TSX–Plus
Installation Guide

Sixth Edition—First Printing—January 1988

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

Contents

Chapter 1

Introduction

The purpose of this guide is to help you install TSX–Plus on your system in the quickest and easiest manner possible. This guide describes the contents of a typical distribution kit, necessary system resources, the circumstances that require a TSX–Plus system generation, and information about how to perform a full system generation to customize TSX–Plus to your needs.

This guide is intended for the person installing TSX–Plus and assumes some familiarity with the commands and utilities of RT–11 including: handling of magnetic media, use of command files, the COPY and PRINT commands, use of one of the system editors (KED, EDIT, or TECO), assembling and linking a pre-written MACRO program, and the R[UN] command. It is not necessary to be able to write MACRO programs, but you should be familiar with the assembly and linking process. If you elect to perform a TSX–Plus system generation, it is also useful to be familiar with the usage of comments and parameters in MACRO programs.

It is necessary to have access to configuration information about the various peripheral devices attached to your computer, especially to know which device handlers are needed and which vectors and addresses are used by these devices and by time-sharing terminals. This configuration information should have been written down and left with the system by the person who installed the hardware. If it is not available, ask the person who installed the hardware to enter the configuration information for your system in the form in Appendix B of this manual.

The TSX–Plus operating system is designed to provide the familiar RT–11 operating environment to multiple users. This goal necessitates several extensions to the command language of RT–11, and makes a few features unavailable. The *TSX–Plus User's Reference Manual* and the *TSX–Plus Programmer's Reference Manual* describe all of the functional differences between RT–11 and TSX–Plus, including additional keyboard commands, unsupported commands, and other minor variations between the two operating systems.

The TSX–Plus manuals are intended to be used in conjunction with the RT–11 manuals; refer to the RT–11 manuals for descriptions of features which are not discussed in the *TSX–Plus User's Reference Manual* or the *TSX–Plus Programmer's Reference Manual*.

Chapter 2

Getting Started

2.1 Distribution Kit

2.1.1 Documentation

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

- *TSX–Plus Installation Guide.*
- *TSX–Plus System Manager's Guide*, which provides information needed by the system administrator.
- *TSX–Plus User's Reference Manual*, which describes the features of TSX–Plus.
- *TSX–Plus Programmer's Reference Manual*, which describes the programming features of TSX–Plus.
- *TSX–Plus Release Notes.*

Version updates will only include the Release Notes. Orders for new licenses will include all of the manuals.

2.1.2 Software

The TSX–Plus distribution package should also contain magnetic media (reversible RX01 diskettes, RX50 diskettes, RL02 cartridge disk pack, 1600 bpi magnetic tape, or TK50 tape cartridge). The media should contain the following files:

1. 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, MU, NL, PC, RF, RK, VM, XL. All of these files have the extension ".TSX". Note that the VM provided is not the DEC VM handler.
2. Source language patch files are provided for the rare situations in which it is necessary to rebuild the distributed device handlers. All of these files have the extension ".SLP".
3. The following object modules, together with your edited and assembled TSGEN, are used by the command file TSXLNK to build the executable program TSX: TSCASH, TSCLO, TSDBUG, TSDUMP, TSEM2, TSEM3, TSEM4, TSEM5, TSEXC2, TSLOCK, TSMIO, TSMSG, TSPLAS, TSRTX, TSSLE, TSSPOL, TSSWAP, TSTIOX, TSTTY, TSTTY2, TSUSR, TSWIN, TSX1. The object modules named TSX2, TSKM2A, TSKM2B, TSKM2C, TSKST1, TSKST2 and TSKSHO are used to build the TSKMON executable program. TSXMOD, TSXMOA, TSXMOB and TSXMOT are used to build the TSXMOD utility. SYSMON is used to build the SYSMON utility. All of these files have the extension ".OBJ".

4. The following files are also included in the distribution:

CCL.SAV	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	TSX-Plus logon program
SETSIZ.COM	Command file to set memory size for system programs
SETSIZ.SAV	Program to store memory size info into SAV files
SYSODT.REL	Program used by system developers to debug TSX-Plus
TSAUTH.SAV	TSX-Plus account management program
TSGEN.MAC	TSX-Plus parameter module, macro source file
TSXDB.SAV	Program used by system programmers 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
WINPRT.TSX	Command file to start printwindow program
WINPRT.SAV	Printwindow program

2.2 Copying files from the TSX–Plus distribution

2.2.1 From diskettes

To copy your TSX–Plus distribution from RX01 or RX50 diskettes, type:

```
.COPY ddn:*.*  dvn:*.*
```

where *ddn:* is the name and number of the diskette drive (DX1:, DY1:, DU1:) and *dvn:* is the name of the device where you will be generating your system (DL1:, DU2:, LD3:).

You must repeat this for all four surfaces of your diskette distribution. If you received your distribution on RX01s you will need to copy both sides of the diskettes.

2.2.2 From an RL02

To copy your TSX–Plus distribution from RL02 disks, type:

```
.COPY DLn:*.* dvn:*.*
```

where DL*n* is the DL device number and *dvn:* is the name of the device on which you will be generating your TSX–Plus system (DL1:, DU2:, LD3:).

2.2.3 From Reel to Reel Mag Tape

To copy your TSX–Plus distribution from magnetic tape, type:

```
.COPY tdn:*.*/POS:-1 dvn:*.*
```

where *tdn:* is the name of the tape device (MT:, MS:, MM:) and *dvn:* is the name of the device where your system will be generated (DL1:, DU2:, LD3:).

2.2.4 From TK50 Mag Tapes

Distributions on TK50 tape cartridges are in RT-11 BUP format.

In order to retrieve a TSX-Plus distribution, you must first create a logical disk of 2600 blocks, and initialize it. This may be done as follows:

Replace "DU1" with the actual name of the device to which you wish to restore the file. Note that wildcards are not permitted in BUP command lines.

```
.CREATE DU1:TSX.DSK/ALLOCATE:2600
.MOUNT LD2 DU1:TSX62.DSK
.INITIALIZE LD2:
```

Once this is done, you need to use the RT-11 "Backup Utility Program" (BUP) to retrieve your distribution. This may be done as follows:

```
.R BUP
*LD2:=MU:/I/X
Mount input volume 1 in MU0; Continue? Y
?BUP-I-Restore operation started from volume 1
?BUP-I-Copy operation is complete
*^C
```

NOTE: Do not attempt to restore your TK50 distribution to a device with any files on it. To do so will cause a loss of the current directory on the device.

If you wish to restore a single file from your distribution, the procedure is similar. For example, to recover the distributed TSGEN.MAC file:

```
.R BUP
*DU1:TSGEN.MAC=MU:TSGEN.MAC/I/X/F
Mount input volume 1 in MU0; Continue? Y
?BUP-I-Restore operation started from volume 1
?BUP-I-Copy operation is complete
*^C
```

File restoration does not cause the loss of the disk's current directory.

2.3 System Prerequisites

2.3.1 Hardware

TSX-Plus is implemented on the PDP-11 series of computers manufactured by Digital Equipment Corporation. TSX-Plus uses the memory management hardware available on most DEC computers. Specifically, TSX-Plus may be used on the following computers: LSI-11/23, PDP-11/23-Plus, LSI-11/73, PDP-11/83, MICRO/PDP-11, PDP-11/24, PDP-11/34A, PDP-11/44, PDP-11/60 and PDP-11/84. (TSX-Plus may not be used on LSI-11/2, PDP-11/03 or VAX computers.) Some older machines which support memory management may also use TSX-Plus. All of these computers must support memory up to at least 256 Kb. TSX-Plus requires at least 192 Kb of memory, with a minimum of 256 Kb of memory recommended.

On the LSI-11 bus, the PDP-11/23-Plus, LSI-11/73, and MICRO/PDP-11 support more than 256 Kb of memory whereas the LSI-11/23 may support more than 256 Kb of memory only if the backplane is configured for 22-bit addressing. The only Digital devices currently supported with 22-bit DMA addressing on the LSI-11 bus are those using the DL handler (RL01 and RL02), the DU handler (RA80, RC25, RD51, RD51, RD53, RX50, and RX33), the MS handler (TSV05), and the MU handler (TK50). All other DMA

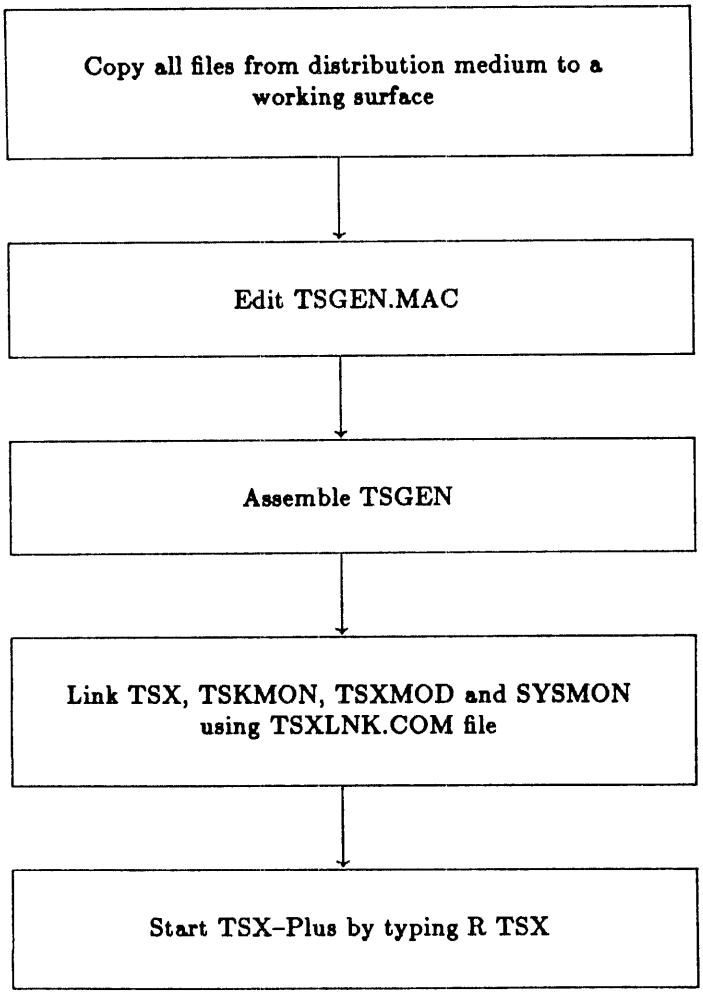
device handlers only support 18-bit addressing on the LSI-11 bus and therefore must use the TSX-Plus I/O mapping facility if the system has more than 256 Kb of installed memory. We do not recommend that devices which use the DM device handler use system I/O mapping. On the UNIBUS, the PDP-11/24, PDP-11/44 and PDP-11/84 support more than 256 Kb of memory with the necessary hardware.

2.3.2 Software

Although TSX-Plus completely replaces the RT-11 kernel during operation of TSX-Plus, it must be started from the RT-11 Single-Job or Baseline monitor. The utilities and device handlers necessary to run TSX-Plus are provided as a part of RT-11, and for this reason each TSX-Plus site must also be properly licensed to run RT-11.

TSX-Plus supports many of the keyboard commands of RT-11 Version 5.4, but it may be run with any RT-11 Version 5. Attempts to use features which require RT-11 Version 5.4 utilities will generate error messages when used with previous RT-11 version 5 utilities. The VM (virtual memory) pseudo-disk may *not* be used in conjunction with RT-11 Version 4. Both the Logical Subset Disk and Single Line Editor features are separately implemented in the TSX-Plus kernel; they do not use the LD or SL handler and are available whether TSX-Plus is used with either RT-11 Version 4 or 5.

