is the two character source file name and "nnnnnn" is the patch checksum from the following table for RT-11 version 5 handlers:

Patch file checksums for RT-11 version 5 handlers

DD	005747	DM	036211
DX	147610	DY	131466
FSM	140676	TJ	124673
TM	012771	TS	000444

For example, the following command would patch the DM version 5 source code:

```
SLP DM.MAC=DM.OLD,DMTSX.SLP/C:036211
```

For more information on the use of SLP, refer to Chapter 21 of the RT-11 System Utilities Manual. Note: Do not apply the patch to the original RT-11 distribution media as any RT-11 patches will require the original source file.

Whether or not patching is required, most handlers may be built (when necessary) 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 (dd.TSX files) for all devices included in your TSGEN DEVDEF list, including the system disk, must be on the system disk when TSX-Plus is started.

On the LSI-11 bus, only the DL, 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 256 Kb 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 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 which do not use direct memory access do not require 22-bit handlers or controllers or system I/O mapping.

The following RT-11 device handlers are unsupported under TSX-Plus: BA (resident batch handler), EL (SJ error logging pseudohandler), and PD (PDT-11/130/150 handler). The single line editor (SL) and logical disk support (LD) are implemented in TSX-Plus as overlay regions and do not require device handlers.

2.6.5 Device handler 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.

TSX-Plus stores the number of the job issuing an I/O request in bit positions 11 through 15 of the fourth word of the queue element. A job number of zero implies the I/O request was initiated from the operating system. User job numbers correspond to the TSX-Plus time-sharing line number.

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 or .FORK request.

Kernel page address register 5 is also available for use in device drivers. 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 startup) they must be saved and restored by the handler.

2.6.6 .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.

2.7 Setting the memory allocation for system programs

SETSIZ.SAV is a program that can be used to store information in a SAV file about how much memory TSX-Plus should allocate for the program when it is run. The method used to store this information in the SAV file does not affect the execution of the program when being run under RT-11. See Appendix A of the TSX-Plus Reference Manual for complete information about the SETSIZ program.

A command file named SETSIZ.COM is provided with TSX-Plus. It contains the necessary commands to cause the SETSIZ program to set appropriate allocation sizes for most of the commonly used system programs. To execute this command file, make sure the SETSIZ.SAV program and SETSIZ.COM command file is on the system disk then type:

```
@SETSIZ
```

The allocation sizes set by this command file should be adequate for most sites but a particular site might wish to alter them based on special requirements. Note that it is not necessary to execute the SETSIZ.COM file every time TSX-Plus is started since the size information is stored permanently in the program SAV files.

3. SYSTEM AND FILE ACCESS SECURITY

TSX-Plus provides a number of system security options that allow the site manager to control access to the system by timesharing users. By selecting the appropriate combination of options the system manager can control who can log onto the system, which files or devices each user can access and can also lock users to application programs. There are six facilities that can be used to control system access, all of which are described in this chapter:

1. Start-up command files.
2. Log-off command files.
3. The RUN/LOCK switch.
4. The ACCESS command.
5. The SET MAXPRIORITY command.
6. Operator privilege.
7. The LOGON and account authorization programs.

In addition to these security features, TSX-Plus also provides a use accounting facility that keeps track of the number of timesharing sessions and the total connect time that each user uses.

3.1 Start-up command files

When TSX-Plus is generated, the system manager may specify for each timesharing line the name of a command file that is to be executed each time the line is started. The command file name is specified by using a "CMDFIL" macro within the line definition block in TSGEN. This is explained in Chapter 2. Different command files may be specified for each line and any or all lines may be generated without start-up command files.

If a line has a start-up command file, that command file is started each time the line is initialized (e.g., when a user presses carriage return on an inactive line). Start-up command files are different than 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 a start-up command file to be executed to completion regardless of the actions of the timesharing user. However, if the command file aborts for some other reason, the line may be granted access to the system without proper initialization. This may 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 FATAL
    .
    .
    .
R/LOCK LOGON
OFF
```

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

A start-up command file can contain any keyboard command and can run one or more programs. Control-C resumes its normal function when start-up command file is terminated or a program initiated 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 operator privilege (see below). Note, if "TSX" is used as the file extension, it must be specified with the file name in the CMDFIL macro as 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.

3.2 Log-off Command Files

It is possible to declare 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 only legal 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.

3.3 The RUN/LOCK Switch

The "R" and "RUN" commands accept a "/LOCK" switch that causes the program being run to be "locked" to the timesharing line. A locked program executes in the normal fashion, and may chain to other programs (which become 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 an .SCCA EMT or by using the TSX-Plus "D" program controlled terminal option (see Chapter 6 in the TSX-Plus Reference Manual for information on defining activation characters).

In a situation in which a timesharing 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 entry in the file a "RUN/LOCK program" command.

3.4 The ACCESS Command

The ACCESS keyboard command is used to limit access to devices and files. The ACCESS command is unique in that it 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 "dev:file.ext" expressions may be specified. Each logical subset disk mounted also counts toward the limit of twenty entries in the access table.

If no ACCESS command is executed, the timesharing user is allowed to access all devices and files on the system (with the exception of SYS and TSX-Plus files--see Operator Privilege, below). If any 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 spooled line printer. Access privilege is needed to read, create, delete, or rename a file. A device can only be initialized (directory zeroed) if full access to the device is 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.

Because 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, 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, 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 the system library file (SY:SYSLIB.OBJ or 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 (20 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. Files on LD2 may be accessed for reading only. The user may also create nested logical disks within LD1.

3.5 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 to use 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.

3.6 Operator Privilege

Certain system facilities and keyboard commands are only available to timesharing users who are granted "Operator Privilege." The following list summarizes those system facilities that are restricted to users having operator privilege.

1. The $STOP, $SHUTDOWN and BOOT keyboard commands. If a user without operator privilege attempts to use one of these commands, TSX-Plus displays the message "?KMON-F-You're not privileged for that command." The $STOP, $SHUTDOWN and BOOT commands either halt the system or boot RT-11, depending on your system configuration. See the TSX-Plus Reference Manual for further information.

2. Real-time programming facilities such as the ability to access the I/O page and connect real-time interrupts to completion routines. Note that operator privilege is necessary to run the SYSMON utility, because it uses some real-time facilities.

3. Creation or execution of files with the extension ".TSX" or ".SYS".

4. Use of the TSAUTH account authorization program.

5. The ability to use the .PEEK and .POKE EMT's to access the I/O page or low memory areas.

6. The use of the EMT to set the user name (operator privilege is not required to determine the user name).

7. The ability to set the system date or time (DATE and TIME commands and the .SDTTM EMT).

8. Use of the SYSMON system status display program.

Operator privilege can be granted in either of two ways. If the LOGON program is used, individual accounts can be granted or denied this privilege (see Chapter 4, Account Authorization). If the LOGON program is not used, this privilege is specified on a line-by-line basis during TSX-Plus system generation by including $PRIV in the FLAGS macro (see Chapter 2 on line definitions). Note that even if the FLAGS macro for a given line includes the $PRIV flag and the LOGON program is run from that line, then operator privilege is still controlled by the account authorization system.

3.7 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 user by user basis.

To use the LOGON facility the system manager must first use the account authorization program (see Chapter 4) 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 FATAL
    .
    .
    .
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) should 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 "^(" can 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 FATAL
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.

4. ACCOUNT AUTHORIZATION PROGRAM

TSAUTH, the TSX-Plus account authorization program, is used to authorize project-programmer numbers 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 operator privilege to be allowed 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 operator 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 project-programmer numbers (PPN's) 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.

The format of the account authorization file changed with TSX-Plus version 4.0. If your authorization file was created with an earlier version of TSAUTH, the AUTCVT program, which is described at the end of this chapter, must be used to convert the file to the new format.

Once the authorization file is found or built, TSAUTH prints an asterisk indicating it is waiting for a command. Commands are entered as a single letter followed by a space and (for most commands) a user name or a project-programmer number typed with a comma separating the project number from the programmer number. The L(IST), K(ILL), U(SE) and R(ESET) commands allow a wildcard asterisk character to be used in place of the project number, programmer number or both. This allows sets of project-programmer numbers to be dealt with as a group. These commands also allow use of the user name instead of the PPN, allowing one to do these operations without knowing the PPN.

TSX-Plus gives no special significance to the grouping of accounts by project and programmer numbers. However, for convenience in using the TSAUTH program, it is suggested that a common project number be used for all PPN's associated with the same project or class and the programmer number be used to identify an individual.

User names given to accounts should be unique. As TSAUTH and LOGON read the ACCESS.TSX file sequentially, only the first occurrence of a given user name will be recognized at logon time or during TSAUTH file updating. For example, if two users have the user name SMITH and the one with the later record in the access file attempts to log on by using his name, LOGON will not recognize his password, as it is expecting the password for the first SMITH. However, he can still log in by specifying his project-programmer number, since PPN's must be unique.

4.1 Command summary

The following table lists a brief summary of the commands accepted by TSAUTH:

Command	Function
A proj,prog	Authorize a new account
K user-name	De-authorize an existing account
L user-name	List authorization status
U user-name	List account usage statistics
C	Create charge file (DK:CHARGE.TSX)
R user-name	Reset account usage statistics
E	Exit TSAUTH

Commands which accept a user-name will also accept project-programmer numbers or wildcards. The A command accepts only PPN's.

4.2 Authorizing a project-programmer number

The "A" (Authorize) command is used to add a new PPN to the authorization file. The form of this command is:

A proj,prog

A wildcard ("*") may not be substituted for either the project or programmer number. A project-programmer number must be deauthorized by the use of the kill command before it can be reauthorized. Project and programmer numbers must be decimal values in the range 1 to 65535.

TSAUTH responds to the "A" command by asking a series of questions as follows:

a) User Name:

Enter the user name to be associated with the PPN as a 1 to 12 character alphanumeric string. User names may be composed of letters and digits but may not contain spaces. The user name must be unique to the PPN; you may not have more than one PPN with a given user name. This is necessary as you can log in with the user name as well as with the PPN.

b) Password:

Enter a 1 to 8 character alphanumeric string that is the password to be associated with the PPN.

c) Start-up file:

Enter the name (dev:file.ext) of the start-up command file to be executed whenever this user logs on. Press return if no startup command file is desired. The default device is SY and the default extension is COM.

Note that for each job, two startup command files may be executed; the file associated with the line and the file associated with the user. Usually, the start-up command file associated with the line will contain little more than R/LOCK LOGON, whereas logical assignments and ACCESS commands will be located in the command file associated with the user, as defined in this parameter.

d) Virtual lines:

Respond with "Y" for yes or "N" for no, indicating whether the account is to be allowed to use the TSX-Plus virtual line facility.

e) Detached jobs:

Respond Yes/No indicating whether the account is to be allowed to use detached jobs.

f) Operator commands:

Respond Yes/No indicating whether the account has operator privilege. Generally, this should be restricted to a small set of users.

g) Maximum execution priority:

Respond with a (decimal) number in the range of 1 to 127 to restrict this user's maximum job execution priority. The default value is 50.

Example: In the following example a PPN 107,423 is authorized with the user name "DAGWOOD" and the password "SECRET" and the start-up command file named "SY:SU107.TSX" is specified. The maximum execution priority defaults to 50 since no value is entered.

Account Authorization

