TSX-Plus version 2.0

System Release Notes

The following new features and problem corrections are included in TSX-Plus version 2.0.

1. Support for real-time programs is now provided. The basic facilities provided include the ability to access the I/O page; the ability to connect real-time interrupts to completion routines; the ability to lock a job in memory. See the attached documentation for a full description of the real-time facility.

   If real-time program support is wanted, it must be selected when TSX-Plus is generated. The TSGEN parameter "RTVECT" is used to specify the maximum number of real-time interrupts that can be connected to completion routines. If RTVECT is set to a value greater than zero the real-time support facility is included in the generated TSX-Plus system. If RTVECT is set to zero, the real-time facility is not included in the generated system. Inclusion of the real-time facility causes the size of the generated system to be increased by about 800 bytes.

2. Support is now provided for shared run-time systems. This facility allows multiple jobs to map a portion of their virtual address space to a memory-resident shared run-time system or common data area which may be accessed by multiple jobs. The attached documentation provides complete information about how shared run-times are associated with jobs.

   If shared run-time systems are going to be used, they must be specified using the RTDEF macro in TSGEN. See the description in TSGEN of how this macro is used to specify information about shared run-times.

3. Support is now provided for caching of data blocks for shared files. This can provide a significant speed boost for COBOL-Plus and DBL ISAM files (the typical speed increase is a factor of 1.6). If you want to enable data caching, set the NUMDC TSGEN parameter to a non-zero value (the recommended value is 8.). The NUMDC parameter controls how many 512-byte buffers are set aside in system memory space for data caching. 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. It is recommended that NUMDC be set to zero if less than 192Kb of memory is installed on the system of if shared files are not heavily accessed.

4. A program called SETSIZ is now provided that can be used to store into a SAV file information about how much memory space should be allocated for the program when it is run. See the attached documentation for further information about use of the SETSIZ program. A command file named SETSIZ.COM is supplied that contains SETSIZ commands to set appropriate memory allocations for most of the standard system utility programs.

5. Device handler time-out support is now provided. Handlers may be generated with or without timer support -- TSX-Plus will handle either case.

6. A terminal "TAPE" mode is now provided. This is useful when input to a time-sharing line is being provided by paper-tape, cassette tape, floppy disk or other devices that responds to x-on/x-off control characters to start and stop transmission. Setting a line to TAPE mode has three effects: 1) An x-off character (ctrl-s) is sent by TSX-Plus whenever the line input buffer fills to the point that there are only 10 remaining free character positions; 2) An x-on character (ctrl-q) is sent by TSX-Plus when a program is about to enter an input wait state and an x-off has previously been sent; 3) Line-feed characters are ignored -- this is done so that each line of input can be terminated by a carriage-return line-feed pair. TAPE mode can be selected by including the "$TAPE" sysgen option with the line definition or by use of the "SET TT [NO]TAPE" keyboard command.

7. An EMT is now available to allow a program to set the amount of memory that will be allocated for the job. The form of the EMT is

```
EMT  375
```

with R0 pointing to the following argument area:

```
.BYTE     0,141
.WORD     .top-address
```

Where .top-address is the highest address to be included in the program space. On return from the EMT, R0 contains the highest address granted to the program. The size allocated to a job reverts to the size specified by the most recent "MEMORY" keyboard command when the job exits or chains.

8. The .GVAL EMT may be used to obtain TSX-Plus system values by using negative offset values. The following offset values may be used to obtain the specified values.

offset	value
-2.	Job number
-4.	"Lead-in" character used for terminal controlled options.
-6.	1 if job is privileged; 0 if non-privileged.
-8.	1 if PAR 7 is mapped to I/O page; 0 if PAR 7 is mapped to simulated RMON.
-10.	Project number job is logged on under.
-12.	Programmer number job is logged on under.

9. Terminal character buffers have been moved into the extended memory area. The result is that the size of the critical low-memory portion of TSX-Plus has been reduced and it is possible to generate a system with more options and lines without running into memory size problems.

10. The use of the "CORTIM" sysgen parameter has been changed. When a job is swapped into memory a timer is started, the job is not eligible to be

outswapped until CORTIM units of time have elapsed or the job has entered a state where it is waiting for some system resource other than I/O completion. Increasing the value of CORTIM reduces the job swapping rate at the expense of increasing the response time.