2.4 System Generation Overview



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 User's Reference Manual*. There are a number of features provided by TSX–Plus that are not available in RT–11 (deferred character echoing, subprocesses, process windowing, 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 all communication equipment and device controllers that will be used by TSX–Plus. Once you have done this you can proceed with the TSX–Plus system generation as described in this chapter.

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

1. Copying all of the files from the TSX–Plus distribution media to a work disk/surface.
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, TSX-MOD.SAV and SYSMON.SAV.

Chapter 3

Editing the TSGEN module

The TSGEN module of TSX-Plus is supplied in MACRO 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, 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:

- Numeric values are assumed to be octal unless the number is terminated with a decimal point.
- When using editors that recognize them, the TSGEN module is divided into several pages by formfeed characters.
- On some parameters 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 or CMDFIL macros.

Setting parameters in TSGEN

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 may easily be located by searching for a string of three equal signs (===).

3.1 General parameters

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 line may use. The virtual memory handler (VM) may not be used to contain the swap file due to the nature of completion routine processing. Unpredictable results (which may include system halts and fatal system errors) will occur if VM is used as the swap device.

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. The virtual memory handler (VM) may not be used to contain the spool file due to the nature of completion routine processing. Unpredictable results (which may include system halts and fatal system errors) will occur if VM is used as the spool device. 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).

UCLDAT This is the name of the file used to store user defined commands for all jobs. The default name is SY:TSXUCL.TSX. This file is only used (and allocated) if the U$CL parameter is non-zero. The amount of disk space required for this file is equal to: *file size(in blocks) = (total number of processes** UCLMNC*94)/512.

INDFIL This is the name of the file used to store information for IND during the execution of IND command files. The default name is SY:TSXIND.TSX.

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 kilobytes. 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* (in blocks)=(*total lines*)*(HIMEM+6)*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 kilobytes. 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.

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 zero (0) 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=0 the following actions occur:

1. No disk swap file is created (see SWDBLK above).
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 (see RFSBLK above).

SWPSLT This parameter controls the number of job slots allocated in the swap file when job swapping is enabled (SWAPFL=1). SWPSLT should be in the range zero to the total number of jobs. If SWPSLT is set to zero, TSX-Plus will automatically allocate one job slot in the swap file for each job. SWPSLT may be set to a value less than the total number of jobs if a small amount of job swapping is anticipated. However, a system crash will occur if the system needs to swap a job out of memory and no free slot is available in the swap file. This may occur when the system is moving jobs to provide a contiguous memory extent for a job or a PLAS region, even though there is sufficient memory available to hold all jobs. The SWPSLT parameter has no effect on non-swapping systems (SWAPFL=0). The recommended setting is SWPSLT=0.

SEGBLK This parameter specifies the number of 512-byte blocks to allocate for the swap file that is used for extended memory PLAS (Program 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 and no region swap file is allocated or used. 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 zero (0) disables use of PLAS.

NGR This parameter specifies the number of shared global PLAS regions that can be created by all jobs.

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, 11/53, 11/73 and 11/83, and UNIBUS for 11/24, 11/34a, 11/44, 11/60 and 11/84.

MEMSIZ This parameter controls the maximum memory available for TSX–Plus system use. The value is the upper memory limit size specification expressed in number of kilobytes. Memory above this upper limit will not be used by the operating system. If the MEMSIZ parameter is set to zero (0), 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 be 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:

- A device that was specified in TSGEN does not have a TSX handler on the system disk or is not supported by the version of RT–11 being used.
- A time sharing line that was generated into TSX–Plus is not installed on the machine.
- A shared run-time system file could not be found during startup.
- A device declared in the SPOOL macro could not be opened.

UXIFLG This parameter controls the action taken by TSX–Plus when a device interrupt occurs through a vector which is not initialized for use with TSX–Plus. This can occur when the interrupt vector address specified in a device handler does not match the actual interrupt vector selected by the hardware for the device. It can also occur if a real-time interrupt occurs and no connection is established between the interrupt and a real-time program. If this parameter is set to zero (0), then TSX–Plus will ignore unexpected interrupts. If this parameter is set to one (1) and an unexpected interrupt occurs, then TSX–Plus will halt with the fatal error message: "?TSX–F–UEI–Interrupt occurred at unexpected location." The recommended setting for UXIFLG is one (1).

SYSDMP This parameter controls whether the system crash dump facility is included in the system. If SYSDMP is set to one (1) then the system will print some useful internal system data if a system crash occurs. The dump information will be printed to the device whose address is specified by the parameter DMPTCR below. It is recommended that this facility not be included in the system unless you are experiencing system crashes.

DMPTCR This parameter specifies the address of the transmitter control register for the device to which the crash dump will be written. This must be a DL–11 type device controller or a parallel printer controller. Specify 177564 to dump on the console terminal. Specify 177514 to dump to a line printer connected to standard parallel port.

DMPKTP This parameter specifies whether you want a system crash to occur any time a trap occurs within the system. Set it to zero (0) if you want recoverable traps within the system to abort the job but continue execution of the system. The recommended setting is one (1).

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 one (1). The IOABT parameter must be set to one (1) if the XL or CL handlers are used with VTCOM or if the system includes a magtape device. See the *TSX–Plus User's Reference Manual* for more information on the SET IO [NO]ABORT command, which may be used to dynamically alter this parameter during system operation.

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* (in 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 assigned 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 User's 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 or when only physical disks are used.

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 one (1) if the single line editor facility is to be made available to any lines.

KEYMAX This parameter specifies the maximum number of user-defined key definitions supported by the single line editor that may be in effect at one time for each user. The maximum supported value for KEYMAX is 60. See the DEFINE/KEY command in the *TSX–Plus User's Reference Manual* for more information.

MAXWIN This parameter specifies the maximum number of terminal display windows that may be in use by all jobs on the system. The SEGBLK parameter must also be set to a non-zero value for this feature to be included If MAXWIN is set to zero (0), the display window feature is not included in the system. See the SET WINDOW command and the chapter on subprocesses in the *TSX–Plus User's Reference Manual*.

DBGFLG This parameter controls whether support for the program debugging facility is included in the system. If DBGFLG=1, the debugging facility is included in the generated system; if DBGFLG=0, the debugging facility is not included in the system. See the *TSX–Plus Programmer's Reference Manual* for information about the use of the program debugging facility.

NUIP This parameter specifies the number of slots in the INSTALL table to be reserved for user programs. See the *TSX–Plus System Manager's Guide* for more information on installed programs.

QUAN0 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 QUAN0 as the time-slice value. If QUAN0 is set to zero

(0), high-priority jobs are not time-sliced. Specify the value of QUAN0 in 0.1 second units. The SET QUAN0 keyboard command may be used to dynamically alter this parameter during system operation. See the *TSX-Plus User's Reference Manual* for more information about the SET QUAN0 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 User's 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 User's 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 User's 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 User's 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 User's 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 User's 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. See the *TSX-Plus User's Reference Manual* for more information on the SET INTIOC command, which may be used to dynamically alter this parameter during system operation.

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. See the *TSX-Plus User's Reference Manual* for more information on the SET HIPRCT command, which may be used to dynamically alter this parameter during system operation.

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. See the *TSX-Plus User's Reference Manual* for more information on the SET CORTIM command, which may be used to dynamically alter this parameter during system operation.

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 subprocess, the job execution priority of the disconnected process 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 the *TSX-Plus System Manager's Guide* for more information on priority and job scheduling.

MAXSEC This parameter is used to specify the maximum number of subprocesses 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 zero blocks. This parameter does not limit the space that will be allocated to .ENTER requests that specify a size. It only affects .ENTER requests that specify a file size of zero. If a value of zero (0) is specified for MAXFIL, no limit is placed on the size of a file created with a specified size of zero.

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 256 Kb of memory. If generalized data caching is not desired, set CACHE=0. If generalized data caching is desired, set CACHE to the number of 512 byte blocks to be allocated for the cache. This parameter must be less than 4096 due to the implementation of the cache control tables and the PDP-11 architectural design of 8 Kb per mapping register.

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 256 Kb 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 fifty.

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. See the section of system tuning in the *TSX-Plus System Manager's Guide* for more information.

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 should 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).

MAXALC This parameter specifies the maximum number of device units that can be allocated simultaneously by all users for exclusive use through the ALLOCATE command.

MAXMON This parameter controls the maximum number of job monitoring requests which can be simultaneously active system-wide. The TSX–Plus job monitoring facility allows one job to schedule a completion routine to be entered with a status code whenever the status of a job being monitored changes. See the *TSX–Plus Programmer's Reference Manual* for further information about the job monitoring facility. If the job monitoring facility is not desired, set MAXMON to zero (0).

SYSPS All lines which require system password checking, specified by the $SYSPS flag to the FLAGS macro (see below), will be required to enter this password before the standard TSX–Plus greeting message is displayed and the normal line initiation sequence proceeds. The form of this macro is:

```
SYSPS      <password>    <79X>
```

where *password* is a string of up to 20 characters which may contain spaces.

TIMOUT This parameter is only used for lines connected to dial-up telephone equipment. It specifies the length of time that the carrier signal may be lost before the system assumes the connection has been broken and logs off the job. Specify in 0.5 second units.

Setting this parameter to a reasonable value (e.g., one minute) provides time for a disconnected user to reestablish the connection by recalling the computer. You should not make this parameter too large because it provides a window during which another user could dial into the computer and connect to a line that is still logged on.

See also the SET TIMOUT command in the *TSX–Plus User's Reference Manual.*

OFFTIM This parameter is only used for lines connected to dial-up telephone equipment. It is the length of time that a user may remain connected to a dial-up line before logging in. It also controls the length of time that a user may remain connected to a dial-up line between logging off and logging back on. If the user does not log in within this time interval, the Data Terminal Ready (DTR) signal will be dropped causing the phone to be hung up. Specify this parameter in 0.5 second units.

See also the SET OFFTIM command in the *TSX–Plus User's Reference Manual.*

PHONE This parameter is used to determine how $PHONE lines are treated. Modem lines ($PHONE in the LINDEF FLAGS macro) are normally treated as phone lines if the DCD signal (carrier) is present when the lines are started and optionally treated as local lines if the signal is not present. The OFFTIM and TIMOUT parameters are only effective if the line is recognized as a phone line when started. Set PHONE to 0 to allow lines with the $PHONE flag to optionally support local lines. Set PHONE to 1 to force the OFFTIM and TIMOUT parameters to always take effect for lines with the $PHONE flag.

TSLICH This is the *lead-in* character that tells TSX–Plus that the next character sent to the terminal by the program is to be interpreted by TSX–Plus as a special command (e.g., defining a new activation character). See the *TSX–Plus Programmer's Reference Manual* for more information on program controlled terminal options. The default value for this parameter is 35 (29 decimal, CTRL–]).

VLSWCH This is the character used to signal a request to switch terminal control to a (sub)process. The default value is 27 (23 decimal, CTRL–W). See the *TSX–Plus User's Reference Manual* for a discussion of the use of subprocesses.

PWCH This parameter specifies the octal value of the character that is used to request that the current screen window be printed. The default value is 2, CTRL–B. See also the SET PRINTWINDOW command.

CCXTRM This parameter specifies the octal value of the character that is used to terminate a cross-connection between a time-sharing line and a CL line. The default value is 34 (28 decimal, CTRL–\).

CCXCTL This parameter specifies the octal value of the character that is used to signal a special control function during a cross-connection between a time-sharing line and a CL line. The default value is 1, CTRL–A.

CLVRSN This parameter allows you to force CL to report a particular version number. If CLVRSN is set to zero, then a version number appropriate for your version of RT–11 will be automatically assigned. Forcing CL to report a different version number is only necessary when the version automatically selected is incorrect. This version number is normally only checked by VTCOM. The recommended value is 0.