```
*A 107,423
User name:DAGWOOD
Password:SECRET
Start-up file:SY:SU107.TSX
Virtual lines:Y
Detached jobs:N
Operator commands:N
Maximum execution priority:
*
```

4.3 Deauthorizing Accounts

The "K" (Kill) command is used to deauthorize project-programmer numbers. The form of this command is:

```
    K proj,prog
or
    K user-name
```

where a wildcard character ("*") may be substituted for the project number, programmer number or both.

Examples:

1. Deauthorize PPN 107,423.

```
    *K 107,423
```

2. Deauthorize all PPN's with the project number 237.

```
    *K 237,*
```

3. Deauthorize a PPN with the user name DAGWOOD.

```
    *K DAGWOOD
```

4.4 Listing account status

The "L" (List) command is used to list the current authorization status of any or all accounts. The form of this command is:

```
    L proj,prog
or
    L user-name
```

The wildcard character may be substituted for the project and programmer numbers. The information listed includes all of the items that were specified when the account was authorized.

Example:

List the current status of the account with user name DAGWOOD.

```
*L DAGWOOD

PPN:107,423
user-name:DAGWOOD
Password:SECRET
Start-up file:SY:SU107.TSX
Virtual lines:Y
Detached jobs:N
Operator commands:N
Maximum execution priority:50
*
```

4.5 Listing Account Usage Statistics

The "U" (Usage) command can be used to display the account usage statistics which consist of the number of sessions, the connect time, and the CPU time. The form of this command is:

```
U proj,prog
```
or
```
U user-name
```

The wildcard character may be substituted for the project number, programmer number, or both.

Example:

List the usage statistics for all accounts with the programmer number 423.

```
*U *,423

PPN:107,423     # sessions=14   Connect time=01:23:00  CPU=00:03:07.4
PPN:413,423     # sessions=5    Connect time=14:02:00  CPU=01:13:02.5
PPN:21,423      # sessions=10   Connect time=01:11:00  CPU=00:49:18.9
```

4.6 Creating a Charge Information File

The "C" (Charge) command causes TSAUTH to create a file of usage information. The file is named "DK:CHARGE.TSX"; it contains one record for each PPN; 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)

4.7 Resetting Account Usage Statistics The "R" (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 this command is:

```
R proj,prog
```
or
```
R user-name
```

where the wildcard character may be substituted for the project number, the programmer number, or both.

Examples:

1. Reset all PPN's with the project number 21.

```
*R 21,*
```

2. Reset all PPN's.

```
*R *,*
```

3. Reset the PPN with the user name DAGWOOD.

```
*R DAGWOOD
```

4.8 Exiting from the Account Authorization Program The "E" (Exit) command is used to exit from the TSAUTH program to the keyboard monitor. The form of this command is:

```
E
```

4.9 AUTCVT program

The TSAUTH and LOGON programs supplied with TSX-Plus are incompatible with authorization files created prior to version 4.0. A conversion program named AUTCVT is supplied to convert old format authorization files to the new format. It is suggested that a copy of the old format file be made before the conversion is performed since there is no way to convert back to the old format.

The AUTCVT program reads an old format authorization file with the name SY:ACCESS.TSX and creates a new format authorization file with the same name (superseding the old file). This conversion should be done while running under RT-11 before starting TSX-Plus. In the process of converting the file, AUTCVT prints each project-programmer number and requests a corresponding user name to be entered. If you do not wish to specify a user name for some PPN's, press return without entering a name. This will limit the user to logging on using his PPN. If a user name is entered, then either the user name or the PPN may be used in subsequent logons.

5. SYSTEM OVERVIEW

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 system internal organization and operation.

5.1 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 4Kb 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.

5.1.1 System Memory Mapping

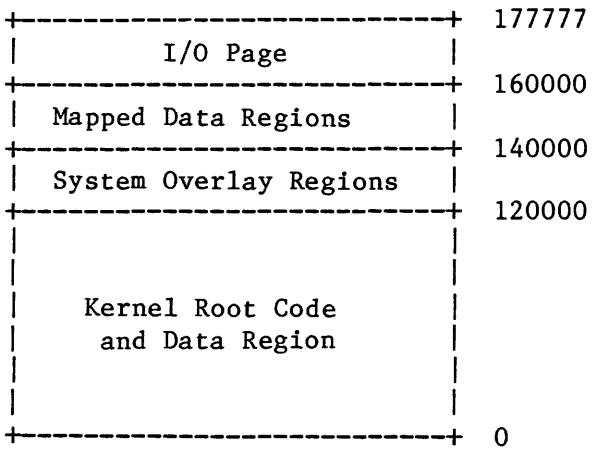
The operating system is divided into four distinct regions: kernel root, system overlays, mapped data, and the I/O page. The kernel root is mapped using kernel PARs (page address registers) 0 through 4. Because of this, the kernel root code region is restricted to a maximum of 40 K bytes. (Each PAR maps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

Each system overlay code region is mapped through kernel PAR 5 and is therefore restricted in size to a maximum of 8K bytes. Only one memory resident overlay 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



5.1.1.1 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 swapper, scheduler, etc.; I/O related processing code; clock and terminal interrupt entry code; and the startup initialization code. To conserve space, TSX-Plus re-uses the memory containing the startup initialization code by allocating data structures which do not require initialization over it after completion of startup. If additional space is necessary, the top of TSX-Plus is extended. These buffers consist of the job information tables (simulated RMON); I/O queue elements; and system message buffers. Device handlers are loaded above these data buffers. The size of the entire kernel root region described here (including device handlers) must not exceed 40Kb. See Appendix E for more information about the size of the system.

5.1.1.2 System Overlay Regions: There are currently eleven 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. Seven 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
3. Directory manipulation requests and directory cache buffers
4. Miscellaneous executive functions such as clock processing and fatal error processing
5. * Program logical address space requests (PLAS)
6. * Device spooling with buffers
7. * Record locking and data structures
8. * Message communication and data structures
9. * Real-time service requests
10. * Mapped I/O servicing
11. * Single line editor

* Denotes optional overlays that are only loaded into memory if the corresponding feature is selected during system generation. See Appendix E for information about the size of the optional system overlays.

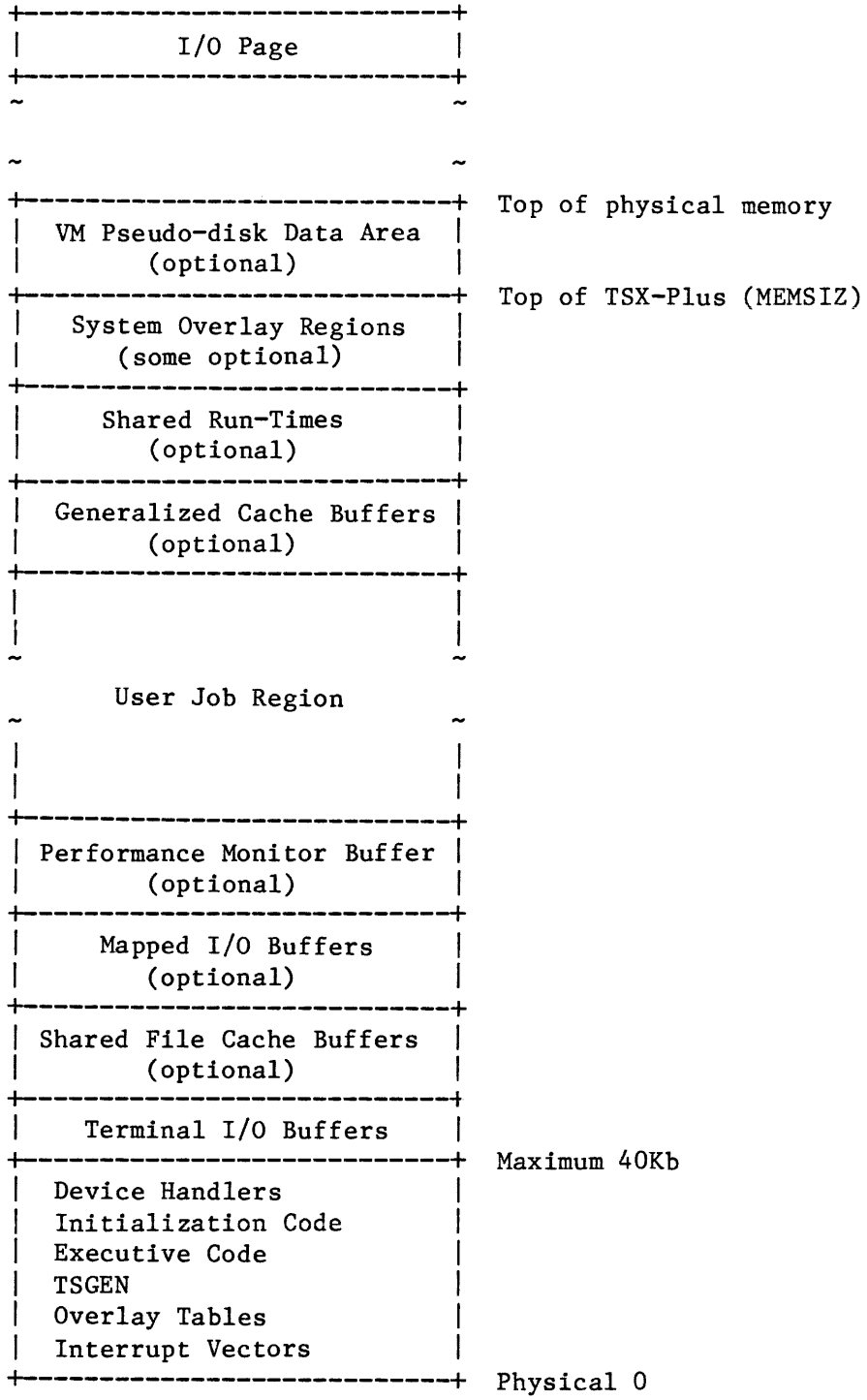
5.1.1.3 Mapped Data Regions: The mapped data regions allocated during startup contain the terminal input and output character buffers, and the following optional buffers: shared data cache buffers, mapped I/O buffers, performance monitor buffers, and generalized data cache buffers.

5.1.1.4 Shared Run-time Systems: In addition to the system regions described, a fifth region, also pre-allocated by the system but not directly used by it, contains user-defined shared run-time systems such as those provided with COBOL-Plus and DBL.

5.1.2 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 40Kb. 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. See Appendix E for information about the size of various optional system features. 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 the buffers used by the generalized data caching facility. Finally, all the physical memory between the mapped data regions and shared run-time systems 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



5.1.3 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 4Kb 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 the 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.

5.2 I/O Mapping

I/O mapping is a facility which allows DMA devices with 18-bit controllers 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 256Kb 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 4Mb 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 256Kb of memory. This would 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 256Kb of memory. This facility may be specified selectively for those DMA devices that only have 18-bit controllers. 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 do not need system buffering and can operate normally.

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 256Kb it moves the data from the user's buffer to/from a system buffer 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

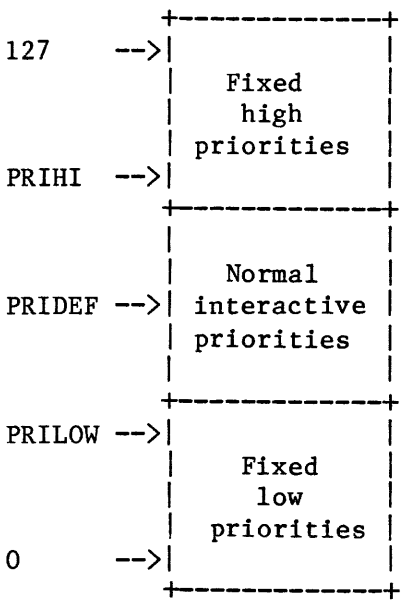
amount of data being transferred is larger than the system buffer. In this case, an I/O operation which would normally be accomplished as a single transfer will be broken down into a series of smaller transfers. When a large operation is broken down into a series of smaller operations time is lost waiting 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.

5.3 Job Scheduling.

TSX-Plus 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.

5.3.1 Job Priorities

The 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:



5.3.1.1 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 sched-

uling 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.

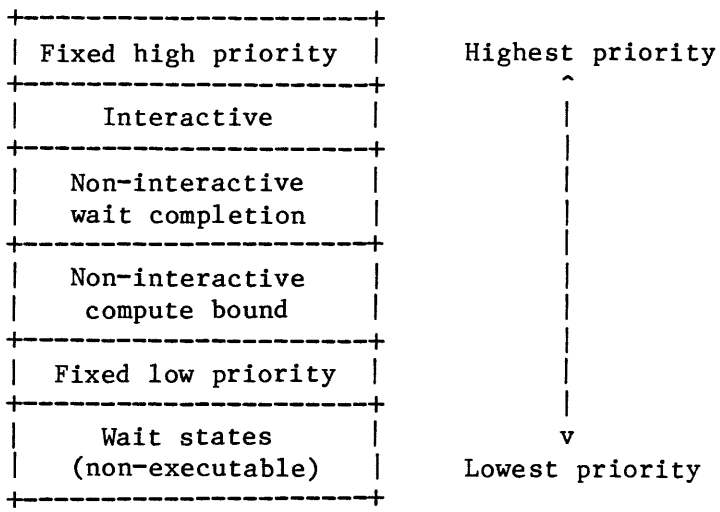
The fixed-high-priority group is intended for use by real-time programs. See the chapter on real-time program support in the TSX-Plus Reference Manual. The fixed-low-priority group is intended for use by very low priority background tasks. Normal time-sharing jobs should not be assigned priorities in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

5.3.1.2 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 virtual line, the priority of the disconnected job 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. The priority reduction is also constrained so that the priority will never be reduced below the value of (PRILOW+1).

5.3.2 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:



5.3.2.1 Wait States: Currently, there are sixteen 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.

5.3.2.2 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 (this is effectively the same as when a job receives an activation character), 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.

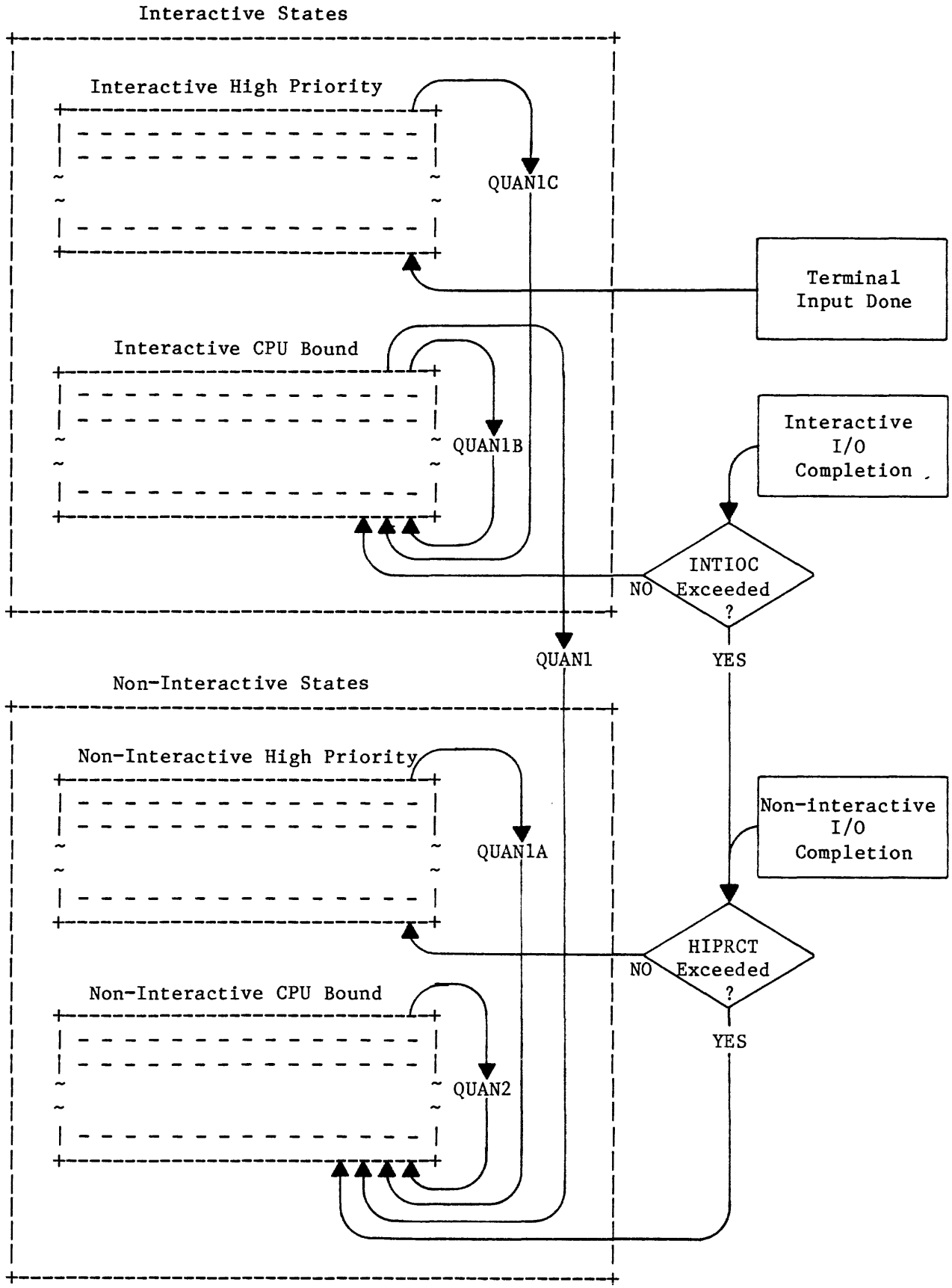
The job remains in the highest priority interactive state until it either has executed for QUAN1C units of time or performs an I/O operation. At that time, the job is rescheduled into the next lower execution state in the interactive group (interactive-CPU). On return from an I/O operation (during which 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

5.3.3 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.

5.4 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 whenever the swapper determines that there is a job in an executable state out of memory and there is a job in a lower priority state in memory. 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 time (clock time, not the job's execution 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, then the system "holds" its I/O. Jobs may also lock themselves in memory by using real-time EMT requests.

5.5 Real-time Interrupt processing:

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

5.5.1 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.

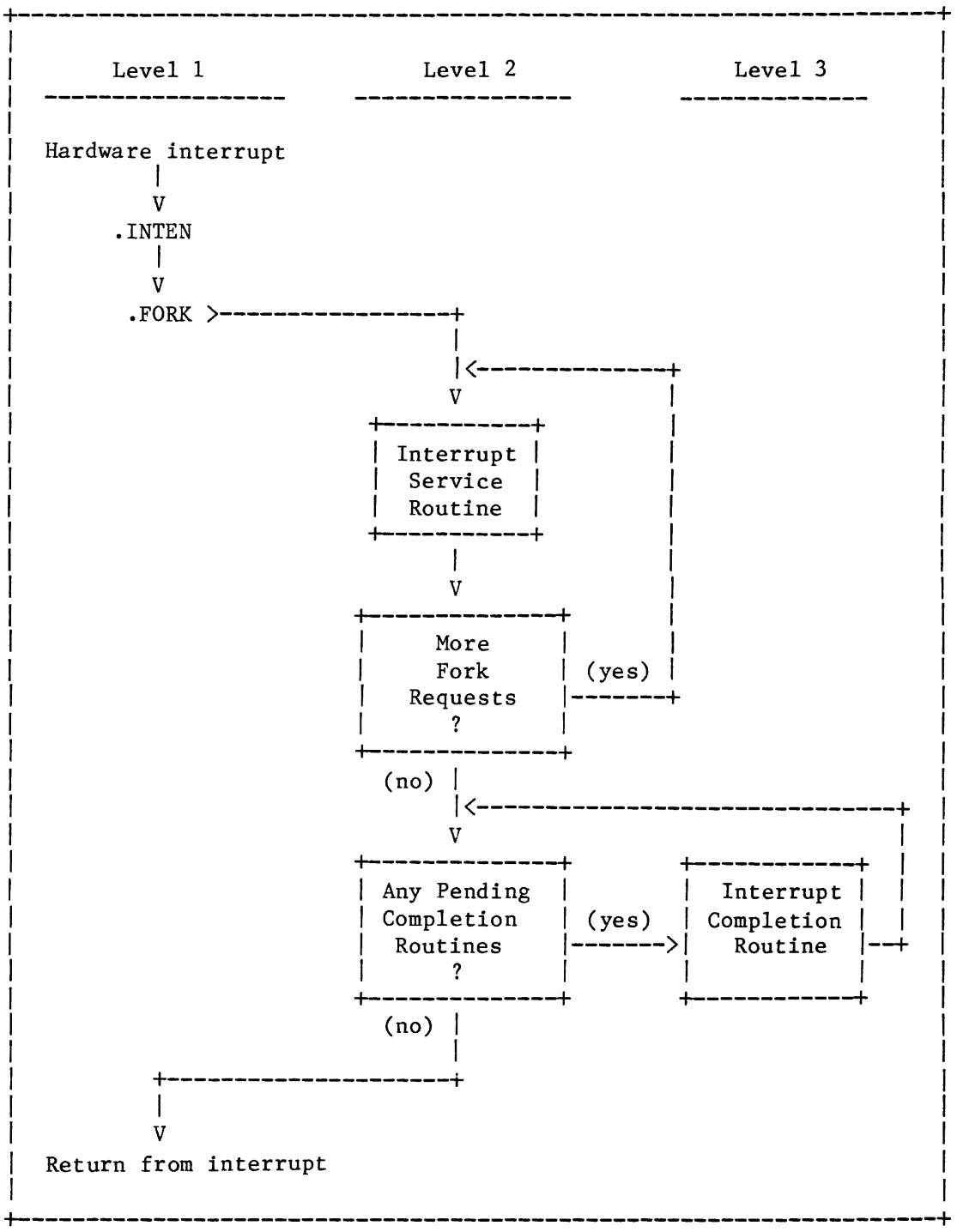
5.5.2 Real-time interrupt completion routines:

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 full 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.

6. 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.

6.1 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 4Mb 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.

6.1.1 System Memory Utilization

The memory utilization of the operating system is discussed and illustrated in Chapter 5. 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, virtual, 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 virtual lines 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). Appendix E 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.

6.1.2 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. The SYSMON utility can also be used to dynamically monitor memory allocation.

6.2 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 parameters such as QUAN1 and INTIOC (as well as QUAN1A, QUAN1B, etc.) 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 such as QUAN1 and INTIOC (as well as QUAN1A, QUAN1B, etc.) 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.

6.3 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.

6.4 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.

6.4.1 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.

6.4.2 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.

6.4.3 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 somewhat 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.

6.4.3.1 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.

6.4.3.2 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 at all. 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.

6.4.3.3 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 requiring no disk I/O to be performed. 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 CACHE sysgen parameter can be set to this value and the system regenerated. A "SHOW CACHE" keyboard command can be used to display the current number of blocks 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).

6.4.4 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.

7. SYSMON - DYNAMIC SYSTEM DISPLAY UTILITY

SYSMON is a dynamic interactive utility program written to display information about system activities at a VT100 or VT52 display terminal. It was written to help the system manager to optimize system resource use and verify that certain operations are taking place. Currently it provides dynamic screen displays of how CPU and I/O times are being used, job status, terminal status, message channel status, user time bar chart, cpu times bar chart, contents of the directory cache, and data cache usage statistics.

SYSMON draws most of its information from the low memory area of TSX-Plus, therefore it will not function without real-time support included in TSGEN as it uses a real-time EMT to access system tables. Operator privilege is required to run the SYSMON program. If a user without operator privilege tries to run this program, or anyone tries to run it without real-time support generated, he will receive the message:

