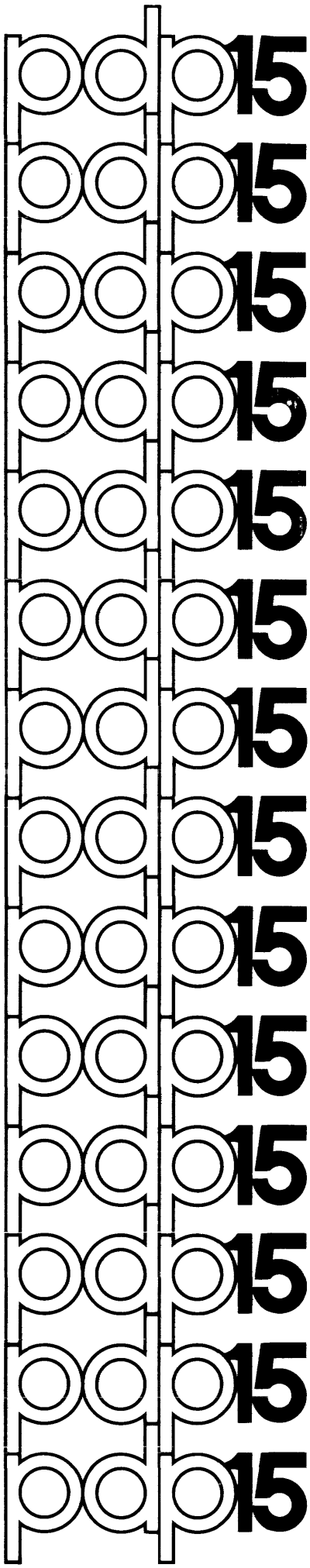
digital



rsx15

real time executive reference manual

digital equipment corporation

DEC-15-GRQA-D

R S X - 1 5

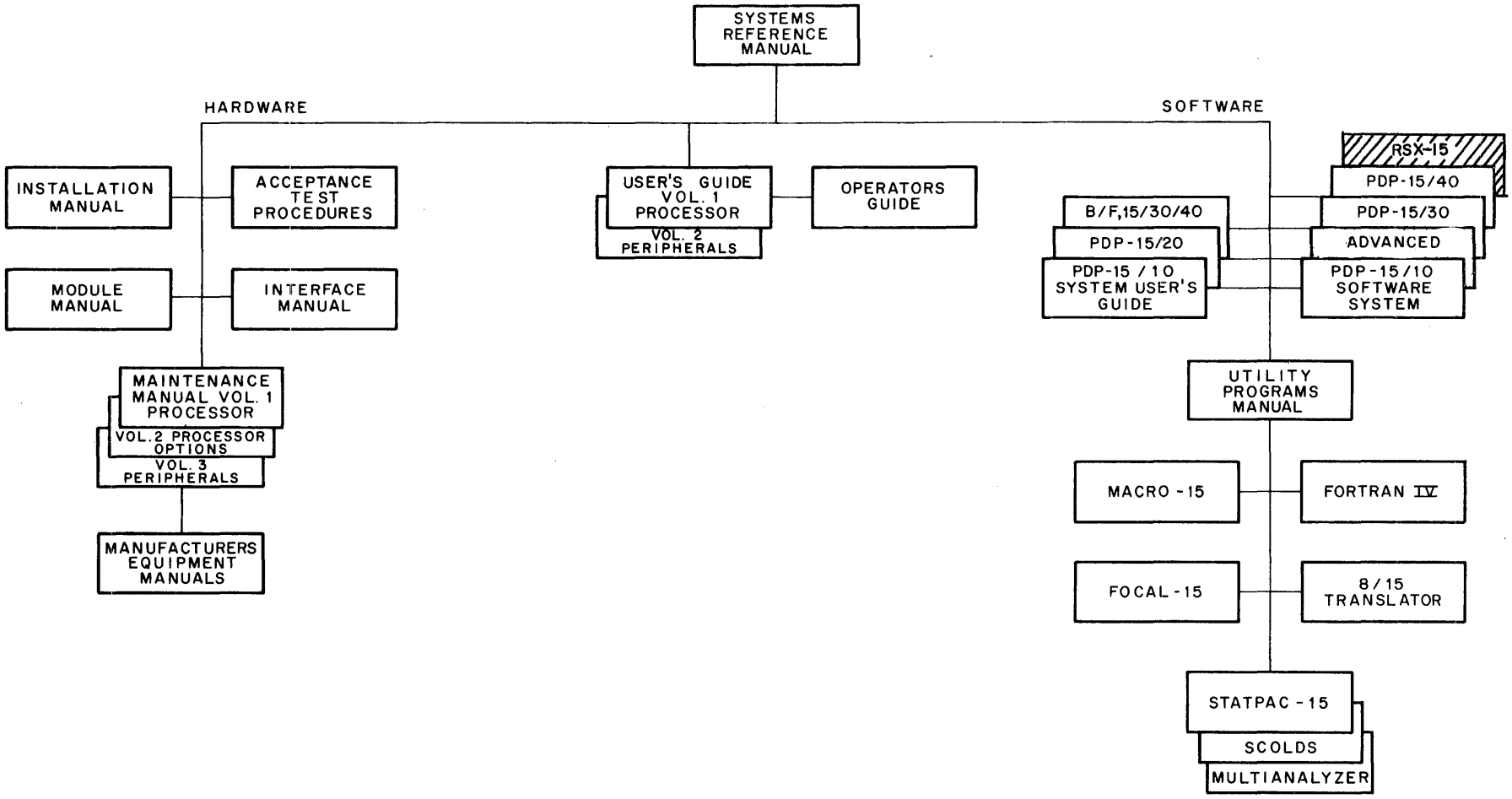
REAL TIME EXECUTIVE

REFERENCE MANUAL

FOR ADDITIONAL COPIES, ORDER DEC-15-GRQA-D FROM
PROGRAM LIBRARY, DIGITAL EQUIPMENT CORPORATION,
146 MAIN STREET, MAYNARD, MASS. 01754

PRICE $7.00

PDP-15 FAMILY OF MANUALS



15-0040

SYSTEM REFERENCE MANUAL - Overview of PDP-15 hardware and software systems and options; instruction repertoire, expansion features and descriptions of system peripherals. (DEC-15-GRAZ-D)

USER'S GUIDE VOLUME 1, PROCESSOR - Principal guide to system hardware includes system and subsystem features, functional descriptions, machine-language programming considerations, instruction repertoire and system expansion data. (DEC-15-H2DA-D)

VOLUME 2, PERIPHERALS - Features functional descriptions and programming considerations for peripheral devices. (DEC-15-H2DA-D)

OPERATOR'S GUIDE - Procedural data, including operator maintenance, for using the operator's console and the peripheral devices associated with PDP-15 Systems. (DEC-15-H2CA-D)

PDP-15/10 SYSTEM USER'S GUIDE - COMPACT and BASIC I/O Monitor operating procedures. (DEC-15-GG1A-D)

PDP-15/20 SYSTEM USER'S GUIDE - ADVANCED Monitor system operating procedures. (DEC-15-MG2B-D)

PDP-15/20/30/40 ADVANCED MONITOR SOFTWARE SYSTEM - ADVANCED Monitor System descriptions; programs include system monitor and language, utility, and application types; operation, core organization, and input/output operations within the monitor environment are discussed. (DEC-15-MR2B-D)

PDP-15/30 and 15/40 BACKGROUND/FOREGROUND MONITOR SOFTWARE SYSTEM - Background/Foreground Monitor description, including the associated language, utility, and application programs. (DEC-15-MR3A-D)

PDP-15/35, RSX-15 REAL TIME EXECUTIVE, REFERENCE MANUAL - Reference manual for the real time, multiprogramming RSX monitor system. (DEC-15-GRQA-D)

MAINTENANCE MANUAL VOLUME 1, PROCESSOR - Block diagram and functional theory of operation of the processor logic. Preventive and corrective maintenance data. (DEC-15-HB2A-D)

VOLUME 2, PROCESSOR OPTIONS - Block diagram and functional theory of operation of the processor options. Preventive and corrective maintenance data. (DEC-15-HB2A-D)

VOLUME 3, PERIPHERALS (Set of Manuals - Block diagram and functional theory of operation of the peripheral devices. Preventive and corrective maintenance data. (DEC-15-HB2A-D)

INSTALLATION MANUAL - Power specifications, environmental considerations, cabling, and other information pertinent to installing PDP-15 Systems. (DEC-15-H2AA-D)

ACCEPTANCE TEST PROCEDURES - Step-by-step procedures designed to ensure optimum PDP-15 Systems operation.

MODULE MANUAL - Characteristics, specifications, timing, and functional descriptions of modules used in PDP-15 Systems. (DEC-15-H2EA-D)

INTERFACE MANUAL - Information for interfacing devices to a PDP-15 System. (DEC-15-H0AA-D)

UTILITY PROGRAMS MANUAL - Utility programs common to PDP-15 Monitor Systems. (DEC-15-YWZA-D)

MACRO-15 - MACRO assembly language for the PDP-15. (DEC-15-AMZA-D)

FORTRAN IV - PDP-15 version of the FORTRAN IV compiler language. (DEC-15-KFZB-D)

FOCAL-15 - An algebraic interactive compiler-level language developed by Digital Equipment Corporation. (DEC-15-KJZB-D)

TABLE OF CONTENTS

CHAPTER ONE
INTRODUCTION

1.1 INTRODUCTION

RSX-15 is a real-time monitor system designed for handling real-time information in a multiprogramming environment. The modular construction of the system allows the user to configure his available hardware and software resources to best fit his requirements.

RSX-15 controls and supervises all operations within the system including any number of core- and disk-resident programs (called Tasks) limited in number only by available space. This control and supervision allows an unlimited number of Tasks to share core and disk memory, input/output device handlers, and other resources of the system.

The execution of Tasks is determined by software priorities, hardware interrupts, timing algorithms, and requests from other Tasks. The user can install a new Task on-line, establish its software priority from any of 512 distinct levels, and then request its activation at any time with an automatic reactivation at any periodic interval of time thereafter.

Utilizing simple time-directed commands, the user can dynamically schedule Tasks from the console terminal or from within a Task.

Device independence in RSX then allows the user to obtain results of that Task immediately on his Teletype* or store them on a mass storage device such as a disk or magnetic tape for future reference.

I/O requests from Tasks are queued and processed by RSX on a priority basis allowing high priority requests access to heavily used devices which have pending lower priority requests. Delays are further reduced by having the actual transfer of data being performed by the I/O Processor independent of the Central Processing Unit, thus allowing concurrent Task execution and I/O processing.

1.2 HARDWARE REQUIREMENTS AND OPTIONS

The minimum hardware configuration required to operate the RSX-15 system on a PDP-15/35**is as follows:

16K of core memory
API - Automatic Priority Interrupt
EAE - Extended Arithmetic Element
Real Time Clock (frequency is 16.7 msec for 60 Hz systems and 20 msec for 50 Hz systems.)
NOTE: The clock must be wired to API hardware level 3.
One (1) RS15 DECdisk (262,000 word fixed head) and one(1) RF15 controller.
One (1) TU56 DECtape unit and controller.
One (1) KSR35 Teletype
High Speed Paper Tape Reader
High Speed Paper Tape Punch

The RSX-15 system supports the following additional hardware:

Addition of core memory in increments of 4K up to 32K.
Addition of one or more disk units. The disk controller is designed to accommodate up to 8 disk units (2 million words).
Addition of one or more DECtape units. The controller is designed to accommodate up to 4 TU56 DECtape units (8 tape drives).
Addition of one or more Teletypes. Up to 16 additional Teletypes, either model KSR33 or KSR35, may be added to the standard system (under LT15/LT19).
One (1) VT01 storage tube display system. (Tektronix model 611 storage tube with interface).
One (1) to Eight (8) TU10 - 7 or 9 track IBM compatible magtape transport (7 and 9 track may not be mixed)
One (1) LP15 - Line Printer.

---

* Teletype is a trademark of the Teletype Corporation.
** A fully ECOed PDP-15 is required.

1.3 SYSTEM SOFTWARE

RSX-15 is a complete system for program preparation, compilation, assembly, debugging, and operation in a system that has been configured to the user's needs.

The RSX-15 system utilizes two separate monitors, the ADVANCED Software Monitor and the Real-Time Monitor. The ADVANCED Software Monitor is the standard monitor for the PDP-15/20 and PDP-15/35 computers.

The ADVANCED Monitor is used in the development, debugging, and building of executable Tasks for the Real-Time Monitor. The system software includes the FORTRAN IV compiler, MACRO assembler, TEXT EDITOR, TASK BUILDER, and numerous Utility programs*.

The TASK BUILDER, TKB, is used to build user's Tasks from relocatable binary files by linking them together along with library functions to constitute an executable Task that runs under control of the Real-Time Monitor. TKB is quite similar to the CHAIN program allowing very elaborate overlay structures to be built. A resultant Task is defined by a name (Task name), default run priority, core partition and common block requirements, and resident code. The Task, which resides either on paper tape or DECtape, is now ready to be incorporated into the real-time operating system under control of the Real-Time Monitor. Chapter five discusses the TASK BUILDER in greater detail.

The Real-Time Monitor is used to supervise and control the execution of real-time Tasks. The real-time software includes the RSX-15 EXECUTIVE, I/O Device Handler Tasks, Resident MCR, and the SYSTEM CONFIGURATOR. The SYSTEM CONFIGURATOR is a Task which is requested by the Real-Time Monitor when the system is initially loaded.

---

* Refer to ADVANCED Software Monitor Manual and Utility Programs Manual.

The CONFIGURATOR is an interactive program which asks the user several questions in order to tailor the RSX-15 EXECUTIVE to suit his particular application and hardware configuration. The user is required to supply information such as the amount of core memory available, number of disk units and Teletypes, partition sizes and locations, common areas, and which I/O Device-units are in the system. Chapter Six discusses the SYSTEM CONFIGURATOR in greater detail.

CHAPTER TWO
EXECUTIVE

2.1 INTRODUCTION

The RSX EXECUTIVE is the heart of the real-time operating system. It coordinates all activities in the system including Task scheduling, I/O supervision, resource allocation, and interactive operator communication.

The core memory of the RSX system is divided into partitions that are occupied by the Real-Time Monitor, Monitor Console Routine (MCR) Function Tasks, I/O Handler Tasks, user written Tasks(programs), and COMMON Blocks used for inter-Task communications. There is no limit to the number of core partitions and COMMON Blocks that can be defined except for the amount of core space available. All Tasks are then executed from these partitions allowing several programs to be in core at any given time (multiprogramming). Normally Tasks reside on the disk, and are brought into their partition (if unoccupied) only when requested, and release their partitions upon exit. However, when desirable, or necessary, a Task may be fixed in core, thereby dedicating a partition to a single Task, but assuring core availability and rapid response. Tasks that can tolerate a response time of 100 milliseconds or more will normally be disk resident rather than

core resident. Since the Task Builder program allows a Task to consist of a resident program with a simple to very elaborate overlay structure, a Task can be both core and disk resident at the same time. The core resident program remains in core once the Task has been activated,and overlay segments are requested when needed. Requested overlay segments will be executed immediately if already in core, or brought in from disk overlaying the previous segment(s) and then executed. When a Task is built using the Task Builder program it can include any number of user written programs and be assigned any core partition (providing the partition was defined at system configuration time and is large enough to contain the Task). The Task can also be assigned any run priority (which may be overridden at run time) from 1 to 512 where 1 is the highest priority.

Task execution occurs because of requests by the operator, requests from a currently executing Task, or by a predefined schedule*. Activated Tasks are defined in the system Active Task List and scheduled Tasks are defined in the Clock Queue to be activated at a predefined time. Tasks can also be installed in the system on-line while other Tasks are currently executing.

2.2 EXECUTIVE ORGANIZATION

2.2.1 CORE AND DISK MANAGEMENT

Core memory in the RSX system is partitioned to allow several Tasks to be active at any given time. All core above the resident EXECUTIVE (first 4K) can be user specified into Partitions and COMMON Blocks during system configuration. All unspecified space above the first 8K of core will then be used to create Partition Blocks and a reservoir

* The scheduling capabilities of RSX will be described later.

of empty nodes called the "Pool". Each node in the Pool consists of ten contiguous memory locations with internal pointers connecting the previous node to the next node resulting in a circular or double ended queue called a deque[1]. The EXECUTIVE uses nodes to create linked lists containing system information. These nodes are removed from and returned to the Pool as needed.

Since the Pool and Partition Blocks are needed to run, sufficient core above the lower 8K core should be left unspecified. Unspecified core below 8K (and above the EXECUTIVE) is unused.

Partitions and SYSTEM COMMON Blocks[2] are fixed at system configuration time and cannot be altered at run time. Tasks are built to execute in specific partitions, and, any number of Tasks may be built to execute in the same partition.

A Partition containing an active Task cannot be used by other Tasks, regardless of priority, until that partition becomes available. When two or more requests for the same partition are made and the partition is occupied, the Task with the highest priority will be serviced first when the partition becomes available. An executing Task releases its partition once it EXITs to the EXECUTIVE.

RSX uses the disk for storage of user written Tasks, MCR Function Tasks and data. Disk space is automatically allocated by the EXECUTIVE when Tasks are installed in the system. The remaining portions of the disk are available to the user. When a Task requires disk space to store data, it must request it through the EXECUTIVE via an ALLOCATE Directive. The actual allocation of disk space is performed in increments of 128_{10} words of contiguous disk storage providing the user with true random access capability. The EXECUTIVE maintains a record of available disk space by using a bit map scheme. A user can relinquish allocated disk space through the use of the

[1]Deque is pronounced "deck".
[2]See Glossary: COMMON BLOCK, SYSTEM.

DEALLOCATE Directive.

2.2.2 SCHEDULING OF REAL-TIME PROGRAMS (TASKS)

The scheduling of Real-Time programs (Tasks) can result from any one of three types of events: the request for the activation of a Task, the request for an Input/Output transfer, or the occurrence of a hardware interrupt. It is important that the reader have a thorough understanding of the meaning of "Significant Event"[1] and the Active Task List in order to understand the concepts of scheduling in RSX.

The Active Task List is a priority ordered list of Active Tasks that is used to drive the system. This list is scanned from high to low priority by the EXECUTIVE as a result of a Significant Event to give control to the highest priority Task that is capable of executing at that time.

Tasks are installed in the system either at the priority given them at Task Building time (default priority) or at the priority specified in the INSTALL Directive. For instance, the user can install a Task named SCAN with default priority 100 by typing:

```
INSTALL SCAN
```

If a new default priority of 78 were desired, the user could have installed SCAN by typing:

```
INSTALL SCAN 78
```

Task priorities can also be altered at run time either by the operator or by a currently executing Task. Once a Task has been installed in the system it can be activated by typing:

```
REQUEST SCAN                    (Task will execute at default
                                priority)
SCHEDULE SCAN 13:3Ø:ØØ 3ØM      (Task will execute at default
                                priority at 1:3Ø P.M. and be resched-
                                uled every 3Ø minutes thereafter)
RUN SCAN 25M                    (Task will execute at default
                                priority 25 minutes from now)
```

[1] Task initiation, task completion, and I/O completion are examples of significant events.

```
SYNC SCAN H 3ØM 2H 78          (Task will be executed at priority
                               78, 3Ø minutes past the hour, and
                               every 2 hours thereafter)
```

A request to activate a Task will be executed providing that a partition is available and a Task with a higher priority is not currently executing. Once a Task is activated it will run to completion unless interrupted by a higher priority Task. An interrupted lower priority Task will be resumed only when higher priority Tasks have completed or have relinquished control. Whenever one Task is interrupted by another, its active registers are automatically saved by the EXECUTIVE and later restored when execution is resumed. Control will be given to a lower priority Task if a currently executing Task is waiting for the completion of an I/O request or by issuing any of the following Directives: WAIT; WAITFOR; and SUSPEND. Control can be given to a higher priority Task by requesting it to be run or by issuing any of the following Directives: REQUEST, RUN, SYNC, and SCHEDULE.

The following example illustrates the mechanism used by the EXECUTIVE to scan the Active Task List. Assume that two Tasks are installed in the system with names TASK1 and TASK2 and have priorities of 50 and 100 respectively[1]. The operator requests TASK1 to be executed one minute from now and TASK2 to be executed immediately. The operator's commands would be:

```
RUN TASK1 1M
REQUEST TASK2
```

The following chart describes the sequences of events during the request and activation of both Tasks.

---

[1]The larger number indicates a lower priority.

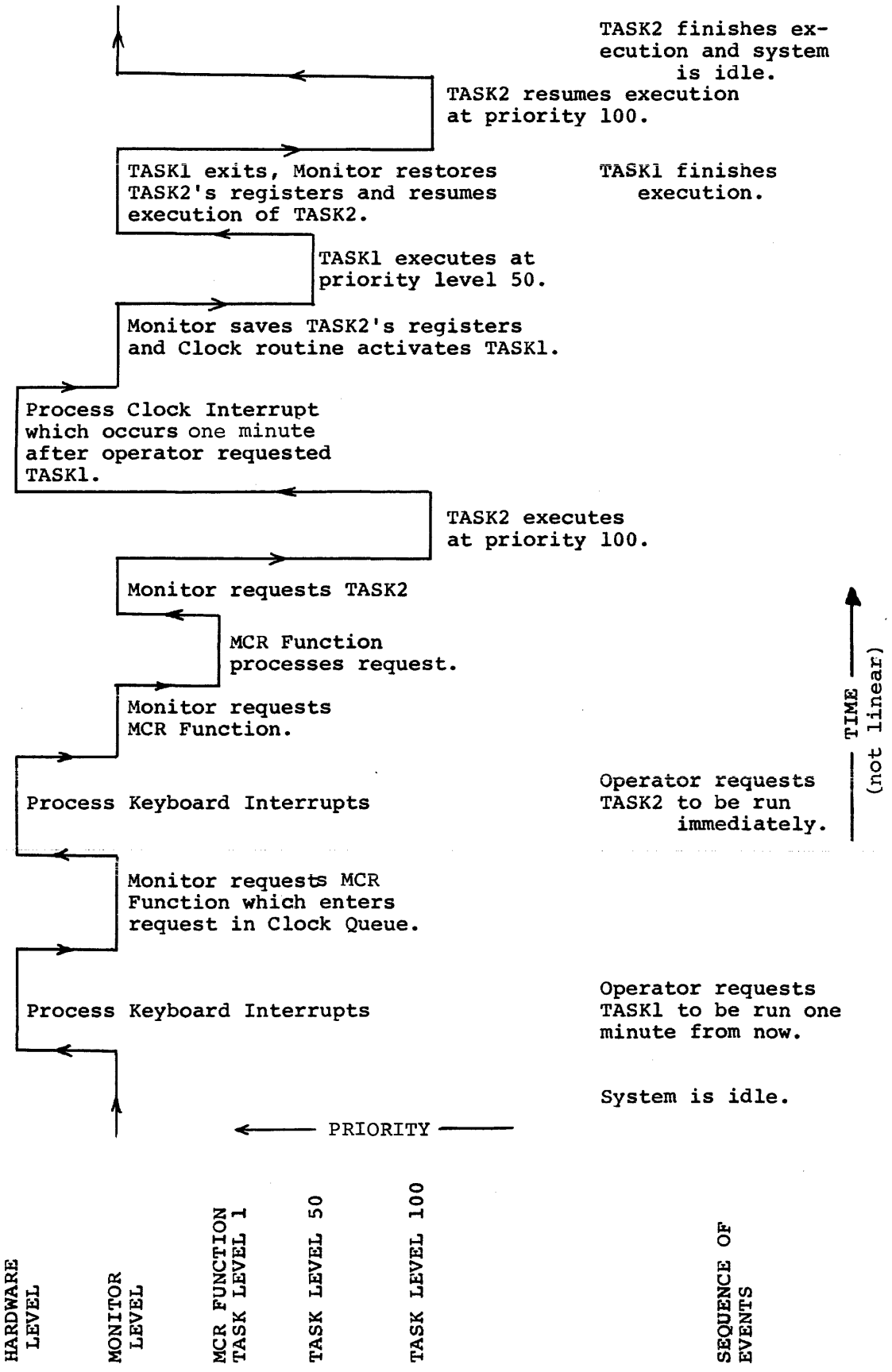

TASK2 finishes execution and system is idle.
TASK2 resumes execution at priority 100.
TASK1 exits, Monitor restores TASK2's registers and resumes execution of TASK2.
TASK1 finishes execution.
TASK1 executes at priority level 50.
Monitor saves TASK2's registers and Clock routine activates TASK1.
Process Clock Interrupt which occurs one minute after operator requested TASK1.
TASK2 executes at priority 100.
Monitor requests TASK2
MCR Function processes request.
Monitor requests MCR Function.
Process Keyboard Interrupts
Operator requests TASK2 to be run immediately.
Monitor requests MCR Function which enters request in Clock Queue.
Process Keyboard Interrupts
Operator requests TASK1 to be run one minute from now.
System is idle.
TIME (not linear)
PRIORITY
HARDWARE LEVEL
MONITOR LEVEL
MCR FUNCTION TASK LEVEL 1
TASK LEVEL 50
TASK LEVEL 100
SEQUENCE OF EVENTS

Figure 2.1

2.2.3 INPUT/OUTPUT OPERATIONS

The RSX EXECUTIVE allows the user device independent programming, reassignment of devices on-line, and the ability to queue I/O requests providing him with an extremely powerful and flexible I/O structure.

The RSX system provides the user with device handlers for standard I/O devices supplied with the system. These handlers are called I/O Device Handler Tasks and can be installed in the system either at system configuration time or on-line. I/O Handler Tasks are connected to the user's environment by means of a logical/physical device relationship. This relationship allows the user to reference a Logical Unit Number (LUN) rather than a physical device when requesting an I/O operation. At run time the user then may reassign the LUN to the desired physical device. Thus, a Task which normally outputs data to LUN 4 which is currently assigned to the teleprinter can output the same results on a paper tape punch or line printer if the user types the following:

```
REASSIGN 4 PP TTØ          (Reassign LUN 4 to the Paper Tape
        (or)               Punch from TTØ)
REA 4 LP PP                (Reassign LUN 4 to the line printer
                           from the Paper Tape Punch)
```

All requests to be serviced by I/O Handler Tasks are entered into a priority ordered queue even though the device may be busy. The priority of the request will be the same as the priority of the Task issuing the request. Once the Task has issued a request, it can either wait for its completion or continue executing and test at any time the current state of the I/O operation it requested.

An Event Variable (software flag) may be associated with I/O requests and its value indicates the current status of the I/O operation. When

an I/O operation completes, the user's Event Variable is set accordingly and a Significant Event is declared. This causes the Active Task List to be scanned and control to be given to the highest priority Task capable of executing.

There are two types of I/O Device Handlers in the RSX System: The Standard I/O Device Handler Task and the System I/O Device Handler Task. The System I/O Handlers are the Disk and Multi-Teletype Handler Tasks which must be core resident and cannot be deleted from the system. Standard I/O Device Handler Tasks are those which are not System I/O Handlers supplied by the manufacturer or created by the user. When a LUN is assigned to a device, the Handler Task is requested and remains in core as long as the LUN is assigned to the device.

A Task can obtain the exclusive use of an I/O Handler Task by issuing an ATTACH Directive. When the ATTACH Directive is accepted by the Handler, only requests from that Task are serviced with requests from other Tasks queued until a DETACH request is serviced.

Example:	ATTACH 2,EV	Attach the device assigned to LUN 2 to the currently executing Task.
	DETACH 2,EV	Detach the Device. EV is the event variable.

2.2.4 DYNAMIC SYSTEM PRIORITY CONTROL

The priority structure of the RSX System includes both hardware and software priority levels. Hardware levels are established by the Automatic Priority Interrupt (API) of the PDP-15 computer and software levels are established by the user and controlled by the EXECUTIVE. There are 8 levels of API of which four are used for hardware I/O devices and four for the EXECUTIVE. API levels 4 and 6 are used exclusively by the EXECUTIVE and level 7 is used for Task execution (level 5 is currently not used). From level 7, the

EXECUTIVE derives its 512 Task priority levels used for Task operations. The following figure illustrates the hierarchy of the entire priority system.

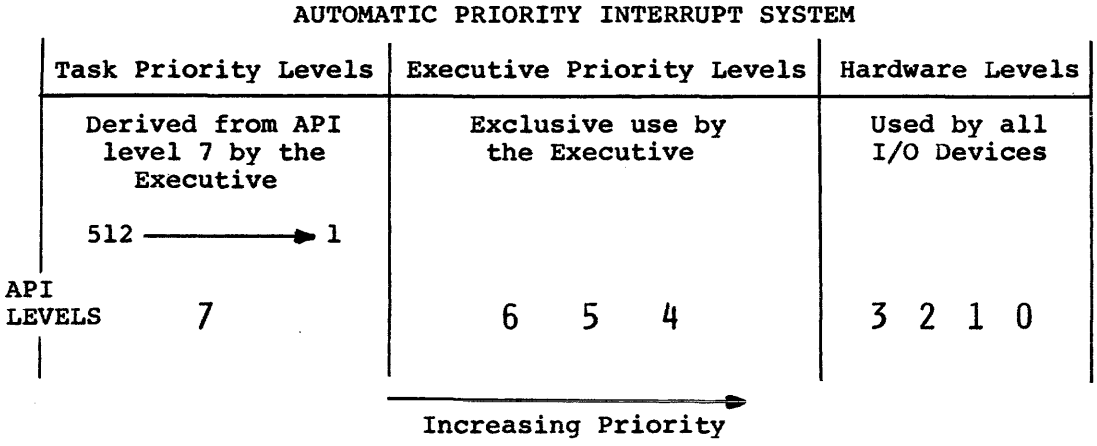


Figure 2.2 RSX Priority Structure

The hardware API levels 0,1,2, and 3 are used to control I/O devices in the system. Each level can have as many as 8 device controllers connected to it allowing a total of 32 devices to be serviced by the API system. Each of the 32 API lines are associated to unique core locations which specify where program control will be transferred when an interrupt signal occurs on that line.

CHAPTER THREE
MCR
MONITOR CONSOLE ROUTINE

3.1 INTRODUCTION

The Monitor Console Routine (MCR) allows the user to communicate on-line with the system from the console teleprinter to dynamically adjust and modify the operation of the system through simple commands (functions). The operator may obtain status information about the system, install or replace Tasks, request Task execution based upon time driven schedules, or fix a Task in core. Other MCR commands permit altering of logical/physical device relationships, examination and modification of core locations, and adjustment of the System Clock and Calendar.

The MCR consists of the Resident MCR Task, which accepts the user's commands, and the MCR Functions, which actually carry out the indicated requests. The MCR Functions are similiar to user created Tasks in that they normally reside on the disk and are brought into a core partition when requested. Although the MCR Functions are built (with the Task Builder) to execute in a predefined core partition, they can be built to run in any partition. Execution of MCR Tasks, like all Tasks, is based upon partition availability and Task priority.

3.1.1 REQUESTING THE RESIDENT MCR

The Resident MCR must be Active in order to receive requests for Function Tasks from the operator at the console teleprinter. To request the Resident MCR, type CTRL C (i.e., simultaneously depressing the CTRL and C keys). When the MCR is ready to accept a command it will output:

```
MCR>
```

(on LUN 2) and wait for a command to be typed immediately to the right of the prompting character (>).

3.1.2 ERROR DETECTION AND HANDLING

Error detection is provided by the various MCR Functions and Resident MCR where applicable. When an error is detected, an appropriate message, prefixed by the name of the issuing MCR Function, is output to LUN 3.

3.1.3 COMMAND STRINGS

When typing MCR command strings the following conventions apply:

a. Command strings are terminated either by a Carriage RETURN or by an ALT MODE. If a Carriage RETURN is typed, the Resident MCR will be requested when the current Function is complete. If an ALT MODE is typed, the Resident MCR will not be requested at the termination of the current Function.

b. Each element of a command string must be separated by either a comma (,) or a space (_).

c. If an error is discovered while typing a command string prior to typing a terminator, the line may be deleted as far back as the prompting character (>) by typing CTRL U (formed by simultaneously typing CTRL and U characters). A commercial

"at" (@) symbol is echoed informing the user that he can retype the command string. The RUBOUT, echoed as a backslash (\), may be used to delete the last character typed in. Every time the RUBOUT is typed, a backslash is echoed and a character is deleted.

d. **Any number of characters (except a comma or space) may be inserted between a Function name and its arguments or command string terminator (Carriage RETURN or ALT MODE). This is useful if the user wishes to improve the readability of his teleprinter copy.**

3.2 SUMMARY OF MCR FUNCTIONS*

Task Name	Function
ETI[ME]	Enter time and date into the system.
TIM[E]	Request current time from the system.
DAT[E]	Request current date and time from the system.
TAS[K LIST]	Request System Task List.
PAR[TITIONS]	Request list of Partition definitions.
COM[MON BLOCKS]	Request list of Common Block definitions.
DEV[ICES AND ASSIGNMENTS]	Request list of LUN device assignments.
INS[TALL]	Install a Task in the system.
REM[OVE]	Remove a Task from the system.
REQ[UEST]	Request immediate activation of a Task.
SCH[EDULE]	Schedule the activation of a Task.
RUN	Request scheduled Task a delta time from now.