11. A program controlled terminal option has been added to control whether or not line-feed characters are echoed when a carriage-return character is received. The lead-in-"Y" function causes TSX-Plus to NOT echo line feed characters when a carriage return is received. It still echoes the carriage return and it passes both a carriage return and a line feed to the program. The "Z" function reenables normal line-feed echoing following carriage-return.

12. SET commands have been provided to allow users with operator command privilege to change some system parameter values during system operation. The form of these SET commands is

    SET parameter value

    Where "parameter" is the name of one of the system parameters and "value" is the decimal value to which the parameter is to be set (same units as specified in TSGEN). The parameters which may be set are QUAN1, QUAN1A, QUAN2, QUAN3, CORTIM, NUMDC.

13. A SHOW MOUNTS command is available to display the names of all devices that have been mounted.

14. A SHOW RUN-TIMES command is available to display the names of shared run-time systems.

15. The privilege level of detached jobs is now handled as follows: Detached jobs initiated automatically when the system starts as a result of being specified in TSGEN run with operator privilege. Detached jobs started by time-sharing users are given operator privilege only if the user initiating them has privilege. Note that detached jobs which have operator privilege may use the real-time support facilities. In many cases detached jobs may be the best way to implement real-time control programs.

16. Support is now provided for user programs to use the IOT instruction.

17. The .PROTECT EMT has been changed to return the no-error status (carry-flag cleared) rather than error status. It still has no actual effect.

18. Floating point traps are now correctly handled.

19. The use of double ctrl-c to abort nested command files has been improved. Double ctrl-c should now abort all executing command files.

20. FILEX now runs correctly under TSX-Plus.

21. The INIT/VOLUMEID:ONLY command now works.

22. The CREATE keyboard command is now supported.

23. It is now possible for us to provide a "demonstration" version of TSX-Plus which will self-destruct after a half-hour or so of execution time.

24. The FORMAT program has been examined to ascertain the feasibility of making it run under TSX-Plus. Unfortunately this does not look possible; FORMAT does all of its I/O and device control by storing directly into the device status registers.

25. If you rebuild the DL handler (DL.TSX) you must install a patch. The DEC patch sequence number is 6.5.1 and was issued 8-Oct-80. The text of the patch is listed below:

```
-130,,/;003/
<tab>CMPB<tab>R2,#FN$SIZ
<tab>BEQ<tab>102$
-137,137,/;003/
102$:<tab>MOV<tab>PC,R3
/
```

Where "<tab>" stands for the tab character. This patch is applied to the source of the handler using the SLP program. Assuming you have created a file named DL.003 containing this patch, you can use the following commands to apply the patch to the handler:

```
.R SLP
*DL.MAC=DL.MAC,DL.003
*^C
```

The version of the DL handler on the TSX-Plus distribution disk (DL.TSX) already contains this patch.

26. The REDUCE program can be made to run under TSX-Plus by applying the following patch to it using the PATCH program:

```
1066/   5000    12700
1070/   5740    177777
```

This patch will not affect its operation under RT-11.

SETSIZ Program

The SETSIZ program can be used to store into a SAV file information about how much memory space should be allocated for the program when it is executed.

There are three ways that the amount of memory allocated to a job can be controlled:

1. The TSX-Plus EMT with function code 141 (described in chapter 7) may be used by a running program to dynamically set the job's size.
2. If a size is specified in a SAV file (by use of the SETSIZ program) the specified amount of memory is allocated for the program when it is started.
3. If no size is specified in the SAV file, the size specified by the last MEMORY keyboard command is used.

Note that the .SETTOP EMT does not alter the amount of memory space allocated to a job but can be used by a running program to determine the amount of memory allocated.

The effect of the SETSIZ program is to store in location 56 of block 0 of the SAV file the number of K-words of memory to allocate for the program when it is running (the LINK "/K:n" switch can also be used to do this). This value has no effect when the SAV file is run under RT-11 but cause TSX-Plus to allocate the specified amount of memory when starting the program.

If a size value is specified in a SAV file it takes precedence over the size specified by the last MEMORY keyboard command. The TSX-Plus EMT with function code 141 may still be used to dynamically alter the memory allocation while the program is running.