MXSPAC TSX–Plus allows running programs to dynamically define activation characters (an activation character is one 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 zero (0) for explicit wildcards or one (1) for implicit wildcards. See the *RT–11 System User's Guide* for further discussion of explicit and implicit wildcards.

3.2 Device definitions

The next section of TSGEN is used to specify all of the devices which are to be available while TSX–Plus is running. Unlike RT–11, TSX–Plus does not allow fetchable, non-resident device handlers. All TSX–Plus device handlers are loaded into memory when the system is started and remain in memory until the system is stopped. Some TSX–Plus device handlers must be loaded into the 40 Kb low-memory system area, but other handlers can be loaded into extended memory regions.

Because of the characteristics of file-structured magtape handlers, they cannot be loaded into extended memory regions on some hardware configurations. Because of their large size, file-structured magtape handlers impose severe constraints on the number of other features which may be included if they are loaded into the low 40 Kb region. To reduce the impact of the magtape handlers' sizes, hardware-only handlers are also included on the distribution. These handlers are much smaller and have less impact on maximum system size. The limitation of a hardware-only magtape handler is that tapes may no longer be directly accessed with standard system utilities which expect a special file structure (e.g. PIP and DIR). If a hardware-only magtape handler is used, it must be accessed with a utility which does not require special file structuring — such as BUP or several third-party tape utilities. If you elect to use a hardware-only magtape handler, you must copy the appropriate hardware-only handler from the distribution media to the system disk. The following table specifies the appropriate copy/rename operation:

Hardware handler	Handler name	Device protocol
MMHRD.TSX	MM.TSX	TJU16, TU45
MSHRD.TSX	MS.TSX	TS11, TSV05
MTHRD.TSX	MT.TSX	TM11, TS03
MUHRD.TSX	MU.TSX	TK50

Device handlers are loaded into memory during TSX-Plus initialization. In order to be loaded, a handler must be specified by a DEVDEF macro. The corresponding handler need not be installed in RT-11, except for the handlers controlling the devices on which the TSX-Plus swap and spool files are located, since these files are accessed or created while RT-11 is still in control. Before a device handler is loaded, the device CSR address is tested—if it does not respond, the handler is not loaded. If the device handler contains installation code that is executed and must succeed, the handler is not loaded if the installation code fails. See the description of the INIABT parameter for more information on the action taken during TSX-Plus initialization in regards to device handlers.

Three macros are used to specify devices to TSX-Plus. The DEVBEG macro begins the device specification section and the DEVEND macro ends the device specification section. The DEVDEF macro is used to specify the name and options for each device. Thus, the DEVDEF macro is used once for each device between the DEVBEG and DEVEND macros.

The devices CL, LD, TT, and SL are an integral part of the TSX-Plus system and should not be specified with the DEVDEF macro. It is recommended that the CL handler be used to drive all serial devices instead of the LS and XL handlers. See the *TSX-Plus Programmer's Reference Manual* for a description of the CL handler.

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>[,option,...,option]
```

where *dd* defines the 2 character device name enclosed in angle brackets ("<" and ">").

The *optional* parameters specify the device characteristics. There are nine allowable device attributes which may be specified in any order. They are as follows:

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

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

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

HANBUF Handler contains an internal I/O buffer used for DMA transfers. Handlers with internal DMA buffers require special coding to be used as a mapped device handler. In addition, when TSX-Plus is evaluating the system definitions and device characteristics for loading device

handlers, it will never map a handler which uses an internal buffer if the handler also requires mapped I/O transfers in Q-Bus systems with more than 256 Kb of memory (MAPIO option), or in UNIBUS systems with more than 256 Kb of memory.

MAPH Load the device handler outside the low memory 40 Kb region and into a mapped handler region. TSX-Plus can place some device handlers within an extended memory region, reducing the size of the low memory kernel region (restricted to 40 Kb). Handlers which are placed in extended memory are known as *mapped* handlers. TSX-Plus communicates with mapped device handlers by mapping PAR 5 to the handler's extended memory base address. As device handlers are loaded, the interrupt entry point is intercepted and directed to a low memory routine which will map to the handler, then enter the handler's interrupt entry code.

Handlers may be mapped under the following conditions:

1. Since only one PAR register is used to access the device handler it must not be larger than 8 Kb.
2. Since handlers are accessed by kernel PAR 5, the handler must not use kernel PAR 5.
3. Since only two device interrupt vectors per handler are redirected, the handler may not connect to more than two device interrupt vectors. In addition, since the redirection is performed only once, during initialization, the handler may not dynamically connect to interrupt vectors.
4. When the device handler contains an internal buffer used for DMA access, it must calculate the correct physical address taking into account it's own mapped address. It must also declare the HANBUF option (see above) which will not allow it to be mapped on UNIBUS systems with more than 256 Kb of memory or on Q-Bus systems when MAPIO is also specified.

See the *TSX-Plus System Manager's Guide* for further information about mapped handlers.

NOMAPH Do not load the handler into a mapped handler region; instead load it into the low memory 40 kilobyte region. Some device handlers are not eligible for mapping into extended memory regions and TSX-Plus will place them in the low memory kernel region. The NOMAPH option can be used to specify that a handler which would ordinarily be mapped should not be mapped. This option takes precedence over the MAPH option.

MAPIO Perform I/O mapping. This parameter should be specified only for Q-Bus DMA devices which 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 not be specified for any device in an 18-bit environment Older LSI-11/23 systems can 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.

NOMOUNT Do not allow mounts for this device.

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

NOSET Do not apply SET option to the copy of handler in memory. When SETs are issued to device handlers, the SET alters the disk image which normally is then reloaded to replace the memory copy. For example, SET DL RETRY=4 permanently changes code in SY:DL.TSX to a retry count of 4. If the device handler is currently loaded, TSX-Plus will reload the memory image of the altered device handler (since DL does not require NOSET). When NOSET is selected, TSX-Plus allows the SET option to modify the system disk image but does not attempt to reload the memory resident copy of the device handler. Currently, two device handlers require this option: DU and MU. This is due to the fact that these handlers use addresses which are calculated and stored in memory locations during the initial loading of the device handler. If TSX-Plus were to reload the disk image, these stored addresses would no longer be correct. TSX-Plus must be rebooted in order to load the new copy of a device handler which has the NOSET characteristic.

The following default options are set by the system for standard devices:

Handler	Default Options
CR	MAPH
CT	MAPH
DB	DMA,MAPH
DD	NOMAPH
DL	DMA,MAPH,HANBUF
DM	DMA,NOMAPH
DP	DMA
DS	DMA
DT	DMA
DU	DMA,NOMAPH,NOSET
DW	MAPH
DX	MAPH
DY	DMA,NOMAPH
DZ	MAPH
FW	DMA
LP	MAPH
LS	MAPH
MM	DMA,MAPH,HANBUF
MS	DMA,MAPH,HANBUF
MT	DMA,MAPH,HANBUF
MU	DMA,NOMAPH,HANBUF,NOSET
NL	MAPH
PC	MAPH
RF	DMA
RK	DMA,MAPH
VM	EVNBUF,NOCACHE,NOMAPH
XC	MAPH
XL	MAPH

If neither the MAPH nor the NOMAPH options is specified for a handler, the handler is not mapped. If both the NOMAPH and MAPH options are specified, the handler is not mapped. Thus it is possible to override the default MAPH option for a standard handler by specifying NOMAPH. It is not possible to override the NOMAPH default.

If a handler is copied or renamed and installed using a non-standard device name, the correct attributes (from the preceeding table) should be placed on its device declaration. Note that MAPIO is never the default for any device, and therefore must be explicity included when necessary (such as DY on a 22-bit bus).

The devices CL, 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. The following are TSX-Plus supported device handlers: CR, CT, DD, DL, DM, DP, DS, DT, DU, DX, DY, LP, LS, MM, MS, MT, MU, NL, PC, RF, RK, VM and XL. The following are for the Professional-300 series computers: DW, DZ, and XC. Any others listed above are commonly used device handler names. See the section on device handlers in the *TSX-Plus System Manager's Guide* for more information.

A maximum of 15 devices can be installed in TSX-Plus as distributed, including CL, 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.

Example

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

It is not necessary to have device handlers installed in RT-11 when TSX-Plus is started. The only device handlers that must be installed are handlers for the devices where TSX-Plus swap and spool files are placed. Device handlers specified by DEVDEF macros are loaded during TSX-Plus initialization if:

1. the device handler file (*dd*.TSX) is available;
2. the device CSR (specified at offset 176 in block 0 of the handler file) responds; and
3. successful completion of handler installation code if the handler contains it.

TSX-Plus will refuse to install new device handlers which were issued by Digital subsequent to the version of RT-11 under which TSX-Plus is being started. In other words you must upgrade to the appropriate version of RT-11 in order to be able to use the newer device handlers with TSX-Plus. Specifically, the following device handlers require the indicated version of RT-11:

Handler	RT-11 version
DU	5.00
XL	5.01
MU	5.04

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 or handlers 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.

3.3 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 zero (0) as the first parameter to the SPOOL macro; the other parameters will be ignored. See the *TSX–Plus User's Reference Manual* for more information on device spooling.

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 zero (0) 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. The recommended setting for **nbuf** is 2 * **ndev**.

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 device-name. The spooled devices must be non-file-structured output devices such as line printers, plotters, or Communication Lines (CL). Note that spooled devices (except for CL lines) must also be specified in the DEVDEF (device definition) list. See the Special Device Handler chapter in the *TSX–Plus User's Reference Manual* for further information about the use of CL lines.

hold Specify 1 for this parameter if the default mode for the spooler is to be HOLD. See the SPOOL command description in the *TSX–Plus User's Reference Manual*. Specify zero (0) 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 User's Reference Manual*.

Example

```
SPOOL     2,10.,4,500.,<LP CL2>,1,10.
```

This SPOOL macro declares that there are 2 spooled devices: LP and CL2; there may be up to 10 active spooled files; four 512 byte buffers are to be used for spooling I/O; the spool file is to be 500 blocks large; default mode is HOLD; and the SPOOL BACK command is to backup 10 blocks.

3.4 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 zero (0).

MAXSF This 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 zero (0) shared file data caching is not done.

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 192 Kb 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.

3.5 Message communication parameters

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

MAXMRB Maximum number of requests for messages that may be held by the system for all jobs. This includes requests for messages that test for a pending message, wait for a message, or call a completion routine when a message arrives.

3.6 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. See the *TSX–Plus Programmer's Reference Manual* for a full discussion of the TSX–Plus real-time facilities. 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 will be restricted, but this method 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 facilities which connect device interrupts to program interrupt service or interrupt completion routines are only included in the system if the RTVECT parameter is non-zero. The remaining real-time facilities are always included in the system.

RTVECT The RTVECT parameter controls whether or not the portion of the TSX–Plus real-time facility which permits programs to connect to interrupts will be included in the generated system. If real-time program interrupt connections are not wanted, set the RTVECT parameter to zero (0). If real-time program interrupt connections are wanted, set RTVECT to the number of real-time interrupt vectors that will be used by all real-time programs.

3.7 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. See the *TSX–Plus Programmer's Reference Manual* for information on use of the performance monitor feature.

PMSIZE This parameter 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 zero (0) to avoid using any memory space for this feature. The maximum value that may be given to PMSIZE is 8192.

3.8 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 RAD50 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. The *r-flag* parameter is either "R" if user programs are to have read-only access to 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 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 the *TSX–Plus User's Reference Manual* for information about using shared run-time systems.

Examples of shared run-time declarations:

```
RTDEF     <SY CBRO60SHR>,R,1.     ;COBOL-Plus runtime
RTDEF     <SY DBLSHRRTS>,R,1.     ;DBL runtime
RTDEF     <RK2COMDT1SAV>,RW,0.
```

3.9 Time-sharing line definitions

TSX–Plus supports communication through serial lines connected to DL(V)11, DZ(Q,V)11, DH(U)11, and DH(Q,V)11 controllers. Terminals, modems, printers, computers, and other external devices may be connected to these lines.

Serial ports may be used in two distinctly different ways—either as time-sharing lines or as communication lines. A time-sharing line allows a user to control the system—the user can log on, edit files, run programs, etc. A communication line (CL) is merely a serial Input/Output port with no more control capability than any other peripheral device. Communication lines are used to drive printers and other serial devices, and to communicate with other computers or external devices.

When the system is generated, serial ports are designated as time-sharing lines or communication lines. Dedicated communication lines require less memory space in the system than time-sharing lines but may only be used as CL lines. Lines generated as time-sharing lines are normally used in that fashion but may be taken over for use as CL lines while the system is running.

In this discussion a serial port is referred to as a *line*. The control facility within TSX–Plus which drives an individual CL line is referred to as a *unit*. Each CL line which is generated into the system has a CL unit initially assigned to it. The CLXTRA parameter (see below) is used to include extra CL units in the system

which are not initially assigned to any line but which may be used to take over time-sharing lines for use as CL lines. The total number of CL units is equal to the number of dedicated CL lines plus the value of the CLXTRA parameter.

CL units are accessed as I/O devices with the names CL0, CL1, ..., CL7 for the first set of 8 units and C10, C11, ..., C17 for the second set of 8 units. It is suggested that dedicated CL lines be used to drive serial devices such as printers and plotters. Extra CL units can be included to allow local lines to connect with modem lines for dial-out.

The CL facility can be used as a replacement for both the LS and XL handlers. Advantages of CL are that a single copy of the handler can drive up to 16 devices connected to serial or multiplexer lines, and that CL provides for bidirectional transfers.

The system may be generated with any combination of time-sharing and dedicated CL lines provided that there are not more than 16 CL units. Time-sharing and CL lines may be mixed on the same multiplexer. See the Device Handlers chapter in the *TSX-Plus User's Reference Manual* for further information about the use of CL lines.

3.9.1 Default values for time-sharing lines

The following parameters establish default values which will be used for all lines unless overridden by parameters specified within individual line definition blocks.

DINSPC This parameter specifies the default number of characters that will be reserved for the input ring buffer for each time-sharing line. This value is used for all subprocesses. See the BUFSIZ macro (below) to override this for an individual line. It must be large enough to hold an entire line of input plus any characters that are typed ahead. Input ring buffers are not used for CL lines.

DOTSPC This parameter specifies the default number of characters that will be reserved for the output ring buffer for each time-sharing line. This value is used for all subprocesses. See the BUFSIZ macro (below) to override this for an individual line. A running program will be suspended when its output ring buffer is filled. The CLORSZ parameter (see below) specifies the default output ring buffer size for dedicated CL lines.

OTRASZ A job's execution is suspended and the job may be swapped out of memory when that line'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 intent is to get the job running again before all of the available output is exhausted.

NCSILO TSX-Plus allocates a character storage area for each time-sharing and CL line to hold characters before they are processed and moved to the terminal character input ring buffer. This storage area is known as a character *silo* because it functions as a *first in first out* holding buffer. The silo buffers are important in that they allow TSX-Plus to store characters that arrive in a burst. There are three parameters which set default values for the character silos: NCSILO, NCXOFF, and NCXON. The SILO macro may be used within a line definition block to specify silo parameters for a specific line which override the default values.

The NCSILO parameter specifies the default silo size. A value of 32 (decimal) is a reasonable silo size for most applications although you may want to use larger silos on lines that receive high-speed input from external devices or other computers. The character silo buffers are allocated in the 40 Kb low-memory portion of TSX-Plus so you should not be overly generous in their allocation. The maximum silo size is 255 characters; the minimum silo size is 32 bytes for time-sharing lines and 16 bytes for dedicated CL lines.

NCXOFF When a character silo fills to the point that only NCXOFF free character positions remain, the system will transmit an XOFF (CTRL-S) to try to stop transmission from the external device. The choice for NCXOFF depends on the speed of transmission, the time it takes for the XOFF to propagate

to the sending device, and the time it takes for the sending device to respond to the XOFF and stop transmitting. A reasonable range of values for NCXOFF is 4 to 16. If characters appear to be lost due to input silo overrun, increase the value of NCXOFF. The SILO macro may be used within a line definition block to specify the XOFF control parameter for a particular line. The maximum value is 1/2 the silo size minus 2.

NCXON If the system sends an XOFF character because a silo buffer becomes nearly full, it will send an XON (CTRL–Q) to restart transmission when there are only NCXON characters remaining in the silo. The value of this parameter is not very critical; the recommended value is 4. The SILO macro may be used within a line definition block to specify the XON control parameter for a particular line. The maximum value is 1/2 the silo size minus 2.

CLXTRA This parameter specifies the number of unattached communication line (CL) units to be included in the system. These CL units are not initially assigned to any line but may *take over* an inactive time-sharing or CL line by use of the SET CL keyboard command or a TSX–Plus system service call (EMT). The total number of CL units included in the system is the sum of the CLXTRA parameter plus the value specified for the fourth parameter to the TBLDEF macro (see below). No more than 16 total CL units may be included in the system. CL units are referenced as I/O devices using the names CL0:–CL7: for the first set of eight units, and C10:–C17: for the next eight units.

CLORSZ Each CL unit has an output character ring buffer which is used to transfer characters from the output data buffer to the communications controller. CL units do not use input ring buffers, they transfer characters directly from the silo to the requesting program. The output ring buffer allows the system to overlap the transmission of the final characters for one write operation with the time needed to get the next write operation started. The CLORSZ parameter specifies the default output ring buffer size for CL units. The recommended value for CLORSZ is ((3**baud rate*)/1000+2). For example, the recommended value for a 9600 baud line is ((3*9600)/1000+2) or 31. The BUFSIZ macro can be used within a CL line definition block to specify the output buffer size for an individual CL line.

NRMFLG The NRMFLG parameter is used to specify a default set of options to apply to all lines unless the FLAGS macro is used within a line definition block. The value specified for the NRMFLG parameter must be the logical sum of a set of option symbols. When more than one option is specified, the names of the option symbols should be joined together with exclamation marks ("!") which is the MACRO assembler syntax symbol for the logical OR operation. The valid option symbols that may be used to form the value for NRMFLG are described below.

$8BIT If this flag is set, 8 bits of each received character are passed to the user's program. If this flag is not set, only the low order 7 bits of each character are passed to the program. Regardless of the setting of this flag, the null character (000) is never passed to the program. Also, if the VT*nnn* escape sequences are declared to be a TSX–Plus activation condition, the character 377 (octal) also may not be received. If 8 bit mode is selected, the terminals should be set to transmit and receive 8 bit characters without parity. The SET TT [NO]EIGHTBIT keyboard command can also be used to control this option. This flag controls software masking to 7 or 8 bits. The SPEED macro (see below) controls hardware character length selection.

$AUTO This flag enables automatic baud rate selection (autobaud) for the line. This may only be used with lines connected to hardware controllers that support programmable speed selection such as DZ(Q,V)11, DH(Q,V)11, and DLV11–E. When autobaud is specified for a line the line's speed is set to match the speed of the terminal connected to the line. When the carriage return character is received to start up a line, the system examines the (possibly garbled) character and uses the received character value to determine the speed of the transmitting terminal. The line's speed is then set to that value and the line is started. The following baud rates are supported by automatic speed selection: 110, 300, 1200, 2400, 4800, 9600, and 19200. It is necessary to type two consecutive carriage returns to start a line with autobaud if the speed is less than 1200 baud. A speed of 19200 is not recommended for machines slower than an 11/73 or 11/44 and

this speed is not supported by some hardware controllers such as the DEC DZ(Q,V)11. The SET TT [NO]AUTOBAUD keyboard command can also be used to control this option. If a SPEED macro (see below) and a $AUTO flag are both specified for a line, then the $AUTO flag will take precedence.

$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 SET TT DEFER command in the *TSX-Plus User's Reference Manual* for an explanation of deferred character echoing. Deferred echoing is recommended.

$ECHO Echo characters to the terminal. This flag is normally included and would be omitted only when using half-duplex communications. See also SET TT [NO]ECHO command.

$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. See also, SET TT [NO]FORM command.

$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 during program execution. See also SET TT [NO]LC command.

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

$NOSUB If this flag is specified, the line will not be allowed to use the TSX-Plus subprocess facility. That is, the line will always be connected to its primary process.

$PAGE This flag is specified to select whether CTRL-Q and CTRL-S (XON/XOFF) characters are intercepted by the operating system and used for flow control, or whether they are passed to the program like all other characters. Most lines should have this option set. The $PAGE setting is automatically selected if any of the following terminal types are specified for a line: VT100, VT200, VT52, LA120, Diablo, Qume. See also SET TT [NO]PAGE command.

$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. This flag may be specified for a local terminal. See previous section on the PHONE parameter and the *TSX-Plus System Manager's Guide* for further information about modem control. See also SET TT [NO]PHONE command in the *TSX-Plus User's Reference Manual*.

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

$SCOPE Terminal is a CRT type terminal and DELETE is to echo as *backspace space backspace*. See also SET TT [NO]SCOPE command.

$START If this flag is set the line will automatically be initiated when TSX-Plus is started. If the flag is not set, the line will not be initiated until carriage-return is pressed at the terminal. This flag is not recommended for lines with modems attached.

$SYSPS If this flag is set, then entry of the system password will be required before the standard TSX-Plus greeting message will be displayed. This is useful for modem lines where it is not desirable to display information about the site or the system to unauthorized users. See the SYSPS macro (above) for information regarding the setting and use of this password.

$TAB Do not simulate tabs by inserting spaces. Use with terminals whose hardware responds to tab characters, such as VT100 terminals. See also SET TT [NO]TAB command.

$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 another computer that is transmitting lines of data terminated by carriage-return and line-feed. When tape mode is selected, the system discards incomming line-feed characters. 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.

Note that the FLAGS macro can be used to specify options on a *line by line* basis. Most of the settings may be changed after the line is initialized by use of a SET TT keyboard command.

3.9.2 Specifying the number of lines

Each line that is to be used as a TSX–Plus time-sharing or communications line must be declared in TSGEN. The total number of lines is first declared by setting the proper values as arguments to the TBLDEF macro. The form of the TBLDEF macro is:

```
TBLDEF   num-real,num-subprocess,num-detached,num-cl
```

where *num-real* is the number of real (physical) time-sharing lines (this value should equal the number of LINDEFs specified), *num-subprocess* is the number of subprocess jobs, *num-detached* is the number of job slots to allocate for the execution of detached jobs, and *num-cl* is the number of dedicated communication lines (this value should equal the number of CLDEFs specified).

See the *TSX–Plus User's Reference Manual* for more information on subprocesses and detached jobs.

TSX–Plus will support any number of total lines, including time-sharing lines, CL lines, subprocesses, and detached jobs. One is limited only by the amount of space available in low-memory for job dependent tables. However, you should not define more lines than necessary for the system. Performance with a large number of lines generated is highly dependent on the system configuration, specifically the 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 and dedicated CL lines. Subprocesses and detached jobs are not described by LDB's.

A Line Definition Block for a time-sharing line begins by using the LINDEF macro and ends by using the LINEND macro. A line definition block for a dedicated CL line begins by using the CLDEF macro and ends by using the CLEND macro. Each LDB *must* have matching calls to LINDEF and LINEND or CLDEF and CLEND. Other optional macros may be used within a line definition block to specify parameters for that line.

TSX–Plus supports lines connected to DL11 and DLV11 serial communication cards and lines connected to DZ11, DZV11, DZQ11, DH11, DHU11, DHV11, and DHQ11 multiplexers. TSX–Plus will support a mixture of dial-up and local lines. Unless otherwise stated, there is no distinction between DL11 and DLV11 support, nor between DZ11 and DZ(Q,V)11 support. However, DH11 support is different from DHV11 and DHU11 support.

3.9.3 Specifying multiplexer lines

The line definition blocks for lines connected to DZ(Q,V)11, DH11, DHU11, and DH(Q,V)11 multiplexers are enclosed within a Multiplexer Definition Block (MDB). An MDB begins with a DZDEF, DHDEF, DHVDEF or DHUDEF macro call, contains the LDB's for all lines connected to the multiplexer, and ends with a MUXEND macro call. Thus the general model for defining a group of lines connected to a multiplexer is:

```
DZDEF or DHDEF or DHVDEF or DHUDEF   ;Start of lines for this multiplexer
LINDEF                               ;Start first line definition block
:                                    ;Parameters relating to first line
LINEND                               ;End of first line definition block
LINDEF                               ;Start of next LDB
:                                    ;Parameters relating to this line
LINEND                               ;End of LDB
MUXEND                               ;End of lines for this multiplexer
```

The DZDEF macro (for DZ11 multiplexers), the DHVDEF macro (for DHV11 and DHQ11 multiplexers), and the DHUDEF macro (for DHU11 multiplexers) all require two arguments and are similar in form:

```
DZDEF     vector,csr
DHVDEF    vector,csr
DHUDEF    vector,csr
```

where *vector* specifies the address of the receiver interrupt vector, and *csr* specifies the 16-bit address of the Control and Status Register (CSR).

The DHDEF macro is used to specify values for a DH11 multiplexer. It requires four parameters and has the following form:

```
DHDEF     dhvector,dhcsr,dmvector,dmcsr
```

where *dhvector* is the address of the receiver interrupt vector for the DH11, *dhcsr* specifies the address of the Control and Status Register (CSR) for the DH11, *dmvector* specifies the address of the interrupt vector for the associated DM11 modem control unit, and *dmcsr* specifies the address of the CSR register for the DM11. A DM11 is an optional unit which provides modem control for DH11 lines. If there is not a DM11 associated with the DH11, specify zero (0) for *dmvector* and *dmcsr*.

Note that the system interface to a DH11 multiplexer is different from that to a DHV11 and DHU11 multiplexer as they have a different control register structure. Thus a DHV11 and DHU11 is *not* the same as a DH11 compatible multiplexer installed on a Q-Bus system. Therefore, choose the DHDEF, DHVDEF, or DHUDEF macros based on the type of multiplexer, not on the type of system on which they are installed. The DHV11 and DHU11 multiplexers have identical control register structures and therefore may be used interchangably.

TSX-Plus uses the DMA output capabilities of the DH11, DHV11, and DHU11 multiplexers. This reduces system overhead for character output by about 14% compared with a DZ11 driving a time-sharing line or about 20% for a DZ11 driving a CL line.

TSX-Plus will support up to eight multiplexers. If there is more than one multiplexer, each must be defined using a separate multiplexer definition block beginning with a DZDEF, DHDEF, DHUDEF or DHVDEF macro and ending with a MUXEND macro. Note that multiplexer port numbers (as specified with the LINDEF or CLDEF macros for multiplexer lines) start at 0 for each multiplexer.

3.9.4 DL11 line definitions

Line definition blocks for lines connected to DL(V)11 type controllers are different than multiplexer line definition blocks. In the case of DL(V)11 lines, the addresses of the DL(V)11 interrupt vector and Control and Status Register (CSR) are specified with the LINDEF macro that starts the line definition block.

Different model DL(V)11 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 DL(V)11 cards normally start at 300 and increase by 10 (octal) per line.

Note on DLV11-J controllers, port 3 is special and can be configured by a wirewrap jumper to have its vector and CSR address at the normal operator console location (60 and 177560) or alternatively at a location above the addresses for port 2. If you have more than one DLV11-J controller make sure you do not have more than one line configured as the console terminal.

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. Make sure the vector and CSR addresses of your time-sharing line controllers do not conflict with each other or with those for other devices such as LS or LP.

3.9.5 Time-sharing line definition blocks

The line definition block for a time-sharing line begins with a LINDEF macro and ends with a LINEND macro. The LINDEF macro requires different parameters depending on whether it describes a DL(V)11 type line or a multiplexer line. The LINDEF macro for a DL(V)11 type line requires two parameters and has the form:

```
LINDEF      vector,csr
```

where *vector* is the address of the input (receiver) interrupt vector, and *csr* is the address of the receiver control and status register (CSR).

A LINDEF macro for a multiplexer line requires only a single argument and has the form:

```
LINDEF      muxport
```

where *muxport* is the number of the port within the multiplexer group. DZ(Q,V)11 lines are numbered from 0 to 3, DZ11 and DH(Q,V)11 lines are numbered 0 to 7, and DH11 and DHU11 lines are numbered 0 to 15. Note that a decimal point should be specified with the port number to indicate that the value is decimal.

The LINDEF macro also accepts a third (second for multiplexer lines) optional parameter. One terminal may be declared to be the operator 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 console.

3.9.6 CL line definition blocks

The line definition block for a CL line begins with a CLDEF macro and ends with a CLEND macro. The CLDEF macro requires different parameters depending on whether it describes a DL(V)11 type line or a multiplexer line. The CLDEF macro for a DL(V)11 type line requires three parameters and has the form:

```
CLDEF       unit,vector,csr
```

where *unit* specifies the CL I/O device unit number by which this line will be referenced, *vector* is the address of the input (receiver) interrupt vector, and *csr* is the address of the receiver control and status register (CSR). For example, *unit* number 3 is referred to by the name "CL3:". The units may be specified in any order but the unit numbers must be less than the total number of CL units (i.e., (TBLDEF parameter *num-cl*)+CLXTRA). The CL *unit* numbers must be unique.

A CLDEF macro for a multiplexer line requires two parameters and has the form:

```
CLDEF       unit,muxport
```

where *unit* is the CL unit number and *muxport* is the number of the port within the multiplexer group.

The BUFSIZ, FLAGS, NAME, SILO, and SPEED macros may be used within a CL line definition block to specify parameter values for the line.

3.9.7 Line definition control macros

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

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

```
FLAGS     flag-values
```

where *flag-values* is the logical sum of those flags that are to be set for the line. The symbolic flag names are the same as specified for the NRMFLG parameter (see page 26). 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 inclusive OR operation.

Absence of a FLAGS macro within a line definition block is equivalent to specifying NRMFLG for the line.

The FLAGS macro may be used within a CL line definition block. CL lines only use XON/XOFF flow control. Only the $8BIT, $TAB and $FORM flags are significant.

Note that the FLAGS macro sets the default line characteristics when each line is started and that the SET TT 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 indicate that the terminal is not a CRT device.

SPEED The SPEED macro is used to specify the transmit/receive baud rate, number of bits, and parity for lines connected to hardware controllers that support programmable baud rates such as DLV11-E, DZ(Q,V)11 and DH(Q,V)11. The form of the SPEED macro is:

```
SPEED     speed-code[,length,parity]
```

where *speed-code* is a symbolic name formed from concatenating the letter "S" with the baud rate. The following *speed-codes* may be used: S75, S110, S134.5, S150, S300, S600, S1200, S1800, S2000, S2400, S3600, S4800, S7200, S9600, and S19200. A baud rate of 19200 is not supported by DL11 and DZ11 lines. Baud rates of 2000, 3600, and 7200 are not supported by DH11 lines. Baud rates of 3600 and 7200 are not supported by DHV11 or DHU11 lines.

Length is either 7 or 8 (decimal), indicating the number of data bits in each character. *Length* specifies the hardware character length as opposed to the $8BIT flag which controls software character masking.

Parity is either EVEN, ODD, or NONE. If *length* and *parity* are not specified, the default is 8 bits and no parity.

If a line definition block does not contain a SPEED macro, the speed specified for the last defined line will be used. If a SPEED macro and a $AUTO flag are both specified for a line, then the $AUTO flag will take precedence.

NAME The NAME macro is used to specify a descriptive name for the line. The use of the NAME macro is optional. The form of the NAME macro is:

```
NAME      <descriptive text>
```

where *descriptive text* can be up to 12 characters long and should be enclosed in angle brackets ("<" and ">"). *Descriptive text* has no effect on the operation of the system except that it is displayed in the *Job Name* column of the SHOW TERMINALS command display and is shown in the SYSMON screen for terminal line status.

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
```

where *terminal* is one of the following:

Name	Terminal type
VT200	DEC VT200 family terminal
VT100	DEC VT100 family terminal
VT52	DEC VT52 terminal
LA36	DEC LA36 terminal
LA120	DEC LA120 terminal
HAZEL	Hazeltine brand terminal
ADM3A	Lear Siegler ADM3A terminal
DIABLO	Diablo brand terminal (with XON/XOFF protocol)
QUME	Qume brand terminal (with XON/XOFF protocol)
	(Diablo and Qume are treated as equivalent)

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 XON/XOFF (DC1/DC3, CTRL-Q/CTRL-S) protocol are acceptable. See the SET TT command in the *TSX-Plus User's Reference Manual* for more information on the meaning of each terminal type.

BUFSIZ The BUFSIZ macro is used to specify the size of the line's input and output character ring buffers. The form of the macro is:

```
BUFSIZ      inputsize,outputsize
```

where *inputsize* is the number of characters in the input character ring buffer, and *outputsize* is the number of characters in the output character ring buffer. If a BUFSIZ macro is not used in a Line Definition Block, the default sizes are the same as specified for DINSPC and DOTSPC. If the BUFSIZ macro is used within a CL line definition block, *inputsize* is ignored but *outputsize* is used to control the size of the CL unit output buffer. If no BUFSIZ macro is used within a CL line definition block, the value of the CLORSZ parameter controls the size of the output ring buffer.

SILO The SILO macro is used to specify values for parameters related to character input silo buffers. The NCSILO, NCXOFF, and NCXON parameters (see above) specify default silo values for all lines unless a SILO macro is used within a line definition block to specify values for that line. The form of the SILO macro is:

```
SILO      ncsilo,ncxoff,ncxon
```

where *ncsilo* is the silo size, *ncxoff* is the number of free character positions remaining at which point an XOFF is transmitted, and *ncxon* is the number of remaining characters at which point an XON is transmitted to restart the sender.

A SILO macro may be used in line definition blocks for both time-sharing and dedicated CL lines. The silo size is constrained to 32.–255. for time-sharing lines, and to 16.–255. for dedicated CL lines. Space in memory may be saved by specifying a silo size of zero for CL lines that will be used for output only (e.g., a CL line used to drive a printer).