* Square brackets of the form ([]) specify optional characters.

SYN[C]	Schedule and synchronize the activation of a Task.
CAN[CEL]	Cancel the activation of a Task.
RES[UME]	Resume execution of a suspended Task.
FIX	Fix Task in core (Task becomes core resident).
UNF[IX]	Unfix Task in core.
DIS[ABLE]	Disable Task (reject future Task activation directives).
ENA[BLE]	Enable a disabled Task.
REA[SSIGN]	Change LUN assignment(s).
SAV[E]	Save image of core on the disk.
OPE[N]	Open register for examination or modification.

3.3 MCR FUNCTION DESCRIPTIONS

The following paragraphs describe the form and function of the MCR Functions. To simplify the interpretation of the various command strings, the following symbols are used to represent the non-printing teleprinter operations:

```
ʃ = Carriage RETURN
↓ = LINE FEED
∇ = Terminator (either Carriage Return or ALT MODE)
_ = Space
```

Square brackets of the form ([]) specify optional characters and/or arguments. The ampersand sign (&) is used for concatenation of a numeric argument to an alphabetic letter.

3.3.1 ENTER TIME

The Enter Time Function is used to set the System Clock and Calendar.

```
Form:      ETI[ME]_Hr:Min:Sec[_Mo/Day/Yr]∇

Variables:Hr  = Hours (Ø-23)
          Min = Minutes (Ø-59)
          Sec = Seconds (Ø-59)
          Mo  = Month (1-12)
```

Day = Day of Month (1-31)
Yr = Year (last two digits Ø-99)

NOTE: The European form has the month and day reversed.

Example: The time is 3Ø seconds past 3:45 P.M. and the date is March 23, 1971.

```
MCR>ETIME 15:45:3Ø 3/23/71
MCR>
```

3.3.2 TIME

The Time Function outputs the time of day on LUN 3.

Form: TIM[E]V

Example: The time is 41 seconds past 3:45 P.M.

```
MCR>TIME
15:45:41
MCR>
```

3.3.3 DATE

The Date Function outputs the System Calendar and the time of day on LUN 3.

Form: DAT[E]V

Example: The date is March 23, 1971 and time of day is 52 seconds past 3:45 P.M.

```
MCR>DATE
Ø3/23/71  15:45:52
MCR>
```

3.3.4 TASK LIST

The Task List Function outputs to LUN 3 a description of each Task which has been Installed in the system. The description consists of the following information (printed left to right, one line per Task): Task Name, Partition Name, Priority (decimal), Disk Unit Number (octal), Head Track Address (octal) and Task Size (octal). Output may be prematurely terminated by typing CTRL C.

Form: TAS[K LIST]V

Example:

```
MCR>TASK LIST
 RX       P14.6    512   0 102200 00530
 PP....   IO.2     002   0 101200 00617
 DT....   IO.1     002   0 076600 02261
 PR....   IO.2     002   0 075600 00736
 LP....   IO.2     001   0 075000 00460
 ...DAT   MCR      002   0 074600 00176
 ...OPE   MCR      002   0 073600 00624
 ...SAV   MCR      002   0 073200 00214
 ...REA   MCR      002   0 072000 01040
 ...DIS   MCR      002   0 071600 00166
 ...RAR   MCR      002   0 071400 00166
 ...UNF   MCR      002   0 071000 00211
 ...FIX   MCR      002   0 070400 00302
 ...RES   MCR      002   0 070000 00250
 ...CAN   MCR      002   0 067600 00166
 ...SYN   MCR      002   0 067200 00377
 ...RUN   MCR      002   0 066600 00361
 ...SCH   MCR      002   0 066000 00407
 ...REQ   MCR      002   0 065400 00335
 ...REM   MCR      002   0 065000 00305
 ...INS   MCR      002   0 063200 01516
 ...DEV   MCR      002   0 062400 00533
 ...COM   MCR      002   0 062000 00263
 ...PAR   MCR      002   0 061400 00230
 ...TAS   MCR      002   0 060600 00406
 ...TIM   MCR      002   0 060400 00145
 ...ETI   MCR      002   0 060000 00350
MCR>
```

3.3.5 PARTITIONS

The Partition Function outputs to LUN 3 a description of all core partitions defined in the system. The description consists of (printed from left to right, one line per partition): Partition Name, Partition Base Address (octal), and Partition Size (octal). Output may be prematurely terminated by typing CTRL C.

Form: PAR[TITIONS]V

Example:

```
MCR>PARTITIONS
 MCR     10000 01600
 IO.1    11600 03000
 P14.6   14600 03200
 P21.0   21000 05500
 P26.5   26500 06500
 IO.2    35200 01000
 P40.0   40000 15000
MCR>
```

3.3.6 COMMON BLOCKS

The COMMON Blocks Function outputs to LUN 3 a description of all System COMMON Blocks defined in the system. The description consists of (printed from left to right, one line per COMMON Block): COMMON Block Name, COMMON Block Base Address (octal), and COMMON Block Size (octal). Output may be prematurely terminated by typing CTRL C.

Form: COM[MON_BLOCKS]∇

Example:

```
MCR>COMMON BLOCKS
 .XX     20000 00700
 FLAG    36200 00600
MCR>
```

3.3.7 DEVICES AND ASSIGNMENTS

The Devices and Assignments Function outputs to LUN 3 a list of physical device units and the Logical Unit Numbers assigned to them. Output may be prematurely terminated by typing CTRL C.

Form: DEV[ICES_AND_ASSIGNMENTS]V

Example:

```
MCR>DEVICES AND ASSIGNMENTS
 DK0     1
 TT0     2,3,5,10,11,12,13,14,15,16,17,18,19
         20,21,22,23,24,25,26,27,28,29,30,31
         32
 TT1
 DT0     4
 DT1
 DT2
 DT3
 DT4
 DT5     6
 DT6
 DT7
 PR0     7
 PP0     9
 LP0     8
MCR>
```

3.3.8 INSTALL

The Install Function is used to input a Task into the RSX System. The Task to be added must be a binary file (TSK extension) produced by the Task Builder. TSK files are installed from LUN 5.

Form: INS[TALL]_TSKNAM[_P]V

Variables: TSKNAM = Name of Task to be Installed (1 - 6 characters)
P = Task priority (1 - 512)

Examples: Install Task SCAN whose default priority defined at Task Building time is 48.

```
MCR>INS SCAN,
    (or)
MCR>INS SCAN 10,          (SCAN is now redefined
                          with a priority of 10)
```

3.3.9 REMOVE

The Remove Function is used to delete a Task from the RSX System.

Form: REM[OVE]_TSKNAM↵

Variables: TSKNAM = Name of Task to be Removed (1 - 6 characters)

Example: The Task SCAN is no longer required and it is desired to remove it from the System.

```
MCR>REM SCAN↵
```

3.3.10 REQUEST

The Request Function is used to request the execution of a Task at an indicated software priority level. Actual Task execution depends upon priority and partition availability.

Form: REQ[UEST]_TSKNAM[_P]↵

Variables: TSKNAM = Name of Task (1 - 6 characters)
P = Task priority (1 - 512)

Examples: Request the execution of SCAN whose default priority defined at Task Building or Installation time is 48.

```
MCR>REQ SCAN↵
    (or)
MCR>REQ SCAN 1Ø↵          (SCAN is requested with
                          a priority of 1Ø)
```

3.3.11 SCHEDULE

The Schedule Function is used to schedule the execution of a Task at some time in the future, specified in time-of-day, at an indicated software priority level, and with periodic rescheduling.

Form: SCH[EDULE]_TSKNAM_Hr:Min:Sec[_RI&RU][_P]↵

Variables: TSKNAM = Name of Task (1 - 6 characters)
Hr = Hours (Ø - 23)
Min = Minutes (Ø - 59)
Sec = Seconds (Ø - 59)
RI = Reschedule Interval (up to 1 day)
RU = Reschedule Units (T=Ticks, S=Seconds, M=Minutes, and H=Hours)
P = Task Priority (1 - 512)

Examples: Schedule the execution of SCAN at 1:3Ø P.M. and reschedule it every 3Ø minutes thereafter at its default priority.

```
MCR>SCH SCAN 13:3Ø:ØØ 3ØM↵
```

Schedule the execution of SCAN at 8:3Ø A.M. and reschedule it every 2/6Øth's of a second (6Ø cycle clock) at priority level 1Ø.

```
MCR>SCH SCAN 8:3Ø:ØØ 2T 1Ø↵
```

3.3.12 RUN

The Run Function is used to make a Task active at some future time, specified in delta time from now, at an indicated software priority and with periodic rescheduling.

Form: RUN_TSKNAM_SI&SU[_RI&RU][_P]∇

Variables:
- TSKNAM = Name of Task (1 - 6 characters)
- SI = Schedule Interval (up to one day)
- SU = Schedule Units (T=Ticks, S=Seconds, M=Minutes, and H=Hours)
- RI = Reschedule Interval (up to one day)
- RU = Reschedule Units (T=Ticks, S=Seconds, M=Minutes, and H=Hours)
- P = Task Priority (1 - 512)

Examples: Schedule the execution of SCAN 3Ø minutes from now and reschedule it every hour thereafter.

```
MCR>RUN SCAN 3ØM 1H↵
```

Schedule the execution of SCAN 1Ø minutes from now and reschedule it every 32 seconds thereafter at priority level 28.

```
MCR>RUN SCAN 1ØM 32S 28↵
```

3.3.13 SYNCHRONIZE

The Sync Function is used to activate a Task at some future time following the occurrence of the next tick, second, minute, or hour. The Task is executed at the indicated software priority and with periodic rescheduling. This Function is particularly useful for minimizing the peak loading of a system which can occur when many Tasks are scheduled for execution at the same time.

Form: SYN[C]_TSKNAM_SZ_SI&SU[_RI&RU][_P]∇

Variables:
- TSKNAM = Name of Task (1 to 6 characters)

SZ = Synchronization Units (T=Ticks, S=Seconds, M= Minutes, and H=Hours)
SI = Schedule Interval from Synchronization time (up to one day)
SU = Schedule Units (T=Ticks, S=Seconds, M=Minutes, and H=Hours)
RI = Reschedule Interval (up to one day)
RU = Reschedule Units (T=Ticks, S=Seconds, M=Minutes, and H=Hours)
P = Task Priority (1 - 512)

Example: Schedule the execution of SCAN 3 minutes after the next hour and reschedule it every hour thereafter at its default priority level.

```
MCR>SYN SCAN H 3M 1H↵
```

Schedule the execution of SCAN 1Ø seconds after the next minute and reschedule it every hour thereafter at priority 21.

```
MCR>SYN SCAN M 1ØS 1H 21↵
```

3.3.14 CANCEL

The Cancel Function is used to cancel all scheduled requests for activation of a particular Task by removing those requests from the Clock Queue. Cancellation does not affect the current execution of the given Task nor does it affect schedule requests made in the future. The latter case is covered by the DISABLE Function. However, schedule requests which have already been made (and entered in the Clock Queue) are discarded.

Form: CAN[CEL]_TSKNAM⍛

Variables: TSKNAM = Name of Task (1 - 6 characters)

Example: Cancel the activation of Task SCAN.

```
MCR>CAN SCAN↵
```

3.3.15 RESUME

The Resume Function is used to resume the execution of a Task which has been SUSPEND'ed.

Form: RES[UME]_TSKNAM[_Resumption address]⍛

Variables:TSKNAM = Name of Task (1 - 6 characters)

Example: Task SCAN has been previously SUSPEND'ed and it is desired to resume its execution.

```
MCR>RES SCAN,
```

3.3.16 FIX IN CORE

The Fix Function is used to fix an inactive Task into a free partition. This dedicates a partition to a Task and provides for a faster response to the REQUEST, SCHEDULE, RUN, and SYNC Directives as well as responses to external interrupts.

Form: FIX_TSKNAM

Variables:TSKNAM = Name of Task (1 - 6 characters)

Example: Fix Task SCAN in core.

```
MCR>FIX SCAN,
```

3.3.17 UNFIX FROM CORE

The Unfix Function is used to nullify a FIX Directive. If a FIXed Task is active when an UNFIX Directive is issued, the partition will be freed when the Task EXITs.

Form: UNF[IX]_TSKNAM

Variables:TSKNAM = Name of Task (1 - 6 characters)

Example: Unfix Task SCAN from its partition.

```
MCR>UNF SCAN,
```

3.3.18 DISABLE

The Disable Function is used to instruct the system to reject further REQUEST, SCHEDULE, RUN, or SYNC Directives or periodic rescheduling for an indicated Task. This Function renders the specified Task incapable of responding to other Directives except ENABLE. A Disabled Task is not deleted from the system. (cf. REMOVE.)

Form: DIS[ABLE]_TSKNAM∇

Variables: TSKNAM = Name of Task (1 - 6 characters)

Example: Disable the Task SCAN.

```
MCR>DIS SCAN,
```

3.3.19 ENABLE

The Enable Function is used to re-enable a DISABLEd Task.

Form: ENA[BLE]_TSKNAM∇

Variables: TSKNAM = Name of Task (1 - 6 characters)

Example: Task SCAN has been previously DISABLEd and it is desired to re-enable it.

```
MCR>ENA SCAN,
```

3.3.20 REASSIGN

The Reassign Function is used to alter the logical/physical device relationships by deassigning a Logical Unit Number (LUN) from a device and reassigning it to another device. This Function causes the REQUESTing and EXITing of I/O Device Handler Tasks.

Form: REA[SSIGN]_LUN_ND_OD[/LUN_ND_OD][/LUN_ND_OD]....∇

Variables: LUN = Logical Unit Number to be Reassigned
ND = Device to which the LUN is to be assigned
OD = Device from which the LUN is to be deassigned.

Examples: Assume the following LUN assignments currently exist: LUN 2≡TTØ, LUN 3≡TTØ, LUN 4≡DT5, and LUN 33≡LP. It is now desired to reassign those LUN's to the following devices: LUN 2≡TT1, LUN 3≡TT1, LUN 4≡DT7, and LUN 33≡TTØ.

```
MCR>REA 3 TT1 TTØ,
MCR>REA 2 TT1 TTØ,
MCR>REA 4 DT7 DT5,
MCR>REA 33 TTØ LP,
        (or)
MCR>REA 2 TT1 TT/3 TT1 TT,
MCR>REA 4 DT7 DT5/33 TT LP,
```

Note: The MCR Functions, including the REASSIGN Function, use LUN's 2 and 3 for command input and output, respectively. Therefore, it is recommended when the user REASSIGN's these to another device, he should do so by REASSIGNing both LUN's on the same

line as shown in the second set of examples. The first set of examples are valid when altering LUN's 2 and 3 because the output from the REASSIGN Function Task was altered first and further commands can still be input from LUN 2. Device names associated with I/O Handler Tasks provided with the system are:

```
TT[n] = Teletype (n is the unit number which, if omitted,
                  will be assumed zero)
DT[n] = DECtape
DK    = DECdisk
PR    = High Speed Paper Tape Reader
PP    = High Speed Paper Tape Punch
LP    = Line Printer
```

3.3.21 SAVE

The Save Function is used to record a core image of an RSX System (from location 30_8 to the top of core memory) at the beginning of disk zero. The purpose of this Function is to provide a means for updating the system after Tasks have been added or deleted. The updated system can then be restored at any time by simply loading the RSX Bootstrap. This Function should only be executed when the system is quiescent, i.e., no Tasks should be active and no I/O should be in progress.[1] Since the entire RSX system is recorded on the disk, when the user reloads the system, the System Calendar and clock will reflect the previous settings at the time the Save was done. Therefore the user should reset them to the correct date and time after reloading the system.

Form: SAV[E]↵

Example: Save a copy of the RSX system on disk zero.

```
MCR>SAV[E],
```

3.3.22 OPEN REGISTER

The Open Register Function permits the user to access any core location for the purpose of examination and/or modification. The user may optionally enter a signed number in any opened location using either

[1]The former does not imply the latter.

octal or decimal notation. Furthermore he may open and examine the register whose address is specified by the low order 15 bits of the currently open register or he may continue examination either in ascending or descending address order.

Form: OPE[N]_ADR[Dn]∇

Variables:ADR = Address of location to be examined. If the specified address is valid, the address and contents of that register are output followed by the prompting character (>).
D = The letter "D" signifies a disk address (ADR) rather than a core address to open.
n = Disk unit number.
Note: User's response follows the prompting character (>). The (|) symbol denotes "EXCLUSIVE OR".

Form: ADDRESS/CONTENTS>[new contents]|[*∇]|[↑∇]|∇

Variables:new contents = A number (1 - 6 digits) which is to replace the contents of the currently opened register. The number may be optionally signed (+ or -) and/or a radix operator (D = decimal, O = octal).

EXAMPLES: Decimal -39
-D39 (or) -DØØØØ39

Octal 32
O32 (or) +O32 (or) 32

Terminators and special characters may be any of the following:

⏎	- Close the current register and open the next higher register.
ALT MODE	- Close the current register and terminate the Function.
↑⏎	- Close the current register and open the next lower register.
*⏎	- Close the current register and open the register specified by the lower 15 bits of the current register.

Example: Comments follow the slash character and are used only to describe the different operations.

```
MCR>OPE 242⏎                        /open register 242
>ØØ242/ØØ2325⏎                     /open register 243
>ØØ243/ØØ3432*⏎                    /open register 3432
>Ø3432/ØØØØ5Ø ØØØØ47⏎              /change contents of
                                    /  3432 to 47.
>Ø3433/ØØØ432 -D1Ø(ALT MODE)       /change contents of
                                    /  3433 to -1Ø decimal
                                    /  and terminate
                                    /  sequence.
MCR>OPE 243⏎                        /open register 243
>ØØ243/ØØ3432↑⏎                    /open register 242
>ØØ242/ØØ2325 -6 (ALT MODE)        /change contents to -6
MCR>
```

CHAPTER FOUR
RSX SYSTEM DIRECTIVES

4.1 INTRODUCTION

Communication to the RSX System from the user is accomplished by the use of system "Directives". Directives may be issued from within a Task or indirectly by an operator via the teleprinter and the Monitor Console Routine (MCR). The manner in which a Directive may be issued varies according to its function and use.

Directive routines are structured to be reentrant and may be used to direct the Executive to schedule and reschedule a Task, provide status information for a Task, or queue I/O Handler Tasks to perform indicated I/O operations.

4.2 SUMMARY OF RSX DIRECTIVES & SYSTEM MACROS

RSX DIRECTIVES

CAL FUNCTION CODE (octal)	MACRO CALL	FORTRAN CALL	SYSTEM DIRECTIVE
Ø1	REQUEST	REQST	Request Task execution
Ø2	SCHEDULE	SCHED	Schedule Task execution
Ø3	RUN	RUN	Run Task in delta time
14	SYNC	SYNC	Sync Task execution
Ø4	CANCEL	CANCEL	Cancel scheduled requests
Ø6	SUSPEND	SUSPND	Suspend Task execution
Ø7	RESUME	RESUME	Resume Task execution

13	MARK	MARK	Set Event Variable in delta time
2Ø	WAITFOR	WAITFR	Wait for an Event Variable to be set
Ø5	WAIT	WAIT	Wait for next Significant Event
1Ø	EXIT	EXIT	Terminate execution of the Task
11	CONNECT	----	Connect to interrupt line
12	DISCONNECT	----	Disconnect from interrupt line
ØØ	READ	READ	Read from I/O Handler Task
ØØ	WRITE	WRITE	Write to I/O Handler Task
ØØ	DSKAL	DSKAL	Allocate disk storage
ØØ	DSKDAL	DSKDAL	Deallocate disk storage
ØØ	DSKPUT	DSKPUT	Put data on disk
ØØ	DSKGET	DSKGET	Get data from disk
ØØ	ATTACH	ATTACH	Attach Device-Unit to a Task
ØØ	DETACH	DETACH	Detach Device-Unit from a Task
ØØ	SEEK	SEEK	Seek file
ØØ	ENTER	ENTER	Enter file
ØØ	DELETE	DELETE	Delete file
ØØ	CLOSE	CLOSE	Close file
ØØ	HINF	HINF	Handler information
21	DISABLE	DISABL	Disable Task
22	ENABLE	ENABLE	Enable Task
15	FIX	FIX	Fix Task in core
16	UNFIX	UNFIX	Unfix Task in core

SYSTEM MACROS

MACRO CALL	FORTRAN CALL	SYSTEM FUNCTION
DECLAR	DECLAR	Declare a Significant Event
TIME	TIME	Obtain Time from Executive
DATE	DATE	Obtain Time and Date from Executive
INTENTRY	----	Interrupt Entry (register save routine)
INTEXIT	----	Interrupt Exit (register restore routine)

4.3 DESCRIPTION OF DIRECTIVES

The RSX Directives are implemented as CAL instructions* which point to argument blocks (CAL Parameter Blocks). As a convenience to the assembly language programmer, the Directives have been defined as macro instructions and are commonly referred to as System Macros. FORTRAN Tasks use Directives through standard CALL statements to a group of FORTRAN Library Routines which themselves issue the Directives.

* See Glossary

The RSX System allows Task names of one to six characters in length, however, not more than five characters may be used in Task names in FORTRAN calls in order to comply with PDP-15 FORTRAN conventions.

Most Macro and FORTRAN calls to RSX Directives include the Task priority and its Event Variable (EV). The Task priority is indicated by a decimal number between 1 (highest priority) and 512 (lowest priority). A priority value of zero instructs the Executive that the Task's default priority is to be used. Directives usually have EV's associated with them which provide information concerning the results after the issuance of the Directive. If an EV is not specified when issuing a Directive, the Executive does not attempt to provide any information concerning the operation. Event Variables are set positive upon successful completion, zero when the request is pending, and negative to indicate rejection or failure. Appendix D provides a complete list and explanation of the EV values returned by the system.

In the following sections which describe the Directives, square brackets of the form ([]) are used to specify optional arguments.

4.3.1 REQUEST

This Directive instructs the Executive to initiate the execution of a Task based on an indicated software priority. The actual execution depends upon the priority and partition availability. The Event Variable, or the Event Variable and priority, may be omitted. A Task cannot request itself. REQUEST may be issued from an interrupt service routine. Event Variables returned are: +1, -2Ø1, -2Ø2, -2Ø4, and -777.

```
SYSTEM MACRO:   REQUEST TSKNAM[,P[,EV]]

Variables:      TSKNAM = Name of Task (1 - 6 characters)
                P      = Task Priority (1 -512)
                EV     = Event Variable Address
```

Examples: Request the execution of SCAN whose default priority is 48.

```
REQUEST SCAN,Ø,EV
      (or)
REQUEST SCAN,48,EV
```

Request the execution of SCAN at a priority of 2Ø.

```
REQUEST SCAN,2Ø,EV
      (or)
REQUEST SCAN,2Ø        /In this case the testing of the
                       /Event Variable is not desired.
```

FORTRAN CALL: CALL_REQST(nHTSKNAM,IP[,IEV])

Variables:

n	=	Number of characters in Task Name
TSKNAM	=	Name of Task (1 - 5 characters)
IP	=	Task Priority (1 - 512) May be either a Variable name or a direct constant.
IEV	=	Event Variable

Examples: Request the execution of SCAN whose default priority is 48.

```
CALL REQST (4HSCAN,Ø,IEV)
      (or)
IP=48
CALL REQST (4HSCAN,IP,IEV)
```

Request the execution of SCAN at default priority and no Event Variable is desired.

```
CALL REQST (4HSCAN,Ø)
```

4.3.2 SCHEDULE

This Directive instructs the Executive to initiate the execution of a Task at an absolute time of day and to reactivate it continuously at a specified interval thereafter based on the indicated priority. If the Reschedule Interval is zero, the Task is executed only once at the time of day specified. A Task may SCHEDULE itself, however, the SCHEDULE Directive may not be issued from an interrupt service routine. Event Variables returned are: +1, -2Ø1, -2Ø3, -2Ø4, and -777.

SYSTEM MACRO: SCHEDULE_TSKNAM,SH,SM,SS[,RI,RU[,P[,EV]]]

Variables:

TSKNAM	=	Name of Task (1 - 6 characters)
SH	=	Schedule Hour (Ø - 23)
SM	=	Schedule Minute (Ø - 59)

```
SS      = Schedule Second (Ø - 59)
RI      = Reschedule Interval (up to one day)
RU      = Reschedule Units (1=Ticks, 2=Seconds,
          3=Minutes, and 4=Hours)
P       = Task Priority (1 - 512)
EV      = Event Variable Address
```

Examples: Schedule Task SCAN to run at 4:3Ø P.M. and every 5 minutes thereafter at a priority of 2ØØ.

```
SCHEDULE SCAN,16,3Ø,Ø,5,3,2ØØ,EV
         (or)
SCHEDULE SCAN,16,3Ø,Ø,5,3        /If Task's default
                                 /priority was 2ØØ and
                                 /Event Variable was
                                 /not desired.
```

Schedule Task ALPHA to run at its default priority at 7:15 A.M. with no rescheduling and no Event Variable.

```
SCHEDULE ALPHA,7,15
```

FORTRAN CALL: CALL_SCHED(nHTSKNAM,IT,IP[,IEV])

Variables:

```
n      = Number of characters in Task Name
TSKNAM = Name of Task (1 - 5 characters)
IT     = Name of 5 word (integer) array to describe
         the time of scheduling and rescheduling.
         The array is described below:
             IT(1) = Schedule Hour (Ø - 23)
             IT(2) = Schedule Minute (Ø - 59)
             IT(3) = Schedule Second (Ø - 59)
             IT(4) = Reschedule Interval (up to one
                     day)
             IT(5) = Reschedule Units (1=Ticks,
                     2=Seconds, 3=Minutes, and
                     4=Hours)
IP     = Task Priority (1 - 512)
IEV    = Event Variable
```

Examples: This example is equivalent to the first example in this section using FORTRAN.

```
DIMENSION IT(5)
IT(1) = 16
IT(2) = 3Ø
IT(3) = Ø
IT(4) = 5
IT(5) = 3
IP    = 2ØØ
IEV   = Ø
CALL SCHED(4HSCAN,IT,IP,IEV)
          (or)
CALL SCHED(4HSCAN,IT,2ØØ,IEV)
```

Schedule SCAN as above with no Event Variable and at its default priority.

```
IP = Ø
CALL SCHED(4HSCAN,IT,IP)
```

To schedule SCAN only once, set the reschedule interval equal to zero.

```
IT(4) = Ø
IP    = Ø
CALL SCHED(4HSCAN,IT,IP)
```

4.3.3 RUN

This Directive instructs the Executive to initiate the execution of a Task at a specified time interval from the time that the Directive is issued and reactivate the Task continuously at a specified interval thereafter. If the Reschedule Interval is zero, the Task is executed only once. A Task may use this Directive to reschedule itself, but the Directive may not be issued from an interrupt service routine. Event Variables returned are: +1, -2Ø1, -2Ø3, -2Ø4, and -777.

SYSTEM MACRO: RUN_TSKNAM,SD,SU[,RI,RU[,P[,EV]]]

Variables:
- TSKNAM = Name of Task (1 - 6 characters)
- SD = Schedule Delta time from now (up to one day)
- SU = Delta Units (1=Ticks, 2=Seconds, 3=Minutes, and 4=Hours)
- RI = Reschedule Interval (up to one day)
- RU = Reschedule Units (1=Ticks, 2=Seconds, 3=Minutes, and 4=Hours)
- P = Task Priority (1 - 512)
- EV = Event Variable Address

Examples: Run the Task INITS 5 seconds from now and every 1Ø minutes thereafter at priority of 512.

```
RUN INITS,5,2,1Ø,3,512,EV
```

Run the same Task at its default priority with no Event Variable specified.

```
RUN INITS,5,2,1Ø,3
```

FORTRAN CALL: CALL_RUN(nHTSKNAM,IT,IP[,IEV])

Variables:
- n = Number of characters in Task Name
- TSKNAM = Name of Task (1 - 5 characters)
- IT = Name of 4 word (integer) array to describe the time of scheduling and rescheduling. The array is described below:
  - IT(1) = Schedule Delta time from now (up to one day)

```
                    IT(2) = Delta Schedule Units (1=Ticks,
                            2=Seconds, 3=Minutes, and
                            4=Hours)
                    IT(3) = Reschedule Interval (up to one
                            day)
                    IT(4) = Reschedule Units (1=Ticks,
                            2=Seconds, 3=Minutes, and
                            4=Hours)
          IP      = Task Priority (1 - 512)
          IEV     = Event Variable
```

Example: Run the Task INITS 5 seconds from now and every 10 minutes thereafter at a priority of 512.

```
DIMENSION IT(4)
IT(1) = 5
IT(2) = 2
IT(3) = 10
IT(4) = 3
IP    = 512
CALL RUN(5HINITS,IT,IP,IEV)
```

4.3.4 SYNC

This Directive causes the Executive to execute a Task at a specified interval after the next hour, minute, second, or tick and to reinitiate the Task continuously at a specified interval.

A Schedule Delta Time value of zero causes the named Task to be started on the next occurrence of the Synchronization Unit. A Reschedule Interval of zero causes the Task to be executed only once. This Directive may not be issued from an interrupt service routine. A Task may use SYNC to reschedule itself. Event Variables returned are: +1, -201, -203, -204, and -777.

```
SYSTEM MACRO:  SYNC_TSKNAM,SZ,SD,SU[,RI,RU[,P[,EV]]]

Variables:     TSKNAM = Name of Task (1 - 6 characters)
               SZ     = Synchronization Units (1=Ticks, 2=Seconds,
                        3=Minutes, and 4=Hours)
               SD     = Schedule Interval from synchronization
                        time (up to one day)
               SU     = Schedule Units (1=Ticks, 2=Seconds,
                        3=Minutes, and 4=Hours)
```

RI = Reschedule Interval (up to one day)
RU = Reschedule Units (1=Ticks, 2=Seconds, 3=Minutes, and 4=Hours)
P = Task Priority (1 - 512)
EV = Event Variable Address

Examples: Assuming the time is now 14:27:47, run Task FRED at 14:28:Ø9 at a priority of 2Ø and reschedule it every 4 minutes thereafter.

```
SYNC FRED,3,9,2,4,3,2Ø,SYNEV
```

Schedule the execution of SCAN 1Ø seconds after the next minute mark and reschedule it every hour thereafter at priority 21.

```
SYNC SCAN,3,1Ø,2,1,4,21,SYNEV
```

FORTRAN CALL: CALL_SYNC(nHTSKNAM,IT,IP[,IEV])

Variables:
n = Number of characters in Task Name
TSKNAM = Name of Task (1 - 5 characters)
IT = Name of 5 word (integer) array to describe the time of synchronization, scheduling, and rescheduling. The array is described below:
IT(1) = Synchronization Units (1=Ticks, 2=Seconds, 3=Minutes and 4=Hours)
IT(2) = Schedule Interval from synchronization time (up to one day)
IT(3) = Schedule Units (1=Ticks, 2=Seconds, 3=Minutes, and 4=Hours)
IT(4) = Reschedule Interval (up to one day)
IT(5) = Reschedule Units (1=Ticks, 2=Seconds, 3=Minutes, and 4=Hours)
IP = Task Priority (1 - 512)
IEV = Event Variable

Examples: Schedule the execution of Tasks FRED and SCAN as described in the examples for the System Macros.

```
INTEGER FREDEV,SCANEV,SCANP,FREDP
DIMENSION IT(5)
IT(1) = 3
IT(2) = 9
IT(3) = 2
IT(4) = 4
IT(5) = 3
FREDEV = Ø
FREDP = 2Ø
.
CALL SYNC(4HFRED,IT,FREDP,FREDEV)
SCANEV = Ø
SCANP  = 21
IT(2)  = 1Ø
IT(4)  = 1
```

```
      IT(5)  = 4
      .
12    CALL SYNC(4HSCAN,IT,SCANP,SCANEV)
C     INSURE BOTH SYNC REQUESTS WERE ACCEPTED
      IF(SCANEV.OR.FREDEV.LT.Ø) GO TO 1Ø
C     BOTH SYNC REQUESTS WERE ACCEPTED AT THIS POINT
      .
      .
C     REPORT FAILURE OF SYNC REQUESTS TO BE ACCEPTED TO
      CONSOLE OPERATOR.
1Ø    WRITE(3,11)
11    FORMAT(32H TASKS FRED OR SCAN NOT SYNC'ED.//)
      STOP
      END
```

4.3.5 CANCEL

This Directive instructs the Executive to remove all entries which appear in the Clock Queue for a specified Task. Cancel may not be issued from an interrupt service routine and has no effect on an active Task. Event Variables returned are: +1 and -2Ø1.