Most programs allocate memory in two ways: 1) A static region that includes the program code and data areas of fixed size; 2) A dynamic region that is allocated above the static region -- usually the .SETTOP EMT is used to determine how much dynamic space is available to the program. The size of the static region for a program is fixed at link time. If the program is overlayed the static region includes space for the largest overlay segment as well as the program root. Location 50 in block 0 of the SAV file is set by the linker to contain the address of the higest word in the static region of the program.

The amount of memory space to allocate for a SAV file can be specified to the SETSIZ program in either of two ways: 1) As the total amount of memory for the program which includes space for the static plus dynamic regions; 2) As the amount of memory for the dynamic region only, in which case SETSIZ automatically adds the size of the static region.

0.1 -- Running the SETSIZ program

The SETSIZ program is started by use of the command

```
.R SETSIZ
```

it responds by printing an asterisk to prompt for a command line. The form of the command line is

```
*filespec/switch:value
```

Where "filespec" is a file specification of the standard form dev:name.ext with the default device being DK and the default extension being SAV.

If a file specification is entered without a switch the effect is to cause SETSIZ to display information about the size of the SAV file; the SAV file is not altered.

Examples:

```
.R SETSIZ
*TSTPRG
Static program size is 22Kb
Allocation size is 28Kb
*SY:PIP
Static program size is 7Kb
Allocation size is 18Kb
*PROG1
Static program size is 31Kb
No allocation is specified
```

0.2 -- Setting total allocation for a SAV file

The "/T" switch is used to specify the total amount of memory space to be allocated for a program when it is run. The form of the /T switch is "/T:value." where "value" is the number of K-bytes of memory to allocate. Note that a decimal point must be specified with the value if it is entered as a decimal value.

If the "/T" switch is used without a value, the effect is to clear the TSX-Plus size allocation information in the SAV file.

```
.R SETSIZ
*SY:PIP/T:18.
*TSTPRG/T:32.
*PROG1/T
```

0.3 -- Setting amount of dynamic memory space

The "/D" switch is used to specify the amount of dynamic memory space to be reserved for a program. The SETSIZ program calculates the total amount of space to allocate for the program by adding the static size (stored in location 50 of the SAV file by LINK) to the specified dynamic size. The form of the /D switch is "/D:value." where "value" is the number of K-bytes of dynamic memory space to

reserve. If a program does not use any dynamic memory space, the /D switch may be used without an argument value to cause the total memory space allocation to be set equal to the static size of the program. FORTRAN programs use dynamic space for I/O buffers and the exact amount required depends on the number of I/O channels used. However, 4Kb of dynamic memory space seems to be adequate for most FORTRAN programs.

Examples:

```
.R SETSIZ
*SY:PIP/D:11.
*TSTPRG/D:4.
*PROG1/D
```

1. SHARED RUN-TIME SYSTEM SUPPORT

TSX-Plus provides a facility that allows one or more shared run-time systems or data areas to be mapped into the address space of multiple TSX-Plus time-sharing jobs. There are two primary uses of this facility:

1. Memory space can be saved by having multiple jobs that use the same run-time system access a common copy rather than having to allocate space within each job for a copy.

2. Programs can communicate with each other through the use of a common shared memory region to which all of the communicating jobs have direct access.

To use this facility, information about all of the shared run-time systems must be declared when the TSX-Plus system is generated. During system initialization the shared run-time system files are opened and read into memory. These shared run-time systems remain in memory as long as the system is running and are never swapped out of memory even if there are no jobs actively using them.

The EMT's described below can be used to associate one or more shared run-time systems with a job. When such an association is made, a portion of the job's virtual memory space is mapped so as to allow access to part or all of one or more run-time systems.

1.1 Associating a run-time system with a job

The following EMT is used to associate a shared run-time system with a job. The form of the emt is:

```
EMT  375
```

with R0 pointing to the following argument area:

```
.BYTE     0,143
.WORD     .name-pointer
```

Where .name-pointer is the address of a two-word cell containing in RAD50 form the six character name of the shared run-time system file. This name corresponds to the file name which was specified for the run-time system in TSGEN. If the name pointer value is zero, the effect of the EMT is to disassociate all shared run-time systems from the job and to reestablish normal memory mapping for the job.

If the run-time system whose name is specified with this EMT is not recognized, the Carry-flag is set on return from the EMT and an error code of 1 is returned.