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:filnam.ext
```

where *dev:filnam.ext* is the name of a command file. This argument must be included in order to use the TSX–Plus LOGON facility.

The CMDFIL macro should not be used in the line definition block for a dedicated CL line.

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

3.9.8 Line definition example

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

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

     TBLDEF   5.,2.,1.,1.        ;5 real, 2 subprocesses, 1 detached, 1 CL

; Define DL11 lines

; Define DL11 line #1
     LINDEF   300,175610         ;DL11-E line. Vector=300, CSR=175610
     CMDFIL   SY:START.COM       ;Start up command file
     FLAGS    NRMFLG!$SCOPE      ;Control flag options
     NAME     <Work room>        ;Descriptive name for line
     SPEED    S9600              ;Speed = 9600 baud
     TRMTYP   VT200              ;Terminal type
     BUFSIZ   120.,300.          ;Input and output ring buffers
     LINEND                      ;End of DL11 line definition block

; Define DL11 line #2
     LINDEF   60,177560,OPER     ;Console terminal
     FLAGS    NRMFLG!$START      ;Line with auto startup
     NAME     <Console>          ;Descriptive name for line
     SPEED    S9600              ;9600 Baud
     TRMTYP   VT200              ;Terminal is a VT200
     LINEND

; Define DZ11 lines
     DZDEF    310,177620         ;Start of DZ11 definition block

; Define DZ11 line # 0
     LINDEF   0                  ;MUX line #0
     CMDFIL   SY:LINE1.TSX
     FLAGS    NRMFLG!$SCOPE
     NAME     <Room 1203>        ;Descriptive name for line
     SPEED    S1200,7.,EVEN      ;1200 baud, 7 data bit, even parity
     TRMTYP   VT100              ;Terminal type
     LINEND
```

```
; Define DZ11 line # 1
     LINDEF   1                  ;MUX line #1
     CMDFIL   SY:LOGON.TSX
     NAME     <Room 1204>        ;Descriptive name for line
     SPEED    S300,8.,NONE       ;300 baud, 8 data bits, no parity
     TRMTYP   LA36
     LINEND

; Define DZ11 line # 6
     LINDEF   6                  ;MUX line #6
     CMDFIL   SY:LINE5.TSX
     FLAGS    NRMFLG!$PHONE!$AUTO ;Phone line with autobaud speed select
     NAME     <123-4567>         ;Descriptive name for line
     TRMTYP   VT100
     LINEND

; Define DZ11 line # 7 as CL unit 0
     CLDEF    0,7                ;MUX line #7 is CL0:
     FLAGS    $FORM              ;Printer supports form feeds
     NAME     <Printer>          ;Descriptive name for line
     SPEED    S4800              ;4800 baud
     CLEND

; End of MUX lines
     MUXEND                 ;End of MUX definition block
```

3.10 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:filnam.ext [parameters...]
```

where *dev:filnam.ext* 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. Optional *parameters* may be specified in the same manner as explained in the *TSX-Plus User's Reference Manual* for the DETACH keyboard command. The total number of characters specified for *dev:filnam.ext* and *parameters* may not exceed eighty (80).

There may be one DETACH macro specified 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.

Any command file invoked by the DETACH macro when the system is started is classified as a start-up command file. This means that all the privileges and restrictions that apply to start-up command files apply to these command files as well.

3.10.1 An example of using the DETACH macro

The following is an example of how to specify two commonly used programs to start as detached jobs when TSX-Plus is initiated:

```
DETACH      SY:RTSORT.TSX
DETACH      SY:WINPRT.TSX
```

The WINPRT program must be executing as a detached job in order to process a *print window* request. The command file WINPRT.TSX might include the following:

```
SET PROCESS/NAME=SYSTEM
MEMORY 24
R WINPRT
```

The RTSORT program must be executing in message mode as a detached job in order to process sort requests. The command file RTSORT.TSX might include the following:

```
SET PROCESS/NAME=SYSTEN/PRIV=(STANDARD,GETCXT)
R RTSORT
/MESSAGE/GETCXT
```

Chapter 4

Linking and Starting TSX–Plus

4.1 Assembling the modified TSGEN module

Once the TSGEN module has been modified to contain the desired parameter settings, it must be assembled using the RT–11 MACRO assembler. The command to do this is:

```
MACRO TSGEN
```

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

```
MACRO/LIST TSGEN
```

No errors should be reported during the assembly. If an error is reported, determine the location of the error from the TSGEN listing, and correct the problem. The *RT–11 MACRO Assembler Manual* is useful for deciphering assembler messages.

4.2 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
```

This command file creates four SAV files (program images): TSX.SAV, TSKMON.SAV, TSXMOD.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 the *TSX–Plus System Manager's Guide* for more information about SYSMON.

The linker will print the warning message "?LINK-W-Multiple definition of $OVRH" while linking TSKMON and TSXMOD. This is expected and is not an error.

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.***

also, conflicting PSECT attributes due to KM3PAT

4.3 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 all devices declared in TSGEN by the device definition macro.
3. The RT–11 device handlers for the devices on which the TSX–Plus swap and spool files will be placed 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.) See the *TSX–Plus System Manager's Guide* for more information on TSAUTH.
5. If user-defined commands are to be allowed, the TSXUCL.SAV 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. If any shared run-time systems were specified in the system generation, they must be on the appropriate disks before the system is started.
8. TSX–Plus may be started under RT11SJ (single job) or RT11BL (base line). Any attempt to start it under the foreground-background (FB) or extended-memory (XM) versions of RT–11 are likely to result in a "?KMON-F-Insufficient memory" error message. This message may also occur if RT11SJ has been sysgenned to include multi-terminal support or other features which make it too large to be able to start TSX–Plus, 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 version 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 specified to start automatically ($START flag) should print the TSX–Plus greeting message. Other lines will be initiated when carriage return is pressed at the terminal.

During its initialization TSX–Plus performs a test to make sure that 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, and the INIABT flag is set to 1, TSX–Plus displays the message:

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

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

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 DL(V)11 interrupt vectors and receiver status registers. Different models of DL(V)11 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 56 Kb. 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 the *TSX–Plus User's Reference Manual* for information on error messages received during TSX–Plus startup and operation.

4.4 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 Programmer's 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 are 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 these sizes 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 each program SAV file.

Chapter 5

TSXMOD: TSX–Plus Configuration Utility

5.1 What is TSXMOD?

TSXMOD is a specialized patching utility which allows reconfiguration of TSX–Plus without performing a system generation. Although regenerating TSX–Plus is relatively simple (typically requiring only a few minutes), sometimes the minimal desired changes do not seem to warrant even a simple regeneration. TSXMOD alters the disk image of TSX.SAV so that when TSX–Plus is next restarted the modified information is used.

TSX–Plus is distributed as a collection of files, including several object modules and a configuration file (TSGEN.MAC). System generation involves editing parameters in TSGEN.MAC to reflect site specific hardware characteristics and operating preferences. This is then assembled and linked with the object modules provided to create the executable image of the operating system, TSX.SAV. Because TSXMOD alters specific relative locations in TSX.SAV, it must be linked with the symbol table for TSX.SAV. This is done automatically by the TSXLNK command file when building TSX–Plus. If you wish to regenerate TSX–Plus, but preserve the old file (TSX.SAV) and be able to use TSXMOD to patch it at a later time, then you must also save the TSXMOD which was linked with the TSX.SAV. The four files which are dependent on changes in TSGEN are: TSX.SAV, TSKMON.SAV, SYSMON.SAV and TSXMOD.SAV. They must always be saved and restored together.

When TSX–Plus is started, it uses the configuration information from TSGEN to select most operating characteristics. Although many of these characteristics may be dynamically modified by keyboard commands while TSX–Plus is executing, some of the information is used during initialization and cannot be changed during execution. It is this type of information which may be modified by TSXMOD. Most, but not all, of the parameters in the user-modifiable portion of TSGEN can be changed by TSXMOD. Those parameters which affect the amount of memory reserved in TSGEN are generally not alterable without revising TSGEN.MAC and rebuilding TSX–Plus.

5.2 Running TSXMOD

Since TSX–Plus does not refer to its disk image during execution, TSXMOD may be used to modify the TSX.SAV file while running either RT–11 or TSX–Plus. TSXMOD requires approximately 32 Kb of memory. If TSXMOD has been copied to your system device, then use the command:

```
R TSXMOD
```

or, if TSXMOD is on your current default disk, use:

```
RUN TSXMOD
```

or, you may always specify the device where TSXMOD can be found:

```
RUN dev:TSXMOD
```

where *dev* represents your actual device name and unit number, such as DL0 or DM2.

When TSXMOD is started, it first tries to load DK:TSX.SAV. A file is only loaded successfully if it can be opened and read, and if it was linked with the appropriate symbol table. If DK:TSX.SAV cannot be loaded, an informational message is displayed and TSXMOD tries to load SY:TSX.SAV next. When a valid input file is found and successfully loaded at startup, an informational message is displayed. For example:

```
!TSXMOD-I-Input file -- DV0:TSX.SAV
```

If neither file can be loaded, then there is nothing to be patched, so the only valid commands are READ and QUIT. When DK: or SY:TSX.SAV is loaded into memory, it becomes the *current file*. When a READ or WRITE command is issued without a file name, then the file affected is the *current file*. If either the READ or WRITE command specifies a file name, then that file becomes the *current file*. When TSXMOD is EXITed, if any changes have been made since the last WRITE, then the new parameters are rewritten to the *current file*.

5.3 TSXMOD Commands

There are two types of commands to TSXMOD. The first type is used to configure TSX–Plus, and the other is used to control TSXMOD. The format of configuration commands is identical to the format for those same commands in TSGEN.MAC. (Note: the TSXMOD prompt is an exclamation point "!") Some configuration command examples are:

```
!SWDBLK: .RAD50 /SY TSXSWPTSX/
!HIMEM = 64
!BUSTYP = QBUS
```

The current values of all parameters and macro calls may be displayed with the SHOW command followed by the name of the parameter or macro call. For example:

```
!SHOW QUANO
QUANO   = 2
!SHOW NRMFLG
NRMFLG  = $DEFER+$LC+$PAGE+$ECHO
```

Commands which control TSXMOD are used to READ and WRITE files, exit TSXMOD with or without saving current data, or SHOW such information as current values or an abbreviated link map. For example:

```
!SHOW FILE
!Current file is DK :TSX    .SAV
!QUIT
```

5.4 Creating TSXMOD command files

Commands to TSXMOD may be accepted from either the terminal or a command file. This allows the output from a previous session of TSXMOD, which was recorded using the terminal logging facility of TSX–Plus, to be edited and serve as a means of restoring a given configuration without rebuilding TSX–Plus or maintaining full copies of TSX.SAV or even the shorter copies preserved with a TSXMOD WRITE command.

We also recommend creating a log file while running TSXMOD and issuing the SHOW ALL command in order to maintain a record of your current configuration. In fact, when communicating a problem report or inquiry regarding operation of TSX–Plus, it is useful to include the information from a monitor SHOW ALL command and another from within TSXMOD. The following command file demonstrates this:

```
SET LOG FILE=CONFIG  !OPEN LOG FILE
SHOW ALL             !SHOW RUNNING CONFIGURATION
RUN TSXMOD           !RUN TSXMOD
SHOW ALL             !SHOW CURRENT TSGEN CONFIGURATION
QUIT                 !QUIT TSXMOD
SET LOG CLOSE        !CLOSE LOG FILE
PRINT CONFIG.LOG     !PRINT THE LOG FILE
```

In order to use a log file as a command file to reconfigure TSX–Plus: while running TSX–Plus, start a log file, run TSXMOD, issue the SHOW ALL command, QUIT TSXMOD, and close the log file. With some minor editing of this log file, it is simple to reconfigure TSX–Plus or restore a known configuration by invoking the log file as a command file. The following example indicates those commands or characters which should be removed through editing the file, along with any other desired parameter changes.

```
.SET LOG FILE=MOD.COM   !THIS LINE WILL NOT BE IN LOG FILE
.RUN TSXMOD             !REMOVE THE . PROMPT
!TSXMOD-I-Input file -- DK :TSX   .SAV

!SH ALL
SWDBLK: .RAD50 /SY TSXSWPTSX/
SPLBLK: .RAD50 /SY TSXSPLTSX/
RSFBLK: .RAD50 /SY TSXRSFTSX/
HIMEM  = 64.
...
!TSDHIO   050354
!TSCLR    052360
!TSINIT   052606
! INITOP  100324
!QUIT
!TSXMOD-I-Session not saved.
.SET LOG CLOSE              !REMOVE THIS LINE
EXIT                        !ADD THIS LINE
```

This use of a logged and edited TSXMOD session as a command file to reconfigure TSX–Plus is also the reason why the exclamation point is used as the TSXMOD prompt character and prefixes most of its text lines. When accepting input from a command file, TSXMOD ignores everything on a command line after an exclamation point.