SYSTEM MACRO: CANCEL_TSKNAM[,EV]

Variables:
- TSKNAM = Name of Task (1 - 6 characters)
- EV = Event Variable Address

Example: Cancel the activation of Task SCAN.

```
      CANCEL SCAN,EV
```

FORTRAN CALL: CALL_CANCEL(nHTSKNAM[,IEV])

Variables:
- n = Number of characters in Task Name
- TSKNAM = Name of Task (1 - 5 characters)
- IEV = Event Variable

Example: Cancel the activation of Task SCAN

```
      CALL CANCEL(4HSCAN,IEV)
```

4.3.6 SUSPEND

This Directive instructs the Executive to suspend execution of the Task issuing this Directive. The Task remains active in its core partition but execution is not permitted until the system receives a RESUME Directive. The Executive ignores this Directive if it is

issued from an interrupt service routine.

SYSTEM MACRO: SUSPEND

FORTRAN CALL: CALL SUSPND

4.3.7 RESUME

This Directive instructs the Executive to resume execution of a Task which has been SUSPENDed. Task execution continues either at a specified Resumption Address or, if not specified, at the address immediately following the SUSPEND Directive. Event Variables returned are: +1, -2Ø2, and -2Ø5.

SYSTEM MACRO: RESUME_TSKNAM[,RA[,EV]]

Variables:
- TSKNAM = Name of Task (1 - 6 characters)
- RA = Resumption Address (octal)
- EV = Event Variable Address

Example: Resume Task TSKA at location RSTRT.

```
RESUME TSKA,RSTRT,EVA
```

FORTRAN CALL: CALL_RESUME(nHTSKNAM[,IEV])

Variables:
- n = Number of characters in Task Name
- TSKNAM = Name of Task (1 - 5 characters)
- IEV = Event Variable

Example: Resume Task TSKA.

```
CALL RESUME(4HTSKA,IEV)
```

Note: The RESUME subroutine permits a SUSPENDed Task to resume only at the location immediately following the CALL SUSPND statement.

4.3.8 MARK

This Directive instructs the Executive to clear a specified Event Variable and set it to a non-zero value after a specified time interval has elapsed. If the request is accepted, the Event Variable is

cleared. After the specified interval has elapsed, the Event Variable is set to +1 and a Significant Event is declared. The user may test the Event Variable as desired. The time interval indicates time from the execution of the Directive. The Mark Directive is ignored if issued from an interrupt service routine. Event Variables returned are: +1, -2Ø3, and -777.

SYSTEM MACRO: MARK_MI,MU,EV

Variables:
- MI = Delta Interval (up to one day)
- MU = Delta Units (1=Ticks, 2=Seconds, 3=Minutes, and 4=Hours)
- EV = Event Variable Address

Example: Set Event Variable TSTEV 5 minutes from now.

```
MARK 5,3,TSTEV
```

FORTRAN CALL: CALL_MARK(IT,IEV)

Variables:
- IT = Name of 2 word (integer) array to describe the time. The array is described below:
  - IT(1) = Delta Interval (up to one day)
  - IT(2) = Delta Units (1=Ticks, 2=Seconds, 3=Minutes, and 4=Hours)
- IEV = Event Variable

Example: Same as above example except in FORTRAN.

```
DIMENSION IT(2)
IT(1) = 5
IT(2) = 3
CALL MARK(IT,IEV)
```

4.3.9 WAITFOR

This Directive instructs the Executive to examine a specified Event Variable and, if zero, suspend execution of the issuing Task until the Event Variable is found to be non-zero. The Event Variable is examined at each occurrence of a Significant Event. Once a non-zero value is detected, the suspended Task is resumed (contingent on priority) at the address immediately following the Directive (CAL). If WAITFOR is issued from an interrupt service routine, it will be ignored.

SYSTEM MACRO: WAITFOR_EV

Variables: EV = Event Variable Address

Example: In this example, the Mark Directive requests that an Event Variable (MRKEV) be zeroed for 5 minutes and then set non-zero. The WAITFOR detects the fact that the Event Variable is zero and suspends Task execution until the value becomes non-zero (i.e., in 5 minutes). The Task is then resumed at the instruction immediately following the WAITFOR.

```
MARK 5,3,MRKEV
WAITFOR MRKEV
```

FORTRAN CALL: CALL_WAITFR(IEV)

Variables: IEV = Event Variable

Example: Same as preceding example except in FORTRAN.

```
      DIMENSION IT(2)
      IT(1) = 5
      IT(2) = 3
C     IEV WILL AUTOMATICALLY BE CLEARED BY MARK
      CALL MARK(IT,IEV)
      CALL WAITFR(IEV)
```

4.3.10 WAIT

This Directive instructs the Executive to suspend execution of the issuing Task until the next Significant Event occurs. The Task is resumed (contingent upon priority) at the next Significant Event and continues at the location immediately following the WAIT. It is the responsibility of the Task issuing the WAIT to determine the meaningfulness of the Significant Event which caused it to be resumed. This Directive is ignored if issued from an interrupt service routine.

SYSTEM MACRO: WAIT

FORTRAN CALL: CALL_WAIT

4.3.11 EXIT

This Directive causes the Executive to terminate execution of the issuing Task. If the issuing Task is not fixed-in-core (via FIX) the

core partition occupied by the Task becomes available to other Tasks. This Directive should not be issued until all transfers to the partition (e.g. I/O transfers, Task-to-Task transfers, Event Variable settings, etc.) have been completed. This Directive is ignored if issued from an interrupt service routine.

SYSTEM MACRO: EXIT

FORTRAN CALL: CALL_EXIT

4.3.12 CONNECT

This Directive instructs the Executive to create a linkage between a specified API (Automatic Priority Interrupt) trap address and a specified entry point to an interrupt service routine (there is one trap address for each of the 32 API lines in the PDP-15). Event Variables returned are: +1, -3Ø1, and -3Ø2. The following is a list of API line assignments

LINE NUMBER	DEVICE	LINE NUMBER	DEVICE
Ø4	DECtape	24	Diskpack
Ø5	Magtape	25	Plotter
Ø6	(unused)	26	(unused)
Ø7	(unused)	27	(unused)
1Ø	Paper Tape Reader	3Ø	(unused)
11	*Clock	31	UDC15
12	Power Failure	32	AFC15
13	Memory Parity	33	(unused)
14	VP15 Display	34	*LT15/LT19 Printers
15	Card Reader	35	*LT15/LT19 Keyboards
16	Line Printer	36	DECtape (additional)
17	A/D Converter	37	Dataphone(additional)
2Ø	Interprocessor Buffer	4Ø	*Console TTY (Keyboard)
21	(unused)	41	*Console TTY (Printer)
22	Dataphone	42	Paper Tape Punch
23	*Disk	43	Memory Protect

*These lines are always connected to the system.

SYSTEM MACRO: CONNECT_LN,CL[,EV]

Variables:
LN	=	Interrupt line Number (octal)
CL	=	Entry Address of interrupt service routine
EV	=	Event Variable Address

Example: Connect an interrupt service routine for an A/D Converter (entry point called ADINT) to interrupt line 31.

```
CONNECT 31,ADINT,ADEV
```

FORTRAN CALL: No subroutine is provided for this Directive since FORTRAN is not an appropriate language for writing interrupt handling routines.

4.3.13 DISCONNECT

This Directive instructs the Executive to remove the linkage created between an Automatic Priority Interrupt trap address and an interrupt service routine entry by the CONNECT Directive. Event Variables returned are: +1, -3Ø1, and -3Ø2.

SYSTEM MACRO: DISCONNECT_LN,CL,EV

Variables:
LN	=	Interrupt Line Number (octal)
CL	=	Entry Address of interrupt service routine
EV	=	Event Variable Address

Example: Disconnect the A/D Converter from interrupt line 31.

```
DISCONNECT 31,ADINT,ADEV
```

FORTRAN CALL: No subroutine is provided for this Directive since FORTRAN is not an appropriate language for writing interrupt handling routines.

4.3.14 READ

READ generates a form of the QUEUE I/O Directive which causes input of formatted ASCII or Binary to a specified buffer via the I/O Device Handler Task assigned to the indicated Logical Unit Number. The Event Variable specified is set to zero when the request is accepted and subsequently is set to an appropriate value indicating the status

of the operation. Event Variables returned are: +2, +1, -5, -7, -11, -12, -16, -23, -1Ø1, -1Ø2, -1Ø3, and -777.

SYSTEM MACRO: READ_LUN,MODE,BUFF,SIZE[,EV]

Variables:
- LUN = Logical Unit Number (decimal)
- MODE = I/O Data Mode (Ø=IOPS BINARY, 1=IMAGE BINARY, 2=IOPS ASCII, and 3=IMAGE ASCII)
- BUFF = Starting Address of user's buffer
- SIZE = Maximum number of words to transfer (octal)
- EV = Event Variable Address

Example: Read 256 (decimal) words in IOPS ASCII from the device assigned to LUN 3 and store them in a buffer called TXTBF.

```
READ 3,2,TXTBF,4Ø2,INDEV
WAITFOR INDEV
```

FORTRAN CALL: No subroutine is necessary to implement this Directive. The standard READ statement as described in the PDP-15 FORTRAN IV Manual is used.

Example:

```
      DIMENSION TXTBF(256)
      READ (3,1Ø) TXTBF
1Ø    FORMAT(256A1)
```

4.3.15 WRITE

WRITE generates a form of the QUEUE I/O Directive which causes output of formatted ASCII or Binary from a specified buffer to the I/O Device Handler Task assigned to the indicated Logical Unit Number. The Event Variable specified is set to zero when the request is accepted and subsequently set to an appropriate value indicating the status of the operation. Event Variables returned are: +2, +1, -6, -7, -11, -12, -15, -16, -23, -1Ø1, -1Ø2, -1Ø3, and -777.

SYSTEM MACRO: WRITE_LUN,MODE,BUFF[,EV]

Variables:
- LUN = Logical Unit Number (decimal)
- MODE = I/O Data Mode (Ø=IOPS BINARY, 1=IMAGE BINARY, 2=IOPS ASCII, and 3=IMAGE ASCII)
- BUFF = Starting Address of user's buffer
- SIZE = Maximum number of words to transfer (octal)
- EV = Event Variable Address

FORTRAN CALL: No subroutine is necessary to implement this Directive. The standard WRITE statement as described in the PDP-15 FORTRAN IV Manual is used.

```
Example:      DIMENSION TXTBF(256)
              WRITE(6,1Ø)TXTBF
        1Ø    FORMAT(256A1)
```

Write out "WARNING, XFC OSCILLATING AT" followed by a frequency on LUN 3.

```
              WRITE(3,1Ø)IFQ
        1Ø    FORMAT(28H WARNING, XFC OSCILLATING AT, I6//)
```

4.3.16 DSKAL

DSKAL generates a form of the QUEUE I/O Directive to reserve a disk storage area of a specified size. If the space is available, the starting address, physical disk number, and actual amount of space allocated[1] (in increments of 128 decimal words) is returned to a Control Table which is contained within the issuing Task or in a COMMON Block. The actual allocation does not occur instantaneously; hence, one must test the Event Variable to determine completion. Event Variables returned are: +1, -15, -1Ø4, and -777.

SYSTEM MACRO: DSKAL_CTB[,EV]

Variables:

CTB = Address of a Control Table of the following format:

- Word 1: Desired amount of disk storage which is replaced by the actual amount allocated if allocated.
- Word 2: Physical disk unit number (returned at completion of operation).
- Word 3: Absolute starting address of the space allocated relative to the physical disk unit number (returned at completion of the operation).

EV = Event Variable address

[1] which may exceed the amount requested.

Example: Request 7ØØ decimal words of disk storage. (Since allocation is given in increments of 128 decimal words, the actual allocation will be 768 decimal words.)

```
LAC     (1274)      /SETUP FIRST WD OF CONT TBL.
DAC     CTB+0
DSKAL   CTB,EV      /REQUEST ALLOCATION OF 700
WAITFOR EV          /WORDS OF DISK STORAGE.
```

FORTRAN CALL: CALL_DSKAL (ICTB,NW[,IEV])

Variables:

ICTB = Control Table (integer array). The Control Table is described below:

- Word 1: Actual amount of space allocated (returned at completion of the operation).
- Word 2: Physical disk unit number (returned at completion of operation).
- Word 3: Absolute starting address of the space allocated relative to the physical disk unit number (returned at completion of the operation).

IEV = Event Variable

NW = Desired storage in words

Example: Same as above except in FORTRAN.

```
DIMENSION ICTB(3)
CALL DSKAL (ICTB,700,IDKEV)
CALL WAITFR (IDKEV)
```

Note: Space will not be allocated across disk unit bounds (i.e., from one unit to another). No more than 13Ø,944 words may be allocated by a single DSKAL command.

4.3.17 DSKDAL

DSKDAL generates a form of the QUEUE I/O Directive to release a disk storage area, which had previously been allocated by DSKAL, from the Disk. Event Variables returned are: +1, -15, -1Ø4, and -777.

SYSTEM MACRO: DSKDAL_CTB[,EV]

Variables:

CTB = Control Table Address. This address should be the same as that used by DSKAL which originally allocated the space.

EV = Event Variable Address

Example: Request deallocation of the disk storage allocated in the previous section. There is no concern for when the disk space is actually freed.

```
DSKDAL  CTB
```

FORTRAN CALL: CALL_DSKDAL(ICTB[,IEV])

Variables:
- ICTB = Control Table (integer array). This address should be the same as that used by DSKAL which originally allocated the space.
- IEV = Event Variable

Example: Same as above except in FORTRAN

```
DIMENSION ICTB(3)
.
.
CALL DSKDAL(ICTB,IEV)
```

4.3.18 DSKPUT

DSKPUT generates a form of the QUEUE I/O Directive to output data onto the disk from a specified area in core. This Directive is used when total freedom in data structuring and random access capabilities are desired. Event Variables returned are: +1 and -N, where N is the contents of the disk status register if a disk error occurs.

SYSTEM MACRO: DSKPUT_CTB[,EV]

Variables:
- CTB = Address of a Control Table of the following format:
  - Word 1: Disk unit number
  - Word 2: Starting address on disk
  - Word 3: Starting address in core
  - Word 4: Length of transfer in words
- EV = Event Variable address

FORTRAN CALL: CALL_DSKPUT(ICTA,IOA,NW,ARRAY[,IEV])

Variables:

ICTA	=	Device Control Table (integer array). This array must be the same as that used to allocate the space onto which the data is being written since this uses information in the Control Table obtained via DSKAL
IOA	=	Disk offset address. The relative position (in words) within an array at which the transfer to the disk is to begin.
NW	=	Number of words (decimal) to transfer.
ARRAY	=	The name of the array containing the data to be transferred.
IEV	=	Event Variable

Example: Allocate 128Ø decimal words of disk storage and write out 256 words on the disk from BUF. Writing on disk is to begin 128 words beyond the starting address of the beginning of the disk storage area.

```
DIMENSION ICTA(3),BUF(256)
CALL DSKAL(ICTA,128Ø,IDKEV)
CALL WAITFR(IDKEV)
CALL DSKPUT(ICTA,128,256,BUF,IDKEV)
```

4.3.19 DSKGET

DSKGET generates a form of the QUEUE I/O Directive to read data from the disk into a specified area in core. This Directive is used where total freedom in data structuring and random access capabilities are desired. Event Variables returned are: +1 and -N, where N is the contents of the disk status register if a disk error occurs.

SYSTEM MACRO: DSKGET_CTB[,EV]

Variables:

CTB	=	Address of a Control Table of the following format: Word 1: Disk unit number Word 2: Starting address on disk Word 3: Starting address in core Word 4: length of transfer in words
EV	=	Event Variable address

FORTRAN CALL: CALL_DSKGET(ICTA,IOA,NW,ARRAY[,IEV])

Variables:

- ICTA = Device Control Table (integer array). Array must be the same as that used to allocate the space from which the data is being read since this uses information in the Control Table obtained via DSKAL.
- IOA = Disk offset address. The relative position (in words) within an array at which the transfer from the disk is to begin.
- NW = Number of words (decimal) to transfer.
- ARRAY = The name of the array where data is to be transferred.
- IEV = Event Variable

Example: Allocate 512 decimal words of disk storage and later read in the last 256 decimal words into BUF.

```
DIMENSION ICTA(3),BUF(256)
CALL DSKAL (ICTA,512,IEV)
CALL WAITFR(IEV)
.
.
CALL DSKGET(ICTA,256,256,BUF,IEV)
```

Example: This final FORTRAN example allocates 1024 words of disk storage, writes 256 words from four different arrays, later reads the last array of 256 words, and then deallocates the disk space and EXITs.

```
      COMMON BUF2(128),ICTA(3),BUF1(128),BUF3(128),BUF4(128)
C
C ---- ALLOCATE 1Ø24 WORDS OF DISK STORAGE
C
      CALL DSKAL (ICTA,1Ø24,IEV)
      CALL WAITFR (IEV)
C
C ---- INSURE ALLOCATION WAS MADE
C
      IF (IEV .GT. Ø) GO TO 2Ø
C
C ---- STORAGE NOT ALLOCATED, TYPE MESSAGE & EXIT
C
      WRITE (3,1Ø)
1Ø    FORMAT (2ØH ALLOCATION NOT MADE)
      CALL EXIT
C
C ---- ALLOCATION MADE, WRITE OUT ARRAYS
C
2Ø    CALL DSKPUT (ICTA,Ø,256,BUF1,IEV)
      CALL DSKCK (IEV)
      CALL DSKPUT (ICTA,256,256,BUF2,IEV)
      CALL DSKCK (IEV)
      CALL DSKPUT (ICTA,512,512,BUF3,IEV)
      CALL DSKCK (IEV)
```

```
      ...
      ...
      ...
C
C ---- READ IN LAST ARRAY FROM DISK
C
      CALL DSKGET (ICTA,768,256,BUF4,IEV)
      CALL DSKCK (IEV)
      ...
      ...
      ...
C
C ---- RELEASE DISK SPACE & EXIT
C
      CALL DSKDAL (ICTA)
      STOP
      END


      SUBROUTINE DSKCK (IEV)
      CALL WAITFR (IEV)
      IF (IEV .LT. Ø) GO TO 1Ø
      RETURN
1Ø    WRITE (3,2Ø)
2Ø    FORMAT (11H DISK ERROR)
      CALL EXIT
      END
```

4.3.20 ATTACH

ATTACH generates a form of the QUEUE I/O Directive which requests the exclusive use of an I/O device. Once the Directive is accepted, no other Task may use the device regardless of priority. All requests by other Tasks, however, will be queued and processed whenever the device becomes free (DETACHED). The REASSIGN MCR Function, however, overrides the ATTACH. Event Variables returned are: +1, -6, -24, -101, -102, -103, and -777.

```
SYSTEM MACRO:  ATTACH_LUN[,EV]

Variables:     LUN    = Logical Unit Number (decimal)
               EV     = Event Variable Address

Example:  Attach device assigned to LUN 32.

              ATTACH 32,ATEV

FORTRAN CALL:  CALL_ATTACH(LUN[,IEV])

Variables:     LUN    = Logical Unit Number
               IEV    = Event Variable

Example:  Same as above except in FORTRAN.

              CALL ATTACH(32,IEV)
```

4.3.21 DETACH

DETACH generates a form of the QUEUE I/O Directive which releases a device from the exclusive use of the issuing Task. Previous requests which were queued by the I/O Handler Task while ATTACHed will now be processed. The Task issuing the DETACH Directive must be the Task which ATTACHed the device. Event Variables returned are: +1, -6, -101, -102, -103, and -777.

```
SYSTEM MACRO:  DETACH_LUN[,EV]

Variables:     LUN    = Logical Unit Number (decimal)
               EV     = Event Variable Address
```

Example: Detach device assigned to LUN 23.

```
DETACH 23,DTEV
```

FORTRAN CALL: CALL_DETACH(LUN[,IEV])

Variables:
- LUN = Logical Unit Number
- IEV = Event Variable

Example: Same as above except in FORTRAN.

```
CALL DETACH(23,IEV)
```

4.3.22 SEEK (OPEN FILE FOR INPUT)

SEEK generates a form of the QUEUE I/O Directive which requests the I/O Handler Task assigned to the indicated Logical Unit Number to search the device's file directory for a specified file name. This Directive is used to initiate file-oriented transfers using the READ Directive. Once the SEEK has been accepted by the I/O device, it effectively attaches the LUN to the issuing Task. Event Variables returned are: +1, -6, -1Ø, -12, -13, -1Ø1, -1Ø2, -1Ø3, and -777.

SYSTEM MACRO: SEEK_LUN,FLNAM,EXT[,EV]

Variables:
- LUN = Logical Unit Number
- FLNAM = File name (1 - 6 characters)
- EXT = File name extension (1 - 3 characters)
- EV = Event Variable Address

Example: Search the directory of the file-oriented device associated with LUN 6 for a file named DATA SRC.

```
SEEK 6,DATA,SRC,EV
```

FORTRAN CALL: CALL_SEEK(LUN,nHFLNAM,nHEXT[,IEV])

Variables:
- LUN = Logical Unit Number
- n = Number of characters in file name or extension.
- FLNAM = File Name (1 - 5 characters)
- EXT = File Name Extension (1 - 3 characters)
- IEV = Event Variable

Example: Same as above except in FORTRAN.

```
      CALL SEEK(6,4HDATA,3HSRC,IEV)
C     WAIT FOR SEEK TO COMPLETE
      CALL WAITFR(IEV)
```

4.3.23 ENTER (OPEN FILE FOR OUTPUT)

ENTER generates a form of the QUEUE I/O Directive which requests the I/O Handler Task assigned to the indicated Logical Unit Number to search the device's file directory for a free Directory Entry Block in which to place the file name specified. This Directive is issued prior to issuing a WRITE Directive to a file-oriented device. The actual recording of the file name does not occur until the CLOSE. Once the ENTER has been accepted by the I/O device, it effectively attaches the LUN to the issuing Task. Event Variables returned are: +1, -6, -1Ø, -12, -14, -15, -1Ø1, -1Ø2, -1Ø3, and -777.

SYSTEM MACRO: ENTER_LUN,FLNAM,EXT[,EV]

Variables:

LUN	= Logical Unit Number (decimal)
FLNAM	= File Name (1 - 6 characters)
EXT	= File Name Extension (1 - 3 characters)
EV	= Event Variable Address

Example: Enter into the directory of the file-oriented device associated with LUN 6 the file name DATA SRC.

```
ENTER 6,DATA,SRC,EV
```

FORTRAN CALL: CALL_ENTER(LUN,nHFLNAM,nHEXT[,IEV])

Variables:

LUN	= Logical Unit Number
n	= Number of characters in file name or extension.
FLNAM	= File Name (1 - 5 characters)
EXT	= File Name Extension (1 - 3 characters)
IEV	= Event Variable

Example: Same as above except in FORTRAN.

```
CALL ENTER(6,4HDATA,3HSRC,IEV)
```

4.3.24 DELETE

DELETE generates a form of the QUEUE I/O Directive which requests the I/O Handler Task assigned to the indicated Logical Unit Number to remove the indicated file name from the device's file directory. Event Variables returned are: +1, -6, -1Ø, -12, -1Ø1, -1Ø2, -1Ø3, and -777.

SYSTEM MACRO: DELETE_LUN,FLNAM,EXT[,EV]

Variables:
LUN	=	Logical Unit Number (decimal)
FLNAM	=	File Name (1 - 6 characters)
EXT	=	File Name Extension (1 - 3 characters)
EV	=	Event Variable Address

Example: Delete the file DATA SRC from the directory of the file-oriented device associated with LUN 6.

```
DELETE 6,DATA,SRC,EV
```

FORTRAN CALL: CALL_DELETE(LUN,nHFLNAM,nHEXT[,IEV])

Variables:
LUN	=	Logical Unit Number
n	=	Number of characters in file name or extension.
FLNAM	=	File Name (1 - 5 characters)
EXT		File Name Extension (1 - 3 characters)
IEV	=	Event Variable

Example: Same as above except in FORTRAN.

```
CALL DELETE(6,4HDATA,3HSRC,IEV)
```

4.3.25 CLOSE

CLOSE generates a form of the QUEUE I/O Directive which instructs the appropriate I/O Handler Task that the issuing Task has completed an I/O operation to the named file which resides on the device. Once a CLOSE is issued, subsequent transfers to or from the CLOSEd file are not possible until an appropriate SEEK or ENTER is again issued. Event Variables returned are: +1, -6, -11, -12, -15, -1Ø1, -1Ø2, -1Ø3, and -777.

SYSTEM MACRO: CLOSE_LUN,FLNAM,EXT[,EV]

Variables:
LUN	=	Logical Unit Number (decimal)
FLNAM	=	File Name (1 - 6 characters)
EXT	=	File Name Extension (1 - 3 characters)
EV	=	Event Variable Address

Example: Close the file DATA SRC on the file-oriented device associated with LUN 6.

```
CLOSE 6,DATA,SRC,EV
```

FORTRAN CALL: CALL_CLOSE(LUN,nHFLNAM,nHEXT[,IEV])

Variables:

LUN	=	Logical Unit Number
n	=	Number of characters in file name or extension.
FLNAM	=	File Name (1 - 5 characters)
EXT	=	File Name Extension (1 - 3 characters)
IEV	=	Event Variable

Example: Same as above except in FORTRAN.

```
CALL CLOSE(6,4HDATA,3HSRC,IEV)
```

4.3.26 HANDLER INFORMATION

This Directive provides rudimentary information about the physical device and the I/O handler associated with a particular Logical Unit Number (LUN). Handler information is coded into a single word, which is stored in the requestor's Event Variable as follows:

Bit Ø	UNUSED	This bit is unused to allow a handler to return a value of -6 if this function was not implemented.
Bit 1	INPUT	Set to 1 if data can be input from the device to the computer.
Bit 2	OUTPUT	Set to 1 if data can be output from the computer to the device.
Bit 3	FILE-ORIENTED	Set to 1 if the I/O handler treats the device as being "file-oriented". A device is "file-oriented" if SEEK and ENTER are required prior to READ and WRITE, respectively. "File-oriented" implies, but does not guarantee, the existence of a file directory or that the device is bulk or mass storage.
Bits 4 thru 11	UNIT	Unit number.
Bits 12 thru 17	DEVICE CODE	These six bits allow up to 63 decimal devices (zero is not a legal device code). The codes listed below are fixed for standard DEC devices. Users should assign codes to their own devices starting with 63 and working towards lower numbers.

1 TT -- The TTY terminals (console, LT15, and LT19)
2 DK -- The RF15 fixed-head DECdisk
3 DP -- The RPØ2 disk pack
4 DT -- The TCØ2D DECtape
5 MT -- The TC59 MAGtape

```
                        6   PR -- The PC15 Paper Tape Reader
                        7   CD -- The CRØ3B Card Reader
                        1Ø  PP -- The PC15 Paper Tape Punch
                        11  LP -- The LP15 Line Printer
                        12  VP -- The VP15 Storage Scope
                        13  VT -- The VT15 Display

SYSTEM MACRO:  HINF_LUN,EV

Variables:     LUN    = Logical Unit Number (decimal)
               EV     = Event Variable Address

FORTRAN CALL:  CALL_HINF(LUN,IEV)

Variables:     LUN    = Logical Unit Number
               IEV    = Event Variable
```

4.3.27 DISABLE

This Directive causes the Executive to render the specified Task incapable of responding to other Directives except ENABLE. The Task is not deleted from the system. If the Task is active, it will continue to execute, however, schedule activations for that Task will be ignored when they come due. When the Task is subsequently ENABLEd, previously established rescheduling for the Task will continue in effect. Event Variables are: +1 and -2Ø1.

```
SYSTEM MACRO:  DISABLE_TSKNAM[,EV]

Variables:     TSKNAM = Name of Task (1 - 6 characters)
               EV     = Event Variable Address

Example:  Disable Task named SCAN.

              DISABLE SCAN,EV

FORTRAN CALL:  CALL_DISABL(nHTSKNAM[,IEV])

Variables:     n      = Number of characters in Task Name
               TSKNAM = Name of Task (1 - 5 characters)
               IEV    = Event Variable

Example:  Same as above except in FORTRAN.

              CALL DISABL(4HSCAN,IEV)
```

4.3.28 ENABLE

This Directive causes the Executive to restore the specified Task to its normal state (i.e., as it was before DISABLE was issued). Event Variables returned are: +1 and -2Ø1.

SYSTEM MACRO: ENABLE_TSKNAM[,EV]

Variables:
TSKNAM = Name of Task (1 - 6 characters)
EV = Event Variable Address

Example: Enable Task SCAN which is currently disabled.

```
ENABLE SCAN,EV
```

FORTRAN CALL: CALL_ENABLE(nHTSKNAM[,IEV])

Variables:
n = Number of characters in Task Name
TSKNAM = Name of Task (1 - 5 characters)
IEV = Event Variable

Example: Same as above except in FORTRAN.

```
CALL ENABLE(4HSCAN,IEV)
```

4.3.29 FIX

This Directive instructs the system to load an inactive Task into an available partition. The Task is not executed, but is fixed-in-core and may therefore respond rapidly to a request for execution. FIX does not wait for a Task to be loaded before setting EV to +1. The Directive may not be issued to an active Task. Event Variables returned are: +1, -2Ø1, -2Ø2, -2Ø4, -2Ø7, and -21Ø.

SYSTEM MACRO: FIX_TSKNAM[,EV]

Variables:
TSKNAM = Name of Task (1 - 6 characters)
EV = Event Variable Address

FORTRAN CALL: CALL_FIX(nHTSKNAM[,IEV])

```
Variables:      n      = Number of characters in Task Name
                TSKNAM = Name of Task (1 - 5 characters)
                IEV    = Event Variable
```

4.3.30 UNFIX

This Directive instructs the Executive to nullify a FIX Directive thereby freeing a partition for use by other Tasks. If UNFIX is issued to a Task which is currently running, the Task will be allowed to run to completion before the Directive becomes effective. Event Variables returned are: +1, -2Ø1, and -2Ø7.

```
SYSTEM MACRO:  UNFIX_TSKNAM[,EV]

Variables:     TSKNAM = Name of Task (1 - 6 characters)
               EV     = Event Variable Address

FORTRAN CALL:  CALL_UNFIX(nHTSKNAM[,IEV])

Variables:     n      = Number of characters in Task Name
               TSKNAM = Name of Task (1 - 5 characters)
               IEV    = Event Variable
```

4.3.31 DECLARE A SIGNIFICANT EVENT

The DECLAR Directive provides the means for declaring to the Executive that a Significant Event has occurred. The occurrence of a Significant Event signals the Executive to initiate a scan of the Active Task List with control passing to the Task having the highest priority. DECLAR is particularly useful for intertask signalling and synchronization.

```
SYSTEM MACRO:  DECLAR

FORTRAN CALL:  CALL_DECLAR
```

4.3.32 TIME

The TIME System Macro and FORTRAN subroutine obtain the time of day from the Executive's internal clock and deposit these values in three

locations specified by the issuing Task.

```
SYSTEM MACRO:  TIME_Hr,Min,Sec

Variables:     Hr     = Hours   (Ø - 23)
               Min    = Minutes (Ø - 59)
               Sec    = Seconds (Ø - 59)

FORTRAN CALL:  CALL_TIME(ITIME)

Variables:     ITIME  = Three word Integer array as follows:
                        ITIME(1) = Hours   (Ø - 23)
                        ITIME(2) = Minutes (Ø - 59)
                        ITIME(3) = Seconds (Ø - 59)
```

4.3.33 DATE

The DATE System Macro and FORTRAN subroutine obtain the current time (hours, minutes, and seconds) and date (month, day, and year) from the Executive's internal clock and calendar. The values obtained are deposited in six locations specified by the issuing Task.

```
SYSTEM MACRO:  DATE_Hr,Min,Sec,Mon,Day,Yr

Variables:     Mon    = Month   (1 - 12)
               Day    = Day     (1 - 31)
               Yr     = Year    (Ø - 99)
               Hr     = Hours   (Ø - 23)
               Min    = Minutes (Ø - 59)
               Sec    = Seconds (Ø - 59)

               Note:  The month and day reversed in European
                      format.

FORTRAN CALL:  CALL_DATE(IDATE)

Variables:     IDATE  = Six word Integer array as follows:
                        IDATE(1) = Month   (1 - 12)
                        IDATE(2) = Day     (1 - 31)
                        IDATE(3) = Year    (Ø - 99)
                        IDATE(4) = Hours   (Ø - 23)
                        IDATE(5) = Minutes (Ø - 59)
                        IDATE(6) = Secnnds (Ø - 59)
```

4.3.34 INTENTRY

The INTENTRY System Macro results in an entry to the Executive's Register

Save Routine. The Save Routine obtains the current contents of all active system registers, including the AC, Index and Limit Registers, first four Autoincrement Registers, and deposits them in a save area created by the Macro Assembler during expansion of the System Macro. This Directive may only be issued from within an interrupt service routine and must be the first instruction of the interrupt routine. The saved registers are restored by execution of the INTEXIT Directive. Appendix E provides a listing of all registers which are saved.