```
Real-time has not been generated
```

7.1 Creating and Running SYSMON

SYSMON is created every time TSX-Plus is relinked using the TSXLNK.COM command file supplied in the distribution. This is necessary as offsets will change every time TSGEN changes. If the link is not done on the system disk, copy the file SYSMON.SAV from the generation disk used for linking 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. 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 page.

7.2 SYSMON Menu

```
+-------------------------------------------------------------------------+
|                                                                         |
|        License : 999SP  - 44          S & H Computer Systems            |
|                                                                         |
|                                                                         |
|                                                                         |
|                                                                         |
|  Enter selection :                      Sample time :                   |
|  (RETURN to exit)                       (RETURN defaults to 10. seconds)|
|                                                                         |
|                                                                         |
|                                                                         |
|  1. System status display               6. CPU modes display            |
|                                                                         |
|  2. Job execution status display        7. Directory cache display      |
|                                                                         |
|  3. Terminal status display             8. Shared file cache display    |
|                                                                         |
|  4. Message queue display               9. Generalized data cache display|
|                                                                         |
|  5. User times display                                                  |
|                                                                         |
+-------------------------------------------------------------------------+
```

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. Beware that on some systems (notably 11/23 based systems) that 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.

7.3 System Status Display

```
License : 999SP  - 44          S&H Computer Systems

                ***** System Status Display *****

Total Uptime      00:35:41.0     Cur      Total        System Parameters

User Job 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 =    41
                                                         CORTIM =     2
Swap I/O Time     00:00:00.6 *   0.0%      0.0%          IOABT  =     0

* - 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.

Seven rows of information are presented. The first is Total Uptime, that is, the amount of time since the system was booted. The User Job time is the time used in computation, that is, actual CPU activity. The I/O Wait time is the amount of time user jobs spend waiting on information from various I/O devices. The Swap Wait time is the time the system spends waiting for a job 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 jobs spend performing I/O. Finally, the Swap I/O time is the amount of time the operating system spends swapping user jobs 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 job as another job 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 get interrupted while computing I/O wait time, during which the times for the subsequent display items may change.

System scheduling parameters are displayed across the bottom of the screen. The values of these parameters can be dynamically changed during system operation by use of the keyboard SET command.

7.4 Job Execution Status Display

```
License : 999SP  - 44          S & H Computer Systems

          *****  Job Execution Status Display  *****

 Job   Line   Pri  Program  User Name     Run   Size  State
 ---   ------ ---  -------  ---------     ---   ----  -----
  1    1(0)   50   KMON     TSX                 32    Wait-TT input
  3    3(0)   50   KMON     CRAIG               32    Wait-TT input
  4    4(0)   50   SYSMON   SAM           I     20    TT input done
  6    6(0)   50   KMON     DIRECTOR            32    Wait-TT input
  7    7(0)   50   KED      SOFTWARE            34    Wait-TT input
  9    Det.   50   RTSORT   [000,000]          60    Wait-message
 11    4(1)   40   KED      SAM                 34    Wait-TT input
 12    4(2)   40   MACRO    SAM           C     60    CPU bound job
```

The job execution status display provides a variety of useful information. For each job on the system, it provides:

1. Job Number - the TSX-Plus job number assigned to that particular primary line, virtual line, or detached job.

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

3. Priority - the current priority for this job.

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

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

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

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

8. Current execution state - What the job is currently doing. The execution states are:

State Displayed	Meaning
Real time state	Executing high priority real-time job
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 Job compute	Interactive priority job executing
Timed wait completion	Finished executing .TWAIT or .MRKT request
TT output buffer low	Ready for program to continue output
I/O completion	I/O transfer completed
CPU bound job	Normal job executing
Low Priority Computation	Low priority job 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-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 locked
Wait-system message buf	Waiting for free system message 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 done
Wait-timed interval	Waiting for interval to finish
Wait-memory expansion	Waiting for memory expansion to complete

7.5 Terminal Status Display

```
+-------------------------------------------------------------------------+
|                                                                         |
|        License : 999SP  - 44          S & H Computer Systems             |
|                                                                         |
|                    ****   Terminal Status Display  *****                |
|                                                                         |
|                                                                         |
| Terminal line number = 12                                               |
| Terminal type = VT100, virtual, primary =  4, logged on                 |
| Single line editor status = Enabled, NOKED, TTYIN, Inactive             |
| Terminal Characteristics = DEFER ECHO NOFORM NOFORMO NOGAG LC PAGE PRIV |
| NOQUIET SCOPE NOSINGLE TAB NOTAPE WAIT                                  |
|                                                                         |
| Rubout filler character .  " "       Escape-letter activation   disabled |
| Virtual line . . . . . .  enabled    Transparent output . . .  disabled |
| Command file input . . .  enabled    Field width activate . .    0       |
| High-efficiency mode . .  disabled   Field limit activate . .    0       |
| Echo LF after CR . . . .  enabled    Default terminal editor .  KED       |
| Single char activation .  disabled   Inhibit terminal wait   .  disabled |
| UCL setting . . . . . .  middle                                         |
| Activation characters . .  none                                         |
| Input buffer contents . .  Unavailable - SL active.                     |
|                                                                         |
|                                                                         |
+-------------------------------------------------------------------------+
```