The effect of this EMT is to associate a particular shared run-time system with the job. However, this EMT does not affect the memory mapping for the job

or make the run-time system visible to the job; that is done by the EMT described below. If some other run-time system has been previously mapped into the job's region, that mapping is unaffected by this EMT. Thus it is possible to have multiple run-time systems mapped into the job's region by associating and mapping them one at a time into different regions of the job's virtual memory space.

1.2 Mapping a run-time system into a job's region

Once a shared run-time system has been associated with a job by use of the previous EMT, the run-time system (or a portion thereof) can be made visible to the job by mapping it into the job's virtual address region. The form of the EMT to do this is:

```
EMT  375
```

with R0 pointing to the following argument area

```
.BYTE     1,143
.WORD     .par-region
.WORD     .run-time-offset
.WORD     .mapped-size
```

If an error is detected during execution of the EMT, the carry-flag is set on return and the following error codes indicate the nature of the error:

error code	meaning
1	There is no run-time system associated with the job

The parameter .par-region is a number in the range 0 to 7 that indicates the Page Address Register (PAR) that is to be used to access the run-time system. The PAR number selects the region of virtual memory in the job that will be mapped to the run-time system. The following correspondence exists between PAR numbers and virtual address regions within the job:

PAR	Address Range
0	000000 - 017777
1	020000 - 037777
2	040000 - 057777
3	060000 - 077777
4	100000 - 117777
5	120000 - 137777
6	140000 - 157777
7	160000 - 177777

The .run-time-offset parameter specifies which portion of the run-time system is to be mapped into the PAR region. The .run-time-offset specifies the number of the 64-byte block within the run-time system where the mapping is to begin. This makes it possible to access different sections of a run-time system at different times or through different regions. The .mapped-size parameter

specifies the number of 64-byte blocks to be mapped. If this value is larger than can be contained within a single PAR region, multiple PAR regions are automatically mapped as necessary to contain the entire specified section of the run-time.

This EMT only affects the mapping of the PAR region specified in the argument block (and following PAR's if the size so requires). It does not affect the mapping of any other PAR regions that may previously have been mapped to a run-time system. Thus a job may have different PAR regions mapped to different sections of the same run-time system or to different run-times. Real-time programs may map PAR 7 to the I/O page and map other PAR regions to shared run-time systems.

The memory size of a job is not affected by the use of the shared run-time control EMT's. That is, mapping a portion of a job's virtual address space to a shared run-time system neither increases nor decreases the size of memory occupied by the job. If a job's size is such that a portion of its normal memory is under a PAR region that is mapped to a run-time system, that section of its normal memory becomes inaccessible to the job as long as the run-time system mapping is in effect but the memory contents are not lost and may be reaccessed by disassociating all run-time systems from the job.

Note that any of the PAR regions may be mapped to a run-time system including PAR 7 (160000-177777) which is normally not accessible to a job.

1. REAL-TIME PROGRAM SUPPORT

TSX-Plus provides a real-time program support facility that allows multiple real-time programs to be run concurrently with normal time-sharing operations. 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 completion routines. These real-time completion routines run at user-selectable real-time priority levels that preempt execution of normal time-sharing jobs.

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

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

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

6. 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).

These real-time support features are only available if the real-time support facility is included in TSX-Plus when the system is generated.

A program must have operator privilege to use any of the real-time features described in this chapter. The real-time facilities are available to both normal jobs controlled by time-sharing lines and to detached jobs. Note that detached jobs that are specified during system generation for automatic startup run with operator privilege; detached jobs started by time-sharing users have operator privilege only if the user starting them does.

1.1 Accessing the I/O page

A basic facility required by most real-time programs is the ability to access the PDP-11 I/O page (160000-177777) which contains the device control and status registers. A normal TSX-Plus time-sharing job does not have access to this page; its virtual memory space in the range 160000 to 177777 is instead mapped to a simulated RT-11 RMON (resident monitor). This is done so that programs that directly access offsets in RMON will run correctly.

A TSX-Plus real-time program can access the I/O page in one of two ways: It can cause the program's virtual address region in the range 160000 to 177777 to be mapped directly to the I/O page so that it can directly access device

registers; or it can leave the virtual address range mapped to the simulated RMON and use a set of EMT's to peek, poke, bit-set and bit-clear registers in the I/O page. It is much more efficient to directly access the device control registers by mapping the I/O page into the program's virtual address region than to use EMT's to perform each access. However, this technique will not work if the program must also directly access offsets inside RMON. The correct way for a program to access RMON offsets is to use the .GVAL EMT which will work even if the I/O page is mapped into the program region.