SYSTEM MACRO: INTENTRY_CL

Variable: CL = Interrupt service routine entry address. (connect location)

FORTRAN CALL: No subroutine is provided to implement this since FORTRAN is not an appropriate language for writing interrupt handling routines.

Example: See example in next section (4.3.35)

4.3.35 INTEXIT

The INTEXIT System Macro results in an entry into the Executive's Register Restore Routine. This routine restores all active registers saved by the INTENTRY Directive, Debreaks, and returns to the interrupted Task. INTEXIT may only be issued from within an interrupt service routine.

SYSTEM MACRO: INTEXIT_CL

Variable: CL = Interrupt service routine entry address (connect location)

FORTRAN CALL: No subroutine is provided to implement this since FORTRAN is not an appropriate language for writing interrupt handling routines.

Example: An interrupt service routine named ADINT for an A/D Converter is to use the INTENTRY and INTEXIT System Macros.

```
INTENTRY ADINT      /MUST BE PLACED AT THE ENTRANCE
                    /  TO THE INTERRUPT ROUTINE.
   .
   .                /SECTION TO SERVICE INTERRUPT.
   .
INTEXIT ADINT       /RESTORE REGISTERS, DEBREAK,
                    /  AND RETURN TO INTERRUPTED
                    /  TASK.
```

CHAPTER FIVE
TASK BUILDER

5.1 INTRODUCTION

The TASK BUILDER program, TKB, is an ADVANCED Software System's program used to build user's Tasks from relocatable binary files. TKB is quite similar to the CHAIN program allowing very elaborate overlay structures to be built.

The process of Task building is one where relocatable binary files are linked together along with library functions to constitute an executable Task that runs under the control of the Real-Time Monitor (RSX-15). A resultant Task is defined by its name (Task name), default run priority, core partition and common block requirements, and resident code. Once a Task has been built it may be incorporated into the real-time operating system under control of the Real-Time Monitor from DECtape or paper tape.

5.2 TASK BUILDER DESCRIPTION

The following description of TKB assumes the reader has a thorough understanding of the ADVANCED Software System CHAIN* program since only

*TKB and CHAIN have identical Input/Output Device Assignments and loading procedures. (See CHAIN & EXECUTE Manual DEC-15-YWZA-DN2)

minor differences exist between the two. Only areas which are not part of the CHAIN program will be amplified in this section. Answers to all questions, as for the CHAIN program, must end with an ALT MODE.

LIST OPTIONS

SZ to output size in load maps, GM to output Global Symbol & File names instead of program names in load maps, NM for no load map, and PAR & PAL for pause after outputting resident code and pause after outputting each Link.

NAME TASK

Identical to NAME XCT FILE in CHAIN.

SPECIFY DEFAULT PRIORITY

This is the default priority of the Task which will be assumed at INSTALL time. Default priority is optional and can be any number from 1 (highest priority) to 512 (lowest priority).

DESCRIBE PARTITION

This is the name of the core partition in which the Task is to be executed. The form of the partition description is: NAME(BASE ADDRESS,SIZE). The NAME is the name of a Partition defined in the RSX System, BASE is the octal start address of the partition, and SIZE is the size of the partition which the Task is to occupy.

DESCRIBE SYSTEM COMMON BLOCKS

These are the names of the Common Blocks which are referenced by the Task but are common to all Tasks in the RSX System. The form of the Common Blocks is: NAME(BASE ADDRESS,SIZE). NAME is the name of the Common Block defined in the RSX System, BASE is the octal start address of the Common Block, and SIZE is the maximum size of the Common Block in which data is to be placed. Additional Common Block descriptions (maximum of four) may be specified by separating the descriptions with commas. Blank and Named Common declared in FORTRAN programs will be included in the Task's partition block if not specified in a DESCRIBE SYSTEM COMMON BLOCKS description. Blank Common assumes the default name of .XX.

DEFINE RESIDENT CODE

Identical to CHAIN program.

DESCRIBE LINKS & STRUCTURE

Identical to CHAIN program.

5.3 EXAMPLE USING THE TASK BUILDER

```
 ↑C

KM9-15 V5A

$A DK1 -4/DT1 -6

$TKB


TASK BUILDER V1A

LIST OPTIONS
>SZ
NAME TASK
>SUM
SPECIFY DEFAULT PRIORITY
>40
DESCRIBE PARTITION
>P40.0(40000,15000)
DESCRIBE SYSTEM COMMON BLOCKS
>FLAG(36200,600)
DEFINE RESIDENT CODE
>MAIN,F1,F2,MAC
DESCRIBE LINKS & STRUCTURE
>
MAIN    40020-40047 00030
F1      40050-40117 00050
F2      40120-40255 00136
MAC     40256-40303 00026
WAFF.1  40304-40316 00013
DSGF.2  40317-40360 00042
EXIF.1  40361-40363 00003
.DA     40364-40432 00047
.DAA    40433-40501 00047
BCDIO   40502-43537 03036
.SS     43540-43617 00060
STO.3   43620-43631 00012
FIO.3   43632-44311 00460
OTS.5   44312-44440 00127
.SP.3   44441-44560 00120
INTEAE  44561-44674 00114
RELEAE  44675-45736 01042
.CB     45737-45756 00020

BLANK COMMON
.XX     45757-46266 00310

CORE REQ'D
        40000-46266 06267


KM9-15 V5A

$
```

```
 ↑C

KM9-15 V5A

$A DK1 -4/DT1 -6

$TKB


TASK BUILDER V1A

LIST OPTIONS
>SZ
NAME TASK
>STORE
SPECIFY DEFAULT PRIORITY
>35
DESCRIBE PARTITION
>P40.0(40000,15000)
DESCRIBE SYSTEM COMMON BLOCKS
>FLAG(36200,600)
DEFINE RESIDENT CODE
>STORE
DESCRIBE LINKS & STRUCTURE
>
STORE  40020-40317 00300
WAFF.1 40320-40332 00013
RUNF.2 40333-40404 00052
EXIF.1 40405-40407 00003
DSAF.2 40410-40433 00024
DSPF.2 40434-40475 00042
FTS.2  40476-40543 00046
.BC    40544-40617 00054
.EE    40620-40710 00071
.EF    40711-41026 00116
.EC    41027-41072 00044
.DA    41073-41141 00047
.DAA   41142-41210 00047
BCDIO  41211-44246 03036
.SS    44247-44326 00060
STO.3  44327-44340 00012
FIO.3  44341-45020 00460
OTS.5  45021-45147 00127
.SP.3  45150-45267 00120
INTEAE 45270-45403 00114
RELEAE 45404-46445 01042
.CB    46446-46465 00020

BLANK COMMON
.XX    46466-47000 00313

CORE REQ'D
       40000-47000 07001


KM9-15 V5A

$
```

CHAPTER SIX
SYSTEM CONFIGURATOR

6.1 INTRODUCTION

The RSX-15 System is supplied on DECtape (RSX COLD START MASTER TAPE) with each PDP-15/35 computer as a very generalized software package. The Master Tape of the system consists of the RSX Executive, Multi-Teletype Handler, Disk Handler, Monitor Console Routine(MCR) Function Tasks, and one running Task called the SYSTEM CONFIGURATOR. The SYSTEM CONFIGURATOR enables the user to tailor his software to fit his requirements.

The SYSTEM CONFIGURATOR allows the user to specify core size, disk size, number of Teletypes, clock frequency, Partition descriptions, system COMMON Block descriptions, and a description of peripheral I/O units. Partitions and COMMON Blocks may be defined anywhere between the top of the Executive (over the SYSTEM CONFIGURATOR) thru the top of core (as long as they do not overlap each other). Partition Blocks and the "Pool of Empty Nodes" are constructed in core above the 8K that has not been defined as a part of a Partition or COMMON Block.

After the Pool and Partition Blocks have been constructed, all Tasks recorded on DT-Ø that can be installed in the newly configured system, are installed. After installation from DT-Ø, the number of empty nodes in the Pool is typed out, and the system is left running.

6.2 INSTALLING THE RSX SYSTEM

When the user receives the RSX COLD START MASTER TAPE he should perform the following steps to configure the RSX Software to best fit his needs and requirements:

1) Mount the RSX COLD START MASTER TAPE onto DECtape unit zero (WRITE LOCK).

2) Read into location ØØØØØ the RSX DECTAPE BOOTSTRAP from the High Speed Paper Tape Reader. This will cause the Cold Start image to be read in from the Master Tape and the SYSTEM CONFIGURATOR started.

3) Answer all questions asked by the SYSTEM CONFIGURATOR (See section 6.3).

4) When the system configuration has completed* and the message "MCR>" is printed on the console Teletype, the user should install his own Tasks into the system and issue the "SAVE" MCR Function command to save an image of the new system on the disk.

5) To make a backup copy of the system, mount scratch tapes on DECtape units one and two (one at a time if only a limited number of DECtapes are available) and read into location ØØØØØ the "DISK TO DECTAPE" from the High Speed Paper Tape Reader to copy an image of the disk(s) onto the DECtapes. This backup system can be restored by reading in "DECTAPE TO DISK" from the High Speed Paper Tape Reader. NOTE: Two DECtapes are required for each disk unit saved. Writing begins on DECtape unit one for the first half of the first disk and automatically transfers to DECtape unit two when unit one is filled. This process is automatically repeated (from DECtape one to two) until all disks have been saved. (Rewinding and unloading DECtapes between save/restore operations is performed by the save and restore programs.) A similar procedure is used to restore the system from DECtape to disk. The tapes may be restored in any order with transfer beginning from DECtape unit one and continuing with DECtape unit two. Each DECtape will contain $131,Ø72_{1Ø}$ disk words followed by a descriptor block.

---

* The SYSTEM CONFIGURATOR is a one-time Task that automatically removes itself once the system has been configured.

6) Read into location ØØØØØ the RSX DISK (WARM START) BOOTSTRAP from the High Speed Paper Tape Reader. This will cause the restored RSX System to be brought in from the disk and the message "MCR>" to be printed on the console Teletype. The System is now ready to accept commands from the user.

6.3 STEP BY STEP SYSTEM CONFIGURATION PROCEDURE

QUESTION ANSWER

SPECIFY CORE SIZE>

16K, 2ØK, 24K, 28K, or 32K (Size of user's core memory)

SPECIFY NUMBER OF DISK UNITS>

1 to 8 (Number of physical disk units)

SPECIFY NUMBER OF TTY'S>

1 to 17 (Number of Teletypes connected to the system.)

SPECIFY NUMBER OF CLOCK TICKS PER SECOND>

1 - 1ØØØ (This is the line frequency used to set the Real-Time Clock frequency in the RSX Executive, and will normally be 5Ø or 6Ø.)

DEFINE PARTITIONS "NAME(BASE,SIZE)"

These are the names (NAME) of all partitions in the System along with their base addresses (BASE) and sizes (SIZE). A line with only a terminator (carriage return or ALTMODE) will terminate the response.

DEFINE SYSTEM COMMON BLOCKS "NAME(BASE,SIZE)"

These are the names of COMMONs to be used for inter-task communication or extra-task data storage. Core is permanently allocated and these COMMONs are always available. The Names, Base Addresses, and Sizes are specified. A line with only a terminator (carriage return or ALTMODE) terminates the response.

SPECIFY DEVICE NAMES AND UNIT NUMBERS (ONE PER LINE)

List only devices which will be used by the user. Device names are two characters in length followed by a unit number. (The Teletypes and Disk are specified in the system.) The following are names of devices for which I/O Handler Tasks are supplied with the system:

LP = Line Printer
DTn = DECtape (n=Ø to 7)
PR = High Speed Paper Tape Reader
PP = High Speed Paper Tape Punch

A line with only a terminator (carriage return or ALTMODE) terminates the response.

INSTALLATION OF TASKS FROM DTØ

This process requires no response from the user. All Tasks on DTØ that can be installed, are installed in the System at the default priority defined at Task Building time. When all Tasks have been installed, the CONFIGURATOR continues with:

nnnnn NODES IN POOL

This is the number (nnnnn) of empty nodes (in decimal) in the POOL available for queueing and scheduling. The CONFIGURATOR continues with:

SYSTEM IS RUNNING

This indicates to the user that the system is running and the Resident MCR responds by typing "MCR>". The System is now ready to accept user's commands from the console Teletype.

6.4 EXAMPLE OF A SYSTEM CONFIGURATION PROCEDURE

```
RSX SYSTEM CONFIGURATION

SPECIFY CORE SIZE >24K

SPECIFY NUMBER OF DISK UNITS >1

SPECIFY NUMBER OF TTY'S >4

SPECIFY NUMBER OF CLOCK TICKS PER SECOND >60

DEFINE PARTITIONS  "NAME(BASE,SIZE)"
 >MCR(10000,1600)
 >IO.1(11600,3000)
 >IO.2(35200,1000)
 >P14.6(14600,3200)
 >P21.0(21000,5500)
 >P26.5(26500,6500)
 >P40.0(40000,15000)
 >

DEFINE SYSTEM COMMON BLOCKS  "NAME(BASE,SIZE)"
 >.XX(20000,700)
 >FLAG(36200,600)
 >
SPECIFY DEVICE NAMES & UNIT NUMBERS (ONE PER LINE)
 >DT0
 >DT1
 >DT2
 >DT3
 >PR
 >PP
 >

INSTALLATION OF TASKS FROM DT-0

00171 NODES IN POOL

SYSTEM IS RUNNING


MCR>SAVE
```

6.5 DESCRIPTION OF SYSTEM CONFIGURATOR ERROR MESSAGES

††† WOULD OVERFLOW ADJACENT AREAS -- RETYPE

A Partition and/or COMMON Block has overflowed in an adjacent area. The user must redefine the COMMON or Partition.

††† SYNTAX ERR AT "X" -- RETYPE

The illegal character "X" was found in the command string, retype the line.

††† INVALID SIZE -- RETYPE

Illegal core size. Must be either 16K, 20K, 24K, 28K, or 32K.

††† INVALID NUMBER -- RETYPE

Illegal number of disk units, Teletype units, or clock frequency setting.

††† NAME ALREADY USED -- RETYPE

The name of the Partition or COMMON Block is already defined in the system. Partitions and COMMON Blocks may not have the same name.

††† DEVICE NAME/UNIT ERR -- RETYPE

Device name and unit already defined.

*** DISK READ ERR

A disk read error has occurred. The system will halt and wait for the user to manually depress the continue switch to retry the disk read.

*** DISK WRITE ERR

A disk write error has occurred. The system will halt and wait for the user to manually depress the continue switch to retry the disk write.

*** RE-ENTRANT ECO PACKAGE NEEDED

The user's machine does not have the RE-ENTRANT ECO PACKAGE required to run RSX. Contact your local field service office.

*** INSUFFICIENT FREE CORE -- RE-STRUCTURE

Insufficient free core for storage of all Partition Blocks or out of nodes in the Pool.

TASK "XXXXXX" NOT INSTALLED, TASK ALREADY IN SYSTEM

The Task XXXXXX is already installed in the system.

TASK "XXXXXX" NOT INSTALLED, PARTITION NOT IN SYSTEM

The Task XXXXXX was built for a partition which is not defined in the system.

TASK "XXXXXX" NOT INSTALLED, TASK WOULD OVERFLOW PARTITION

The Task XXXXXX is larger than the partition defined for it in the system.

TASK "XXXXXX" NOT INSTALLED, OUT OF DISK STORAGE

The Task XXXXXX is larger than the amount of available Disk storage required to install the Task.

TASK "XXXXXX" NOT INSTALLED, INPUT CHECKSUM ERR

An input checksum error occurred while installing Task XXXXXX from DECtape.

TASK "XXXXXX" NOT INSTALLED, INPUT PARITY ERROR

An input parity error occurred while installing Task XXXXXX from DECtape.

TASK "XXXXXX" NOT INSTALLED, SYSTEM COMMON BLOCK ERR

COMMON Block not defined in system or COMMON Block BASE and/or SIZE specified incorrectly (to the Task Builder).

TASK "XXXXXX" NOT INSTALLED, READ ERROR

An illegal DECtape block number was found or a DECtape error exists.

TASK "XXXXXX" NOT INSTALLED, NO DEFAULT PRIORITY

The Task XXXXXX was not given a default priority at Task Building time. The INSTALL MCR Function can be used to install the Task once System Configuration has completed.

CHAPTER SEVEN
SYSTEM ORGANIZATION

7.1 INTRODUCTION

The RSX System is organized into several units consisting of: The Executive, Partitions, Partition Blocks, System COMMON Blocks, several linked lists, and a pool of empty list nodes. The Executive, or heart of the system, lies entirely in the lower 4K memory bank and consists of the Resident MCR Task, Teletype and Disk I/O Handler Tasks, and assorted routines to properly carry out the functions of a real-time operating system. The remaining area of core memory is available for Partitions, Partition Blocks, System COMMON Blocks, and the Pool of Empty Nodes. The following sections describe the system in more detail.

7.2 RSX BOOTSTRAPS

The RSX DECTAPE BOOTSTRAP is read in at location ØØØØØ and starts a DECtape to core transfer from tape block zero into core location 3Ø. The size of the image loaded is the same as the image that was recorded and is determined by the word count and current address registers 3Ø & 31. This bootstrap is normally used to initiate a COLD START, but can also be used to initiate a WARM START of a system that does not use the disk. i.e., to load an image of a system where all Tasks have been

fixed-in-core. viz., an emergency system in case of disk failure.

The RSX DISK (WARM START) BOOTSTRAP is a program used to restore a core image of the system (recorded by a SAVE MCR Function), from disk unit zero into core memory.

The DISK bootstrap is read into location ØØØØØ, clears the disk controller, and begins transfer starting from the beginning of disk zero and core location ØØØ3Ø. Transfer continues until the entire core memory has been restored. When the system has been successfully restored, control is transferred to the address specified by R1 (absolute location 10_8 in the System Communications table) causing the system to be started.

7.3 RSX MEMORY MAP USAGE

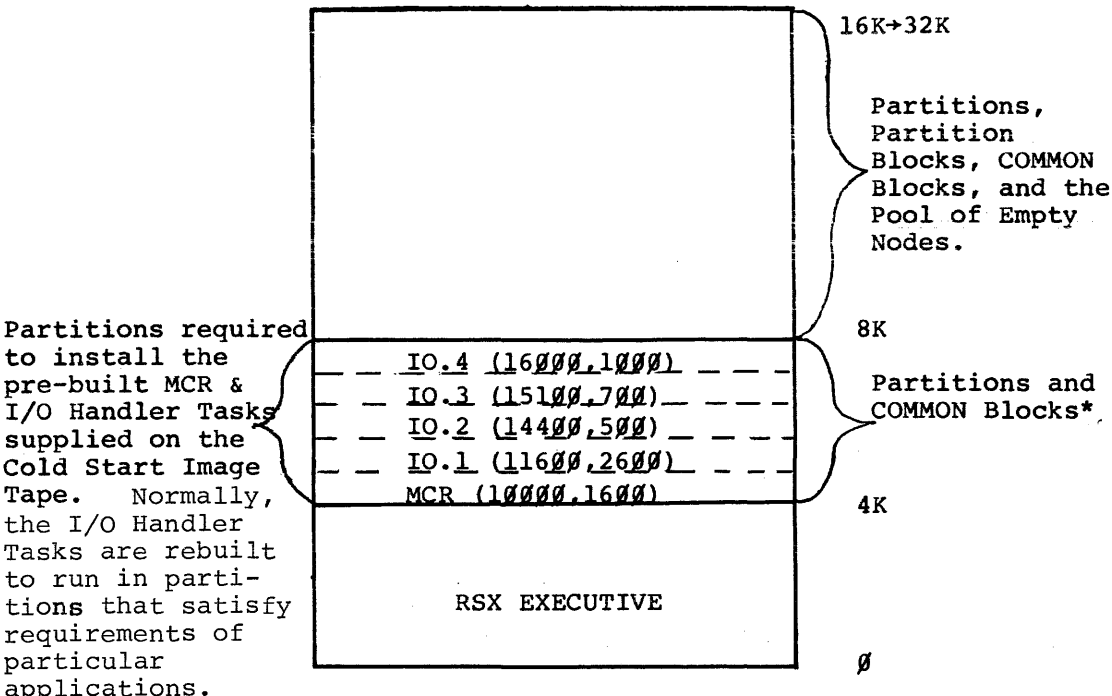


Figure 7.1

* Unspecified core space between Common Blocks and Partitions which exist between the RSX Executive and the 8K memory boundary is not used by the system.

7.4 SYSTEM DEQUES

The RSX System uses linked lists, rather than tables, to maintain system information. These lists are linked together as Double Ended Queues called deques. Each deque consists of a listhead and list elements, or nodes, circularly linked by both forward and backward pointers. The first word of a node or listhead is a forward pointer containing the address of the next node (or the listhead) looking forward. The second word of a node or listhead is a backward pointer containing an address of the previous node (or the listhead) looking backward. The listhead consists of only the two pointers. All nodes in a deque consist of the two pointers followed by eight words of data. Some of the major deques used in the RSX System are the Active Task List (ATL), the System Task List (STL), the Clock Queue, and the Physical Device List (PDVL).

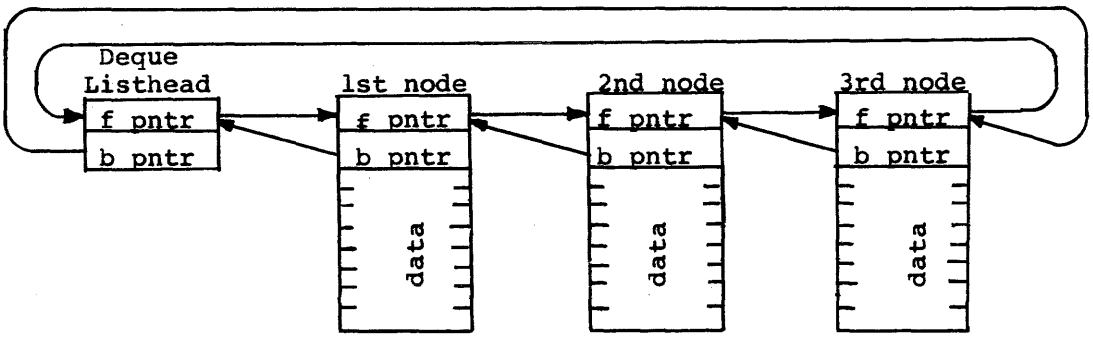


Figure 7.2 A three node deque

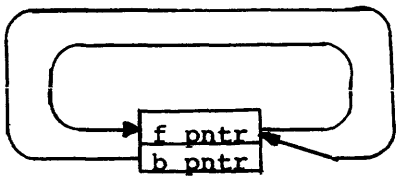


Figure 7.3 An empty deque

7.4.1 POOL

During System Configuration, core which has not been specified by the user for other purposes (viz., Partitions and COMMON Blocks), is divided into ten-word blocks (empty ten-word nodes) and linked together forming a deque called "The Pool of Empty Nodes" or "Pool". When a node is needed to expand a list, it is taken ("Taken" implies changing the node pointers, not moving ten words of data) from the Pool. When a node is no longer needed, it is returned to the Pool.

7.4.2 THE SYSTEM TASK LIST (STL)

The System Task List (STL) is a directory of Tasks in the system. The STL is a deque consisting of one node for each Task currently in the system. An STL node has the following format:

```
Word  0 -- Forward pointer
Word  1 -- Backward pointer
Word  2 -- Task name (first half in .SIXBT)
Word  3 -- Task name (second half in .SIXBT)
Word  4 -- Flags and Default priority
Word  5 -- Partition Block Address
Word  6 -- Disk address of Task image
Word  7 -- Size of resident image
Word 10 -- Disk storage allocated
Word 11 -- Task entry point
```

Word 4, the Flags and Default priority, has the following bit designations:

```
Bit 0 -- set when the Task is active
Bit 1 -- Unused
Bit 2 -- set when the Task is disabled
Bit 3 -- set when the Task is "FIXed in Core"
Bit 4 -- Unused
Bit 5 -- Unused
Bits 6 - 17 -- Task's default priority
```

Word 6, the Disk address, contains the disk unit number in bits 15-17.

Nodes are added to the STL whenever a Task is INSTALLed into the system, and deleted from the STL when a Task is REMOVEd from the system.

7.4.3 THE ACTIVE TASK LIST (ATL)

The Active Task List (ATL) is a priority ordered list of Active Tasks. The ATL is a deque consisting of one node for each Active Task in the system. An ATL node has the following format:

```
Word  0 -- Forward pointer
Word  1 -- Backward pointer
Word  2 -- Task name (first half .SIXBT)
Word  3 -- Task name (second half .SIXBT)
Word  4 -- Task run priority
Word  5 -- Partition block address
Word  6 -- STL node address
Word  7 -- Task status indicator
Word 10 -- Start or resumption address
Word 11 -- Event variable address
```

The ATL is ordered by the priority of the Active Tasks and is used to drive the system. The order in which Tasks are considered is determined by scanning the list, and the action to be taken is determined by examining the Task status word. There are six levels of status, each of which is described below:

Status 1: Task image is on the disk. If its partition is available, flag partition unavailable and proceed to status two; otherwise, service next Task in ATL.

Status 2: Task image is on the disk and the partition is available for its use. Queue disk read request with Event Variable in ATL (Word 11) and proceed to status three.

Status 3: Waiting for an Event Variable. If the Event Variable, whose address is in the ATL, is non-zero, proceed to status four; otherwise service next Task in the ATL.

Status 4: Task is ready to be started or resumed. In order that its environment will be saved if it is interrupted by the Executive, set status five, and start or continue Task execution. (Status four may be set by the WAIT or RESUME Directives.)

Status 5: Task has been interrupted by the Executive (environment saved in partition block). Restore environment and return control to Task.

Status 6: Task has been suspended. (Status six is set only by the SUSPEND Directive.)

7.4.4 THE CLOCK QUEUE

The Clock Queue is a deque consisting of one node for each item to be

done at some time in the future. These items are: scheduling of Tasks (SCHEDULE, RUN, and SYNC Directives), rescheduling of Tasks (Clock interrupt service routine), and setting of Event Variables after elapsed time periods (MARK Directive). The nodes are linked in the order in which they come due, and have the following format:

```
Word  0 -- Forward Pointer
Word  1 -- Backward pointer
Word  2 -- Type indicator (TS,MT)
Word  3 -- Unused
Word  4 -- Run priority (TS) or Event Variable address (MT)
Word  5 -- STL node address (TS) or zero (MT)
Word  6 -- Schedule interval seconds (TS,MT)
Word  7 -- Schedule interval ticks (TS,MT)
Word 10 -- Reschedule interval seconds (TS)
Word 11 -- Reschedule interval ticks (TS)

           TS -- Task Scheduling usage
           MT -- Mark Time usage
```

Word 2, the Type indicator, is set as follows:

```
0 -- Task scheduling with no rescheduling
1 -- Task scheduling with periodic rescheduling
5 -- Mark time request
6 -- Null node (result of cancellation)
```

The schedule interval in all nodes, except the first node, is relative to the previous node. The schedule interval in the first node is relative to "now" and is decremented and examined at each clock tick. Two words are used to record the schedule interval: "schedule ticks" and "schedule seconds". The schedule ticks is only zero when a node is to come due at the same time as the previous node and is never greater than the number of ticks per second. When an interval of more than one second is represented, the schedule seconds indicates the number of additional whole seconds. The "reschedule ticks" and "reschedule seconds" are the schedule interval reset values when periodic Task rescheduling has been requested.

7.4.5 THE PARTITION BLOCKS DESCRIPTION LIST (PBDL)

Partition Blocks serve three functions: (1) They contain partition description information to assure that a Task being installed into the

system has been built for an existent partition; (2) they provide core for an Event Variable and disk GET (DSKGET) control table necessary to load Tasks into partitions; and (3) they provide for saving a Task's environment when it is interrupted by the Executive. The Partition Blocks are generated by the System Configurator and are linked together into a deque called the Partition Blocks Description List with abnormal nodes having the following format:

```
Word  0 -- Forward pointer
Word  1 -- Backward pointer
Word  2 -- Partition name (first half .SIXBT)
Word  3 -- Partition name (second half .SIXBT)
Word  4 -- Partition base (address)
Word  5 -- Partition size
Word  6 -- Flags word (bit Ø indicates partition is occupied)
Word  7 -- Register save routine entry point (operand address for
           wd. 12)
Word 10 -- Interrupt connect location (JMS here upon interrupt)
Word 11 -- DBA instruction
Word 12 -- JMS* .-3 instruction (transfer to save routine)
Word 13 -- AC buffer
Word 14 -- XR buffer
Word 15 -- LR buffer                 (Words 15 thru 21 are used by
Word 16 -- MQ buffer                 the Executive during status two
Word 17 -- SC buffer                 and three to store the disk read
Word 20 -- R1 buffer                 Event Variable and Control
Word 21 -- R2 buffer                 Table)
Word 22 -- R3 buffer                 (R1 thru R6 are pseudo registers
Word 23 -- R4 buffer                 used by re-entrant system
Word 24 -- R5 buffer                 routines)
Word 25 -- R6 buffer
Word 26 -- X10 buffer                (X10 thru X13 are autoincrement
Word 27 -- X11 buffer                registers 10 thru 13)
Word 30 -- X12 buffer
Word 31 -- X13 buffer
Word 32 -- L20 buffer                (CAL return parameters)
Word 33 -- SKP                       SKP is an indicator to the
                                     register save routine to trans-
                                     fer control to the Executive
                                     (NOP indicates transfer to an
                                     interrupt service routine).
```

7.4.6 THE PHYSICAL DEVICE LIST (PDVL)

When a logical I/O unit is assigned to a physical unit, the address of a node describing the device and unit is set in a logical unit table entry corresponding to the Logical Unit Number (LUN). These nodes are constructed by the System Configurator and linked together into a deque

called the Physical Device List. Each PDVL node has the following format:

```
Word  0 -- Forward pointer
Word  1 -- Backward pointer
Word  2 -- Device name (first half .SIXBT)
Word  3 -- Device name (second half/always zero)
Word  4 -- Device Attach flag
Word  5 -- Unit number
Word  6 -- Device request queue (deque listhead) forward pointer
Word  7 -- Device request queue (deque listhead) backward pointer
Word 10 -- Trigger Event Variable address
Word 11 -- Assign inhibit flag
```

7.4.7 THE SYSTEM COMMON BLOCK DEFINITION LIST (SCDL)

The System COMMON Block Definition List (SCDL) is a deque built by the System Configurator consisting of a description of each COMMON Block in the System. An SCDL node has the following format:

```
Word  0 -- Forward pointer
Word  1 -- Backward pointer
Word  2 -- COMMON Block name (first half .SIXBT)
Word  3 -- COMMON Block name (second half .SIXBT)
Word  4 -- Unused
Word  5 -- Base of COMMON Block (address)
Word  6 -- Size of COMMON Block
Word  7 -- Unused
Word 10 -- Unused
Word 11 -- Unused
```

7.5 INPUT/OUTPUT OPERATIONS

Input/Output operations in the RSX System are device independent, with I/O requests being made to Logical Device Units rather than Physical Device Units. Logical Units are equivalenced to Physical Device Units via a Logical Unit Table (LUT). The LUT is a block of contiguous core with a one word entry, or slot, for each LUN. LUN slots are designated sequentially from one and will contain a zero if unassigned (assigned to NONE). The LUT is maintained by the REASSIGN MCR Function.