5.5 Commands used to control TSXMOD

These commands are used to control the operation of TSXMOD, such as selecting the file to be modified or exiting the program.

Command	Explanation
DELETE	Prefixes the DEVDEF, RTDEF or DETACH configuration commands in order to remove an entry and replace it with an unused entry.
EXIT	Exits TSXMOD after writing memory to the current file. (Does not automatically write unless changes have been made.)
READ [*file*]	Loads memory from the current file or optionally from the specified *file*. If *file* is specified, it becomes the current file.
SET TT *n* SET TERMINAL *n*	Selects the current time-sharing line definition block. The SHOW TT *n* and SHOW LINDEF *n* commands also set the current line number. Time-sharing lines are numbered in the order in which they are defined in TSGEN.MAC.
SHOW *keyword*	Prefixes a configuration *keyword* (the parameter or macro name) in order to display the current value. The value in memory is the current value and will not reflect the value in TSX.SAV if it has been changed until memory is rewritten to the disk file with a WRITE command or upon EXIT.
SHOW ALL	Displays all configuration parameters and teeir current values and the TSX-Plus module entry point addresses.
SHOW FILE	Displays the current file name.
SHOW MAP	Displays TSX-Plus module entry point addresses. (The most useful information from the link map.)
SHOW PHYSICAL	Displays the current file name with any logical device assignments translated to their equivalent physical device.
QUIT	Terminates TSXMOD without rewriting to a disk file.
WRITE [*file*]	Causes memory to be rewritten to the current file or optionally to the *file* specified. If *file* is specified, it becomes the current file.

5.6 TSXMOD Configuration Commands

Parameters which are normally configured in TSGEN by an equate statement (e.g., QUAN0=1) or by a macro call (e.g., RTDEF <SY CBR060SHR>,R,1) may be configured by entering the same type of statement as a command to TSXMOD. (We will refer to these types of equate statements and macro calls in the user-modifiable section of TSGEN.MAC as TSGEN statements.) Current values of all TSGEN statements may be displayed with SHOW commands.

5.6.1 Simple numeric parameters

The most common type of parameter is simply equated to a numerical value. (Note that decimal is the default radix for all input.) For example:

```
!SHOW DFLMEM
DFLMEM = 56.
!DFLMEM = 64.
!SHOW DFLMEM
DFLMEM  = 64.
```

5.6.2 Simple non-numeric parameters

Some parameters are equated to symbolic named values. For example:

```
!SHOW UCLORD
UCLORD  = MIDDLE
!UCLORD = LAST
!SHOW UCLORD
UCLORD  = LAST
```

5.6.3 RAD50 parameters

Some parameters which declare file names use the MACRO syntax for declaring a .RAD50 value. For example:

```
!SHOW SWDBLK
SWDBLK:         .RAD50  /SY TSXSWPTSX/
!SWDBLK:        .RAD50  /DLOSWPFILTSX/
!SHOW SWDBLK
SWDBLK:         .RAD50  /DLOSWPFILTSX/
```

5.6.4 Device definitions

Device handlers to be loaded by TSX–Plus are declared with the DEVDEF macro. The two-character device name must be enclosed by angle brackets and may be followed by several optional symbolic parameters. For example:

```
!SHOW DEVDEF
DEVDEF <DL>
DEVDEF <DX>
DEVDEF <LP>,MAPH
DEVDEF <NL>
```

If unused device definition slots are available, new devices may be added simply by declaring a new DEVDEF. For example:

```
!DEVDEF <RK>
!SHOW DEVDEF
DEVDEF <DL>
DEVDEF <DX>
DEVDEF <LP>,MAPH
DEVDEF <NL>
DEVDEF <RK>
```

Device definitions may also be removed with the DELETE command. For example:

```
!DELETE DEVDEF <RK>
!SHOW DEVDEF
DEVDEF <DL>
DEVDEF <DX>
DEVDEF <LP>,MAPH
DEVDEF <NL>
```

5.6.5 SPOOL

Devices (usually printers) which are to be spooled by TSX-Plus are declared by the SPOOL macro. This macro is not changeable by TSXMOD. However, the spool disk file size (the fourth parameter to the SPOOL macro) may be changed by adjusting a parameter named SPLSIZ. For example:

```
!SHOW SPOOL
!SPOOL       2.,20.,2.,2000.,<LP CL2>,0,5.
SPLSIZ = 500
!SHOW SPOOL
!SPOOL       2.,20.,2.,500.,<LP CL2>,0,5.
```

5.6.6 Shared run-times (RTDEF)

Shared run-time areas are declared using the RTDEF macro. The name of the file to be loaded is specified in angle brackets, followed by "R" or "RW" if the shared run-time memory is read-only or read-write access, and then the number of blocks to skip when loading the disk file into memory. RTDEFs may be deleted with the delete command. If free RTDEF slots are available, new RTDEFs may also be declared. For example:

```
!SHOW RTDEF
RTDEF    <SY CBRO60SHR>,R,1.
!RTDEF   <DLOSHAREDRTS>,RW,1
!SHOW RTDEF
RTDEF    <SY CBRO60SHR>,R,1.
RTDEF    <DLOSHAREDRTS>,RW,1.
!DELETE RTDEF <SY CBRO60SHR>,R,1
!SHOW RTDEF
RTDEF    <DLOSHAREDRTS>,RW,1.
!DELETE RTDEF <DLOSHAREDRTS>,RW,1
!SHOW RTDEF
!No shared run-times (RTDEF) declared.
```

5.6.7 Unchangeable statements

Not all TSGEN statements may be configured by TSXMOD. The following TSGEN statements may not be changed by TSXMOD: MAXSEC; MAXALC; MXSPAC; MXTTCT; WILDFL; SPOOL (although spool disk file size may be changed by adjusting SPLSIZ); RTVECT; PMSIZE; default time-sharing line parameters (DINSPC, DOTSPC, OTRASZ, NCSILO, NCXOFF, NCXON, CLXTRA, CLORSZ, NRMFLG); TBLDEF; LINDEF; LINEND; MUXDEF; DHDEF; DHVDEF; DZDEF; MUXEND; CLDEF; CLEND. For example:

```
!CLXTRA = 4
!This parameter not adjustable without rebuilding TSX-Plus.
CLXTRA = 2.
```

5.6.8 Time-sharing line definitions

Neither terminal line definitions nor multiplexer definitions may be changed by TSXMOD, but the optional macro calls within each line definition block may be changed. Current terminal line characteristics may be displayed with the SHOW LINDEF [*n*] command, or an abbreviated display may be obtained with the

SHOW TERMINAL [*n*] command. The optional number [*n*] indicates which time-sharing line is to be selected. If *n* is zero (0), then all lines will be displayed. For example:

```
!SHOW TERMINAL 0
!Line  Type     Vector     CSR     Terminal  Speed     Line Name
!----  -----  ----------  ------  --------  --------  ------------
!1     OPER   DL -   060  177560  VT200     9600  8N  Console
!2     Local  DH -0  330  160060  VT200     Auto  8N  Charles G Br
!3     Local  DH -1  330  160060  VT100     9600  8N  Sir NDippity
!4     Local  DH -4  330  160060  VT100     Auto  8N  VAX link
!5     Local  DH -5  330  160060  VT100     9600  8N
```

5.6.9 Time-sharing line options

In order to change line options, you must select a current line with one of the commands: SHOW LINDEF *n*; SHOW TERMINAL *n*; SET TT *n*; or SET TERMINAL *n*. You may then set any line option by issuing the appropriate TSGEN statement. For example:

```
!SHOW LINDEF 2
SET TT 2.
!  LINDEF   0.
   CMDFIL   LINE1.TSX
   FLAGS    NRMFLG+$AUTO+$FORM+$TAB
   NAME     <Charles G Br>
   SPEED    S19200,8.,NONE
   TRMTYP   VT200
!  LINEND
!TRMTYP VT100
!SHOW TRMTYP
   TRMTYP   VT100
```

The syntax for the FLAGS macro differs slightly from that used in TSGEN.MAC where the exclamation point is used to separate individual flags. In TSXMOD the "+" or "–" symbols separate the individual flags. For example:

```
!FLAGS NRMFLG+$SCOPE+$PAGE
!SHOW FLAGS
   FLAGS    NRMFLG+$SCOPE
!SHOW NRMFLG
NRMFLG = $DEFER+$LC+$PAGE+$ECHO
```

In addition to declaring a completely new set of flags for a line, you may also turn on or off individual flags. If the first flag is not preceded by a "+" or "–", then all flags are cleared, each desired flag must be included, and flags must be separated by a "+" sign. To make individual flag changes, simply use a "+" or "–" before the first flag. The existing flags are then not cleared, flags preceded by a "+" sign will be added, and flags preceded by a "–" sign will be deleted. For example:

```
!FLAGS -$SCOPE+$FORM
!SHOW FLAGS
   FLAGS    NRMFLG+$FORM
```

5.6.10 DETACH

Command files which are to be automatically executed on detached lines whenever TSX-Plus is started from RT-11 are declared with the DETACH macro. The total number of detached jobs which may be running at any given time is declared with the TBLDEF macro, which may not be changed by TSXMOD. Start-up detached jobs may be declared with the DETACH command by TSXMOD if free start-up detached job slots are available. For example:

```
!DETACH SY:STRTUP.TSX
```

To remove an automatically started detached job, use the DELETE command. Only the minimum number of characters needed to uniquely identify the detached job are necessary. Lower case characters are not automatically converted to upper case. For example:

```
!SHOW DETACH
DETACH <SY:DETACH.TSX>
DETACH <SY:STRTUP.TSX>
!DELETE DETACH SY:DE
!SHOW DETACH
DETACH <SY:STRTUP.TSX>
```

If parameters (up to 80 characters) are to be passed to the start-up detached job, the command file name and the parameters must be enclosed in angle brackets. For example:

```
!DETACH <SY:NEWDET.COM Tuesday 13:00>
```

Chapter 6

Patching and Building TSX–Plus Device Handlers

6.1 Device Handlers for TSX–Plus

TSX–Plus generally uses standard RT–11 XM device handlers, however, several of the device handlers as supplied with RT–11 require minor modifications to function correctly with TSX–Plus. The necessary handler modifications are supplied as SLP (Source Language Patch) files with TSX–Plus. *These SLP files have already been applied and are included in the dd.TSX handlers supplied with TSX–Plus.*

If you ordinarily need to make no modifications to the handlers supplied by Digital on your system, then you may use the handlers provided with the TSX–Plus distribution. Most common changes can be accommodated through device SET options. However, if you need to change the handlers supplied with RT–11, you may need to apply some patches before using them. An example of a change that requires regenerating a device handler is adding a second controller (vector and CSR) to the MT handler.

6.2 Patching device handlers

Handlers which do not require patches for use with TSX–Plus are: CR, DP, DS, DT, LP, NL, PC, RF, and RK. Some device handlers (DD, DL, DM, DU, DX, DY, LS, MM, MS, MT, MU and XL) require minor modifications to execute properly with TSX–Plus. When using the file structured magtape device handlers, the file structured module (FSM) as well as the device specific handlers (TJ, TM, and TS) must be patched.

The DM handler is not supported as a MAPIO device. The distributed DM handler performs 22-bit (as well as 18-bit) Q–Bus I/O with the DILOG DQ215 and Emulex SC02C controllers. A standard delay following each subsystem clear operation is included for some non-DEC controllers.