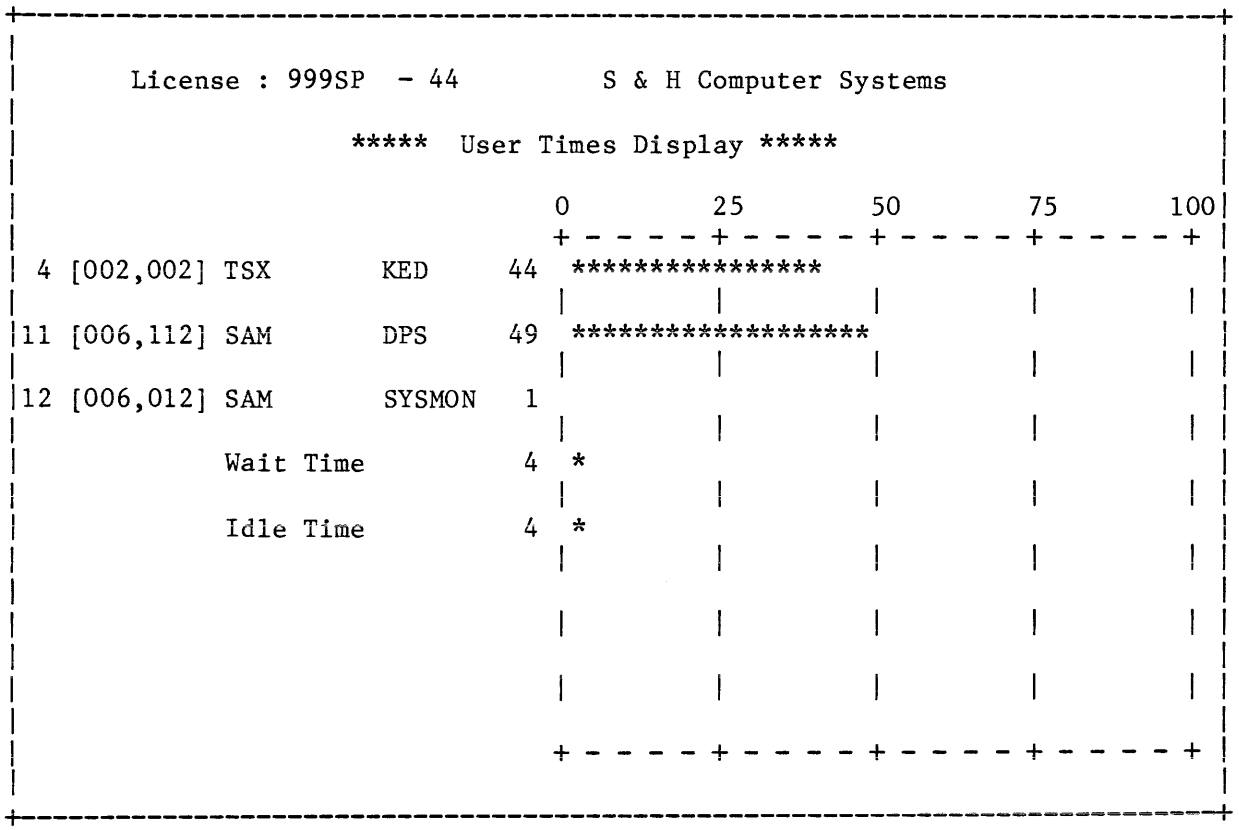
The terminal status display shows the parameters that are currently set on a given terminal line, various terminal characteristics, the terminal's owner, and type of job the terminal is connected to. This display will prompt you first for a job number - this can be obtained from the TSX-Plus SYSTAT command or the job execution status display. If you do not enter a number or enter an invalid line number, the parameters for your job will be displayed. For information on the parameters displayed, see the TSX-Plus Reference Manual, Chapters 2, 4, and 6.

7.6 Message Queue Display

```
License : 999SP  - 44          S & H Computer Systems

                  *****  Message Queue Display  *****

Channel  Message (only 70 characters printed)

TSTMSG   THIS MESSAGE SUBMITTED TO TEST SEND MESSAGE EMT
SAM      LUNCH TIME

Jobs waiting (Channel name, job number)
RTSORT    9
```

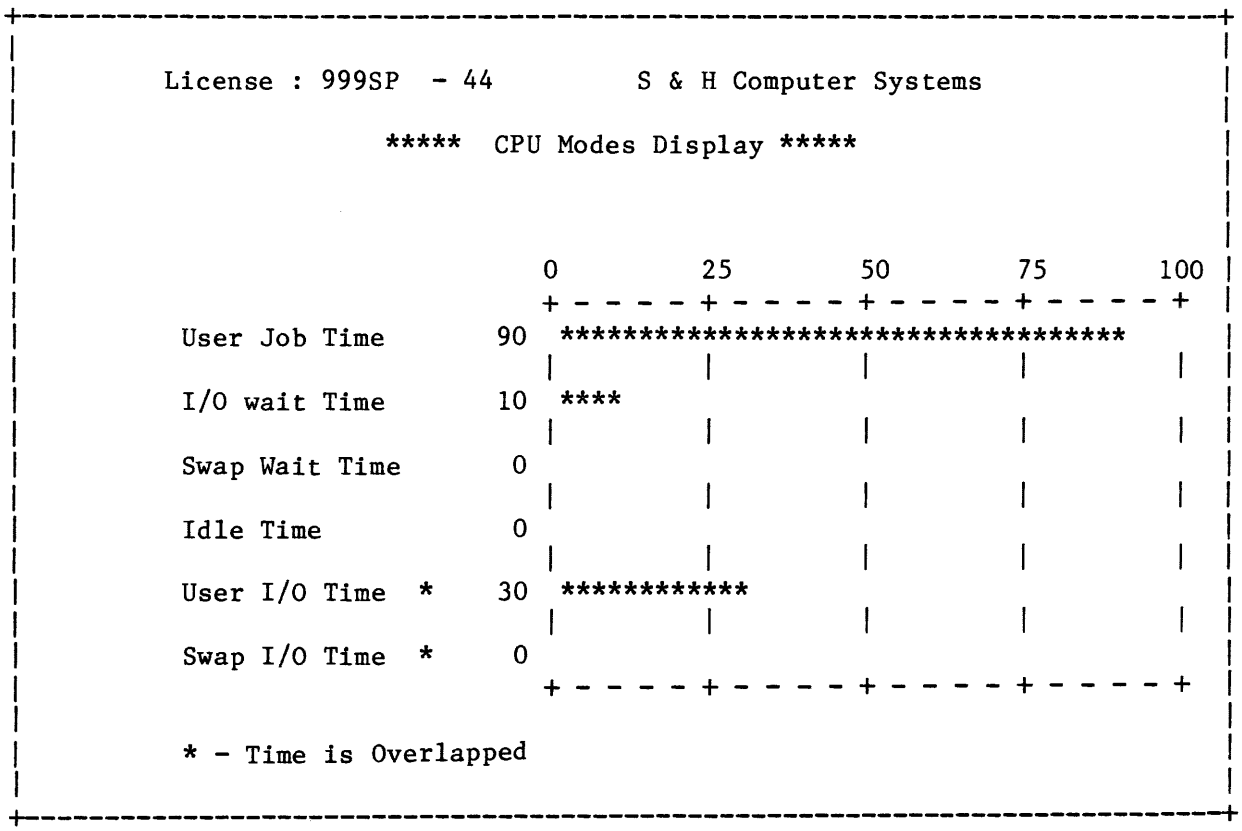
The message queue display shows both what messages are waiting in what message channels, and what jobs are waiting on what channels for messages. This information is useful to verify that proper information is being sent in the proper channels. For example, if you send a sort command string on the RTSORT message channel without RTSORT running, you can verify that the proper information is being sent, as the message will sit in its channel without a job to read it. Similarly, you can verify that a job is waiting on the proper channel. In this example, RTSORT is running as a detached job, running the messages queued display shows RTSORT waiting on a message channel named RTSORT on the first available detached line. For more information on message channels see the TSX-Plus Reference Manual, Chapter 10.

7.7 User Times Display



The user times display shows, for each job that used at least one percent of the CPU time during the last sample period: the job number, project-programmer number, user name, program running, percentage of time used by that job during the last sample period, and a bar graph depiction of that percentage. If a job uses less than one percent of the time during the period, it will not be displayed for that period. This display is useful for determining how CPU time is being used, and the interrelationship between job time, wait time (which includes both I/O wait and swap wait time), and idle time.

7.8 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.

7.9 Directory Cache Display

```
+-------------------------------------------------------------------------+
|                                                                         |
|        License : 999SP  - 44         S & H Computer Systems             |
|                                                                         |
|                    *****  Directory Cache Display *****                 |
|                                                                         |
|                                  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 files 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. Upon initial entry to this display, you will be asked for the number of pages to display. If you enter 1 or simply hit return, only the first 38 entries in the directory cache will be displayed. Otherwise, the display will cycle over as many pages as you request, until there are no more active cache entries. In this mode, 34 entries are displayed per page. 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.

7.10 Shared File Data Cache Display

```
License : 999SP  - 44          S & H Computer Systems

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


                                      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 as 16 bit 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.

7.11 Data Cache Display

License : 999SP - 44 S & H Computer Systems

***** Data Cache Display *****

	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		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 is readily usable, but to leave sufficient free memory both for other purposes, such as job memory or VM, and to minimize swapper activity. In this context, swapper activity includes both swapping of jobs in and out of memory, and the moving of jobs around in memory to enable all jobs to get the memory they need when job sizes are changing. Note that both the SET CACHE command and the RESET command will reset the cache counters.

7.12 Exiting SYSMON

To leave 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 file "SY:dd.TSX" cannot be opened (where dd represents a two character device driver name). TSX-Plus requires a device driver for every device listed in TSGEN. If the device name was not correctly specified, correct the device name and generate a new system. Standard device drivers are provided on the distribution media. If this is a standard device driver, find the distribution media and copy dd.TSX to SY. If this is a user written driver, the device handler must be generated for TSX-Plus. Review the System Manager's Guide for building device drivers for TSX-Plus.

Cannot find "SY:CCL.SAV" file

The file "SY:CCL.SAV" cannot be opened. This file exists on the distribution media. Find the distribution media and copy this file to SY.