Physical Device Units are represented by nodes in a deque called the Physical Device List (PDVL). When a LUN is assigned to a Physical

Device Unit, the corresponding LUT slot contains the address of the appropriate Physical Device List node. Corresponding to the LUT is an Attach-Flag-Table (AFT) with a two word entry for each LUT slot. Whenever a LUN is attached to a Task, the Task name is set in the corresponding AFT slot. Whenever a LUN and Device Unit are both attached to a Task, the Device attach flag in the PDVL points to the appropriate AFT slot.

7.5.1 I/O HANDLER TASK INITIALIZATION

All I/O Handlers are RSX Tasks and are called I/O Handler Tasks (IOHT's). They differ from most other Tasks in that they contain an interrupt service routine and that a naming convention exists.*

When a LUN is assigned to a Physical Unit, the appropriate I/O Handler Task is REQUESTed (by the REASSIGN MCR Function). The Handler Task then initializes itself and instructs the system (using the WAITFOR Directive) to suspend its execution until its Trigger Event Variable is set. Handler Task initialization consists of CONNECTing to an interrupt line and setting the address of the Task's Trigger Event Variable in the corresponding Physical Device List node(s).

A Handler Task normally services all Units of a Device.

7.5.2 I/O REQUESTS

I/O requests are made using the QUEUE I/O Directive. This Directive expects to find a PDVL node address in the LUT slot indicated by the LUN, and a Trigger Event Variable address in the PDVL node. If the LUT slot contains a zero, the request is rejected because the LUN has not been assigned to a Physical Unit. If the Trigger Event Variable address is zero, the request is rejected because the Handler Task has not yet been initialized. If the LUT slot and Trigger Event Variable

---

* See section 8.5

address have been set, a request node is formed and inserted into a request queue, and the Handler Task is "triggered" by setting the Trigger Event Variable and declaring a Significant Event. If a Handler Task is triggered while it is servicing a request, the trigger is ignored; however, if a Handler Task is idle, the trigger will bring it back into service.

There are separate I/O request queues for each Physical Device Unit. These queues are deques with their listheads in the PDVL nodes for the physical units. Requests are normally serviced in order of priority by simply picking up the front node from the request queue. I/O requests are processed at API level 7 and are de-queued by priority of the requestor (software priority 1-512) with the highest priority request at the front of the deque. Requests of equal priority are inserted in the order that the requests were made.

If a Physical Device Unit is ATTACHed, only requests from the Task that issued the ATTACH will be serviced, however, I/O requests from all Tasks are queued. When the DETACH request is serviced, pending I/O requests from other Tasks in the queue will then be serviced.

The QUEUE I/O Directive allows an I/O requestor to specify an Event Variable to be set to indicate the status of a request. If a request cannot be queued, the requestor's Event Variable is set to one of the following negative values:

```
-1Ø1 -- Illegal (out of range) LUN
-1Ø2 -- LUN not assigned to a physical unit
-1Ø3 -- Handler not resident and initialized
-777 -- Request node not available (pool empty)
```

If a request is queued, the requestor's Event Variable is zeroed to indicate that the request is pending and the Handler Task will set it non-zero. If a Handler Task cannot successfully complete a request, it will set the requestor's Event Variable to a negative value. (See Appendix D for a complete list of returned Event Variables.

When an I/O request is successfully completed, the requestor's Event Variable is set positive, normally to one (+1).

7.5.3 I/O FUNCTIONS

The following is a description of CAL Parameter Block operands for the QUEUE I/O Directive. The FUNCTION CODE WORD contains the CAL Function Code for the QUEUE I/O Directive (ØØ) in bits 12-17 and the I/O Function code in bits 3-11. An Event Variable address of zero implies "no Event Variable specified". ALLOCATE, DEALLOCATE, GET, and PUT are device dependent functions, and the address of a table of control information is a part of the request, i.e., the Control Table is not queued.

```
ALLOCATE (4 words)
     FUNCTION CODE WORD   (1500)
     EVENT VARIABLE ADDRESS
     LOGICAL UNIT NUMBER
     CONTROL TABLE ADDRESS

DEALLOCATE  (4 words)
     FUNCTION CODE WORD   (1600)
     EVENT VARIABLE ADDRESS
     LOGICAL UNIT NUMBER
     CONTROL TABLE ADDRESS

ATTACH  (3 words)
     FUNCTION CODE WORD   (2400)
     EVENT VARIABLE ADDRESS
     LOGICAL UNIT NUMBER

DETACH  (3 words)
     FUNCTION CODE WORD   (2500)
     EVENT VARIABLE ADDRESS
     LOGICAL UNIT NUMBER

READ  (6 words)
     FUNCTION CODE WORD   (2600)
     EVENT VARIABLE ADDRESS
     LOGICAL UNIT NUMBER
     DATA MODE INDICATOR
     CORE BUFFER ADDRESS
     BUFFER SIZE (max words transferred)

WRITE  (5 words)
     FUNCTION CODE WORD   (2700)
     EVENT VARIABLE ADDRESS
     LOGICAL UNIT NUMBER
     DATA MODE INDICATOR
     CORE BUFFER ADDRESS
```

```
GET (4 words)
     FUNCTION CODE WORD  (3000)
     EVENT VARIABLE ADDRESS
     LOGICAL UNIT NUMBER
     CONTROL TABLE ADDRESS

PUT (4 words)
     FUNCTION CODE WORD  (3100)
     EVENT VARIABLE ADDRESS
     LOGICAL UNIT NUMBER
     CONTROL TABLE ADDRESS

SEEK (6 words)
     FUNCTION CODE WORD  (3200)
     EVENT VARIABLE ADDRESS
     LOGICAL UNIT NUMBER
     FILE NAME (first half)
     FILE NAME (second half)
     FILE NAME EXTENSION

ENTER (6 words)
     FUNCTION CODE WORD  (3300)
     EVENT VARIABLE ADDRESS
     LOGICAL UNIT NUMBER
     FILE NAME (first half)
     FILE NAME (second half)
     FILE NAME EXTENSION

CLOSE (3 words)
     FUNCTION CODE WORD  (3400)
     EVENT VARIABLE ADDRESS
     LOGICAL UNIT NUMBER

DELETE (6 words)
     FUNCTION CODE WORD  (3500)
     EVENT VARIABLE ADDRESS
     LOGICAL UNIT NUMBER
     FILE NAME (first half)
     FILE NAME (second half)
     FILE NAME EXTENSION

HINF (3 words)
     FUNCTION CODE WORD  (3600)
     EVENT VARIABLE ADDRESS
     LOGICAL UNIT NUMBER
```

The following Control Table formats are expected by the Disk Driver:

```
ALLOCATE (3 words)
     REQUIRED STORAGE (IN WORDS)
     DISK UNIT*
     TRACK & HEAD*
```

---

* Set by the Disk I/O Handler, not the requestor.

```
DEALLOCATE (3 words)
     STORAGE ALLOCATED
     DISK UNIT
     TRACK & HEAD

GET (4 words)
     DISK UNIT
     TRACK & HEAD
     CORE ADDRESS
     WORD COUNT

PUT (4 words)
     DISK UNIT
     TRACK & HEAD
     CORE ADDRESS
     WORD COUNT
```

7.5.4 HANDLER TASK EXIT

When there are no Logical Unit Numbers assigned to a physical device, the REASSIGN MCR Function sets the "assign inhibit flag" and clears the Trigger Event Variable address in the Physical Device List node for each unit of the device and inserts an EXIT request in the I/O queue for one of the device's units. The Handler Task services the EXIT request by: 1) DISCONNECTing from an interrupt line, 2) clearing the assign inhibit flag, and 3) EXITing.

7.5.5 DISK STRUCTURE

A disk unit (platter) in the RSX System contains a total of 262,144 decimal words which is divided into 2048 decimal blocks of 128 decimal words each for purposes of storage allocation. The disk is word addressable for data transfer purposes. Recorded on each disk platter is a bit map that indicates which areas of the disk are free (to be ALLOCATEd) and those which are reserved (already ALLOCATEd). These maps are initialized (cleared) by the System Configurator indicating that all blocks* are free. Bit maps consist of 128 decimal words

---

* Excluding one bit map block per platter and save area on platter zero.

(1 disk block) beginning at location $7776ØØ_8$ on each platter. Each word in the bit map represents 16 decimal consecutive blocks. Bits 16 & 17 of the PDP-15 word are unused. A Ø-bit indicates a block is free and a 1-bit indicates it is occupied.

Since there may exist up to 8 disk platters, it is possible for an ALLOCATE request to require 9 disk transfers (approximately 3ØØ milliseconds). To prevent holding off high priority disk requests for this length of time, the Disk handler consists of two Tasks: "DSK", the Disk Driver, and "DSA", a lower priority Task that handles Disk ALLOCATE/DEALLOCATE requests. Whenever "DSK" encounters an ALLOCATE or DEALLOCATE request in its I/O request queue, it simply moves the request node from its own queue to another queue belonging to "DSA". Then it sets "DSA's" trigger event variable and declares a significant event so that "DSA" will run ("DSA" runs at a priority level lower than "DSK").

7.5.6 I/O DATA MODES

The following I/O data modes are supported in the RSX System:

```
IOPS BINARY     (mode Ø)
IMAGE BINARY    (mode 1)
IOPS ASCII      (mode 2)
IMAGE ASCII     (mode 3)
```

The data modes, including line buffer construction, have identical meanings to those used in the ADVANCED Software System (see PDP-15 ADVANCED Software System Monitors Manual section 2.2 and 2.3).

I/O Handler Tasks provided by DEC are listed below along with the data modes each is capable of handling:

```
DT  -- All
DK  -- None*
```

* Data modes do not apply to the GET and Put functions.

```
LP  -- IOPS ASCII & IMAGE ASCII
TT  -- IOPS ASCII & IMAGE ASCII
PP  -- All
PR  -- All
```

7.5.7 INTERRUPT PROCESSING

Interrupt processing under the RSX System consists of hardware interrupts having various levels of priorities. The hardware interrupts normally suspend the execution of other functions in the System including the Executive; however, the Executive delays the servicing of hardware interrupts while it is completing internal operations which cannot be interrupted. These operations are always short in duration and involve the updating of the various lists of system information maintained by the Executive. This delay is never greater than 30 usecs.

Real-Time programs connect themselves to hardware interrupt lines with the use of System Directives; and when hardware interrupts occur on those lines, control is transferred by the hardware (API) directly to the interrupt service routines. Once an interrupt service routine has been entered, it can either save its active registers or use the Executive's Register Save and Restore routines to preserve the contents of the active system registers. The user, however, might or might not decide to save the registers of an interrupted Task depending on timing constraints. Some cases may only require the saving of the Accumulator (AC) which would be done by the interrupt service routine itself. The Executive's Save and Restore routines, however, save and restore several system registers including the Accumulator, Link, MQ, first four Autoincrement, Limit, and Index Registers. The decision whether or not to use the Executive's Save and Restore routines depends on two considerations. First, can the interrupt service routine tolerate delays incurred by using these routines (each operation requires about 7Ø usecs). Second, how many of the system's active

registers are used by the interrupt service routine?

To illustrate the different methods the user can use for saving and restoring system registers, two examples are given. Example one illustrates an interrupt service routine which only requires saving the Accumulator, and example two illustrates the use of the Executive's Save and Restore routines when several registers are required in the interrupt service routine.

Example 1: Assume the interrupt service routine requires only the Accumulator and does not desire to use the Executive's routines because of timing constraints. The interrupt service routine could be structured as follows:

```
SERDEV Ø            /ENTRY POINT TO ROUTINE
       DBA          /ENTER PAGE ADDRESSING MODE
       DAC  SAVEAC  /SAVE ACCUMULATOR
        .     .     /REAL-TIME PROGRAM EXECUTES AT HARDWARE
        .     .     /    PRIORITY LEVEL.
        .     .
       LAC  SAVEAC  /RESTORE ACCUMULATOR
       DBR          /DEBREAK FROM HARDWARE LEVEL
       JMP* SERDEV  /RETURN TO INTERRUPTED TASK
SAVEAC Ø            /TEMPORARY STORAGE FOR ACCUMULATOR
```

Example 2: Assume the interrupt service routine requires several system registers and desires the use of the Executive's Save and Restore routines.

```
INTENTRY SERDEV     /ENTRY POINT TO ROUTINE, INTENTRY IS A
    .               /    SYSTEM MACRO TO CALL THE EXECUTIVE'S
    .               /    SAVE ROUTINE.
    .               /BODY OF INTERRUPT SERVICE ROUTINE
```

```
INTEXIT SERDEV      /INTEXIT IS A SYSTEM MACRO TO CALL THE
                    /    EXECUTIVE'S RESTORE ROUTINE, DEBREAK
                    /    FROM HARDWARE LEVEL, AND RETURN TO
                    /    INTERRUPTED TASK.
```

Interrupt service routines are an integral part of a Task and must be connected to and disconnected from hardware interrupt lines before use. That is, before a Task can process hardware interrupts the Task must first connect itself to a particular API line. Likewise, when a Task no longer requires the use of an interrupt line it should disconnect and release it to the system. Note that even though several Tasks can connect and disconnect themselves to the same API line, only one Task can be connected to it at any given time. The following example illustrates an interrupt service routine that uses the System Directives CONNECT and DISCONNECT to connect and disconnect itself from an interrupt line.

Example 3:

```
          .
          .           /CODE TO INITIALIZE A TASK
          .
          .           /CONNECT INTERRUPT SERVICE ROUTINE,
CONNECT SERDEV,26,EV/   SERDEV, TO INTERRUPT LINE 26.
          .           /THE VALUE OF EV, THE EVENT VARIABLE,
          .           /    SHOULD BE TESTED TO INSURE THE
          .           /    CONNECTION WAS MADE.
          .
INTENTRY SERDEV       /ENTRY POINT TO INTERRUPT ROUTINE, SAVE
          .           /    ACTIVE REGISTERS.
          .           /BODY OF INTERRUPT SERVICE ROUTINE
          .
          .
INTEXIT SERDEV        /RESTORE SAVED REGISTERS, DEBREAK FROM
          .           /    HARDWARE LEVEL, AND RETURN TO IN-
          .           /    TERRUPTED TASK.
          .
          .           /MAIN PORTION OF TASK WHICH OPERATES AT
          .           /    TASK PRIORITY LEVEL.
          .
DISCONNECT SERDEV,26,EV/DISCONNECT INTERRUPT SERVICE ROUTINE
          .           /    FROM LINE 26.
          .
EXIT                  /END OF TASK
```

CHAPTER EIGHT
TASK CONSTRUCTION

8.1 INTRODUCTION

Task construction in the RSX environment falls into one of four distinct categories: (1) a Task which includes computation and/or requests to I/O Handler Tasks; (2) an MCR Function Task; (3) a Front-End Interrupt Driver Task; and (4) an I/O Handler Task. All Tasks, regardless of priority, must be built with the Task Builder before installing into the RSX System.

When building Tasks, the following conventions must be adhered to for successful operation of the RSX System:

(1) All hardware registers are available to the programmer except the last 4 Autoincrement Registers (14-17) which are used by the system.

(2) A naming convention exists for Tasks in categories 2 and 4 (see sections 8.2 and 8.4, respectively).

(3) Tasks should not EXIT while I/O, Mark Time, or Event Variable settings are still pending since the Task may be overlayed by another Task before the operation has completed.

(4) All Directives (viz., the issuing of the CAL instruction) result in a loss of the original contents of the following registers: AC, XR, LR, MQ, LINK, SC, Autoincrement Registers 1Ø-13, system registers R1-R6, and location 2Ø. Unexpected interrupts which suspend normal Task execution always save and restore active registers before use.

The following sections describe the different Task categories in

greater detail.

8.2 COMPUTATIONAL TASK

The computational Task is the more common type of user written Task since it includes programs written in FORTRAN and assembly language which do not have interrupt routines and I/O drivers internal to the Task. All necessary Input/Output is referenced through LUN slots.

Computational Tasks require no naming conventions except the name must be 1-6 characters in length (Tasks called by FORTRAN programs must be 1-5 characters in length).

8.3 MCR FUNCTION TASK

The Monitor Console Routine (MCR) consists of a resident Task called the Resident MCR Task, and a set of MCR Function Tasks. The Resident MCR Task reads a line of input from LUN 2 and REQUESTs the appropriate MCR Function Task which performs the MCR function.

MCR Functions normally all share the same core partition dedicated to MCR Functions, however, they may be "built" to run in any partition. The name of the Resident MCR is "...MCR" and the name of an MCR Function is three periods followed by the first three characters of the name of the MCR Function (e.g., the TIME MCR Function is named "...TIM").

The Resident MCR Task is REQUESTed either by the Teletype Handler Task in response to a CTRL C (from LUN 2) or by an MCR Function Task.

Two subroutines, with entry points in the System Communications (SCOM) area, are used by both Resident MCR and MCR Function Tasks. One (FAC) is used to Fetch-A-Character from a line of command input, and the other (IFAC) to Initialize the Fetch-A-Character subroutine by

reading a line of command and setting the appropriate pointers. Before reading a line, "MCR>" is output (on LUN-2) to indicate that the MCR is waiting for input.

The name of an MCR Function Task is formed by reading a line of command input (IFAC), fetching the first three characters (FAC), and preceding them with three periods. After forming the MCR Function Task Name, the Resident MCR Task continues to fetch characters until either a SPACE, COMMA, CAR RTN, or ALTMODE is found. This is done so that only as few as the first three characters of an MCR Function need be input. After "flushing thru the first break character", the MCR Function Task is REQUESTed and the Resident MCR Task EXIT's. If more information is contained in the first line of input, it will be read by the Function Task using the FAC subroutine. If additional lines of input are required by the Function Task, they are read using the IFAC and FAC subroutines.

Also included in the SCOM area is the MCR Request Inhibit flag (MCRRI) which is examined and set by both the Teletype Handler Task and MCR Function Tasks. If MCRRI=Ø and a CTRL C is typed in, the Teletype Handler Task will REQUEST ...MCR and set MCRRI=1. If MCRRI≠0 and a CTRL C is typed in, the Teletype Handler Task will set MCRRI=-1. MCRRI is cleared by MCR Function Tasks, normally just before they exit, or at least after they have finished fetching characters from the input line. MCRRI is set negative whenever a CTRL C does not result in REQUESTing ...MCR so that CTRL C may also be used to imply "premature termination" to an MCR Function with lengthy output.

When an MCR Function has been performed, and the first line of command input (the line read by the Resident MCR) has been terminated by a CAR RTN, the Resident MCR Task is REQUESTed by the MCR Function Task. When the first line of command input is terminated by an ALTMODE, the Resident MCR Task is not REQUESTed at the completion of an MCR

Function, and a CTRL C typein is necessary to re-establish MCR dialogue.

The MCR Function Tasks are normal Tasks that adhere to the above conventions related to REQUESTing the Resident MCR Task. A user may build his own MCR Function Tasks and is restricted only in naming it (the name must start with three dots).

The following example illustrates the structure of a typical MCR Function Task (TIME MCR Function). Note that the section between line numbers 73-78 (cross-reference line numbers) shows the standard EXIT procedure from an MCR Task.

```
                        / EDIT #5
                        /
                        / COPYRIGHT 1970, DIGITAL EQUIPMENT CORP., MAYNARD, MASS.
                        /
                        / MCR FUNCTION: TIME          25 MAR 71          R. MCLEAN
                        /
                        / TASK NAME:  "...TIM"   TO TYPE LUN=3  TIME ON THE
                        /  IN RESPONSE TO THE MCR "TIME" REQUEST.
                        /
                        / THE FIRST LINE OF COMMAND INPUT FOR ANY MCR FUNCTION IS READ
                        / BY THE RESIDENT MCR TASK ("...MCR"),  FOR THE "TIME" FUNCTION,
                        / THE SYNTAX OF THE FIRST LINE IS:
                        /
                        /       SYNTAX = "TIM"$<CHARACTER> <CR>/<AM>
                        /           <CHARACTER> = <LETTER>/<DIGIT>
                        /           <CR> = CAR RTN
                        /           <AM> = ALTMODE
                        /           $ "ANY NUMBER OF, INCLUDING ZERO"
                        /
                        / THE RESIDENT MCR READS A LINE, FETCHES THE FIRST THREE CHARACTERS
                        / TO FROM THE MCR FUNCTION TASK NAME ("...TIM"), FLUSHES CHARACTERS
                        / THRU THE FIRST BREAK OR TERMINAL CHAR, REQUESTS "...TIM" AND EXITS.
                        /
                        / LINE TERMINATOR,  NO DATA IS TAKEN FROM THE COMMAND INPUT LINE.
                        / THE TASK "...TIM" FLUSHES ALL CHARACTERS THRU THE END OF THE INPUT LINE,
                        / AND SAVES THE LINE TERMINATOR,  NO DATA IS TAKEN FROM THE INPUT LINE.
                        /
                        / THE TIME IS TYPED OUT  "HH:MM:SS"
                        /
                        / IF THE INPUT LINE IS TERMINATED BY A CAR RTN, THE RESIDENT MCR
                        / IS REQUESTED, AND THE FUNCTION TASK EXITS.
                        /
                        / IF THE INPUT LINE IS TERMINATED BY AN ALTMODE, THE FUNCTION
                        / TASK ("...TIM") EXITS WITHOUT REQUESTING THE RESIDENT MCR,
                        / A ^C TYPEIN IS NECESSARY TO RE-ESTABLISH MCR DIALOGUE.
```

```
                        .TITLE *** MCR FUNCTION 'TIME'
                        /
                 000163 A    SS=163
                 000164 A    MM=164
                 000165 A    HH=165
                 000171 A    MCRRI=171
                 000174 A    FAC=174
                        /
                 705522 A    .INH=705522     /INHIBIT INTERRUPTS
                 705521 A    .ENB=705521     /ENABLE INTERRUPTS
                        /
00000 R 120113 R    TIME    JMS*    (FAC)   /FLUSH INPUT THRU TERMINATOR, AND
00001 R 540114 R            SAD     (015)
00002 R 600006 R            JMP     TIM3
00003 R 540115 R            SAD     (175)
00004 R 600006 R            JMP     TIM3
00005 R 600000 R            JMP     TIME
00006 R 040034 R    TIM3    DAC     TERM
                        /
00007 R 705522 A            .INH            ///FETCH  TIME
00010 R 220116 R            LAC*    (HH)    ///
00011 R 040101 R            DAC     MHH     ///
00012 R 220117 R            LAC*    (MM)    ///
00013 R 040104 R            DAC     MMM     ///
00014 R 705521 A            .ENB            ///
00015 R 220120 R            LAC*    (SS)    ///
00016 R 040107 R            DAC     MSS
                        /
00017 R 200121 R            LAC     (MHH)   /CONVERT TIME TO DECIMAL AND
00020 R 040067 R            DAC     CONX    /STORE IN IMAGE ALPHA BUFFER
00021 R 100044 R            JMS     CON
00022 R 100044 R            JMS     CON
00023 R 100044 R            JMS     CON
                        /
00024 R 000071 R            CAL     TYPCPB  /TYPE TIME
00025 R 000035 R            CAL     WAITLP  /WAIT FOR TTY TO FINISH
                        /
```

```
 73          00026 R 200034 R    EXIT    LAC     TERM    /IF CAR RTN TERMINATED INPUT LINE, REQUEST
 74          00027 R 540114 R            SAD     (015)   /RESIDENT MCR TASK & EXIT,  IF ALTMODE TERMINATED
 75          00030 R 000037 R            CAL     REQMCR  /INPUT LINE, CLEAR *C REQUEST INHIBIT FLAG
 76          00031 R 540115 R            SAD     (175)   /AND EXIT,
 77          00032 R 160122 R            DZM*    (MCRRI)
 78          00033 R 000123 R            CAL     (10)
 79                                  /
 80          00034 R 000000 A    TERM    0
 81                                  /
 82          00035 R 000020 A    WAITLP  20
 83          00036 R 000076 R            TYPEV   /EVENT VARIABLE ADDRESS
 84                                  /
 85          00037 R 000001 A    REQMCR  1               /REQUEST "...MCR" CPB
 86          00040 R 000000 A            0
 87          00041 R 565656 A            .SIXBT  "..."
 88          00042 R 150322 A            .SIXBT  "MCR"
 89          00043 R 000000 A            0
 90                                  / CON -- SUBROUTINE TO CONVERT AN INTEGER TO ITS TWO DIGIT
```



```
 91                                  / DECIMAL EQUIVALENT (IMAGE ALPHA).  'CONX' POINTS TO THE
 92                                  / BINARY WORD.  THE BINARY WORD IS REPLACED BY THE TENS
 93                                  / DIGIT AND THE UNITS DIGIT IS STORED IN THE FOLLOWING WORD.
 94                                  / 'CONX' IS INCREMENTED BY THREE.
 95                                  /
 96          00044 R 000000 A    CON     0
 97          00045 R 220067 R            LAC*    CONX
 98          00046 R 160067 R            DZM*    CONX
 99          00047 R 040070 R    CON1    DAC     CONB
100          00050 R 723766 A            AAC     -12
101          00051 R 741100 A            SPA
102          00052 R 600055 R            JMP     CON2
103          00053 R 460067 R            ISZ*    CONX
104          00054 R 600047 R            JMP     CON1
105          00055 R 220067 R    CON2    LAC*    CONX
106          00056 R 240124 R            XOR     (60)
107          00057 R 060067 R            DAC*    CONX
```

```
108          00060 R 440067 R                ISZ      CONX
109          00061 R 200070 R                LAC      CONB
110          00062 R 240124 R                XOR      (60)
111          00063 R 060067 R                DAC*     CONX
112          00064 R 440067 R                ISZ      CONX
113          00065 R 440067 R                ISZ      CONX
114          00066 R 620044 R                JMP*     CON
115                                  /
116          00067 R 000000 A    CONX        0
117          00070 R 000000 A    CONB        0
118                                  /
119          00071 R 002700 A    TYPCPB      2700     /FUNCTION
120          00072 R 000076 R                TYPEV    /EV ADR
121          00073 R 000003 A                3        /LUN
122          00074 R 000003 A                3        /MODE
123          00075 R 000077 R                MES12    /BUFFFER ADR
124                                  /
125          00076 R 000000 A    TYPEV       0        /EVENT VARIABLE
126                                  /
127          00077 R 006003 A    MES12       006003   /HEADER
128          00100 R 000000 A                000000
129          00101 R 000000 A    MHH         000      /HOURS
130          00102 R 000000 A                000
131          00103 R 000072 A                072      /COLON
132          00104 R 000000 A    MMM         000      /MINUTES
133          00105 R 000000 A                000
134          00106 R 000072 A                072
135          00107 R 000000 A    MSS         000      /SECONDS
136          00110 R 000000 A                000
137          00111 R 000015 A                015      /CR
138          00112 R 000012 A                012      /LF
139                                  /
140                    000000 R              .END     TIME
             00113 R 000174 A *L
             00114 R 000015 A *L
             00115 R 000175 A *L
             00116 R 000165 A *L
             00117 R 000164 A *L
```

```
PAGE    4      TIM,5  SRC      *** MCR FUNCTION 'TIME'

               00120 R 000163 A *L
               00121 R 000101 R *L
               00122 R 000171 A *L
               00123 R 000010 A *L
               00124 R 000060 A *L
                   SIZE=00125      NO ERROR LINES
```

```
PAGE    5     TIM,5  CROSS REFERENCE

CON      00044    66     67     68     96*   114
CONB     00070    99    109    117*
CONX     00067    65     97     98    103    105    107    108    111    112
                 113    116*
CON1     00047    99*   104
CON2     00055   102    105*
EXIT     00026    73*
FAC     000174    42*    47
HH      000165    40*    56
MCRRI   000171    41*    77
MES12    00077   123    127*
MHH      00101    57     64    129*
MM      000164    39*    58
MMM      00104    59    132*
MSS      00107    62    135*
REQMCR   00037    75     85*
SS      000163    38*    61
TERM     00034    53     73     80*
TIME     00000    47*    52    140
TIM3     00006    49     51     53*
TYPCPB   00071    70    119*
TYPEV    00076    83    120    125*
WAITLP   00035    71     82*
.ENB    705521    45*    60
.INH    705522    44*    55
```

8.4 FRONT-END DEVICE DRIVER TASK

The Front-End Interrupt Driver Task is a Task which has both computational and interrupt processing capabilities. Unlike the Computational Task, the Front-End Task has an internal interrupt routine; but it does not require the QUEUE I/O Directive to control it as do I/O Handler Tasks.

An example Front-End Task used to generate straight line vectors on the VP15 storage scope is given at the end of this section (VP.6). This particular Task is a subroutine with four entry points for CONNECTing and DISCONNECTing from the interrupt line, erasing the display, and plotting a straight line vector.

The following paragraphs describe the separate sections of the VP.6 Task.

Line Numbers*	Label	Description
25-31	CINT	Connect display interrupt routine, VPINT, to interrupt line 14. Notice that if the Event Variable (EV) is negative, the Task EXITs since the connection could not be made. If a successful connection is made, the EV is cleared before return-to the caller.
4Ø-47	DINT	Disconnect display interrupt routine, VPINT, from interrupt line 14. The testing of the EV is not required here; hence, the address of EV in the CAL Parameter Block, line number 45, is zero.
51-54	ERASE	Erase the face of the storage scope. This operation (EST) generates an interrupt once the display has been erased and requires waiting till completion. This is done by issuing a WAITFOR EV from routine WDINT

*Line Numbers (decimal) along the left hand column of the VP.6 Task listing.

