RSX-11D SPEC

TO: RSX-11D Distribution = no online compilation

FROM: H. Krejci

DATE: 16 Jun 72

SUBJ: RSX-11D SYSTEM OVERVIEW

DOC: 130-101-035-00

The material included in this functional specification, including but not limited to, instruction times and operating speeds is for information purposes only. All such material is subject to change without notice. Concequently DEC makes no claim and shall not be liable for its accuracy.

INTRODUCTION

RSX-11D is a Real Time Operating System for the PDP-11/45, that is philosophically similar to RSX-15 with the addition of: Fixed and/or dynamic partitioning, re-entrant libraries, output spooling, Checkpointing, and run-time Partition selection. A major difference from RSX-15 is the use of "Event Flags" instead of "Event Variables", and the ability to WAITFOR a logical combination of events.

SYSTEM DESCRIPTION

RSX-11D is an event driven, fixed/variable partition, disk based, software priority multi-programming system that provides flexible scheduling, program protection, priority queued I/O, and fast interrupt response.

The system is designed for use in industrial process control and laboratory applications where disk storage of programs and data, efficient and convenient scheduling of operations, and rapid response to interrupts is required.

The basic program unit under RSX-11D is called a "Task" and consists of a program or set of programs that have been written in FORTRAN and/or PDP-11 assembly language. Relocatable object modules are created either on-line or off-line, and are "Installed" into an RSX-11D system on-line. This on-line process results in the recording of the Task on the system disk in absolute memory image form (ready for execution).

PARTITIONS

Partitions are areas of contiguous real memory that are used for Task execution. There are two modes of Partition usage: (1) "User Control" where only one task at a time may occupy the Partition, and (2) "System Control" where the system controls allocation of memory (within the Partition) for execution of one or more Tasks. The name, base address, size, and mode of each Partition is specified at System Configuration time, and cannot be changed on-line. Tasks are installed to run in a particular Partition, but, upon specific request, may run in any Partition that is large enough.

DISK USAGE

Part of the executive code is a RF and/or RK Disk Driver which provides Storage Allocation and Block Transfer functions for Privelaged requestors.

When a Task Image is built, a disk area is allocated to contain the Task's load image, and, if the Task is declared "Checkpointable", a swapout area is also allocated. The Task Image is written such that a single transfer can be used to load its image into a Partition. Except for Global Common and Resident Library Routines (which are not a part of the disk image), a Task requires contiguous real memory.

MULTIPROGRAMMING

Effective multiprogramming depends upon multiple memory residency of Tasks, and upon Tasks spending some of their residency waiting for I/O completion, waiting for synchronization with other Tasks, or in some way being unable to continue execution. Thus, while one or several Tasks are waiting, another Task may utilize the central processor's resources.

Under RSX-11D, Tasks are run at a software priority level ranging from a low of one through a high of 255, and the highest priority Task capable of execution is granted the central processor resources.

When a Task becomes ready to execute, and is of a higher priority than the currently executing Task, the lower priority Task is interrupted and the higher priority Task is allowed to run. Execution of the interrupted Task will continue when it once again becomes the highest priority Task capable of execution. The environment of an interrupted Task is preserved, and, except for elapsed time, interruption is transparent to an interrupted Task.

This multiprogramming normally applies only to memory-resident Tasks, i.e., once a Task is in memory, it is normally allowed to run to completion in a multiprogramming fashion even if its memory becomes required for the execution of a higher priority Task. However, when it is desirable to free a Partition for execution of a higher priority Task, a Task may be declared "checkpointable" when it is installed. Checkpointable Tasks are swapped-out when their Partition is required for a higher priority Task, and swapped-in when they once again become the highest priority Task requiring its Partition.

Normally, a Task is brought into memory only upon a request for its execution, and several Tasks may use the same memory. However, when desirable, a Task may be "fixed-in-memory" permitting faster response to requests for execution, but dedicating a Partition, or part of one, to a single Task.