Cannot locate "SY:SYSODT.REL" file

The file "SY:SYSODT.REL" cannot be located when attempting to run the system debugger. This file exists on the distribution media. Find the distribution media and copy SYSODT.REL to SY. Review the instruction necessary to run the debugger.

Cannot find "SY:TSKMON.SAV" file

The file "SY:TSKMON.SAV" cannot be opened. This file is built during the generation process. Review the system generation process to determine why the file was not created.

Cannot locate "SY:TSX.SAV"

The file "SY:TSX.SAV" could not be opened. This file is built during the generation process. Review the system generation process to determine why the file was not created.

Cannot open program swap file
Number of contiguous blocks needed = nnnnnn

The program swap file (specified in TSGEN) cannot be created because the assigned disk does not have enough contiguous blocks. The number of contiguous blocks required is specified by nnnnnn. Delete any unnecessary files from the disk assigned to contain the swap file and squeeze the disk to consolidate the empty space. When enough free space is obtained, then run TSX-Plus.

Cannot open shared run-time file: dev:file.ext

The shared run-time file "dev:file.ext" specified in TSGEN could not be opened. If the file was incorrectly specified, correct TSGEN and generate a new system. Locate the shared run-time file and copy it to dev.

Startup Error Messages

Cannot open spooled device: dd
The spooled device dd cannot be opened by TSX-Plus. Review TSGEN to determine if the device specified was correct. Make sure the RT-11 device handler "SY:dd.SYS" exists and that dd is installed in RT-11.

Error on read of SYSODT rel file
An error occurred when reading the system debugger file "SY:SYSODT.REL" into memory. Check the system disk and the hardware involved.

Error reading device handler file: dd
An 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 error occurred when reading 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 generated is too large to load and run. Remove any unnecessary features, decrease excessive parameters, and generate a smaller system.

Handler for SY device was not loaded
The handler for SY was not specified in TSGEN. Correct TSGEN and generate a new system.

Handler not generated with extended memory support: dd
The device handler file "SY:dd.TSX" was not generated with support for memory management. Make sure the device handler was written to support memory management and review the System Manager's Guide for building device drivers for TSX-Plus.

Insufficient disk space for spool file
The spool file (specified in TSGEN) cannot be created because the assigned disk does not have enough contiguous blocks. The number of contiguous blocks required was specified in TSGEN. Either decrease the number of blocks required in TSGEN; or delete any unnecessary file from the disk assigned to contain the spool file and squeeze the disk to consolidate the empty space. When a block of free space exists that is greater than or equal to the number of blocks specified in TSGEN, then run TSX-Plus.

Insufficient memory to load all mapped system regions
There is not enough memory to load TSX-Plus. Review the Software Product Description to determine if you have enough memory. Review the system generation (TSGEN), remove any unnecessary features, decrease excessive parameters, and generate a smaller system.

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. Purchase enough memory to load all shared run-time systems or review the system generation (TSGEN), remove any unnecessary run-time systems and features, decrease excessive parameters, and generate a smaller system.

Invalid interrupt vector address for T/S line:
Line # = nn

The line numbered nn has an invalid vector address. Lines are numbered sequentially in TSGEN starting with the first line definition. Correct the vector address and generate a new system.

Invalid status register address for T/S line:
Line # = nn

The line numbered nn has an invalid status register. Lines are numbered sequentially in TSGEN starting with the first line definition. Correct the status register and generate a new system.

RT-11 doesn't recognize device: dd

The dd handler is not recognized by RT-11 and therefore cannot be loaded by TSX-Plus. Install the RT-11 device driver for dd. When the driver has been successfully installed in RT-11, then run TSX-Plus.

System is not equipped with extended memory management hardware

TSX-Plus does not find extended memory management hardware. Purchase extended memory hardware or specify less than or equal to 248Kb of memory in TSGEN and generate a new system.

System is not equipped with memory management hardware

TSX-Plus cannot be run on the current hardware configuration because it requires memory management support. Purchase memory management hardware to run TSX-Plus. Check the Software Product Description for TSX-Plus to determine if any other hardware is required.

TSX is already running

TSX-Plus is currently running and therefore cannot be started again.

Appendix B - System Error Messages

The error messages listed below are fatal and once reported, the operating system halts. All messages are displayed as four line messages with the following format:

```
<BELL>?TSX-F-Fatal system error at nnnnn
EEE-Error message displayed here (see below)
Arg. value = xxxxxx
Seg. value = yyyyyy
```

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.

INO-Interrupt occurred at location 0

The hardware has asserted an interrupt at location 0 where no device resides.

KRE-KMON read error

An input error occurred when attempting to read the file "SY:TSKMON.SAV" into memory. This indicates a probable hardware malfunction with the system disk.

KTP-Kernel mode trap

A trap through vector 4, 10, or 250 occurred in kernel space while at interrupt level. The argument value indicates the address at which the trap occurred and the segment value indicates the mapped memory resident overlay region.

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 queue elements were available.

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.

SIE-Swap file I/O error
An input or output error occurred either reading or writing into the program swap file.

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 system stacks has overflowed. The argument value reports a number that indicates which stack has created the failure.

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 definition, a device handler, or a real-time completion routine. The argument value indicates the vector location.

Appendix D - Device CSR and vector address table

System PDP-11/73 ORION CPU & Serial # _____

Memory installed 1MB Peripherals installed RX50, RL01 (DEC 880), RX02 (DSD), DLV-11J

Interface device(s) installed DLV-11J

Device	Type	Handler	Vector	CSR/RSR	Speed	Parity	Stop-bits

Appendix C - Device CSR and Vector Address Table

System _____ CPU & Serial # _____

Memory installed _____ Peripherals installed _____

_____

Interface device(s) installed _____

_____

Device	Type	Handler	Vector	CSR/RSR	Speed	Parity	Stop-bits

Appendix D - Device Driver Source Language Patch Files

The following source language patch files are provided in the event it becomes necessary to rebuild a device handler. The device handler files supplied on the distribution medium have already been patched using these SLP's and linked with the appropriate conditional file to create the dd.TSX files.

D.1 RT-11 Version 5 Patches

The following SLP's are for application to the RT-11 version 5, uncommented source files. These patches are also provided on the distribution medium as files with names of the form - dd.SLP. See the section in this manual on rebuilding device handlers before attempting to apply these patches.

DD SLP Patch File

```
-48,48
        KISAR1  = 172352                ;USE KERNEL PAR 5
-174
        .IF NE TSX$P                    ;CONDITIONAL CODE FOR TSX-PLUS
        MOV     DDCQE,R4                ;GET POINTER TO THE QUEUE ENTRY
        SUB     #20000,Q$BUFF(R4)       ;ADJUST BUFFER ADDRESS FOR PAR 5
        .ENDC                           ;END OF TSX-PLUS CONDITIONAL CODE
-235
        .FORK   PDFBLK                  ;DO THE .FORK
/
```

DM SLP Patch File

```
-219,219
        CMP     Q$BUFF(R5),#160000;CHECK PAR 6 UPPER BOUNDARY
/
```

DX SLP Patch File

```
-178
        BR      0.BAD                   ;DISABLE /[NO]WRITE UNDER TSX-PLUS
-293,293
        ADD     #2,R0                   ;INCREMENT BUFFER ONE WORD
-372,372
        CMP     Q$BUFF(R4),#140000      ;CHECK PAR 6 LOWER BOUNDARY
/
```

DY SLP Patch File

```
-193
        BR      0.BAD                   ;DISABLE /[NO]WRITE UNDER TSX-PLUS
-314,314
        ADD     #2,R0                   ;INCREMENT BUFFER ONE WORD
-384,384
        CMP     Q.BUFF-Q.BLKN(R4),#140000;CHECK PAR 6 LOWER BOUNDARY
/
```

Device Handler SLP Files

FSM SLP Patch File

```
-873
        CLR     EXTADR          ;CLEAR 18-BIT ADDRESS EXTENSION
-937
        CLR     EXTADR          ;CLEAR 18-BIT ADDRESS EXTENSION
/
```

TJ SLP Patch File

```
-296,296
        BIC     #177740,R2      ;ISOLATE THE JOB NUMBER
        .IF     NE, MMG$T
        MOVB    R2,JOBNM        ;STORE JOB NUMBER IN JOBNM
        .ENDC
-426,426
        BEQ     7$              ;BR IF NO ACTIVE QUEUE ELEMENT
-432,432
        BIC     #177740,R3      ;ISOLATE THE JOB NUMBER
-434,434
        BNE     7$              ;BR IF NOT THE ABORTING JOB
/
```

TM SLP Patch File

```
-264,264
        BIC     #177740,R2      ;ISOLATE THE JOB NUMBER
        .IF     NE, MMG$T
        MOVB    R2,JOBNM        ;STORE THE JOB NUMBER IN JOBNM
        .ENDC
-404,404
        BEQ     7$              ;BR IF NO ACTIVE QUEUE ELEMENT
-410,410
        BIC     #177740,R3      ;ISOLATE THE JOB NUMBER
-412,412
        BNE     7$              ;BR IF NOT THE ABORTING JOB
/
```

TS SLP Patch File

```
-337,337
        BIC     #177740,R1      ;ISOLATE THE JOB NUMBER
        .IF     NE, MMG$T
        MOVB    R1,JOBNM        ;STORE THE JOB NUMBER IN JOBNM
        .ENDC
-543,543
        BIC     #177740,R3      ;ISOLATE THE JOB NUMBER
/
```

D.2 RT-11 Version 4 Patches

The following SLP's are for application to the RT-11 version 4, uncommented source files. These patches are only appropriate for RT-11 V4 autopatch level E. See the section in this manual on rebuilding device handlers before attempting to apply these patches.