The dd.TSX device handlers provided with TSX–Plus have *already* been patched using the SLP files provided on the distribution media. These patched handlers have been assembled with the appropriate conditionals (TSX$P=1; MMG$T=1; TIM$IT=1; ERL$G=0, etc.) 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 cannot be accomplished with a device SET option. See the *RT–11 System User's Guide* for a list of the valid SET options for each device handler. Specification of SET options must be done under TSX–Plus in order to alter the SY:*dd*.TSX device handler. Device SETs will alter the disk copy of the handler; most will also alter the memory image for installed device handlers with the exception of code which is altered outside of block one in the handler (such as SET *dd* VECTOR). Device handlers installed with the NOSET option are only altered in their disk image by a SET command; the system must be restarted for that SET to take effect.

SLP files are Source Language Patch files which correct a source module by means of inserting, modifying or deleting code. In order to function properly, the correct version SLP file must be applied to the original RT-11 driver's source code. SLP files are named "*ddvvv*.SLP", where *dd* represents the two or three character source handler's name (e.g., TJ, TM, TS, FSM, etc.) and *vvv* represents the RT-11 version number (e.g., 501, 51B, 51C, 502). SLP files are included for the RT-11 releases from which the device handler was created. The distributed SLP files contain the release and version number of the driver to which they may be applied. This version number must match the version number embedded in the .MODULE declaration (second line) of the original device driver's source file.

If it is necessary to rebuild a device handler, copy the SLP files from the TSX-Plus distribution media. The SLP files have the extension .SLP. Copy the RT-11 handler source, named dd.MAC, from the RT-11 distribution media to a scratch working disk. Apply the patch using a command of the form:

```
SLP  dd.NEW=dd.MAC,ddvvv.SLP/A
```

where *dd* is the two or three character source file name, and *vvv* is the RT-11 version number. For more information on the use of the SLP utility, 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.*

6.3 Building device handlers

When building device handlers, it is necessary to set certain switches before assembling. These switches 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.

The conditional file used to build the device handlers supplied on the TSX-Plus distribution media is named TSXCND.MAC. It is included on the distribution media. Most of the conditionals are device specific, but some are standard for TSX-Plus; these are listed below. See the distributed TSXCND.MAC file for the device specific conditionals.

```
;  TSXCND - Conditional file used to build TSX-Plus device handlers.
;  See the TSX-Plus Installation Guide for more information on
;  patching and building device handlers for execution with TSX-Plus.
;
TSX$P       = 1             ;TSX-Plus support
BF          = 1             ;No SJ support
MMG$T       = 1             ;Memory management support
TIM$IT      = 1             ;Device time out
ERL$G       = 0             ;No Error log support
;
;  See the distributed TSXCND.MAC for device specific conditionals.
```

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 are included to specify device characteristics. For example the conditional:

```
DL$UN = 4.          ;Number of RL01/RL02 units
```

selects support for four units in the DL handler.

The following conditionals are specified for file structured MT support:

```
MT$FSM = 1          ;TM11 magtape file-structure support
MT$UN  = 2.         ;Number of TM11 magtape units
```

Additional MT conditionals could be set to override the default CSR and vector if necessary. The standard magtape CSR and vectors appear as follows:

```
MT$CSR = 172520  ;Define the default CSR address for the MT handler
MT$VEC = 224     ;Define the default vector for the MT handler
```

Refer to the *RT-11 System Generation Guide* for an entire list of device conditionals, their descriptions, and default values. These parameters are not required and will use a default value if left unspecified, except TSX$P and MMG$T which must be set to 1 for TSX-Plus.

Whether or not patching is required, most handlers may be built (when necessary) by the following commands:

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

where *dd* represents the two character device name.

Handlers which implement set code as internally managed overlays, require one additional switch specified on the link command. The following switch should be added to either of the above link specifications:

```
/BOUNDARY:512
```

When this switch is specified, LINK prompts foe the name of the boundary section. SETOVR is the appropriate response. The dialog will be as follows:

```
Boundary? SETOVR
```

The current handlers which implement overlayed set code are DU, DW, LS, and MU.

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

```
MACRO  TSXCND+FSM.NEW/OBJ
MACRO  TSXCND+td.NEW/OBJ
LINK/NOBITMAP/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 used here automatically appends the .TSX file extension. Since TSX-Plus uses handlers with the extension .TSX, the handlers must be linked with that extension rather than with the extension .SYS. This allows the TSX-Plus handlers to coexist on the same system disk with standard RT-11 handlers without conflict. Handlers for all devices included in your TSGEN DEVDEF list, including the system disk, must be on the system disk when TSX-Plus is started.

See the *TSX-Plus System Manager's Guide* for more information on writing device handlers for TSX-Plus. For specific information concerning special device handlers (CL, DM, IB, and VM) see the *TSX-Plus Programmer's Reference Manual.*

Chapter 7

TSX–Plus Installation and System Generation Hints

Various types of errors can occur and be reported during the installation process. The errors discussed in this section are those most commonly encountered when first installing TSX–Plus.

A complete listing of start-up error messages which may be reported by TSX–Plus are included in an appendix of the *TSX–Plus User's Reference Manual*. The *TSX–Plus User's Reference Manual* also lists those fatal system error messages which may be reported at any time during the operation of TSX–Plus; these are usually indicative of an unusual hardware condition.

If an error is reported during the start-up of TSX–Plus which is not described in those two appendices, it will have originated from RT–11 before TSX–Plus has taken control of the system; see the *RT–11 System Message Manual* for descriptions of RT–11 error messages.

Error message: ?PIP-F-Device full

There is not enough free space on the output disk to copy all distributed files. The system disk requires about 2000 free blocks: about 400 blocks for the executable files and handlers, about 1100 blocks for the swap file, and an additional 500 blocks for the spool file. The working disk requires about 1400 free blocks: about 800 blocks for distributed source and object files, and about 800 blocks for files created while generating TSX–Plus. Remove unnecessary files from the system and working disks and squeeze if necessary.

Error message: ?KMON-F-Insufficient memory (RT-11 V5)

The base of the monitor is too low to load the TSX.SAV program. Unload some resident device handlers and SET USR SWAP. TSX–Plus may not be started from the RT–11FB or RT–11XM monitors. The TSX.SAV program must be run from the RT–11 Single-Job or Baseline monitor which has **NOT** been *sysgenned* to include multi-terminal support. Copy the original RT–11 Single-Job monitor (RT11SJ.SYS) from your RT–11 distribution and boot it before running TSX–Plus.

Error message: ?TSX-F-(* error message displayed here ***)**

See the *TSX–Plus User's Reference Manual* for descriptions and remedies for fatal errors which occur during start-up of TSX–Plus.

Error message: ?TSX-F-Generated TSX system is too large

Too many features were included which require memory in the unmapped portion of TSX (low 40 Kb of memory). See Appendix A to determine the amount of low memory required for various system features. Regenerate the system with enough of these features reduced or eliminated to allow the system to start. Note that you cannot simply add the low memory sizes from this table and obtain the unmapped size of TSX. The table does not include various components which may not be removed. Rather, they are intended to identify the amount of change to expect when altering specific features. If you are using magtape, consider using a hardware-only version of the magtape handler. (See the description of the DEVDEF macro for information about using hardware magtape handlers.) Modify the TBLDEF macro to remove unnecessary sub-process and detached job slots.

No response from terminals

The operator console is established with vector=60 and CSR=177560. This and all other time-sharing terminals must have their hardware interface vector and CSR address settings declared to TSX-Plus by parameters in the file TSGEN.MAC before assembling and linking the TSX-Plus system. Every terminal and every device must be assigned its own unique CSR and vector addresses.

```
Error message:   ?TSX-F-Fatal system error @nnnnnn
                 (EEE-error message displayed here)
                 Arg. value = xxxxxx
                 Seg. value = yyyyyy
```

The most common causes of this error are incorrect device vector and address specifications. If the vector is incorrect, the error message will be "UEI-Interrupt occurred at unexpected location." If the address is incorrect, the error will be "KTP-Kernel mode trap." Review the default time-sharing line vectors and addresses above.

The vector and address assigned to the printer may be changed using the keyboard SET command after TSX-Plus has been started, but before printing anything; see the next paragraph. See the *TSX-Plus User's Reference Manual* for explanations of other fatal error messages.

Printer does not respond

The interface hardware for the printer must correspond to the address specified in the device handler LP.TSX. If your printer is not at the default vector and address, then after starting TSX-Plus, but before printing anything, use the keyboard handler SET options to set the correct vector and address.

```
SET LP VECTOR=nnn
SET LP CSR=nnnnnn
```

The printer vector and address must not conflict with those defined for any time-sharing terminal line. After setting the correct vector and address in the LP.TSX file, reboot the system and restart TSX-Plus with the command:

```
R TSX
```

Appendix A

System Sizing Calculations

A.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 40 Kb. 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.

These sizes are only relative to an existing system. The size of TSX–Plus's kernel code is *not* included in this table.

Effect of System Components on Overall System

System Component	Bytes in Unmapped Region	Bytes in Mapped Region
Each additional time-sharing line	290 + *input character silo buffer*	*Terminal input character buffer* + *output character buffer*
Each subprocess	274	0
Each detached job	254	0
CL Handler	150	2940
Each dedicated CL line	130 + *output ring buffer*	0
Each extra CL unit	46 + *output ring buffer*	0
DH11 or DH(Q,V)11	1028 + *num lines**40	0
Unmapped handler	Size of device handler	0
Mapped handler	26	Size of device handler
Job swapper	0	976
PLAS support	NGR*16	3460 + SEGBLK/8
Device spooling	*num spool devices* * (48 + 2 * *num of backup blocks*) + *num spool files* * 26	2642 + *num spool buffers* * 512 + *spool file size* / 8
Shared file record locking	2500	*total num of jobs**8 +MAXSF*24 +(18+2*MXLBLK)*MAXSFC
Shared file data caching	730	NUMDC*530
Generalized data caching	2028	CACHE*528
Directory caching	MAXCSH*18	NMFCSH*18
Inter-job message communication	0	1300 + (MAXMC+MAXMRB)*12 + MAXMSG*(6+MSCHRS)
Real-time support	496 + RTVECT*10	1034
Single line editor	0	4444
Process windowing	0	5720 -- 54*MAXWIN + *num active windows** (24+48**lines***columns*)
Program debugger	0	6304
I/O mapping (18-bit device support on 22-bit Q–Bus systems	22 + MIONBF*16	964 + MIONBF*(MIOBSZ*512)
System crash dump generator	0	1800
Performance analysis monitor	0	PMSIZE
Shared run-time	*num of run times**14	Size of run time system

A.2 Device Handler Sizes

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

Handler	Size (bytes)
CL	(See preceding table)
CR	800
CT	2350
DD	1250
DL	1450
DM	1450
DP	400
DS	300
DT	300
DU	400 root + 1300 extension
DW	900
DX	650
DY	800
DZ	700
LD	See preceding page
LP	450
LS	700
MM	4400 (file struct.)
MMHRD*	2800 (non-file struct.)
MS	4800 (file struct.)
MSHRD*	2100 (non-file struct.)
MT	4000 (file struct.)
MTHRD*	1300 (non-file struct.)
MU	3350 root + 2000 extension
MMHRD*	1000 (non-file struct.)
NL	100
PC	250
RF	250
RK	350
TT	See preceding page
VM	450
XL	1300

* Magtape hardware handlers may only be used when issuing non-file structured I/O (e.g. BUP). They will not function with file structured I/O (e.g. PIP).

Appendix B

Device CSR and Vector Address Table

System _____ CPU and Serial number _____
Memory installed _____
Peripherals installed _____
_____
Interface device(s) installed _____
_____

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

Index