1.1.1 EMT to map the I/O page into the program space

The following EMT can be used to cause the program's virtual address region in the range 160000 to 177777 to be mapped to the I/O page.
The form of the EMT is:

```
EMT  375
```

with R0 pointing to the following argument area:

```
.BYTE     5,140
```

The mapping set up by this EMT remains in effect until the following EMT is executed to remap to RMON or the program exits. Note that completion routines run with the same memory mapping as the main-line code of the job.

1.1.2 EMT to remap the program region to the simulated RMON

The following EMT can be used to cause the virtual address region of the job in the range 160000 to 177777 to be mapped to a simulated RMON.
The form of the EMT is:

```
EMT  375
```

with R0 pointing to the following argument area:

```
.BYTE     6,140
```

1.1.3 EMT to peek at the I/O page

The following EMT can be used to access a word in the I/O page without requiring the job's virtual address region to be mapped to the I/O page.
The form of the EMT is:

```
EMT  375
```

with R0 pointing to the following argument area:

```
.BYTE     1,140
.WORD     .address
```

where .address is the address of the word in the I/O page to be accessed. On return from the EMT the contents of the specified word in the I/O page are returned in R0. If an error is detected during the execution of the EMT the carry-flag is set on return and location 52 contains a code that indicates the nature of the error.

error code	meaning
0	Real-time support not genned in or job does not have operator privilege.

1.1.4 EMT to poke into the I/O page

The following EMT can be used to store a value into a cell in the I/O page.
The form of the EMT is:

```
EMT  375
```

with R0 pointing to the following argument area:

```
.BYTE     2,140
.WORD     .address
.WORD     .value
```

where .address is the address of the cell in the I/O page to be stored into and .value is the value to be stored. If an error is detected during the execution of the EMT the carry-flag is set on return and location 52 contains a code that indicates the nature of the error.

error code	meaning
0	Real-time support not genned in or job does not have operator privilege.

1.1.5 EMT to bit-set a value into the I/O page

The following EMT can be used to perform a bit-set (BIS) operation into a cell in the I/O page.
The form of the EMT is:

```
EMT  375
```

with R0 pointing to the following argument area:

```
.BYTE     3,140
.WORD     .address
.WORD     .value
```

Where .address is the address of the cell to be stored into and .value is the value that will be bit-set into the cell. If an error is detected during the execution of the EMT the carry-flag is set on return and location 52 contains a code that indicates the nature of the error.

error code	meaning
0	Real-time support not genned in or job does not have operator privilege.

1.1.6 EMT to do a bit-clear into the I/O page

The following EMT can be used to perform a bit-clear (BIC) operation into a cell in the I/O page.
The form of the EMT is:

```
EMT  375
```

with R0 pointing to the following argument area:

```
.BYTE     4,140
.WORD     .address
.WORD     .value
```

Where .address is the address of the cell in the I/O page to be accessed and .value is the value to be bit-cleared into the specified cell. If an error is detected during the execution of the EMT the carry-flag is set on return and location 52 contains a code that indicates the nature of the error.

error code	meaning
0	Real-time support not genned in or job does not have operator privilege.

1.2 Connecting a real-time interrupt to a completion routine

The TSX-Plus real-time support facility allows a program to connect a real-time interrupt to a completion routine. If this is done, TSX-Plus causes the completion routine to be executed each time the specified interrupt occurs.

The total number of real-time interrupts that can be connected to completion routines for all active real-time jobs is specified when TSX-Plus is generated. It is possible for several interrupts to be connected to the same completion routine in a job but it is illegal for more than one job to try to connect to the same interrupt. When an interrupt-driven completion routine is entered, R0 contains the address of the interrupt vector that caused the completion routine to be executed.

An execution priority may be specified for each completion routine. This is not the same as the hardware selected priority of the interrupt. All completion routines are synchronized with the job and run at hardware priority level 0. The completion routine priority is used by TSX-Plus to schedule the completion routines for execution. The available priority levels are 0 to 7; the higher the number, the higher the priority. The execution of a real-time completion routine for one job will be interrupted and suspended if an

interrupt occurs that causes a higher-priority completion routine for another job to be queued for execution. However, a completion routine for a given job will never be interrupted to run another completion routine for the same job even if a higher-priority completion routine is pending.