DD SLP Patch File

```
-49,49
KISAR1  = 172354
-144,144
DDINT::;        BCS     1$
/
```

DL SLP Patch File

```
-62
RLBAE   = 10
MONLOW  = 54
CONFG2  = 370
EXTLSI  = 20000
-308,308
DLERJM: JMP     DLERRH
-346,346
DLXFER: ADD     #RLBAE,R4
-348,355
        MOV     @#MONLOW,R3
        BIT     #EXTLSI,CONFG2(R3)
        BEQ     2$
.IF NE MMG$T
        MOV     Q$PAR-Q$WCNT(R5),R3
        ASH     #-4,R3
        BIC     #170000,R3
-357,365
-366
2$:     MOV     (PC)+,R3
DLWTRK:  .WORD  0
        CMP     R3,@R5
        BLOS    1$
        MOV     @R5,R3
1$:     MOV     R3,(PC)+
DLWC:    .WORD  0
        NEG     R3
        MOV     R3,-(R4)
        MOV     (PC)+,-(R4)
DLDA:    .WORD  0
        MOV     -(R5),-(R4)
.IF NE MMG$T
        MOV     Q$PAR-Q$BUFF(R5),R0
        BIC     #^C60,R0
.IFF
        CLR     R0
.ENDC
        BIS     (PC)+,R0
DLCODE:  .WORD  0
        BIS     (PC)+,R0
DLUNIT:  .WORD  0
-397,397
1$:     JMP     DLTRAK
/
```

DM SLP Patch File

```
-34,34
KISAR1  = 172354
/
```

DX SLP Patch File

```
-73,73
KISAR1  = 172354
-154,154
        ADD     #2,R0
/
```

DY SLP Patch File

```
-176,176
        ADD     #2,R0
-251,251
        CMP     Q.BUFF-Q.BLKN(R4),#140000                    ;001
/
```

FSM SLP Patch File

```
-870
        CLR     EXTADR
-934
        CLR     EXTADR
/
```

TJ SLP Patch File

```
-286,286
        BIC     #177740,R2
        .IF     NE,MMG$T
        MOV     R2,JOBNM
        .ENDC
-414,414
        BEQ     7$
-420,420
        BIC     #177740,R3
-422,422
        BNE     7$
```

Device Handler SLP Files

TM SLP Patch File

```
-254,254
        BIC     #177740,R2
        .IF     NE,MMG$T
        MOV     R2,JOBNM
        .ENDC
-392,392
        BEQ     7$
-398,398
        BIC     #177740,R3
-400,400
        BNE     7$
/
```

TS SLP Patch File

```
-311,311
        BIC     #177740,R1
        .IF     NE,MMG$T
        MOV     R1,JOBNM
        .ENDC
-500,500
        BIC     #177740,R3
/
```

Appendix E - System Size Calculation

E.1 System size and sysgen features

The TSX-Plus system is divided into two portions: an unmapped portion that consists of kernel code, device handlers, and job control tables; and a mapped portion that consists of virtual overlays for the monitor, shared run-time systems, and data areas such as data caching buffers and time-sharing terminal character buffers. The unmapped portion of the system is constrained to 40Kb. The mapped portion is only constrained by the physical memory installed on the system and the amount of memory that needs to be made available for time-sharing jobs.

The following table indicates the number of bytes of code and/or data space added to the mapped and unmapped portions of the system by various system features.

Effect of System Components on Overall System Size		
System Component	Bytes in Unmapped Region	Bytes in Mapped Region
Each additional time-sharing line	840	Terminal input character buffer + output character buffer
Each I/O device	Size of device handler	0
Device spooling	18 + (spool file size)/8 + (number spool files)*24 + ((number spool dev.)* (45+2*(num. backup blk.)))	2100 +(num. spool buffers)*512
Shared file record locking and data caching	10 + NUMDC*8	2100 + NUMDC*512 +MAXSF*14 +(num. time share jobs)*6 +MAXSFC*(12+2*MXLBLK)
Generalized data caching	1940	CACHE*528
Directory caching	MAXCSH*14	NMFCSH*18
Inter-job message communication	6	620 + MAXMC*20 + MAXMSG*(4+MSCHRS)
PLAS support	0	2300
Real-time support	490 + RTVECT*10	1010
Single line editor	0	3120
I/O mapping (18-bit device support on 22-bit Q-bus systems)	22 + MIONBF*16	960 + MIONBF*(MIOBSZ*512)
Performance analysis monitor	0	PMSIZE
Shared run-time	(num. of run times)*14	Size of run time system

E.2 Device Handler Sizes

The following table lists the size of the device handlers which are distributed with TSX-Plus.

Handler	Size (bytes)
CR	778
CT	2322
DD	1198
DL	1262
DM	1316
DP	346
DS	246
DT	256
DU	826
DX	594
DY	738
LP	318
LS	590
MM	4238 (file struct.)
MS	4716 (file struct.)
MT	3844 (file struct.)
NL	58
PC	210
PD	238
RF	232
RK	280
VM	306
XL	1170

.CTIMIO requests, 37
.PEEK and .POKE
  Operator privilege, 43
.TIMIO requests, 37
22-bit addressing, 13, 14
  Devices on LSI-11 bus, 36
  EXTMCH parameter, 7
ACCESS command, 41
  use with logical disks, 42
Account authorization, 45
  file conversion, 51
Activation characters
  Max number of - MXSPAC, 12
AUTCVT program, 51
BA handler, 36
Batch support, 36
Baud rate
  Specification for DZ11, 24
Buffers
  Data caching, 16
  Spooling, 15
  Terminal input, 20, 25
  Terminal output, 20, 25
BUFSIZ macro, 25
BUSTYP parameter, 6
CACHE parameter, 11
  optimizing, 80, 97
Caching, 78
  Data, 16, 79, 80
  Directories, 12, 79
CCL sav file, 1, 28
Character echoing
  DEFER flag, 23
Charge information, 49
CMDFIL macro, 26, 39
Command files
  Controlling listing of, 24
  Detached start-up, 26
  log-off, 40
  Start-up, 39
  Start-up for line, 26
Compute-bound time-slice
  QUAN2 parameter, 9
Connect time
  Determination of, 49
CORTIM parameter, 10
  optimizing, 75
CPU Modes display, 94
CPU time
  Determination of, 49
CRT terminal support
  SCOPE flag, 22
CTRL-S/CTRL-Q processing
  PAGE flag, 23
  TAPE flag, 22
Data caching, 79, 80
  affect on system size, 114
  CACHE parameter, 11
  NUMDC parameter, 16
DD Device handler patch, 107, 109
Deauthorizing an account, 48
Default memory allocation, 6
Default system editor, 12
DEFER flag, 23
Deferred character echoing
  DEFER flag, 23
DETACH macro, 26
Detached jobs
  Controlling use of, 23, 47
  Declaring number of, 20
  Start-up command files, 26
DEVDEF macro, 13
  VM handler, 32
  XL handler, 33
Device Driver SLP Files, 107
Device handlers
  and .CTIMIO requests, 37
  and .TIMIO requests, 37
  building, 33
  DD patch, 107
  DEVDEF macro, 13
  DM patch, 107
  DX patch, 107
  DY patch, 107
  Files included in distribution, 1
  FSM patch, 108
  Patch file checksums, 35
  required patches, 34
  restrictions, 36
  size table, 115
  Sysgen requirements, 31
  TJ patch, 108
  TM patch, 108
  TS patch, 108
  Unsupported, 36
  use of PAR 1, 36
  use of PAR 5 and 6, 36
  Use of queue element, 36
  VM, 31, 83
  XL, 32

Device Spooling
  See Spooling
Devices to be spooled, 15
DFLMEM parameter, 6
Dial-up lines
  PHONE flag, 24
  TIMOUT parameter, 12
DINSPC parameter, 20
Directory Cache Display, 95
Directory caching, 79
  affect on system size, 114
  MAXCSH parameter, 12
  NMFCSH parameter, 12
DL Device handler patch, 110
DL11 support, 22
  Interrupt vectors, 22
  Status registers, 22
DM Device handler patch, 107, 111
DMA parameter, 13
DOTSPC parameter, 20
DTSUB DIBOL callable subroutines, 1
DX device handler patch, 107, 111
DY device handler patch, 107, 111
DZ11 support, 21
Echo control
  DEFER flag, 23
  ECHO flag, 22
ECHO flag, 22
EDIT, 12
EDITOR parameter, 12
EL handler, 36
Error logging support
  Device handlers, 31
Error messages
  system, 103
  system startup, 99
ETX/ACK protocol, 25
Example
  Line definitions, 27
Execution states, 59
Extended memory mapping
  EXTMCH parameter, 7
  MEMSIZ parameter, 7
EXTMCH parameter, 7
File access control
  ACCESS command, 41
File access security, 39
File size
  Limiting, 11
Files
  Max number of shared files, 16
  RAD50 specification, 4, 19
FILTIM obtaining file creation time, 1
FLAGS macro, 22, 27, 43
FORM flag, 23
Form-feed control
  FORM flag, 23
FSM device handler patch, 108, 111
FTSUB FORTRAN callable subroutines, 1
Generating a system, 3
Handlers. See Device handlers.
High-priority execution quantum, 8
HIMEM parameter, 5
HIPRCT parameter, 10
  optimizing, 76, 77
Hold mode of spooling, 15
Holding spool files, 15
I/O channels
  Max open to shared files, 16
I/O completion quantum, 9
I/O mapping, 13, 14
  affect on system size, 114
I/O optimization, 76
  Data caching, 78
  Device spooling, 78
  Execution overlap, 76
I/O queue elements, 36
I/O rundown
  IOABT parameter, 7
In-line interrupt service routines, 36
INIABT parameter, 7
Initialization control
  INIABT parameter, 7
Input buffer size
  Default, 20
Interactive job scheduling, 60
  INTIOC parameter, 10
  QUAN1 parameter, 9
  QUAN1B parameter, 9
  QUAN1C parameter, 9
Interprogram communication, 17
  affect on system size, 114
  MAXMC parameter, 17
  MAXMSG parameter, 17
  MSCHRS parameter, 17