		(line 175). The interrupt routine, VPINT, clears the display flag when the erase operation has completed, sets the EV, and declares a Significant Event (Request API level 6). This results in a scan of the Active Task list and a return following the WAITFOR (contingent upon priority).
58-171	VECTOR	This is the straight line vector plot routine which calculates the required points to generate the line and displays them one point at a time. Following each point displayed, a WAITFOR is done to wait for the completion of the displayed point (line 143 and 172).
173-176	WFINT	Subroutine to issue a WAITFOR EV Directive until the point or erase operation has completed. It then clears the EV before returning. (If the EV wasn't cleared, the next WAITFOR EV issued would return immediately since the EV is set.)
181-189	VPINT	Display interrupt service routine which sets the EV signifying the operation is complete and declares a Significant Event (Request API level 6). The display flag is cleared and control returned to the interrupted Task.

```
                                / EDIT #6
                                /
                                / ERASE & VECTOR -- FORTRAN CALLABLE SUBROUTINE TO ERASE
                                / SCOPE, OR TO CONSTRUCT A VECTOR FROM P1(IX1,IY1) TO P2
                                /
                                /CALLING SEQUENCES:
                                /       CALL CINT       [CONNECT INTERRUPT]
                                /       CALL DINT       [DISCONNECT INTERRUPT]
                                /       CALL ERASE
                                /       CALL VECTOR (IX1,IY1,IX2,IY2)
                                /
                                /
            700504 A    LXB=700504
            700604 A    LYB=700604
            700724 A    EST=700724
            700521 A    SDDF=700521
            700722 A    CDDF=700722
            700564 A    LXBD=700564
            700664 A    LYBD=700664
                                /
                                        .GLOBL  CINT,DINT,ERASE,VECTOR,,DA
                                /
                                / CINT -- CONNECT INTERRUPT LINE
                                /
00000 R 000000 A    CINT    0
00001 R 000007 R            CAL     IC
00002 R 200227 R            LAC     EV
00003 R 140227 R            DZM     EV
00004 R 740100 A            SMA
00005 R 620000 R            JMP*    CINT
00006 R 000232 R            CAL     (10)
                                /
00007 R 000011 A    IC      11
00010 R 000227 R            EV
00011 R 000014 A            14
00012 R 000202 R            VPINT
                                /
                                / DINT -- DISCONNECT INTERRUPT LINE
                                /
```

```
40          00013 R 000000 A    DINT    0
41          00014 R 000016 R            CAL     ID
42          00015 R 620013 R            JMP*    DINT
43                                  /
44          00016 R 000012 A    ID      12
45          00017 R 000000 A            0
46          00020 R 000014 A            14
47          00021 R 000202 R            VPINT
48                                  /
49                                  / ERASE -- ERASE STORAGE SCOPE
50                                  /
51          00022 R 000000 A    ERASE   0
52          00023 R 700724 A            EST
53          00024 R 100174 R            JMS     WFINT
54          00025 R 620022 R            JMP*    ERASE
55                                  /
```



```
56                                  / VECTOR -- CONSTRUCT LINE
57                                  /
58          00026 R 000000 A    VECTOR  0
59          00027 R 120231 E            JMS*    .DA     /FETCH ARGUMENT ADDRESSES
60          00030 R 600035 R            JMP     .+5
61          00031 R 000000 A    X1      0
62          00032 R 000000 A    Y1      0
63          00033 R 000000 A    X2      0
64          00034 R 000000 A    Y2      0
65                                  /
66          00035 R 220031 R            LAC*    X1      /DETERMINE DELTA-X & X-INCR POLA
67          00036 R 740031 A            TCA
68          00037 R 360033 R            TAD*    X2
69          00040 R 722000 A            PAL
70          00041 R 741100 A            SPA
71          00042 R 740031 A            TCA
72          00043 R 040213 R            DAC     DELX
73          00044 R 730000 A            PLA
74          00045 R 751100 A            SPA!CLA
75          00046 R 777776 A            LAW     -2
```

```
 76          00047 R 740030 A            IAC
 77          00050 R 040215 R            DAC     XINC
 78                                  /
 79          00051 R 220032 R            LAC*    Y1      /DETERMINE DELTA-Y & Y-INCR POLA
 80          00052 R 740031 A            TCA
 81          00053 R 360034 R            TAD*    Y2
 82          00054 R 722000 A            PAL
 83          00055 R 741100 A            SPA
 84          00056 R 740031 A            TCA
 85          00057 R 040214 R            DAC     DELY
 86          00060 R 730000 A            PLA
 87          00061 R 751100 A            SPA!CLA
 88          00062 R 777776 A            LAW     -2
 89          00063 R 740030 A            IAC
 90          00064 R 040216 R            DAC     YINC
 91                                  /
 92          00065 R 200214 R            LAC     DELY    /IS DELTA-X GREATER THAN OR EQUAL
 93          00066 R 740031 A            TCA
 94          00067 R 340213 R            TAD     DELX
 95          00070 R 741100 A            SPA             /YES -- INITIALIZE FOR HORIZ LARGE
 96          00071 R 600113 R            JMP     V2      /NO -- INITIALIZE FOR VERT LARGE
 97                                  /
 98          00072 R 200213 R            LAC     DELX    /NC=DELX
 99          00073 R 040223 R            DAC     NC
100          00074 R 200214 R            LAC     DELY    /NR=DELY
101          00075 R 040226 R            DAC     NR
102          00076 R 220031 R            LAC*    X1      /LCC=X
103          00077 R 040221 R            DAC     LCC
104          00100 R 220032 R            LAC*    Y1      /SCC=Y
105          00101 R 040217 R            DAC     SCC
106          00102 R 200233 R            LAC     (LXBD)  /LCM=LXBD
107          00103 R 040171 R            DAC     LCM
108          00104 R 200234 R            LAC     (LYB)   /SCM=LYB
109          00105 R 040165 R            DAC     SCM
110          00106 R 200215 R            LAC     XINC    /LCI=XINC
```

```
00107 R 040222 R          DAC     LCI
00110 R 200216 R          LAC     YINC      /SCI=YINC
00111 R 040220 R          DAC     SCI
00112 R 600133 R          JMP     V3
                    /
00113 R 200214 R    V2    LAC     DELY      /NC=DELY
00114 R 040223 R          DAC     NC
00115 R 200213 R          LAC     DELX      /NR=DELX
00116 R 040226 R          DAC     NR
00117 R 220032 R          LAC*    Y1        /LCC=Y
00120 R 040221 R          DAC     LCC
00121 R 220031 R          LAC*    X1        /SCC=X
00122 R 040217 R          DAC     SCC
00123 R 200235 R          LAC     (LYBD)    /LCM=LYBD
00124 R 040171 R          DAC     LCM
00125 R 200236 R          LAC     (LXB)     /SCM=LXB
00126 R 040165 R          DAC     SCM
00127 R 200216 R          LAC     YINC      /LCI=YINC
00130 R 040222 R          DAC     LCI
00131 R 200215 R          LAC     XINC      /SCI=XINC
00132 R 040220 R          DAC     SCI
                    /
00133 R 200223 R    V3    LAC     NC        /NT=NC
00134 R 040224 R          DAC     NT
00135 R 744020 A          RCR               /NA=NC/2
00136 R 040225 R          DAC     NA
                    /
00137 R 220031 R          LAC*    X1        /PLOT INITIAL POINT
00140 R 700504 A          LXB
00141 R 220032 R          LAC*    Y1
00142 R 700664 A          LYBD
00143 R 100174 R          JMS     WFINT
                    /
00144 R 200223 R    PL1   LAC     NC        /NC=0 ?
00145 R 741200 A          SNA
00146 R 620026 R          JMP*    VECTOR    /YES -- EXIT
00147 R 723777 A          AAC     -1        /NO -- NC=NC-1
00150 R 040223 R          DAC     NC
```

```
                                        /
00151 R 200225 R                        LAC     NA      /NA=NA+NR
00152 R 340226 R                        TAD     NR
00153 R 040225 R                        DAC     NA
                                        /
00154 R 200224 R                        LAC     NT      /NA>NT
00155 R 740031 A                        TCA
00156 R 340225 R                        TAD     NA
00157 R 741100 A                        SPA
00160 R 600166 R                        JMP     PL2     /NO -- DO LARGE COUNT MOVEMENT
00161 R 040225 R                        DAC     NA      /YES -- NA=NA-NT & COMBINED MOVE
00162 R 200217 R                        LAC     SCC     /SMALL COUNT MOVEMENT
00163 R 340220 R                        TAD     SCI
00164 R 040217 R                        DAC     SCC
00165 R 740040 A        SCM             XX              /(LYB OR LXB)
                                        /
00166 R 200221 R        PL2             LAC     LCC     /LARGE COUNT MOVEMENT
```



```
00167 R 340222 R                        TAD     LCI
00170 R 040221 R                        DAC     LCC
00171 R 740040 A        LCM             XX              /(LXBD OR LYBD)
00172 R 100174 R                        JMS     WFINT
                                        /
00173 R 600144 R                        JMP     PL1     /TO EXIT TEST
                                        /
00174 R 000000 A        WFINT           0
00175 R 000200 R                        CAL     WFCPB
00176 R 140227 R                        DZM     EV
00177 R 620174 R                        JMP*    WFINT
                                        /
00200 R 000020 A        WFCPB           20
00201 R 000227 R                        EV
                                        /
00202 R 000000 A        VPINT           0
00203 R 040230 R                        DAC     ACBUF
00204 R 440227 R                        ISZ     EV
00205 R 200237 R                        LAC     (401000)
```

```
185          00206 R 705504 A              ISA
186          00207 R 700722 A              CDDF
187          00210 R 200230 R              LAC       ACBUF
188          00211 R 703344 A              DBR
189          00212 R 620202 R              JMP*      VPINT
190                                  /
191          00213 R 000000 A        DELX  0                 /DELTA-X
192          00214 R 000000 A        DELY  0                 /DELTA-Y
193          00215 R 000000 A        XINC  0                 /X INCREMENT (+1 OR -1)
194          00216 R 000000 A        YINC  0                 /Y INCREMENT (+1 OR -1)
195          00217 R 000000 A        SCC   0                 /SMALL COUNT COORDINATE
196          00220 R 000000 A        SCI   0                 /SMALL COUNT INCREMENT
197          00221 R 000000 A        LCC   0                 /LARGE COUNT COORDINATE
198          00222 R 000000 A        LCI   0                 /LARGE COUNT INCREMENT
199          00223 R 000000 A        NC    0
200          00224 R 000000 A        NT    0
201          00225 R 000000 A        NA    0
202          00226 R 000000 A        NR    0
203          00227 R 000000 A        EV    0
204          00230 R 000000 A        ACBUF 0
205                                  /
206               000000 A                    .END
             00231 R 000231 E *E
             00232 R 000010 A *L
             00233 R 700564 A *L
             00234 R 700604 A *L
             00235 R 700664 A *L
             00236 R 700504 A *L
             00237 R 401000 A *L
                 SIZE=00240      NO ERROR LINES
```

```
ACBUF   00230  182   187   204*
CDDF    700722   17*  186
CINT    00000    21    25*   30
DELX    00213    72    94    98   118   191*
DELY    00214    85    92   100   116   192*
DINT    00013    21    40*   42
ERASE   00022    21    51*   54
EST     700724   15*   52
EV      00227    27    28    34   175   179   183   203*
IC      00007    26    33*
ID      00016    41    44*
LCC     00221   103   121   165   167   197*
LCI     00222   111   129   166   198*
LCM     00171   107   125   168*
LXB     700504   13*  126   139
LXBD    700564   18*  106
LYB     700604   14*  108
LYBD    700664   19*  124   141
NA      00225   136   150   152   156   159   201*
NC      00223    99   117   133   144   148   199*
NR      00226   101   119   151   202*
NT      00224   134   154   200*
PL1     00144   144*  171
PL2     00166   158   165*
SCC     00217   105   123   160   162   195*
SCI     00220   113   131   161   196*
SCM     00165   109   127   163*
SDDF    700521   16*
VECTOR  00026    21    58*   146
VPINT   00202    36    47   181*  189
V2      00113    96   116*
V3      00133   114   133*
WFCPB   00200   174   178*
WFINT   00174    53   142   169   173*  176
XINC    00215    77   110   130   193*
X1      00031    61*   66   102   122   138
X2      00033    63*   68
YINC    00216    90   112   128   194*
Y1      00032    62*   79   104   120   140
Y2      00034    64*   81
.DA     00231    21    59
```

8.5 I/O HANDLER TASK

An I/O Handler Task is a Task dedicated to the control of an I/O Device Unit. I/O requests to these Tasks are made to Logical Unit Numbers and are queued at the requestor's priority. (See section 7.5 for a complete description of I/O operations in an I/O Handler Task.)

A naming convention exists for I/O Handler Tasks (Task Building Name), requiring the name to be two characters in length followed by four periods, respectively (e.g., LP...., PP...., and PR....).

An example I/O Handler Task used to drive the LP15C Line Printer is given at the end of this section. The following paragraphs describe the separate sections of the Line Printer Handler, LP.5.

Line Number*	Label	Description
78-1Ø4	START	This is the Handler initialization section required by all I/O Handler Tasks. Between lines 78-84, the Physical Device List (PDVL) is scanned for a node for this device. If found (line 85), the device Name (line 1Ø2) was found in the PDVL and a node is returned in the AC. If not found (line 84), the Task EXITs since no node having the name "LP" was found in the PDVL. Once the node address is returned in the AC, the address of the Trigger Event Variable in the node is calculated and saved (line 87). The interrupt line is then CONNECTed (if no connection was made the Task EXITs) and the address of the Trigger Event Variable is placed in the Physical Device node (line 92). Lines 94 to 96 calculate an address to be used by the Index Register later when obtaining arguments from the PDVL. The Handler then clears the controller and waits for the Trigger Event Variable, TG, to be set (WAITFOR TG).

* Line Numbers (decimal) along the left hand column of the LP.5 Task listing.

115-146	PQ	The Trigger Event Variable has been Triggered. (The CAL Service Routine in the Executive Triggers the Event Variable whenever the Handler has an I/O request.) The Trigger is cleared (line 118) to prevent the Handler from being inadvertently called when the WAITFOR TG is again issued. At line 12Ø the request is de-queued (removed from the queue) and if the queue is empty, the Handler issues a WAITFOR TG which will be set at the next I/O request for this device. If a node was de-queued, the Event Variable and CAL Function are removed and tested. If the user's Event Variable address (line 128) is zero, the handler substitutes an internal Event Variable to handle I/O completion indications. The CAL Function is then tested for ATTACH, DETACH, etc.... When de-queuing a request (line 122), if the de-queue was not made (empty queue) return from DQRQ is immediately following the JMS, otherwise the return is JMS+2 (line 125). If the de-queue was made, the AC will contain the address of the de-queued node. If not, the AC contains either zero, if the queue was empty, or non-zero if the device has been ATTACHed. This is useful when device handlers are multi-unit and the REASSIGN MCR Function removes one of its units from the LUT.
15Ø-173	ATTACH	Routines to ATTACH, DETACH, and return Handler Information (HINF).
174-293	PRINT	Routines to prepare information for and handle the hardware of the LP15C. Notice lines 243-246 declare a Significant Event indicating that a line has effectively been printed.
327-338	INT	This is the interrupt service routine which reads the status of the Line Printer (always non-zero) and saves it in the Handler's Event Variable. A Significant Event is then declared and return given to the interrupted program.
355-38Ø	CCPB	CPB's used by the Handler.

```
                        / EDIT #6
                        /
                        / COPYRIGHT 1971, DIGITAL EQUIPMENT CORP., MAYNARD, MASS.
                        /
                        / RSX PRINTER HANDLER TASK        1/APR/71        H. KREJCI
                        /
                        / THIS HANDLER TASK IS TO DRIVE THE LP15C HARDWARE.  IT IS COMPATABLE
                        / WITH NORMAL OUTPUT FROM FORTRAN & MACRO WRITTEN PROGRAMS.  OUTPUT IN
                        / IMAGE MODE AND OUTPUT NOT BEGINNING WITH A '12', '14', '20', OR '21'
                        / CHARACTER IS PRECEDED BY AN UPSPACE (LINEFEED) AND PRINTED DIRECTLY
                        / FROM THE INDICATED CORE.  ASCII OUTPUT BEGINNING WITH ONE OF THE ABOVE
                        / VERTICAL CONTROL CHARACTERS (E.G., OUTPUT VIA FORTRAN OTS) IS MOVED TO
                        / A BUFFER WITHIN THIS HANDLER WHERE THE HEADER AND POSSIBLY THE LEADING
                        / CONTROL CHARACTER (FOR OVERPRINT) IS MODIFIED AND THE LINE (CONSIDERED
                        / TWO LINES BY THE HARDWARE WHICH TERMINATES LINES AT VERTICAL CONTROL
                        / CHARACTERS) IS PRINTED.
                        /
                        / THERE ARE NO IMPOSED PAGE EJECTS AT PAGE BOTTOMS.
                        /
                        / THE FOLLOWING CAL PARAMETER BLOCKS ARE USED TO QUEUE REQUESTS FOR
                        / PRINTER SERVICE:
                        /
                        /       CPB     3600    /HANDLER INFORMATION (HINF)
                        /               EV
                        /               LUN
                        /
                        /       CPB     2400    ATTACH PRINTER
                        /               EVA
                        /               LUN
                        /
                        /       CPB     2700    PRINT LINE
                        /               EVA
                        /               LUN
                        /               MODE
                        /               LINE
                        /
                        /       CPB     2500    DETACH PRINTER
                        /               EVA
                        /               LUN
```

```
                                        /
                                        / THE REQUESTOR'S EVENT VARIABLE IS CLEARED (ZEROED) WHEN THE REQUEST
                                        / IS QUEUED BY THE "QUEUE I/O" DIRECTIVE.  IF THE REQUEST CAN BE
                                        / PREFORMED, THE EVENT VARIABLE IS SET TO ONE (+1) UPON COMPLETION.
                                        / IF THE REQUEST CANNOT BE PERFORMED, THE EVENT VARIABLE IS SET TO ONE
                                        / OF THE FOLLOWING NEGATIVE VALUES:
                                        /
                                        /        -5 -- DATA MODE (HEADER) DISAGREES WITH REQUEST MODE
                                        /        -6 -- ILLEGAL REQUEST FUNCTION
                                        /       -24 -- LUN HAS BEEN REASSIGNED WHILE REQUEST WAS IN QUEUE
                                        /
                          000012 A      X12=12            /AUTO-INCREMENT REG 12
                          000013 A      X13=13            /AUTO-INCREMENT REG 13
                          000101 A      R1=101            /RE-ENTRANT REGISTER ONE
                          000102 A      R2=102            /RE-ENTRANT REGISTER TWO
                          000107 A      NADD=107          /NODE ADDITION ROUTINE ENTRY POINT


8-22
PAGE  2     LP,6   SRC

                          000123 A      SNAM=123          /NAME SCAN ROUTINE ENTRY POINT
                          000240 A      POOL=240          /LISTHEAD FOR POOL OF EMPTY NODES
                          000252 A      PDVL=252          /LISTHEAD FOR PHYSICAL DEVICE LIST
                          000325 A      ALAD=325          /ATTACH LUN & DEVICE ENTRY POINT.
                          000332 A      DLAD=332          /DETACH LUN & DEVICE ENTRY POINT
                          000337 A      DQRQ=337          /DE-QUEUE REQUEST ENTRY POINT
                          000010 A      D,TG=10           /POSITION OF TRIGGER EVENT VARIABLE IN PDVL NODE
                          000034 A      WCA=34            /WORD COUNT ADDRESS (NOT USED BY LP CONTROLLER)
                          000035 A      CAA=35            /CURRENT ADDRESS REGISTER ADDRESS
                          706541 A      LPP1=706541       /PRINT ONE LINE
                          706521 A      LPPM=706521       /PRINT MULTIPLE LINE
                          706552 A      LPRS=706552       /READ LP STATUS
                          706544 A      LPEI=706544       /ENABLE LP INTERRUPTS
                          706561 A      LPDI=706561       /DISABLE LP INTERRUPTS
                          706621 A      LPCD=706621       /CLEAR LP DONE FLAG
                          706641 A      LPCS=706641       /CLEAR LP STATUS AND ERROR FLAGS
                                        /
                00000 R 002002 A      LBF      002002   /INTERNAL LINE BUFFER HEADER
                00001 R 000000 A               000000   /INITIALIZATION CODE IS USED FOR TEXT BUFFER
```

```
                                /
                                / HANDLER INITIALIZATION
                                /
00002 R 200416 R    START   LAC     (PDVL)  /SCAN PHYSICAL DEVICE LIST FOR FOR NODE
00003 R 060417 R            DAC*    (R1)    /FOR THIS DEVICE.
00004 R 200420 R            LAC     (HNAM)
00005 R 060421 R            DAC*    (R2)
00006 R 120422 R            JMS*    (SNAM)  /(R1, R2, R6, XR, & AC ARE ALTERED)
                                            /NODE FOUND?
00007 R 000423 R            CAL     (10)    /NO -- EXIT
00010 R 040355 R            DAC     PDVNA   /SAVE PDVL NODE ADDRESS
00011 R 723010 A            AAC     *D.TG   /AND TRIGGER EVENT VARIABLE ADDRESS
00012 R 040356 R            DAC     PDVTA   /TRIGGER EVENT VARIABLE ADDRESS ADDRESS.
00013 R 000357 R            CAL     CCPB    /CONNECT INTERRUPT LINE
00014 R 200347 R            LAC     EV      /CONNECT OKAY?
00015 R 741100 A            SPA
00016 R 000423 R            CAL     (10)    /NO -- EXIT
00017 R 200424 R            LAC     (TG)    /YES -- SET TRIGGER EVENT VARIABLE ADDRESS
00020 R 060356 R            DAC*    PDVTA   /IN PHYSICAL DEVICE NODE
00021 R 500425 R            AND     (070000)/DETERMINE "XR=ADJ"
00022 R 740031 A            TCA
00023 R 040342 R            DAC     XADJ
                                /
00024 R 706621 A            LPCD            /CLEAR LP CONTROLLER
00025 R 706641 A            LPCS
00026 R 600070 R            JMP     WFTGR   /WAIT FOR TRIGGER
                                /
00027 R 142000 A    HNAM    .SIXBT "LP####" /HANDLER TASK NAME
00030 R 000000 A
                                /
                                            /END OF INITIALIZATION CODE
                                /
00031 R         A           .BLOCK 66*START-.
                                /
                                / ++++++++++++++++ THE ABOVE CODE IS OVER- ++++++++++++++++++
                                / ++++++++++++++++ LAYED BY OTS ASCII LINES ++++++++++++++++
```

```
                                /
                                / WAIT FOR TASK TO BE TRIGGERED (BY 'I/O CAL' CAL SERVICE ROUTINE)
                                / TO SIGNAL THAT A REQUEST HAS BEEN QUEUED.
                                /
00070 R 000373 R    WFTGR   CAL     WFTCPB  /WAIT FOR TRIGGER EVENT VARIABLE TO BE SET
                                /
                                / THE TASK HAS BEEN TRIGGERED -- PICK A REQUEST FROM QUEUE (IF ANY)
                                /
00071 R 140354 R            DZM     TG      /CLEAR TRIGGER
                                /
00072 R 200355 R    PQ      LAC     PDVNA   /DE-QUEUE A REQUEST
00073 R 060417 R            DAC*    (R1)
00074 R 120426 R            JMS*    (DQRQ)  /(R1, R2, R4, R5, R6, XR, & AC ARE ALTERED)
                                            /WAS A REQUEST FOUND?
00075 R 600070 R            JMP     WFTGR   /NO -- WAIT FOR TRIGGER
00076 R 040353 R            DAC     RN      /YES -- SAVE ADDRESS OF REQUEST NODE
00077 R 340342 R            TAD     XADJ    /SETUP XR TO ACCESS NODE
00100 R 721000 A            PAX
00101 R 210006 A            LAC     6,X     /SAVE ADDRESS OF REQUESTOR'S EVENT VARIABLE
00102 R 741200 A            SNA
00103 R 200427 R            LAC     (RE)
00104 R 040352 R            DAC     RE
                                /
00105 R 210005 A            LAC     5,X     /FETCH CAL FUNCTION CODE
00106 R 500430 R            AND     (777)
00107 R 540431 R            SAD     (024)   /ATTACH REQUEST?
00110 R 600123 R            JMP     ATTACH  /YES -- ATTACH TO A TASK
00111 R 540432 R            SAD     (025)   /NO -- DETACH REQUEST?
00112 R 600132 R            JMP     DETACH  /YES -- DETACH FROM TASK
00113 R 540433 R            SAD     (027)   /NO -- PRINT REQUEST?
00114 R 600143 R            JMP     PRINT   /YES -- WRITE RECORD
00115 R 540434 R            SAD     (036)   /NO -- HANDLER INFO REQUEST?
00116 R 600141 R            JMP     HINF    /YES -- RETURN INFO IN EVENT VARIABLE
00117 R 540430 R            SAD     (777)   /NO -- EXIT (DEASSIGNED) REQUEST?
00120 R 600302 R            JMP     DAEX    /YES -- DEATTACH & EXIT
00121 R 777772 A            LAW     -6      /NO -- UNIMPLIMENTED FUNCTION -- SET
00122 R 600315 R            JMP     SEV     /EVENT VARIABLE TO -6
                                /
```

```
148                             / ATTACH TO A TASK
149                             /
150     00123 R 200355 R    ATTACH  LAC     PDVNA   /ATTACH LUN & DEVICE
151     00124 R 060417 R            DAC*    (R1)
152     00125 R 200353 R            LAC     RN
153     00126 R 060421 R            DAC*    (R2)
154     00127 R 120435 R            JMS*    (ALAD)  /(R3, R4, R5, R6, X10, X11, XR, & AC ARE ALTERED)
155                                                 /WAS LUN ATTACHED?
156     00130 R 600315 R            JMP     SEV     /NO -- SET REQUESTOR'S EVENT VARIABLE TO -24
157     00131 R 600314 R            JMP     REQCMP  /YES -- REQUEST COMPLETED
158                             /
159                             / DETACH FROM A TASK
160                             /
161     00132 R 200355 R    DETACH  LAC     PDVNA   /DETACH LUN & DEVICE
162     00133 R 060417 R            DAC*    (R1)
163     00134 R 200353 R            LAC     RN
164     00135 R 060421 R            DAC*    (R2)
```



```
165     00136 R 120436 R            JMS*    (DLAD)  /(R3, R4, R5, R6, X10, X11, XR, & AC ARE ALTERED)
166                                                 /WAS LUN DETACHED?
167     00137 R 600315 R            JMP     SEV     /NO -- SET REQUESTOR'S EVENT VARIABLE TO -24
168     00140 R 600314 R            JMP     REQCMP  /YES -- REQUEST CONPLETED
169                             /
170                             / RETURN HANDLER INFORMATION IN EVENT VARIABLE
171                             /
172     00141 R 200437 R    HINF    LAC     (100011)
173     00142 R 600315 R            JMP     SEV
174                             /
175                             / PRINT LINE
176                             /
177     00143 R 210010 A    PRINT   LAC     10,X    /SET HEADER ADDRESS
178     00144 R 040350 R            DAC     HX1
179     00145 R 723002 A            AAC     +2      /SET TEXT ADDRESS
180     00146 R 040351 R            DAC     HX2
181     00147 R 220350 R            LAC*    HX1     /GET MODE INDICATOR FROM HEADER
182     00150 R 500440 R            AND     (003)
```

```
183     00151 R 550007 A            SAD     7,X     /DOES DATA MODE AGREE WITH REQUEST MODE?
184     00152 R 600155 R            JMP     .+3
185     00153 R 777773 A            LAW     -5      /NO -- SET EVENT VARIABLE TO -5
186     00154 R 600315 R            JMP     SEV
187     00155 R 540440 R            SAD     (3)     /YES -- IMAGE MODE?
188     00156 R 600174 R            JMP     UBM     /YES -- UNBUFFEDED MODE
189     00157 R 220351 R            LAC*    HX2     /NO -- FETCH FIRST CHARACTER OF TEXT
190     00160 R 500441 R            AND     (774000)/AND TEST FOR LINE TERMINATING CONTROL CHARACTER
191     00161 R 540442 R            SAD     (050000)/LINE FEED (12)?
192     00162 R 600206 R            JMP     BFM     /YES -- BUFFERED MODE WITH LF IN AC BITS 0-6
193     00163 R 540443 R            SAD     (060000)/NO -- FORM FEED (14)?
194     00164 R 600206 R            JMP     BFM     /YES -- BUFFERED MODE WITH FF IN AC BITS 0-6
195     00165 R 540444 R            SAD     (104000)/NO -- DOUBLE SPACE (21)?
196     00166 R 600206 R            JMP     BFM     /YES -- BUFFERED MODE WITH DS IN AC BITS 0-6
197     00167 R 540445 R            SAD     (100000)/NO -- OVERPRINT (20)?
198     00170 R 741000 A            SKP             /YES -- BUFFERED MODE WITH CR IN AC BITS 0-6
199     00171 R 600174 R            JMP     UBM     /NO -- UNBUFFERED MODE
200     00172 R 200446 R            LAC     (064000)
201     00173 R 600206 R            JMP     BFM
202                         /
203                         / UNBUFFERED OUTPUT MODE -- UPSPACE FORMS, AND PRINT A SINGLE
204                         / LINE FROM THE REQUESTOR'S BUFFER IN THE MODE INDICATED BY THE HEADER.
205                         /
206     00174 R 200447 R    UBM     LAC     (LFL)   /PRINT LINEFEED LINE
207     00175 R 100246 R            JMS     PRNT
208     00176 R 706541 A            LPP1
209     00177 R 200350 R            LAC     HX1     /PRINT REQUESTED LINE
210     00200 R 100246 R            JMS     PRNT
211     00201 R 706541 A            LPP1
212     00202 R 600314 R            JMP     REQCMP  /REQUEST COMPLETED
213                         /
214     00203 R 002003 A    LFL     002003  /LINEFEED LINE
215     00204 R 000000 A            000000
216     00205 R 000012 A            000012
217                         /
218                         / BUFFERED OUTPUT MODE -- MOVE 5/7 PACKED LINE TO BUFFER IN HANDLER, IF
219                         / CONTROL CHAR IS '20' (OVERPRINT), CHANGE TO '15', AND PRINT THE TWO
```

```
                                / LINES (CONT CHAR & TEXT) IN ASCII MODE.
                                /
00206 R 040344 R    BFM         DAC     CCBF    /SAVE CONTROL CHAR
00207 R 220350 R                LAC*    HX1     /MOVE TEXT TO INTERNAL BUFFER
00210 R 640510 A                LRS     10
00211 R 500450 R                AND     (776)
00212 R 740031 A                TCA
00213 R 040346 R                DAC     CNT
00214 R 723066 A                AAC     +66
00215 R 740100 A                SMA
00216 R 600221 R                JMP     .+3
00217 R 777712 A                LAW     -66
00220 R 040346 R                DAC     CNT
00221 R 200350 R                LAC     HX1
00222 R 740030 A                IAC
00223 R 060451 R                DAC*    (X12)
00224 R 200452 R                LAC     (LBF+1)
00225 R 060453 R                DAC*    (X13)
00226 R 220012 A                LAC*    X12
00227 R 060013 A                DAC*    X13
00230 R 440346 R                ISZ     CNT
00231 R 600226 R                JMP     .-3
                                /
00232 R 200454 R                LAC     (+1)    /SET REQUESTOR'S EVENT VARIABLE TO +1 AND
00233 R 060352 R                DAC*    RE      /DECLARE A SIGNIFICANT EVENT (LINE HAS BEEN
00234 R 200455 R                LAC     (401000)/EFFECTIVELY PRINTED)
00235 R 705504 A                ISA
                                /
00236 R 200002 R                LAC     LBF+2   /CHANGE CONTROL CHAR TO '15' IF '20'
00237 R 500456 R                AND     (003777)
00240 R 240344 R                XOR     CCBF
00241 R 040002 R                DAC     LBF+2
                                /
00242 R 200457 R                LAC     (LBF)   /PRINT TWO LINES
00243 R 100246 R                JMS     PRNT
00244 R 706521 A                LPPM
                                /
00245 R 600320 R                JMP     RNTP    /RETURN REQUEST NODE TO POOL AND PROCESS NEXT REQUEST
```

```
                                        /
                                        / PRNT -- SUBROUTINE TO PRINT A LINE.  THE LINE BUFFER ADDRESS IS
                                        / IN AC, AND THE IOT TO PRINT IS IN THE LOCATION FOLLOWING THE JMS.
                                        /
00246 R 000000 A    PRNT    0
00247 R 140301 R            DZM     PRNTEF  /CLEAR ERROR FLAG
00250 R 723777 A            AAC     -1      /DETERMINE & SAVE CURRENT ADDRESS
00251 R 040345 R            DAC     CABF
00252 R 060460 R    PRNT1   DAC*    (CAA)   /SET CURRENT ADDRESS
00253 R 160461 R            DZM*    (WCA)   /PREVENT WORD COUNT OVERFLOW
00254 R 420246 R            XCT*    PRNT    /EXECUTE PRINT IOT, CLEAR EVENT VARIABLE,
00255 R 140347 R            DZM     EV      /ENABLE LP INTERRUPT, AND WAIT FOR THE EVENT
00256 R 706544 A            LPEI            /VARIABLE TO BE SET NON-ZERO BY THE INTERRUPT
00257 R 000375 R            CAL     WFECPB  /SERVICE ROUTINE.
                                        /
00260 R 200347 R            LAC     EV      /INTERRUPT HAS OCCURRED -- EXAMINE PRINTER STATUS.
00261 R 500462 R            AND     (200000)/ALARM ERROR?
```



```
00262 R 741200 A            SNA
00263 R 600277 R            JMP     PRNTXT  /NO -- EXIT PRNT SUBROUTINE
00264 R 200301 R            LAC     PRNTEF  /NEW ERROR?
00265 R 740200 A            SZA
00266 R 600273 R            JMP     PRNT2   /NO -- DELAY AND RETRY
00267 R 000377 R            CAL     TEMCPB  /YES -- TYPE ERR MESSAGE
00270 R 000375 R            CAL     WFECPB
00271 R 200454 R            LAC     (1)
00272 R 040301 R            DAC     PRNTEF
                                        /
00273 R 000367 R    PRNT2   CAL     MTCPB   /DELAY
00274 R 000375 R            CAL     WFECPB
00275 R 200345 R            LAC     CABF    /RETRY
00276 R 600252 R            JMP     PRNT1
                                        /
00277 R 440246 R    PRNTXT  ISZ     PRNT    /EXIT PRNT SUBROUTINE
00300 R 620246 R            JMP*    PRNT
                                        /
00301 R 000000 A    PRNTEF  0
```

```
294                                     /
295                                     / EXIT REQUEST (FROM TASK "...REA")
296                                     /
297     00302 R 200463 R        DAEX    LAC     (POOL)  /RETURN REQUEST NODE TO POOL
298     00303 R 060417 R                DAC*    (R1)
299     00304 R 200353 R                LAC     RN
300     00305 R 060421 R                DAC*    (R2)
301     00306 R 120464 R                JMS*    (NADD)
302     00307 R 706561 A                LPDI            /DISABLE LP INTERRUPTS
303     00310 R 000363 R                CAL     DCPB    /DISCONNECT INTERRUPT LINE
304     00311 R 440356 R                ISZ     PDVTA   /CLEAR ASSIGN INHIBIT FLAG IN PDVL NODE
305     00312 R 160356 R                DZM*    PDVTA
306     00313 R 000423 R                CAL     (10)    /EXIT
307                                     /
308                                     / REQUEST COMPLETED -- SET REQUESTOR'S EVENT VARIABLE TO +1
309                                     / AND PICK NEXT REQUEST (IF ANY) FROM QUEUE.
310                                     /
311     00314 R 200454 R        REQCMP  LAC     (+1)
312                                     /
313     00315 R 060352 R        SEV     DAC*    RE      /SET REQUESTOR'S EVENT VARIABLE
314     00316 R 200455 R                LAC     (401000)/DECLARE A SIGNIFICANT EVENT
315     00317 R 705504 A                ISA
316                                     /
317     00320 R 200463 R        RNTP    LAC     (POOL)  /RETURN REQUEST NODE TO POOL
318     00321 R 060417 R                DAC*    (R1)
319     00322 R 200353 R                LAC     RN
320     00323 R 060421 R                DAC*    (R2)
321     00324 R 120464 R                JMS*    (NADD)
322                                     /
323     00325 R 600072 R                JMP     PQ      /PICK ANOTHER REQUEST (IF ANY)
324                                     /
325                                     / INTERRUPT SERVICE ROUTINE
326                                     /
327     00326 R 000000 A        INT     0               /INTERRUPT ENTRY POINT
328     00327 R 707762 A                DBA             /ENTER INDEX (PAGE) MODE
329     00330 R 040343 R                DAC     ACBF    /SAVE AC
```

```
330          00331 R 706552 A                LPRS            /READ STATUS AND SET IN EVENT VARIABLE
331          00332 R 040347 R                DAC     EV
332          00333 R 706641 A                LPCS            /CLEAR STATUS, ERR FLAG, & DONE FLAG
333          00334 R 706621 A                LPCD
334          00335 R 200455 R                LAC     (401000)/DECLARE A SIGNIFICANT EVENT
335          00336 R 705504 A                ISA
336          00337 R 200343 R                LAC     ACBF    /RESTORE AC
337          00340 R 703344 A                DBR             /RETURN TO INTERRUPTED PROGRAM
338          00341 R 620326 R                JMP*    INT
339                                  /
340          00342 R 000000 A    XADJ    0       /XR ADJUST CONSTANT TO SUBTRACT PAGE BITS
341          00343 R 000000 A    ACBF    0       /AC BUFFER
342          00344 R 000000 A    CCBF    0       /CONTROL CHAR BUFFER (BITS 0-6)
343          00345 R 000000 A    CABF    0       /INITIAL CURRENT ADDRESS BUFFER
344          00346 R 000000 A    CNT     0       /COUNTER
345          00347 R 000000 A    EV      0       /EVENT VARIABLE
346          00350 R 000000 A    HX1     0       /HEADER ADDRESS
347          00351 R 000000 A    HX2     0       /TEXT ADDRESS
348          00352 R 000000 A    RE      0       /ADDRESS OF REQUESTOR'S EVENT VARIABLE
349          00353 R 000000 A    RN      0       /ADDRESS OF REQUEST NODE PICKED FROM QUEUE
350          00354 R 000000 A    TG      0       /TRIGGER EVENT VARIABLE
351                                  /
352          00355 R 000000 A    PDVNA   0       /PHYSICAL DEVICE NODE ADDRESS
353          00356 R 000000 A    PDVTA   0       /ADDRESS OF ADR OF TRIGGER EV IN PHY DEV NODE
354                                  /
355          00357 R 000011 A    CCPB    11      /CONNECT CPB
356          00360 R 000347 R            EV
357          00361 R 000016 A            16
358          00362 R 000326 R            INT
359                                  /
360          00363 R 000012 A    DCPB    12      /DISCONNECT CPB
361          00364 R 000000 A            0
362          00365 R 000016 A            16
363          00366 R 000326 R            INT
364                                  /
365          00367 R 000013 A    MTCPB   13      /MARK TIME CPB
366          00370 R 000347 R            EV
367          00371 R 000012 A            12
368          00372 R 000001 A            1
```

```
369                                     /
370         00373 R 000020 A    WFTCPB  20          /WAIT FOR TRIGGER CPB
371         00374 R 000354 R            TC
372                                     /
373         00375 R 000020 A    WFECPB  20          /WAIT FOR EVENT VARIABLE CPB
374         00376 R 000347 R            EV
375                                     /
376         00377 R 002700 A    TEMCPB  2700        /TYEP ERR MESSAGE
377         00400 R 000347 R            EV
378         00401 R 000003 A            3
379         00402 R 000002 A            2
380         00403 R 000404 R            ERRMES
381                                     /
382         00404 R 004002 A    ERRMES  004002; 000000; ,ASCII "*** LP NOT READY"<15>
            00405 R 000000 A
            00406 R 251245 A
```



```
            00407 R 220230 A
            00410 R 501011 A
            00411 R 647650 A
            00412 R 202450 A
            00413 R 540610 A
            00414 R 544320 A
            00415 R 000000 A
383                                     /
384                 000002 R            ,END    START
```

```
00416 R 000252 A *L
00417 R 000101 A *L
00420 R 000027 R *L
00421 R 000102 A *L
00422 R 000123 A *L
00423 R 000010 A *L
00424 R 000354 R *L
00425 R 070000 A *L
00426 R 000337 A *L
00427 R 000352 R *L
00430 R 000777 A *L
00431 R 000024 A *L
00432 R 000025 A *L
00433 R 000027 A *L
00434 R 000036 A *L
00435 R 000325 A *L
00436 R 000332 A *L
00437 R 100011 A *L
00440 R 000003 A *L
00441 R 774000 A *L
00442 R 050000 A *L
00443 R 060000 A *L
00444 R 104000 A *L
00445 R 100000 A *L
00446 R 064000 A *L
00447 R 000203 R *L
00450 R 000776 A *L
00451 R 000012 A *L
00452 R 000001 R *L
00453 R 000013 A *L
00454 R 000001 A *L
00455 R 401000 A *L
00456 R 003777 A *L
00457 R 000000 R *L
00460 R 000035 A *L
00461 R 000034 A *L
00462 R 200000 A *L
00463 R 000240 A *L
00464 R 000107 A *L
    SIZE=00465     NO ERROR LINES
```

```
ACBF     00343  329    336    341*
ALAD    000325   59*   154
ATTACH   00123  136    150*
BFM      00206  192    194    196    201    222*
CAA     000035   64*   266
CABF     00345  265    287    343*
CCBF     00344  222    250    342*
CCPB     00357   88    355*
CNT      00346  227    232    240    344*
DAEX     00302  144    297*
DCPB     00363  303    360*
DETACH   00132  138    161*
DLAD    000332   60*   165
DQRQ    000337   61*   122
D,TG    000010   62*    86
ERRMES   00404  380    382*
EV       00347   89    269    273    331    345*   356    366    374    377
HINF     00141  142    172*
HNAM     00027   80    102*
HX1      00350  178    181    209    223    233    346*
HX2      00351  180    189    347*
INT      00326  327*   338    358    363
LBF      00000   73*   236    248    251    253
LFL      00203  206    214*
LPCD    706621   70*    98    333
LPCS    706641   71*    99    332
LPDI    706561   69*   302
LPEI    706544   68*   270
LPPM    706521   66*   255
LPP1    706541   65*   208    211
LPRS    706552   67*   330
MTCPB    00367  285    365*
NADD    000107   55*   301    321
PDVL    000252   58*    78
PDVNA    00355   85    120    150    161    352*
PDVTA    00356   87     93    304    305    353*
POOL    000240   57*   297    317
PQ       00072  120*   323
PRINT    00143  140    177*
```

```
PRNT     00246  207    210    254    262*  268    290    291
PRNTEF   00301  263    277    283    293*
PRNTXT   00277  276    290*
PRNT1    00252  266*   288
PRNT2    00273  279    285*
RE       00352  130    131    244    313    348*
REQCMP   00314  157    168    212    311*
RN       00353  125    152    163    299    319    349*
RNTP     00320  257    317*
R1      000101   53*    79    121    151    162    298    318
R2      000102   54*    81    153    164    300    320
SEV      00315  146    156    167    173    186    313*
SNAM    000123   56*    82
START    00002   78*   106    384
TEMCPB   00377  280    376*
TG       00354   92    118    350*  371
```



```
UBM      00174  188    199    206*
WCA     000034   63*   267
WFECPB   00375  271    281    286    373*
WFTCPB   00373  114    370*
WFTGR    00070  100    114*   124
XADJ     00342   96    126    340*
X12     000012   51*   235    238
X13     000013   52*   237    239
```