Completion routine priorities in the range 1 to 7 are classified as "real-time" priorities. They are higher than the execution priorities given to any time-sharing jobs and will always preempt the execution of time-sharing jobs. A completion routine running at one of these priorities is not "time-sliced" in the normal time-sharing fashion but rather is allowed to run continuously until one of the following events occurs: 1) The completion routine completes its execution and returns; 2) A higher priority completion routine interrupts its execution -- it is reentered when the higher priority routine exits; 3) The completion routine enters a system wait state such as waiting for an I/O operation to complete or waiting for a timed interval. If a completion routine enters a wait state, it relinquishes its real-time priority and subsequently runs at the priority of a normal time-sharing job.

Note that since a completion routine running at real-time priority is not time-sliced, it can indefinitely block the execution of all lower priority jobs and time-sharing users.

Completion routine priority 0 (zero) operates differently from priorities 1 to 7. Priority 0 is not classified as a "real-time" priority but rather as a very high "normal" priority. The effect of this is that real-time completion routines with priority 0 interrupt normal time-sharing jobs but are time-sliced in the normal fashion and loose their high-priority if they execute longer than QUAN1A (sysgen parameter) length of time.

Jobs that have real-time, interrupt-driven completion routines need not necessarily be locked in memory. If an interrupt occurs while the job is swapped out of memory, it is scheduled for execution like any other job and swapped in before the completion routine is executed.

When a real-time interrupt occurs, a request is placed in a queue to execute the appropriate completion routine. If the interrupt occurs again before the completion routine is entered, another request is placed in the queue so the completion routine will be invoked twice. A fatal TSX-Plus system crash occurs if an interrupt occurs and there are no free completion routine request queue entries.

When a real-time completion routine completes its execution, it must exit by use of a RTS PC instruction (rather than RTI).

The form of the EMT used to connect an interrupt to a completion routine is:

```
EMT  375
```

with R0 pointing to the following argument area:

```
.BYTE     11,140
.WORD     .interrupt-vector
.WORD     .completion-routine
.WORD     .priority
```

Where .interrupt-vector is the address of the interrupt vector, .completion-routine is the address of the completion routine and .priority is the execution priority for the completion routine which must be in the range 0 to 7. If an error is detected during the execution of the EMT the carry-flag is set on return and location 52 contains a code that indicates the nature of the error.

error code	meaning
0	Real-time support not genned in or job does not have operator privilege.
1	Maximum number of interrupts already in use.
2	Some other job is already connected to interrupt.

1.3 Releasing an interrupt connection

A connection between a real-time interrupt vector and a completion routine remains in effect until the job exits or the following EMT is executed to release the connection. The form of the EMT is:

```
EMT  375
```

with R0 pointing to the following argument area:

```
.BYTE     12,140
.WORD     .interrupt-vector
```

Where .interrupt-vector is the address of the interrupt vector whose connection is to be released. If an error is detected during the execution of the EMT the carry-flag is set on return and location 52 contains a code that indicates the nature of the error.

error code	meaning
0	Real-time support not genned in or job does not have operator privilege.

1.4 Locking a job in memory

In time-critical real-time applications where a program must respond to an interrupt with minimum delay, it may be necessary for the job to lock itself in memory to avoid program swapping. This facility should be used with caution since if a number of large programs are locked in memory there may not be enough space left to run other programs.

TSX-Plus provides two program locking facilities: The first moves the program to the low end of memory before locking it; this is done to avoid fragmenting available free memory. This type of lock should be done if the program is going to remain locked in memory for a long period of time. However, this form of locking is relatively slow since it may involve program swapping. The second locking facility simply locks the program into the memory

space it is occuping when the EMT is executed without doing any repositioning. This EMT has the advantage that it is extremely fast but the free memory space may be non-contiguous.

The form of the EMT used to lock a program in low memory (repositioning if necessary) is:

```
EMT  375
```

with R0 pointing to the following argument area:

```
.BYTE     7,140
```

If an error is detected during the execution of the EMT the carry-flag is set on return and location 52 contains a code that indicates the nature of the error.

error code	meaning
0	Real-time support not genned in or job does not have operator privilege.

The form of the EMT used to lock a job in memory without repositioning it is:

```
EMT  375
```

with R0 pointing to the following argument area:

```
.BYTE     13,140
```

If an error is detected during the execution of the EMT the carry-flag is set on return and location 52 contains a code that indicates the nature of the error.

error code	meaning
0	Real-time support not genned in or job does not have operator privilege.