Interrupt processing, 64
Interrupt vectors
  Real-time support, 18
INTIOC parameter, 10
  optimizing, 72, 73
IOABT parameter, 7
  needed for VTCOM, 33
ISAM files
  Optimizing with data caching, 16
Job Execution Status Display, 89
Job scheduling
  HIPRCT parameter, 10
Job swapping, 64
K52, 12
KED, 12
LC flag, 23
LD handler, 36
LDSYS parameter, 8
Lead-in character
  TSLICH parameter, 12
LINDEF macro, 21, 22, 27
Line Definition Block (LDB), 21
LINEND macro, 21, 27
Linking TSX-Plus, 28
LINPRM macro, 24, 27
Listing account status, 48
Locking a program to a line, 40
Locking records
  See Shared files, 16
Log-off command files, 40
Logical disks
  ACCESS command, 42
  Enabling use of, 8
LOGON facility, 44
LOGON sav file, 1
Low priority job time-slice
  QUAN3 parameter, 9
Lower-case character control
  LC flag, 23
MAPIO parameter, 13
MAPIOP parameter, 13
Max memory a job can use, 5
MAXCSH parameter, 12
MAXFIL parameter, 11
MAXMC parameter, 17
MAXMSG parameter, 17
MAXSEC parameter, 11
MAXSF parameter, 16
MAXSFC parameter, 16
Memory allocation
  DFLMEM parameter, 6
  HIMEM parameter, 5
  MEMSIZ parameter, 7
Memory management support
  and device handlers, 31
Memory residency control
  CORTIM parameter, 10
MEMSIZ parameter, 7
  and VM handler, 31
Message communication
  See Interprogram communication
MIOBSZ parameter, 14
MIONBF parameter, 14
MOUNT command
  use with ACCESS command, 42
MSCHRS parameter, 17
Multiplexor support, 21
  LINPRM macro, 24
  MUXDEF macro, 21
MUXDEF macro, 21
MXLBLK parameter, 16
MXSPAC parameter, 12
New authorization file format, 51
NMFCSH parameter, 12
NODET flag, 23
Non-swapping system generation, 6
NONDMA parameter, 13
NOVLN flag, 23
NRMFLG parameter, 24
NUMDC parameter, 16
  optimizing, 79, 96
Object modules in distribution, 2
Operator privilege, 43
  .PEEK and .POKE, 43
  Access to SYS and TSX files, 43
  Granting of, 43
  Granting to an account, 47
  PRIV flag, 24
  Real-time facilities, 43
  Restricted commands, 43
  SYSMON program, 85
  TSAUTH program, 43
  Use of TSAUTH program, 43
Operator's console
  Specification of, 22

Optimizing system parameters, 69
Organization of the system, 53
OTRASZ parameter, 20
Output buffer size
  DOTSPC parameter, 20
Output reactivation count, 20
Overview of the system, 53
PAGE flag, 23
Paper-tape mode
  TAPE flag, 22
PAR 1 use by device handlers, 36
PAR 5 use by device handlers, 36
PAR 6 use by device handlers, 36
Parity control for DZ11, 25
Password specification, 47
Patch file checksums, 35
PD handler, 36
Performance monitor, 19, 75
  affect on system size, 114
  PMSIZE parameter, 19
  TSXPM program, 1
PHONE flag, 24
PLAS
  affect on system size, 114
  Region swap file name, 5
  Specifying file size, 6
PMSIZE parameter, 19
PPN specification, 46
PRIDEF parameter, 10, 59
PRIHI parameter, 10
  job scheduling, 58
PRILOW parameter, 10
  job scheduling, 58
Priority
  and job scheduling, 58
  controlling maximum allowed, 42, 47
  Default value, 10
  Fixed high priorities, 10
  Fixed low priorities, 10
  Virtual job reduction, 11
PRIV flag, 24, 43
Privilege
  Operator, 43
PRIVIR parameter, 11, 59
Project-Programmer specification, 46
QBUS processors
  BUSTYP parameter, 6
QTSET flag, 24
QUAN0 parameter, 8
  optimizing, 74
QUAN1 parameter, 9
  optimizing, 72, 73
QUAN1A parameter, 9
  optimizing, 76, 78
QUAN1B parameter, 9
  optimizing, 72, 74
QUAN1C parameter, 9
  optimizing, 72, 74
QUAN2 parameter, 9
  optimizing, 72, 74
QUAN3 parameter, 9
  optimizing, 74
Queue elements, 36
Queued Message Display, 92
R command
  /LOCK switch, 40
RAD50 file specification, 4, 19
Reactivation count
  For TT output, 20
Real-time support, 18
  affect on system size, 114
  Interrupt completion routines, 65
  Interrupt processing, 64
  Interrupt service routines, 64
  Operator privilege, 43
Record locking
  See Shared files
Resetting account statistics, 50
Resident run-time systems
  See Shared run-time systems
Restricted keyboard commands, 43
RSFBLK parameter, 5
RT-11 version 4 device handlers
  DD patch, 109
  DL patch, 110
  DM patch, 111
  DX patch, 111
  DY patch, 111
  FSM patch, 111
  TJ patch, 111
  TM patch, 112
  TS patch, 112
RTDEF macro, 19
RTVECT parameter, 18
Run command
  /LOCK switch, 40

SCOPE flag, 22
Security of file access
  See system security, 39
SEGBLK parameter, 6
SET CACHE command, 80
SET LOGOFF command, 40
SET MAXPRIORITY command, 42
SET SIGNAL command, 71
SETSIZ command file, 1, 37
SETSIZ program, 1, 37, 71
Shared files, 16
  affect on system size, 114
  Data caching, 16
  MAXSF parameter, 16
  MAXSFC parameter, 16
  MXLBLK parameter, 16
  NUMDC parameter, 16
Shared run-time systems, 19
  affect on system size, 114
  RTDEF macro, 19
Single Line Editor
  affect on system size, 114
  Enabling use of, 8
Size of spool file, 15
Size of system, 113
SL
  see Single Line Editor
SLEDIT parameter, 8
SLP files
  Contents of, 107
  Files included in distribution, 2
  Use of, 35
SLP files for RT-11 version 4
  Contents of, 109
SPLBLK parameter, 5
Spool file name
  Specifying, 5
SPOOL macro, 15
Spooling, 15
  affect on system size, 114
  Back-up blocks, 16
  Devices, 15
  Hold mode, 15
  Number of buffers, 15
  Number of devices spooled, 15
  Number of spooled files, 15
  Spool file size, 15
START flag, 23
Start-up command file
  CMDFIL macro, 26
Start-up command files, 39
  Associating with user, 47
  Controlling listing of, 40, 44
  Interaction, 44, 47
  Use with LOGON program, 44
Start-up of lines, 23
Starting TSX-Plus, 28
Stop bits for DZ11, 25
Swap file name
  Specifying, 5
SWAPFL parameter, 6
Swapping of jobs, 64
SWDBLK parameter, 5
SYSMON Dynamic Display Utility, 85
  CPU Modes display, 94
  Creating and running, 85
  Creating SAV file, 28
  Data Cache Display, 97
  Directory Cache Display, 95
  Enabling use, 18
  Job Execution Status Display, 89
  Message Queue Display, 92
  Operator privilege, 85
  Shared File Data Cache Display, 96
  System Status Display, 87
  Terminal Status Display, 91
  User times display, 93
SYSODT rel file, 1
System generation, 3
System I/O mapping
  see I/O mapping
System overview, 53
System security, 39
  Start-up command files, 39
System size calculation, 113
System Status Display, 87
System tuning, 69
Tab character handling
  TAB flag, 23
TAB flag, 23
TAPE flag, 22
TBLDEF macro, 20
TECO, 12
Terminal Status Display, 91

Terminal type
  Diablo, 25
  Specification of, 25
Time-out support
  Device handlers, 31
Time-sharing lines, 20
  BUFSIZ macro, 25
  CMDFIL macro, 26
  DEFER flag, 23
  ECHO flag, 22
  Example of, 27
  FLAGS macro, 22
  FORM flag, 23
  LC flag, 23
  LINDEF macro, 21
  Line definition block, 21
  LINPRM macro, 24
  NODET flag, 23
  NOVLN flag, 23
  Number of, 20
  NUMFLG parameter, 24
  PAGE flag, 23
  PHONE flag, 24
  PRIV flag, 24
  QTSET flag, 24
  SCOPE flag, 22
  START flag, 23
  TAB flag, 23
  TAPE flag, 22
  TBLDEF macro, 20
  Terminal type, 25
  Total number supported, 20
times display, 93
TIMOUT parameter, 12
TJ device handler patch, 108, 111
TM device handler patch, 108, 112
TRANSF program, 32
TRMTYP parameter, 25
TS device handler patch, 108, 112
TSAUTH program, 45
  Authorize command, 46
  Charge command, 49
  Command summary, 46
  Exit command, 50
  Kill command, 48
  List command, 48
  Operator privilege, 43
  Reset command, 50
  Usage command, 49
TSAUTH sav file, 1
TSGEN module
  Assembling, 4, 28
  Setting parameters in, 5
TSGEN source file, 1
TSLICH parameter, 12
TSLNK3 command file, 1
TSODT object file, 1
TSODT relocatable file, 1
TSXDB sav file, 1
TSXLNK and TSLNK3 command files, 28
TSXLNK command file, 1
TSXPM sav file, 1
TSXUCL program, 29
  data file size, 8
TSXUCL sav file, 1
TSXUCL.TSX, 8
TT buffer sizes
  BUFSIZ macro, 25
  DINSPC parameter, 20
  DOTSPC parameter, 20
TT output reactivation
  OTRASZ parameter, 20
Tuning the system, 69
Twenty-two bit addressing
  EXTMCH parameter, 7
UCL parameter, 8
UCLMNC parameter, 8
UCLORD parameter, 8
UNIBUS processors
  BUSTYP parameter, 6
Unsupported device handlers, 36
Usage information, 49
Usage statistics, 49
User Command Linkage
  TSXUCL program, 29
  UCL parameter, 8
  UCLMNC parameter, 8
  UCLORD parameter, 8
User defined commands
  see User Command Linkage
User names
  Duplicate, 46
  Specification in TSAUTH, 46
User-defined commands
  Maximum number, 8
  Processing order, 8
Virtual lines
  Controlling use of, 23, 47

Declaring number of, 20
Max number of - MAXSEC, 11
Priority reduction, 11
Virtual-line signal character, 12
VLSWCH parameter, 12
VM handler, 83
installing, 31
VTCOM program, 32
Wild cards
Explicit/Implicit, 12
WILDFL parameter, 12
X-ON/X-OFF processing
PAGE flag, 23
TAPE flag, 22
XL handler, 32
installing, 33