8.6 ADDITIONAL INFORMATION

Tasks Written in FORTRAN:

The PAUSE statement results in the Task being SUSPENDed. The RESUME MCR Function is used to continue after a PAUSE.

The STOP statement results in a Task EXIT.

I/O requests to standard I/O Handlers (through LUN's) always wait until the I/O request has completed before continuing.

OTS messages are output on LUN 4.

OTS-2Ø is a FORTRAN READ or WRITE failure.

Tasks Written in MACRO:

The MACRO Assembler pseudo-op .CBD (Common Block Definition) allows the assembly language programmer to declare a COMMON of an indicated name and size, and to specify a word to be set to its base address.

The .CBD pseudo-op takes a COMMON name and its size as arguments, reserves one word of core, and outputs loader codes and parameters to direct the Task Builder to set a vector to the first element of the indicated COMMON in the reserved word. For example, the statement

```
BASE .CBD ABCD 6
```

will provide the base address of COMMON/ABCD/ in the word labeled BASE. (This feature will become available under DOS August 71.)

Normally, 32 LUN's exist; however, this number can be changed by reassembling the system. On a cold start image, LUN 1 is assigned to DSK, LUN's 2, 3, and 4 are assigned to TTØ, and all other LUN's are assigned to NONE.

APPENDIX A

SYNTACTICAL DESCRIPTIONS OF MCR FUNCTIONS

The following is a description of the MCR Functions provided. The syntax is defined in modified Backus Normal Form using the following conventions and definitions:

< >	= Angle brackets delimit metalinguistic variables
" "	= Quote marks delimit a character string
|	= A vertical bar indicates alternation (OR)
	No operator indicates concatenation
( )	= Parens indicate factoring
$	= Indicates any number (including zero) of the following
NUL	= Indicates the empty set
<BC>	= Break character -- blank or comma
<CR>	= Carriage Return
<AM>	= ALTMODE
<NBC>	= Non-break character
<NTC>	= Non-terminal character
<DV>	= Decimal value

A. ENTER TIME

SYNTAX = "ETI" $<NBC><BC><TIME>(<BC><DATE>|NUL) <CR>|<AM>

<TIME> = <HOURS>":"<MINUTES>":"<SECONDS>
<DATE> = <MONTH>"/"<DAY>"/"<YEAR>

B. TIME

SYNTAX = "TIM" $<NTC> <CR>|<AM>

C. DATE

SYNTAX = "DAT" $<NTC> <CR>|<AM>

D. TASK LIST

SYNTAX = "TAS" $<NTC> <CR>|<AM>

E. PARTITIONS

SYNTAX = "PAR" $<NTC> <CR>|<AM>

F. COMMON BLOCKS

SYNTAX = "COM" $<NTC> <CR>|<AM>

G. DEVICES AND ASSIGNMENTS

```
SYNTAX = "DEV" $<NTC> <CR>|<AM>
```

H. INSTALL

```
SYNTAX = "INS" $<NBC><BC><TASK NAME>
         (<BC><DEFAULT PRIORITY>|NUL) <CR>|<AM>

         <DEFAULT PRIORITY> = Decimal value of 1-512
```

I. REMOVE

```
SYNTAX = "REM" $<NTC> <CR>|<AM>
```

J. REQUEST

```
SYNTAX = "REQ" $<NBC><BC><TASK NAME>
         (<BC><RUN PRIORITY>|NUL) <CR>|<AM>

         <RUN PRIORITY> = Decimal value of 1-512
```

K. SCHEDULE

```
SYNTAX = "SCH" $<NBC><BC><TASK NAME><BC><TIME>
         (<BC><RESCHEDULE INTERVAL>|NUL)
         (<BC><RUN PRIORITY>) <CR>|<AM>

         <TIME> = <HOURS>":"<MINUTES>":"<SECONDS>
         <RESCHEDULE INTERVAL> = <DV>("H" "M" "S" "T")
         <RUN PRIORITY> = Decimal value of 1-512
```

L. RUN

```
SYNTAX = "RUN" $<NBC><BC><TASK NAME><BC><SCHEDULE DELTA>
         (<BC><RESCHEDULE INTERVAL>|NULL)
         (<BC><RUN PRIORITY>|NUL) <CR>|<AM>

         <SCHEDULE DELTA> = <DV>("H"|"M"|"S"|"T")
         <RESCHEDULE INTERVAL> = <DV>("H"|"M"|"S"|"T")
         <RUN PRIORITY> = Decimal value of 1-512
```

M. SYNC

```
SYNTAX = "SYN" $<NBC><BC><TASK NAME><BC><SYNC UNIT>
         <SCHEDULE DELTA>(<BC><RESCHEDULE INTERVAL>|NUL)
         (<BC><RUN PRIORITY>|NUL) <CR>|<AM>

         <SYNC UNIT> = "H"|"M"|"S"|"T"
         <SCHEDULE DELTA> = <DV>("H"|"M"|"S"|"T")
         <RESCHEDULE INTERVAL> = <DV>("H"|"M"|"S"|"T")
         <RUN PRIORITY> = Decimal value of 1-512
```

N. CANCEL

SYNTAX = "CAN" $<NTC> <CR>|<AM>

O. RESUME

SYNTAX = "RES" $<NBC><BC><TASK NAME> <CR>|<AM>

P. FIX IN CORE

SYNTAX = "FIX" $<NBC><BC><TASK NAME> <CR>|<AM>

Q. UNFIX

SYNTAX = "UNF" $<NBC><BC><TASK NAME> <CR>|<AM>

R. ENABLE

SYNTAX = "ENA" $<NBC><BC><TASK NAME> <CR>|<AM>

S. DISABLE

SYNTAX = "DIS" $<NBC><BC><TASK NAME> <CR>|<AM>

T. REASSIGN

SYNTAX = "REA" $<NBC><BC><LUN><BC><NEW ASSIGNMENT>
<BC><OLD ASSIGNMENT> <CR>|<AM>

U. SAVE

SYNTAX = "SAV" $<NTC> <CR>|<AM>

V. OPEN REGISTER

SYNTAX = "OPE" $<NBC><BC><NUMBER>(<BC> "D"(<UNIT>|NUL)|NUL)
<CR>|<AM>
<NUMBER> = <SIGN><RADIX><DIGIT>$<DIGIT>
<SIGN> = "+"|"-"|NUL
<RADIX> = "O"|"D"|NUL
<DIGIT> = (Ø|1|2|3|4|5|6|7)|(Ø|1|2|3|4|5|6|7|8|9)
default: positive, octal.
<UNIT> = Disk unit number - default: zero.

APPENDIX B
MACRO EXPANSIONS FOR SYSTEM DIRECTIVES

```
/ EDIT #6
/
/ COPYRIGHT 1971, DIGITAL EQUIPMENT CORP., MAYNARD, MASS.
/
/ RSX-15 MACRO DEFINITIONS      8 APR 71        H. KREJCI
/
/ ABREVIATIONS -- UNLESS OTHERWISE SPECIFIED, ALL PARAMETERS
/                 EXCEPT ADDRESSES ARE GIVEN IN DECIMAL.
/
/       BUFF    CORE BUFFER ADDRESS
/       CL      INTERRUPT CONNECT LOCATION
/       CTB     CONTROL TABLE ADDRESS
/       EV      EVENT VARIABLE ADDRESS
/       FLNAM   FILE NAME (1-6 CHARACTERS)
/       LN      INTERRUPT LINE NUMBER (OCTAL)
/       LUN     LOGICAL UNIT NUMBER
/       MI      MARK TIME INTERVAL (A TICK THRU A DAY)
/       MODE    DATA MODE INDICATOR
/       MU      MARK TIME UNITS
/       RA      RESUMPTION ADDRESS
/       RI      RESCHEDULE INTERVAL (0=1 DAY, WHERE 0
/                   IMPLIES NO RESCHEDULING)
/       RP      RUN PRIORITY (0-512, WHERE 0 IMPLIES
/                   DEFAULT PRIORITY)
/       RU      RESCHEDULE UNITS (H,M,S,T)
/       SD      SCHEDULE DELTA (A TICK THRU A DAY)
/       SH      SCHEDULE HOURS (0-23)
/       SIZE    CORE BUFFER SIZE (OCTAL)
/       SM      SCHEDULE MINUTES (0-59)
/       SS      SCHEDULE SECONDS (0-59)
/       SU      SCHEDULE DELTA UNITS (H,M,S,T)
/       SZ      SYNCHRONIZATION UNIT (H,M,S,T)
/       TASNAM  TASK NAME (1-6 CHARACTERS)
/       EXT     FILE NAME EXTENSION (1-3 CHARACTERS)
/
H=4     /HOURS INDICATOR
M=3     /MINUTES INDICATOR
S=2     /SECONDS INDICATOR
T=1     /TICKS INDICATOR
/
.INH=705522 /INTERRUPT INHIBIT IOT
.ENB=705521 /INTERRUPT ENABLE IOT
/
HH=165  /HRS IN SCOM
MM=164  /MIN IN SCOM
SS=163  /SEC IN SCOM
MO=166  /MON IN SCOM
DA=167  /DAY IN SCOM
YR=170  /YEAR IN SCOM
SAVE=131 /SAVE ENTRY POINT (IN SCOM)
REST=134 /RESTORE ENTRY POINT (IN SCOM)
```

```
/
/ ****  REQUEST TASNAM[,RP[,EV]]
/
        .DEFIN  REQUEST,TN,RP,EV
        CAL     .+2
        JMP     .+6
        01
        EV+0
,,=,;   .SIXBT "TN"
        0; .LOC ..+2
        .DEC
        RP+0
        .ENDM
/
/ ****  SCHEDULE  TASNAM,SH,SM,SS[,RI,RU[,RP[,EV]]]
/
        .DEFIN  SCHEDULE,TN,SH,SM,SS,RI,RU,RP,EV
        CAL     .+2
        JMP     .+13
        02
        EV+0
,,=,;   .SIXBT "TN"
        0; .LOC ..+2
        .DEC
        SH; SM; SS
        RI+0
        RU+0
        RP+0
        .ENDM
/
/ ****  RUN     TASNAM,SD,SU[,RI,RU[,RP[,EV]]]
/
        .DEFIN  RUN,TN,SD,SU,RI,RU,RP,EV
        CAL     .+2
        JMP     .+12
        03
        EV+0
,,=,;   .SIXBT "TN"
        0; .LOC ..+2
        .DEC
        SD; SU
        RI+0
        RU+0
        RP+0
        .ENDM
/
/ ****  SYNC    TASNAM,SZ,SD,SU[,RI,RU[,RP[,EV]]]
/
        .DEFIN  SYNC,TN,SZ,SD,SU,RI,RU,RP,EV
        CAL     .+2
        JMP     .+13
        14
        EV+0
,,=,;   .SIXBT "TN"
        0; .LOC ..+2
        .DEC
        SZ; SD; SU
```

```
        RI+0
        RU+0
        RP+0
        .ENDM
/
/ ****  CANCEL  TASNAM[,EV]
/
        .DEFIN  CANCEL,TN,EV
        CAL     .+2
        JMP     .+5
        04
        EV+0
.,=,;   .SIXBT "TN"
        0; .LOC .,+2
        .ENDM
/
/ ****  SUSPEND
/
        .DEFIN  SUSPEND
        CAL     (6)
        .ENDM
/
/ ****  RESUME  TASNAM[,RAL,EV]]
/
        .DEFIN  RESUME,TN,RA,EV
        CAL     .+2
        JMP     .+6
        07
        EV+0
.,=,;   .SIXBT "TN"
        0; .LOC .,+2
        RA+0
        .ENDM
/
/ ****  MARK    MI,MU,EV
/
        .DEFIN  MARK,MI,MU,EV
        CAL     .+2
        JMP     .+5
        13
        EV
        .DEC; MI; MU
        .ENDM
/ ****  WAITFOR EV
/
        .DEFIN  WAITFOR,EV
        CAL     .+2
        JMP     .+3
        20
        EV
        .ENDM
/
/ ****  WAIT
/
        .DEFIN  WAIT
        CAL     (5)
        .ENDM
```

```
/
/ ****  EXIT
/
        .DEFIN  EXIT
        CAL     (10)
        .ENDM
/
/ ****  CONNECT LN,CL[,EV]
/
        .DEFIN  CONNECT,LN,CL,EV
        CAL     .+2
        JMP     .+5
        11
        EV+0
        LN
        CL
        .ENDM
/
/ **** DISCONNECT  LN,CL[,EV]
/
        .DEFIN  DISCONNECT,LN,CL,EV
        CAL     .+2
        JMP     .+5
        12
        EV+0
        LN
        CL
        .ENDM
/
/ ****  READ    LUN,MODE,BUFF,SIZE[,EV]
/
        .DEFIN  READ,LUN,MODE,BUFF,SIZE,EV
        CAL     .+2
        JMP     .+7
        2600
        EV+0
        .DEC; LUN; .OCT
        MODE
        BUFF
        SIZE
        .ENDM
/ ****  WRITE   LUN,MODE,BUFF[,EV]
/
        .DEFIN  WRITE,LUN,MODE,BUFF,EV
        CAL     .+2
        JMP     .+6
        2700
        EV+0
        .DEC; LUN; .OCT
        MODE
        BUFF
        .ENDM
/
/ ****  DSKAL   CTB[,EV]
/
        .DEFIN  DSKAL,CTB,EV
        CAL     .+2
```

```
        JMP     .+5
        1500
        EV+0
        1
        CTB
        .ENDM
/
/ ****  DSKDAL  CTB[,EV]
/
        .DEFIN  DSKDAL,CTB,EV
        CAL     .+2
        JMP     .+5
        1600
        EV+0
        1
        CTB
        .ENDM
/
/ ****  DSKPUT  CTB[,EV]
/
        .DEFIN  DSKPUT,CTB,EV
        CAL     .+2
        JMP     .+5
        3100
        EV+0
        1
        CTB
        .ENDM
/
/ ****  DSKGET  CTB[,EV]
/
        .DEFIN  DSKGET,CTB,EV
        CAL     .+2
        JMP     .+5
        3000
        EV+0
        1
        CTB
        .ENDM
/
/ ****  ATTACH  LUN[,EV]
/
        .DEFIN  ATTACH,LUN,EV
        CAL     .+2
        JMP     .+4
        2400
        EV+0
        .DEC; LUN; .OCT
        .ENDM
/
/ ****  DETACH  LUN[,EV]
/
        .DEFIN  DETACH,LUN,EV
        CAL     .+2
        JMP     .+4
        2500
        EV+0
```

```
        ,DEC; LUN; ,OCT
        ,ENDM
/
/ ****  SEEK    LUN,FLNAM,EXT[,EV]
/
        ,DEFIN  SEEK,LUN,FLNAM,EXT,EV
        CAL     .+2
        JMP     .+7
        3200
        EV+0
        ,DEC; LUN; ,OCT
.,=,;   ,SIXBT "FLNAM"
        0; ,LOC .,+2
        ,SIXBT "EXT"
        ,ENDM
/
/ ****  ENTER   LUN,FLNAM,EXT[,EV]
/
        ,DEFIN  ENTER,LUN,FLNAM,EXT,EV
        CAL     .+2
        JMP     .+7
        3300
        EV+0
        ,DEC; LUN; ,OCT
.,=,;   ,SIXBT "FLNAM"
        0; ,LOC .,+2
        ,SIXBT "EXT"
        ,ENDM
/
/ ****  DELETE  LUN,FLNAM,EXT[,EV]
/
        ,DEFIN  DELETE,LUN,FLNAM,EXT,EV
        CAL     .+2
        JMP     .+7
        3500
        EV+0
        ,DEC; LUN; ,OCT
.,=,;   ,SIXBT "FLNAM"
        0; ,LOC .,+2
        ,SIXBT "EXT"
        ,ENDM
/
/ ****  CLOSE   LUN[,EV]
/
        ,DEFIN  CLOSE,LUN,EV
        CAL     .+2
        JMP     .+4
        3400
        EV+0
        ,DEC; LUN; ,OCT
        ,ENDM
/
/ ****  HINF    LUN,EV
/
        ,DEFIN  HINF,LUN,EV
        CAL     .+2
        JMP     .+4
```

```
        3600
        EV+0
        .DEC; LUN; .OCT
        .ENDM
/
/ ****  DISABLE TASNAM[,EV]
/
        .DEFIN  DISABLE,TN,EV
        CAL     .+2
        JMP     .+5
        21
        EV+0
.,*,;   .SIXBT "TN"
        0; .LOC .,+2
        .ENDM
/
/ ****  ENABLE  TASNAM[,EV]
/
        .DEFIN  ENABLE,TN,EV
        CAL     .+2
        JMP     .+5
        22
        EV+0
.,*,;   .SIXBT "TN"
        0; .LOC .,+2
        .ENDM
/
/ ****  FIX     TASKNAM[,EV]
/
        .DEFIN  FIX,TN,EV
        CAL     .+2
        JMP     .+5
        15
        EV+0
.,*,;   .SIXBT "TN"
        0; .LOC .,+2
        .ENDM
/
/ ****  UNFIX   TASNAM[,EV]
/
        .DEFIN  UNFIX,TN,EV
        CAL     .+2
        JMP     .+5
        16
        EV+0
.,*,;   .SIXBT "TN"
        0; .LOC .,+2
        .ENDM
/
/ ****  DECLARE
/
        .DEFIN  DECLARE
        LAC     (401000)
        ISA
        .ENDM
```

```
/
/ ****  TIME    HRS,MIN,SEC
/
        .DEFIN  TIME,HRS,MIN,SEC
        .INH
        LAC*    (HH)
        DAC     HRS
        LAC*    (MM)
        DAC     MIN
        LAC*    (SS)
        .ENB
        DAC     SEC
        .ENDM
/
/ ****  DATE    HRS,MIN,SEC,MON,DAY,YEAR
/
        .DEFIN  DATE,HRS,MIN,SEC,MON,DAY,YEAR
        .INH
        LAC*    (HH)
        DAC     HRS
        LAC*    (MM)
        DAC     MIN
        LAC*    (SS)
        DAC     SEC
        LAC*    (MO)
        DAC     MON
        LAC*    (DA)
        DAC     DAY
        LAC*    (YR)
        .ENB
        DAC     YEAR
        .ENDM
/
/ ****  INTENTRY  CL
/
        .DEFIN  INTENTRY,CL
CL      Ø
        DBA
        JMS* (SAVE)
        .REPT 2Ø
        Ø
        NOP
        .ENDM
/
/ ****  INTEXIT CL
/
        .DEFIN  INTEXIT,CL
        LAC     (CL)
        JMP*    (REST)
        .ENDM
/
        .END
```

APPENDIX C

CAL PARAMETER BLOCKS FOR SYSTEM DIRECTIVES

A. QUEUE I/O DIRECTIVE

```
Word  0 -- CAL Function Code (00) in bits 12-17 and
           I/O Function Code in bits 3-11
Word  1 -- Event Variable address
Word  2 -- Logical Unit Number (LUN)
Word  3 -- Unique to I/O Function
Word  4 -- Unique to I/O Function
Word  5 -- Unique to I/O Function
```

B. REQUEST DIRECTIVE

```
Word  0 -- CAL Function Code (01)
Word  1 -- Event Variable address
Word  2 -- Task name (first half)
Word  3 -- Task name (second half)
Word  4 -- Run priority (0-512)
```

C. SCHEDULE DIRECTIVE

```
Word  0 -- CAL Function Code (02)
Word  1 -- Event Variable address
Word  2 -- Task name (first half)
Word  3 -- Task name (second half)
Word  4 -- Schedule hour (0-23)
Word  5 -- Schedule minute (0-59)
Word  6 -- Schedule second (0-59)
Word  7 -- Reschedule interval (0-one day)
Word 10 -- Interval units (1-tks, 2-secs, 3-mins, 4-hrs)
Word 11 -- Run priority (0-512)
```

D. RUN DIRECTIVE

```
Word  0 -- CAL Function Code (03)
Word  1 -- Event Variable address
Word  2 -- Task name (first half)
Word  3 -- Task name (second half)
Word  4 -- Schedule delta (0-one day)
Word  5 -- Delta units (1-tks, 2-secs, 3-mins, 4-hrs)
Word  6 -- Reschedule interval (0-one day)
Word  7 -- Interval units (1-tks, 2-secs, 3-mins, 4-hrs)
Word 10 -- Run priority (0-512)
```

E. SYNC DIRECTIVE

```
Word  0 -- CAL Function Code (14)
Word  1 -- Event Variable address
Word  2 -- Task name (first half)
Word  3 -- Task name (second half)
Word  4 -- Sync units (1-tks, 2-secs, 3-mins, 4-hrs)
Word  5 -- Schedule delta from synchronization (0-one day)
```

Word 6 -- Delta units (1-tks, 2-secs, 3-mins, 4-hrs)
Word 7 -- Reschedule interval (0-one day)
Word 10 -- Interval units (1-tks, 2-secs, 3-mins, 4-hrs)
Word 11 -- Run priority (0-512)

F. CANCEL DIRECTIVE

Word 0 -- CAL Function Code (04)
Word 1 -- Event Variable address
Word 2 -- Task name (first half)
Word 3 -- Task name (second half)

G. SUSPEND DIRECTIVE

Word 0 -- CAL Function Code (06)

H. RESUME DIRECTIVE

Word 0 -- CAL Function Code (07)
Word 1 -- Event Variable address
Word 2 -- Task name (first half)
Word 3 -- Task name (second half)
Word 4 -- Resumption address

I. WAIT DIRECTIVE

Word 0 -- CAL Function Code (05)

J. MARK DIRECTIVE

Word 0 -- CAL Function Code (13)
Word 1 -- Event Variable address
Word 2 -- Delta time (0-one day)
Word 3 -- Delta units (1-tks, 2-secs, 3-mins, 4-hrs)

K. WAITFOR DIRECTIVE

Word 0 -- CAL Function Code (20)
Word 1 -- Event Variable address

L. EXIT DIRECTIVE

Word 0 -- CAL Function Code (10)

M. CONNECT DIRECTIVE

Word 0 -- CAL Function Code (11)
Word 1 -- Event Variable address
Word 2 -- Interrupt line number
Word 3 -- Interrupt transfer address

N. DISCONNECT DIRECTIVE

```
Word  0 -- CAL Function Code (12)
Word  1 -- Event Variable address
Word  2 -- Interrupt line number
Word  3 -- Current interrupt transfer address
```

O. FIX DIRECTIVE

```
Word  0 -- CAL Function Code (15)
Word  1 -- Event Variable address
Word  2 -- Task name (first half)
Word  3 -- Task name (second half)
```

P. UNFIX DIRECTIVE

```
Word  0 -- CAL Function Code (16)
Word  1 -- Event Variable address
Word  2 -- Task name (first half)
Word  3 -- Task name (second half)
```

Q. DISABLE DIRECTIVE

```
Word  0 -- CAL Function Code (21)
Word  1 -- Event Variable address
Word  2 -- Task name (first half)
Word  3 -- Task name (second half)
```

R. ENABLE DIRECTIVE

```
Word  0 -- CAL Function Code (22)
Word  1 -- Event Variable address
Word  2 -- Task name (first half)
Word  3 -- Task name (second half)
```

APPENDIX D

SUMMARY OF RETURNED EVENT VARIABLES

EVENT VARIABLE CONVENTIONS

The following conventions apply to Event Variables in Tasks by the System.

1. Positive values signal successful completion
2. Zero indicates a request is still pending
3. Negative values indicate rejection or unsuccessful completion.