1.5 Unlocking a job from memory

When a job locks itself in memory, it remains locked until the job exits or the following EMT is executed.
The form of the EMT used to unlock a job from memory is:

```
EMT  375
```

with R0 pointing to the following argument area:

```
.BYTE     10,140
```

If an error is detected during the execution of the EMT the carry-flag is set on return and location 52 contains a code that indicates the nature of the error.

error code	meaning
0	Real-time support not genned in or job does not have operator privilege.

1.6 Suspending/Resuming program execution

The RT-11 standard .SPND and .RSUM EMT's are used by TSX-Plus real-time jobs to suspend and resume their execution. Frequently a real-time job will begin its execution by connecting interrupts to completion routines and doing other initialization and then will suspend its execution while waiting for a device interrupt to occur.

The .SPND EMT causes the main-line code in the job to be suspended. Completion routines are not affected and execute when interrupts occur. If a completion routine executes a .RSUM EMT, the main-line code will continue execution at the point following the .SPND when the completion routine exits. Refer to the RT-11 programmer's reference manual for further information about the use of .SPND and .RSUM.

The form of the .SPND EMT used to suspend execution of a job is:

```
.SPND
```

The form of the .RSUM EMT used within a completion routine to cause the main-line code to continue executing from a .SPND is:

```
.RSUM
```

1.7 Converting a virtual address to a physical address

When controlling devices that do direct memory access (DMA), it is necessary to be able to obtain the physical memory address (18-bit) that corresponds to a virtual address in the job. Note that a job should lock itself in memory before performing this EMT.
The form of the EMT is:

```
EMT  375
```

with R0 pointing to the following argument area:

```
.BYTE     0,140
.WORD     .virtual-address
.WORD     .result-buffer
```

Where .virtual-address is the virtual address that is to be converted to a physical address and .result-buffer is the address of a two word area that is to recieve the physical address. The low-order 16-bits of the physical address are stored in the first word of the result buffer and the high-order 2-bits of the physical address are stored in bit positions 4-5 of the second word of the result buffer.

If an error is detected during the execution of the EMT the carry-flag is set on return and location 52 contains a code that indicates the nature of the error.

error code	meaning
0	Real-time support not genned in or job does not have operator privilege.

1.8 Specifying a program-abort I/O reset list

The standard RT-11 .DEVICE EMT is used by TSX-Plus real-time jobs to specify a list of device control registers to be loaded with specified values when the job terminates. This feature is useful in allowing real-time devices to be turned-off if the real-time control program aborts. The TSX-Plus .DEVICE EMT has the same form and options as the standard RT-11 .DEVICE EMT. See the RT-11 programmer's reference manual for further information.

Note that the connection between real-time interrupts and completion routines are automatically dropped when a job terminates.

1.9 Adapting real-time programs to TSX-Plus

The following points should be kept in mind when converting an RT-11 real-time program for use under TSX-Plus.

1. The I/O page (160000-177777) is not accessible to the program unless the program executes the TSX-Plus real-time EMT that maps the job's virtual region to the I/O page.

2. If the program's virtual region 160000 to 177777 is mapped to the I/O page, the program must do .GVAL EMT's to access offsets in the simulated RMON.

3. Real-time interrupts are connected to completion routines by use of the TSX-Plus real-time EMT for that purpose. The program should not try to connect interrupts by storing into the interrupt vector cells.

4. The .PROTECT EMT is a no-op under TSX-Plus and always returns with the carry-flag cleared.

5. Completion routines connected to real-time interrupts should exit by use of a RTS PC instruction rather than RTI.

6. Programs that require very rapid response to interrupts should lock themselves in memory.

7. The program must be sure no real-time interrupts occur unless a completion routine is connected to the interrupt. If a real-time interrupt occurs

with no associated completion routine, a fatal TSX-Plus system crash occurs and the crash "argument value" displays the address of the vector of the interrupt.

8. A higher priority real-time completion routine for one job will interrupt a lower priority completion routine being executed by another job but will not interrupt a lower priority completion routine being executed by the same job.

9. A real-time completion routine running at priority 1 to 7 is not time-sliced and will lock out all time-sharing users until it completes its processing or enters a system wait state.