SIGNIFICANT EVENTS AND EVENT FLAGS

A Task is considered "active" from the time its execution is requested until the time it has EXITed, and the system maintains a priority ordered list of active Tasks called the Active Task List. The system is driven by this list in the following way. A Significant Event is a condition that is declared by a program that recognizes that an event of significance has occurred. Whenever a Significant Event is declared, any executing Task is interrupted and the Active Task List is scanned from the top examining the status of Tasks in decending priority order until a Task capable of execution is found. Execution of the Task is then initiated, or continued, until either: 1) the Task exits, 2) the Task must wait for another event (viz., I/O completion), or 3) a Significant Event occurs and a higher priority Task is capable of execution. Task switching occurs only as a result of a Significant Event, and

Significant Events are declared when events of significance occur, i.e., RSX-11D is an event driven system.

Associated with Significant Events are flags called Event Flags. Declaration of a Significant Event indicates that something has happened, and the setting of a particular Event Flag indicates what has happened. For example, upon completion of I/O requests, Handler Tasks normally set a requestor indicated Event Flag and declare a Significant Event. If the requesting Task has instructed the system that it could not execute until the Event Variable had been set, Tasks of lower priority could be run because scans of the Active Task List could pass the Task waiting for I/O completion until the Significant Event was declared as a result of the I/O completion.

Each Task has access to sixty-four (64) Event Flags of which thirty-two (32) are unique to each Task, and thirty-two are common to all tasks.

SYSTEM TRAPS

The ability to service certain conditions without continuously testing for their existence is provided via "System Traps". These Traps consist of a linkage method to optional in-Task trap service routines. The service routines are installed as a part of the Task, limited by the same restrictions as the Task, and run at the Task's priority as a result of a System Trap condition. This facility also provides a means of responding to the execution of privileged instructions and non-RSX-11D EMTs.

If the system is not explicitly notified of the existence of a System Trap service routine, the System Trap will not occur.

ON-LINE TASK DEVELOPMENT

Program development under RSX-11D is provided as a single-stream Batch capability. Program units may be assembled or compiled (Fortran IV), Task images may be built and installed or removed, and file manipulation utility operations may be performed.

THE MONITOR CONSOLE ROUTINE

Operator interface to the system is provided by a facility called the "MCR" for Monitor Console Routine.

MCR dialogue is established by typing a ^C on a TTY. This causes an MCR dispatch Task to run which outputs an "MCR>" prompting symbol and reads a line of command input. The command input line indicates what function is to be performed and contains parameters when necessary. The dispatch Task causes an MCR Function Task to run, which performs the requested function. There is one MCR Function Task for every MCR Function.

A typical system might have MCR functions to provide system status, perform Task scheduling, change Logical Unit Assignments, etc.

Since normal RSX-11D Tasks are used to implement MCR Functions, special purpose functions to provide added flexibility or convenience for a particular application or installation may be easily developed and added.

SYSTEM DIRECTIVES

System Directives or Directives are instructions to the system to perform an indicated operation. Directives are implemented as EMT's and supported under FORTRAN by library subroutines.

Directives allow Tasks to schedule other tasks, measure time intervals, queue I/O requests, suspend execution in various ways, change logical unit assignments, exit, and perform other utility operations.

I/O HANDLERS

With the exception of the System Disk Driver(s), which are a part of the executive, all I/O is supported by "I/O Handler Tasks". These Tasks differ from most other Tasks in that (1) they contain an interrupt service routine, (2) they run with additional privilege, and (3) that a naming convention exists (viz., the Line Printer Handler Task is named "LP";;,,;").

I/O Handler Tasks are are loaded into, or unloaded from, memory by MCR commands.

I/O requests are queued for each unit by priority (usually requestor task priority), and Handler Tasks pick requests from the top of request queues. Thus, preferential service is given to high priority requestors. However, when appropriate, Handler Tasks accept "attach unit to Task" requests and then de-queue only requests from the attached Task. This continues until a "detach unit from Task" request is de-queued, which causes requests to be de-queued by priority (off the top) once again.

The interface between an I/O Handler Task and the RSX-11D system is accomplished by Directives and by re-entrant system subroutines (viz., to attach, detach, and de-queue), and the major effort in developing an RSX-11D Handler Task is in driving the device, and not in completing an intimate interface to a host system.