-5	Illegal header word read from device (data mode incorrect or data validity bits improperly set) (DVH)
-6	Unimplemented or illegal Function (DVH)
-7	Illegal data mode (DVH)
-1Ø	File still open (DVH)
-11	File not open (DVH)
-12	DECtape error (DVH)
-13	File not found (DVH)
-14	Directory full (DVH)
-15	Medium full (DVH)
-16	Output word-pair-count or input-buffer-size error (DVH)
-23	Input word-pair-count error (DVH)
-24	LUN has been REASSIGNed while an ATTACH or DETACH request was in an I/O request queue (DVH)
-1Ø1	Out of range Logical Unit Number (IO.)
-1Ø2	Un-assigned Logical Unit Number (IO.)
-1Ø3	Non-resident Device Handler (IO.)
-1Ø4	Control Table argument error (DVH)
-2Ø1	Task not in system (RQ., SC., RN., SY., DA., EA., FX., UF., CN.)
-2Ø2	Task is active (RQ., FX.) or not active (RS.)
-2Ø3	CAL not Task issued (SC., RN., SY., MT.)
-2Ø4	Task is DISABLED (RQ., SC., RN., SY., FX.)
-2Ø5	Task not suspended (RS.)
-2Ø7	Task already FIXed (FX.) or not FIXed (UF.)
-21Ø	Partition occupied (FX.)
-3Ø1	Line number rejected (CI., DI.)
-3Ø2	Line is CONNECTed (CI.) or not CONNECTed (DI.)
-777	Pool is empty

DVH -- Device Handler
IO. -- 'QUEUE I/O' Directive
RQ. -- 'REQUEST' Directive
SC. -- 'SCHEDULE' Directive
RN. -- 'RUN' Directive
SY. -- 'SYNC' Directive
CN. -- 'CANCEL' Directive
RS. -- 'RESUME' Directive
CI. -- 'CONNECT' Directive
DI. -- 'DISCONNECT' Directive

FX. -- 'FIX IN CORE' Directive
UF. -- 'UNFIX' Directive
DA. -- 'DISABLE' Directive
EA. -- 'ENABLE' Directive
MT. -- 'MARK' Directive

APPENDIX E

REGISTERS SAVED DURING "SAVE" AND "RESTORE" OPERATIONS

```
Word  0 -- AC buffer (accumulator)
Word  1 -- XR buffer (index register)
Word  2 -- LR buffer (limit register)
Word  3 -- MQ buffer (multiplier-quotient register)
Word  4 -- SC buffer (step counter)
Word  5 -- R1 buffer (absolute location 101)
Word  6 -- R2 buffer (absolute location 102)
Word  7 -- R3 buffer (absolute location 103)
Word 10 -- R4 buffer (absolute location 104)
Word 11 -- R5 buffer (absolute location 105)
Word 12 -- R6 buffer (absolute location 106)
Word 13 -- X10 buffer (autoincrement register 10)
Word 14 -- X11 buffer (autoincrement register 11)
Word 15 -- X12 buffer (autoincrement register 12)
Word 16 -- X13 buffer (autoincrement register 13)
Word 17 -- L20 buffer (location 20 - CAL return)
```

CONVERSION TABLES

SCALES OF NOTATION

2^x IN DECIMAL

x	2^x	x	2^x	x	2^x
0.001	1.00069 33874 62581	0.01	1.00695 55500 56719	0.1	1.07177 34625 36293
0.002	1.00138 72557 11335	0.02	1.01395 94797 90029	0.2	1.14869 83549 97035
0.003	1.00208 16050 79633	0.03	1.02101 21257 07193	0.3	1.23114 44133 44916
0.004	1.00277 64359 01078	0.04	1.02811 38266 56067	0.4	1.31950 79107 72894
0.005	1.00347 17485 09503	0.05	1.03526 49238 41377	0.5	1.41421 35623 73095
0.006	1.00416 75432 38973	0.06	1.04246 57608 41121	0.6	1.51571 65665 10398
0.007	1.00486 38204 23785	0.07	1.04971 66836 23067	0.7	1.62450 47927 12471
0.008	1.00556 05803 98468	0.08	1.05701 80405 61380	0.8	1.74110 11265 92248
0.009	1.00625 78234 97782	0.09	1.06437 01824 53360	0.9	1.86606 59830 73615

$10^{\pm n}$ IN OCTAL

10^n	n	10^{-n}	10^n	n	10^{-n}
1	0	1.000 000 000 000 000 000 00	112 402 762 000	10	0.000 000 000 006 676 337 66
12	1	0.063 146 314 631 463 146 31	1 351 035 564 000	11	0.000 000 000 000 537 657 77
144	2	0.005 075 341 217 270 243 66	16 432 451 210 000	12	0.000 000 000 000 043 136 32
1 750	3	0.000 406 111 564 570 651 77	221 411 634 520 000	13	0.000 000 000 000 003 411 35
23 420	4	0.000 032 155 613 530 704 15	2 657 142 036 440 000	14	0.000 000 000 000 000 264 11
303 240	5	0.000 002 476 132 610 706 64	34 327 724 461 500 000	15	0.000 000 000 000 000 022 01
3 641 100	6	0.000 000 206 157 364 055 37	434 157 115 760 200 000	16	0.000 000 000 000 000 001 63
46 113 200	7	0.000 000 015 327 745 152 75	5 432 127 413 542 400 000	17	0.000 000 000 000 000 000 14
575 360 400	8	0.000 000 001 257 143 561 06	67 405 553 164 731 000 000	18	0.000 000 000 000 000 000 01
7 346 545 000	9	0.000 000 000 104 560 276 41			

$n \log_{10} 2$, $n \log_2 10$ IN DECIMAL

n	$n \log_{10} 2$	$n \log_2 10$	n	$n \log_{10} 2$	$n \log_2 10$
1	0.30102 99957	3.32192 80949	6	1.80617 99740	19.93156 85693
2	0.60205 99913	6.64385 61898	7	2.10720 99696	23.25349 66642
3	0.90308 99870	9.96578 42847	8	2.40823 99653	26.57542 47591
4	1.20411 99827	13.28771 23795	9	2.70926 99610	29.89735 28540
5	1.50514 99783	16.60964 04744	10	3.01029 99566	33.21928 09489

ADDITION AND MULTIPLICATION TABLES

Addition	Multiplication

Binary Scale

Addition	Multiplication
$0 + 0 = 0$	$0 \times 0 = 0$
$0 + 1 = 1 + 0 = 1$	$0 \times 1 = 1 \times 0 = 0$
$1 + 1 = 10$	$1 \times 1 = 1$

Octal Scale

+	01	02	03	04	05	06	07
1	02	03	04	05	06	07	10
2	03	04	05	06	07	10	11
3	04	05	06	07	10	11	12
4	05	06	07	10	11	12	13
5	06	07	10	11	12	13	14
6	07	10	11	12	13	14	15
7	10	11	12	13	14	15	16

×	02	03	04	05	06	07
2	04	06	10	12	14	16
3	06	11	14	17	22	25
4	10	14	20	24	30	34
5	12	17	24	31	36	43
6	14	22	30	36	44	52
7	16	25	34	43	52	61

MATHEMATICAL CONSTANTS IN OCTAL SCALE

π = 3.11037 552421$_8$	e = 2.55760 521305$_8$	γ = 0.44742 147707$_8$
π^{-1} = 0.24276 301556$_8$	e^{-1} = 0.27426 530661$_8$	$\ln \gamma = -$ 0.43127 233602$_8$
$\sqrt{\pi}$ = 1.61337 611067$_8$	$\sqrt{e}$ = 1.51411 230704$_8$	$\log_2 \gamma = -$ 0.62573 030645$_8$
$\ln \pi$ = 1.11206 404435$_8$	$\log_{10} e$ = 0.33626 754251$_8$	$\sqrt{2}$ = 1.32404 746320$_8$
$\log_2 \pi$ = 1.51544 163223$_8$	$\log_2 e$ = 1.34252 166245$_8$	ln 2 = 0.54271 027760$_8$
$\sqrt{10}$ = 3.12305 407267$_8$	$\log_2 10$ = 3.24464 741136$_8$	ln 10 = 2.23273 067355$_8$

POWERS OF TWO

2^n	n	2^{-n}
1	0	1.0
2	1	0.5
4	2	0.25
8	3	0.125
16	4	0.062 5
32	5	0.031 25
64	6	0.015 625
128	7	0.007 812 5
256	8	0.003 906 25
512	9	0.001 953 125
1 024	10	0.000 976 562 5
2 048	11	0.000 488 281 25
4 096	12	0.000 244 140 625
8 192	13	0.000 122 070 312 5
16 384	14	0.000 061 035 156 25
32 768	15	0.000 030 517 578 125
65 536	16	0.000 015 258 789 062 5
131 072	17	0.000 007 629 394 531 25
262 144	18	0.000 003 814 697 265 625
524 288	19	0.000 001 907 348 632 812 5
1 048 576	20	0.000 000 953 674 316 406 25
2 097 152	21	0.000 000 476 837 158 203 125
4 194 304	22	0.000 000 238 418 579 101 562 5
8 388 608	23	0.000 000 119 209 289 550 781 25
16 777 216	24	0.000 000 059 604 644 775 390 625
33 554 432	25	0.000 000 029 802 322 387 695 312 5
67 108 864	26	0.000 000 014 901 161 193 847 656 25
134 217 728	27	0.000 000 007 450 580 596 923 828 125
268 435 456	28	0.000 000 003 725 290 298 461 914 062 5
536 870 912	29	0.000 000 001 862 645 149 230 957 031 25
1 073 741 824	30	0.000 000 000 931 322 574 615 478 515 625
2 147 483 648	31	0.000 000 000 465 661 287 307 739 257 812 5
4 294 967 296	32	0.000 000 000 232 830 643 653 869 628 906 25
8 589 934 592	33	0.000 000 000 116 415 321 826 934 814 453 125
17 179 869 184	34	0.000 000 000 058 207 660 913 467 407 226 562 5
34 359 738 368	35	0.000 000 000 029 103 830 456 733 703 613 281 25
68 719 476 736	36	0.000 000 000 014 551 915 228 366 851 806 640 625
137 438 953 472	37	0.000 000 000 007 275 957 614 183 425 903 320 312 5
274 877 906 944	38	0.000 000 000 003 637 978 807 091 712 951 660 156 25
549 755 813 888	39	0.000 000 000 001 818 989 403 545 856 475 830 078 125
1 099 511 627 776	40	0.000 000 000 000 909 494 701 772 928 237 915 039 062 5
2 199 023 255 552	41	0.000 000 000 000 454 747 350 886 464 118 957 519 531 25
4 398 046 511 104	42	0.000 000 000 000 227 373 675 443 232 059 478 759 765 625
8 796 093 022 208	43	0.000 000 000 000 113 686 837 721 616 029 739 379 882 812 5
17 592 186 044 416	44	0.000 000 000 000 056 843 418 860 808 014 869 689 941 406 25
35 184 372 088 832	45	0.000 000 000 000 028 421 709 430 404 007 434 844 970 703 125
70 368 744 177 664	46	0.000 000 000 000 014 210 854 715 202 003 717 422 485 351 562 5
140 737 488 355 328	47	0.000 000 000 000 007 105 427 357 601 001 858 711 242 675 781 25
281 474 976 710 656	48	0.000 000 000 000 003 552 713 678 800 500 929 355 621 337 890 625
562 949 953 421 312	49	0.000 000 000 000 001 776 356 839 400 250 464 677 810 668 945 312 5
1 125 899 906 842 624	50	0.000 000 000 000 000 888 178 419 700 125 232 338 905 334 472 656 25
2 251 799 813 685 248	51	0.000 000 000 000 000 444 089 209 850 062 616 169 452 667 236 328 125
4 503 599 627 370 496	52	0.000 000 000 000 000 222 044 604 925 031 308 084 726 333 618 164 062 5
9 007 199 254 740 992	53	0.000 000 000 000 000 111 022 302 462 515 654 042 363 166 809 082 031 25
18 014 398 509 481 984	54	0.000 000 000 000 000 055 511 151 231 257 827 021 181 583 404 541 015 625
36 028 797 018 963 968	55	0.000 000 000 000 000 027 755 575 615 628 913 510 590 791 702 270 507 812 5
72 057 594 037 927 936	56	0.000 000 000 000 000 013 877 787 807 814 456 755 295 395 851 135 253 906 25
144 115 188 075 855 872	57	0.000 000 000 000 000 006 938 893 903 907 228 377 647 697 925 567 626 953 125
288 230 376 151 711 744	58	0.000 000 000 000 000 003 469 446 951 953 614 188 823 848 962 783 813 476 562 5
576 460 752 303 423 488	59	0.000 000 000 000 000 001 734 723 475 976 807 094 411 924 481 391 906 738 281 25
1 152 921 504 606 846 976	60	0.000 000 000 000 000 000 867 361 737 988 403 547 205 962 240 695 953 369 140 625
2 305 843 009 213 693 952	61	0.000 000 000 000 000 000 433 680 868 994 201 773 602 981 120 347 976 684 570 312 5
4 611 686 018 427 387 904	62	0.000 000 000 000 000 000 216 840 434 497 100 886 801 490 560 173 988 342 285 156 25
9 223 372 036 854 775 808	63	0.000 000 000 000 000 000 108 420 217 248 550 443 400 745 280 086 994 171 142 578 125
18 446 744 073 709 551 616	64	0.000 000 000 000 000 000 054 210 108 624 275 221 700 372 640 043 497 085 571 289 062 5
36 893 488 147 419 103 232	65	0.000 000 000 000 000 000 027 105 054 312 137 610 850 186 320 021 748 542 785 644 531 25
73 786 976 294 838 206 464	66	0.000 000 000 000 000 000 013 552 527 156 068 805 425 093 160 010 874 271 392 822 265 625
147 573 952 589 676 412 928	67	0.000 000 000 000 000 000 006 776 263 578 034 402 712 546 580 005 437 135 696 411 132 812 5
295 147 905 179 352 825 856	68	0.000 000 000 000 000 000 003 388 131 789 017 201 356 273 290 002 718 567 848 205 566 406 25
590 295 810 358 705 651 712	69	0.000 000 000 000 000 000 001 694 065 894 508 600 678 136 645 001 359 283 924 102 783 203 125
1 180 591 620 717 411 303 424	70	0.000 000 000 000 000 000 000 847 032 947 254 300 339 068 322 500 679 641 962 051 391 601 562 5
2 361 183 241 434 822 606 848	71	0.000 000 000 000 000 000 000 423 516 473 627 150 169 534 161 250 339 820 981 025 695 800 781 25
4 722 366 482 869 645 213 696	72	0.000 000 000 000 000 000 000 211 758 236 813 575 084 767 080 625 169 910 490 512 847 900 390 625

OCTAL-DECIMAL INTEGER CONVERSION TABLE

0000 to 0777 (Octal)	0000 to 0511 (Decimal)

Octal	Decimal
10000 -	4096
20000 -	8192
30000 -	12288
40000 -	16384
50000 -	20480
60000 -	24576
70000 -	28672

	0	1	2	3	4	5	6	7
0000	0000	0001	0002	0003	0004	0005	0006	0007
0010	0008	0009	0010	0011	0012	0013	0014	0015
0020	0016	0017	0018	0019	0020	0021	0022	0023
0030	0024	0025	0026	0027	0028	0029	0030	0031
0040	0032	0033	0034	0035	0036	0037	0038	0039
0050	0040	0041	0042	0043	0044	0045	0046	0047
0060	0048	0049	0050	0051	0052	0053	0054	0055
0070	0056	0057	0058	0059	0060	0061	0062	0063
0100	0064	0065	0066	0067	0068	0069	0070	0071
0110	0072	0073	0074	0075	0076	0077	0078	0079
0120	0080	0081	0082	0083	0084	0085	0086	0087
0130	0088	0089	0090	0091	0092	0093	0094	0095
0140	0096	0097	0098	0099	0100	0101	0102	0103
0150	0104	0105	0106	0107	0108	0109	0110	0111
0160	0112	0113	0114	0115	0116	0117	0118	0119
0170	0120	0121	0122	0123	0124	0125	0126	0127
0200	0128	0129	0130	0131	0132	0133	0134	0135
0210	0136	0137	0138	0139	0140	0141	0142	0143
0220	0144	0145	0146	0147	0148	0149	0150	0151
0230	0152	0153	0154	0155	0156	0157	0158	0159
0240	0160	0161	0162	0163	0164	0165	0166	0167
0250	0168	0169	0170	0171	0172	0173	0174	0175
0260	0176	0177	0178	0179	0180	0181	0182	0183
0270	0184	0185	0186	0187	0188	0189	0190	0191
0300	0192	0193	0194	0195	0196	0197	0198	0199
0310	0200	0201	0202	0203	0204	0205	0206	0207
0320	0208	0209	0210	0211	0212	0213	0214	0215
0330	0216	0217	0218	0219	0220	0221	0222	0223
0340	0224	0225	0226	0227	0228	0229	0230	0231
0350	0232	0233	0234	0235	0236	0237	0238	0239
0360	0240	0241	0242	0243	0244	0245	0246	0247
0370	0248	0249	0250	0251	0252	0253	0254	0255

	0	1	2	3	4	5	6	7
0400	0256	0257	0258	0259	0260	0261	0262	0263
0410	0264	0265	0266	0267	0268	0269	0270	0271
0420	0272	0273	0274	0275	0276	0277	0278	0279
0430	0280	0281	0282	0283	0284	0285	0286	0287
0440	0288	0289	0290	0291	0292	0293	0294	0295
0450	0296	0297	0298	0299	0300	0301	0302	0303
0460	0304	0305	0306	0307	0308	0309	0310	0311
0470	0312	0313	0314	0315	0316	0317	0318	0319
0500	0320	0321	0322	0323	0324	0325	0326	0327
0510	0328	0329	0330	0331	0332	0333	0334	0335
0520	0336	0337	0338	0339	0340	0341	0342	0343
0530	0344	0345	0346	0347	0348	0349	0350	0351
0540	0352	0353	0354	0355	0356	0357	0358	0359
0550	0360	0361	0362	0363	0364	0365	0366	0367
0560	0368	0369	0370	0371	0372	0373	0374	0375
0570	0376	0377	0378	0379	0380	0381	0382	0383
0600	0384	0385	0386	0387	0388	0389	0390	0391
0610	0392	0393	0394	0395	0396	0397	0398	0399
0620	0400	0401	0402	0403	0404	0405	0406	0407
0630	0408	0409	0410	0411	0412	0413	0414	0415
0640	0416	0417	0418	0419	0420	0421	0422	0423
0650	0424	0425	0426	0427	0428	0429	0430	0431
0660	0432	0433	0434	0435	0436	0437	0438	0439
0670	0440	0441	0442	0443	0444	0445	0446	0447
0700	0448	0449	0450	0451	0452	0453	0454	0455
0710	0456	0457	0458	0459	0460	0461	0462	0463
0720	0464	0465	0466	0467	0468	0469	0470	0471
0730	0472	0473	0474	0475	0476	0477	0478	0479
0740	0480	0481	0482	0483	0484	0485	0486	0487
0750	0488	0489	0490	0491	0492	0493	0494	0495
0760	0496	0497	0498	0499	0500	0501	0502	0503
0770	0504	0505	0506	0507	0508	0509	0510	0511

1000 to 1777 (Octal)	0512 to 1023 (Decimal)

	0	1	2	3	4	5	6	7
1000	0512	0513	0514	0515	0516	0517	0518	0519
1010	0520	0521	0522	0523	0524	0525	0526	0527
1020	0528	0529	0530	0531	0532	0533	0534	0535
1030	0536	0537	0538	0539	0540	0541	0542	0543
1040	0544	0545	0546	0547	0548	0549	0550	0551
1050	0552	0553	0554	0555	0556	0557	0558	0559
1060	0560	0561	0562	0563	0564	0565	0566	0567
1070	0568	0569	0570	0571	0572	0573	0574	0575
1100	0576	0577	0578	0579	0580	0581	0582	0583
1110	0584	0585	0586	0587	0588	0589	0590	0591
1120	0592	0593	0594	0595	0596	0597	0598	0599
1130	0600	0601	0602	0603	0604	0605	0606	0607
1140	0608	0609	0610	0611	0612	0613	0614	0615
1150	0616	0617	0618	0619	0620	0621	0622	0623
1160	0624	0625	0626	0627	0628	0629	0630	0631
1170	0632	0633	0634	0635	0636	0637	0638	0639
1200	0640	0641	0642	0643	0644	0645	0646	0647
1210	0648	0649	0650	0651	0652	0653	0654	0655
1220	0656	0657	0658	0659	0660	0661	0662	0663
1230	0664	0665	0666	0667	0668	0669	0670	0671
1240	0672	0673	0674	0675	0676	0677	0678	0679
1250	0680	0681	0682	0683	0684	0685	0686	0687
1260	0688	0689	0690	0691	0692	0693	0694	0695
1270	0696	0697	0698	0699	0700	0701	0702	0703
1300	0704	0705	0706	0707	0708	0709	0710	0711
1310	0712	0713	0714	0715	0716	0717	0718	0719
1320	0720	0721	0722	0723	0724	0725	0726	0727
1330	0728	0729	0730	0731	0732	0733	0734	0735
1340	0736	0737	0738	0739	0740	0741	0742	0743
1350	0744	0745	0746	0747	0748	0749	0750	0751
1360	0752	0753	0754	0755	0756	0757	0758	0759
1370	0760	0761	0762	0763	0764	0765	0766	0767

	0	1	2	3	4	5	6	7
1400	0768	0769	0770	0771	0772	0773	0774	0775
1410	0776	0777	0778	0779	0780	0781	0782	0783
1420	0784	0785	0786	0787	0788	0789	0790	0791
1430	0792	0793	0794	0795	0796	0797	0798	0799
1440	0800	0801	0802	0803	0804	0805	0806	0807
1450	0808	0809	0810	0811	0812	0813	0814	0815
1460	0816	0817	0818	0819	0820	0821	0822	0823
1470	0824	0825	0826	0827	0828	0829	0830	0831
1500	0832	0833	0834	0835	0836	0837	0838	0839
1510	0840	0841	0842	0843	0844	0845	0846	0847
1520	0848	0849	0850	0851	0852	0853	0854	0855
1530	0856	0857	0858	0859	0860	0861	0862	0863
1540	0864	0865	0866	0867	0868	0869	0870	0871
1550	0872	0873	0874	0875	0876	0877	0878	0879
1560	0880	0881	0882	0883	0884	0885	0886	0887
1570	0888	0889	0890	0891	0892	0893	0894	0895
1600	0896	0897	0898	0899	0900	0901	0902	0903
1610	0904	0905	0906	0907	0908	0909	0910	0911
1620	0912	0913	0914	0915	0916	0917	0918	0919
1630	0920	0921	0922	0923	0924	0925	0926	0927
1640	0928	0929	0930	0931	0932	0933	0934	0935
1650	0936	0937	0938	0939	0940	0941	0942	0943
1660	0944	0945	0946	0947	0948	0949	0950	0951
1670	0952	0953	0954	0955	0956	0957	0958	0959
1700	0960	0961	0962	0963	0964	0965	0966	0967
1710	0968	0969	0970	0971	0972	0973	0974	0975
1720	0976	0977	0978	0979	0980	0981	0982	0983
1730	0984	0985	0986	0987	0988	0989	0990	0991
1740	0992	0993	0994	0995	0996	0997	0998	0999
1750	1000	1001	1002	1003	1004	1005	1006	1007
1760	1008	1009	1010	1011	1012	1013	1014	1015
1770	1016	1017	1018	1019	1020	1021	1022	1023

OCTAL-DECIMAL INTEGER CONVERSION TABLE (continued)

	0	1	2	3	4	5	6	7
2000	1024	1025	1026	1027	1028	1029	1030	1031
2010	1032	1033	1034	1035	1036	1037	1038	1039
2020	1040	1041	1042	1043	1044	1045	1046	1047
2030	1048	1049	1050	1051	1052	1053	1054	1055
2040	1056	1057	1058	1059	1060	1061	1062	1063
2050	1064	1065	1066	1067	1068	1069	1070	1071
2060	1072	1073	1074	1075	1076	1077	1078	1079
2070	1080	1081	1082	1083	1084	1085	1086	1087
2100	1088	1089	1090	1091	1092	1093	1094	1095
2110	1096	1097	1098	1099	1100	1101	1102	1103
2120	1104	1105	1106	1107	1108	1109	1110	1111
2130	1112	1113	1114	1115	1116	1117	1118	1119
2140	1120	1121	1122	1123	1124	1125	1126	1127
2150	1128	1129	1130	1131	1132	1133	1134	1135
2160	1136	1137	1138	1139	1140	1141	1142	1143
2170	1144	1145	1146	1147	1148	1149	1150	1151
2200	1152	1153	1154	1155	1156	1157	1158	1159
2210	1160	1161	1162	1163	1164	1165	1166	1167
2220	1168	1169	1170	1171	1172	1173	1174	1175
2230	1176	1177	1178	1179	1180	1181	1182	1183
2240	1184	1185	1186	1187	1188	1189	1190	1191
2250	1192	1193	1194	1195	1196	1197	1198	1199
2260	1200	1201	1202	1203	1204	1205	1206	1207
2270	1208	1209	1210	1211	1212	1213	1214	1215
2300	1216	1217	1218	1219	1220	1221	1222	1223
2310	1224	1225	1226	1227	1228	1229	1230	1231
2320	1232	1233	1234	1235	1236	1237	1238	1239
2330	1240	1241	1242	1243	1244	1245	1246	1247
2340	1248	1249	1250	1251	1252	1253	1254	1255
2350	1256	1257	1258	1259	1260	1261	1262	1263
2360	1264	1265	1266	1267	1268	1269	1270	1271
2370	1272	1273	1274	1275	1276	1277	1278	1279

	0	1	2	3	4	5	6	7
2400	1280	1281	1282	1283	1284	1285	1286	1287
2410	1288	1289	1290	1291	1292	1293	1294	1295
2420	1296	1297	1298	1299	1300	1301	1302	1303
2430	1304	1305	1306	1307	1308	1309	1310	1311
2440	1312	1313	1314	1315	1316	1317	1318	1319
2450	1320	1321	1322	1323	1324	1325	1326	1327
2460	1328	1329	1330	1331	1332	1333	1334	1335
2470	1336	1337	1338	1339	1340	1341	1342	1343
2500	1344	1345	1346	1347	1348	1349	1350	1351
2510	1352	1353	1354	1355	1356	1357	1358	1359
2520	1360	1361	1362	1363	1364	1365	1366	1367
2530	1368	1369	1370	1371	1372	1373	1374	1375
2540	1376	1377	1378	1379	1380	1381	1382	1383
2550	1384	1385	1386	1387	1388	1389	1390	1391
2560	1392	1393	1394	1395	1396	1397	1398	1399
2570	1400	1401	1402	1403	1404	1405	1406	1407
2600	1408	1409	1410	1411	1412	1413	1414	1415
2610	1416	1417	1418	1419	1420	1421	1422	1423
2620	1424	1425	1426	1427	1428	1429	1430	1431
2630	1432	1433	1434	1435	1436	1437	1438	1439
2640	1440	1441	1442	1443	1444	1445	1446	1447
2650	1448	1449	1450	1451	1452	1453	1454	1455
2660	1456	1457	1458	1459	1460	1461	1462	1463
2670	1464	1465	1466	1467	1468	1469	1470	1471
2700	1472	1473	1474	1475	1476	1477	1478	1479
2710	1480	1481	1482	1483	1484	1485	1486	1487
2720	1488	1489	1490	1491	1492	1493	1494	1495
2730	1496	1497	1498	1499	1500	1501	1502	1503
2740	1504	1505	1506	1507	1508	1509	1510	1511
2750	1512	1513	1514	1515	1516	1517	1518	1519
2760	1520	1521	1522	1523	1524	1525	1526	1527
2770	1528	1529	1530	1531	1532	1533	1534	1535

2000	1024
to	to
2777	1535
(Octal)	(Decimal)

Octal	Decimal
10000 -	4096
20000 -	8192
30000 -	12288
40000 -	16384
50000 -	20480
60000 -	24576
70000 -	28672

	0	1	2	3	4	5	6	7
3000	1536	1537	1538	1539	1540	1541	1542	1543
3010	1544	1545	1546	1547	1548	1549	1550	1551
3020	1552	1553	1554	1555	1556	1557	1558	1559
3030	1560	1561	1562	1563	1564	1565	1566	1567
3040	1568	1569	1570	1571	1572	1573	1574	1575
3050	1576	1577	1578	1579	1580	1581	1582	1583
3060	1584	1585	1586	1587	1588	1589	1590	1591
3070	1592	1593	1594	1595	1596	1597	1598	1599
3100	1600	1601	1602	1603	1604	1605	1606	1607
3110	1608	1609	1610	1611	1612	1613	1614	1615
3120	1616	1617	1618	1619	1620	1621	1622	1623
3130	1624	1625	1626	1627	1628	1629	1630	1631
3140	1632	1633	1634	1635	1636	1637	1638	1639
3150	1640	1641	1642	1643	1644	1645	1646	1647
3160	1648	1649	1650	1651	1652	1653	1654	1655
3170	1656	1657	1658	1659	1660	1661	1662	1663
3200	1664	1665	1666	1667	1668	1669	1670	1671
3210	1672	1673	1674	1675	1676	1677	1678	1679
3220	1680	1681	1682	1683	1684	1685	1686	1687
3230	1688	1689	1690	1691	1692	1693	1694	1695
3240	1696	1697	1698	1699	1700	1701	1702	1703
3250	1704	1705	1706	1707	1708	1709	1710	1711
3260	1712	1713	1714	1715	1716	1717	1718	1719
3270	1720	1721	1722	1723	1724	1725	1726	1727
3300	1728	1729	1730	1731	1732	1733	1734	1735
3310	1736	1737	1738	1739	1740	1741	1742	1743
3320	1744	1745	1746	1747	1748	1749	1750	1751
3330	1752	1753	1754	1755	1756	1757	1758	1759
3340	1760	1761	1762	1763	1764	1765	1766	1767
3350	1768	1769	1770	1771	1772	1773	1774	1775
3360	1776	1777	1778	1779	1780	1781	1782	1783
3370	1784	1785	1786	1787	1788	1789	1790	1791

	0	1	2	3	4	5	6	7
3400	1792	1793	1794	1795	1796	1797	1798	1799
3410	1800	1801	1802	1803	1804	1805	1806	1807
3420	1808	1809	1810	1811	1812	1813	1814	1815
3430	1816	1817	1818	1819	1820	1821	1822	1823
3440	1824	1825	1826	1827	1828	1829	1830	1831
3450	1832	1833	1834	1835	1836	1837	1838	1839
3460	1840	1841	1842	1843	1844	1845	1846	1847
3470	1848	1849	1850	1851	1852	1853	1854	1855
3500	1856	1857	1858	1859	1860	1861	1862	1863
3510	1864	1865	1866	1867	1868	1869	1870	1871
3520	1872	1873	1874	1875	1876	1877	1878	1879
3530	1880	1881	1882	1883	1884	1885	1886	1887
3540	1888	1889	1890	1891	1892	1893	1894	1895
3550	1896	1897	1898	1899	1900	1901	1902	1903
3560	1904	1905	1906	1907	1908	1909	1910	1911
3570	1912	1913	1914	1915	1916	1917	1918	1919
3600	1920	1921	1922	1923	1924	1925	1926	1927
3610	1928	1929	1930	1931	1932	1933	1934	1935
3620	1936	1937	1938	1939	1940	1941	1942	1943
3630	1944	1945	1946	1947	1948	1949	1950	1951
3640	1952	1953	1954	1955	1956	1957	1958	1959
3650	1960	1961	1962	1963	1964	1965	1966	1967
3660	1968	1969	1970	1971	1972	1973	1974	1975
3670	1976	1977	1978	1979	1980	1981	1982	1983
3700	1984	1985	1986	1987	1988	1989	1990	1991
3710	1992	1993	1994	1995	1996	1997	1998	1999
3720	2000	2001	2002	2003	2004	2005	2006	2007
3730	2008	2009	2010	2011	2012	2013	2014	2015
3740	2016	2017	2018	2019	2020	2021	2022	2023
3750	2024	2025	2026	2027	2028	2029	2030	2031
3760	2032	2033	2034	2035	2036	2037	2038	2039
3770	2040	2041	2042	2043	2044	2045	2046	2047

3000	1536
to	to
3777	2047
(Octal)	(Decimal)

OCTAL-DECIMAL INTEGER CONVERSION TABLE (continued)

4000 to 4777 (Octal)	2048 to 2559 (Decimal)

Octal	Decimal
10000 -	4096
20000 -	8192
30000 -	12288
40000 -	16384
50000 -	20480
60000 -	24576
70000 -	28672

	0	1	2	3	4	5	6	7
4000	2048	2049	2050	2051	2052	2053	2054	2055
4010	2056	2057	2058	2059	2060	2061	2062	2063
4020	2064	2065	2066	2067	2068	2069	2070	2071
4030	2072	2073	2074	2075	2076	2077	2078	2079
4040	2080	2081	2082	2083	2084	2085	2086	2087
4050	2088	2089	2090	2091	2092	2093	2094	2095
4060	2096	2097	2098	2099	2100	2101	2102	2103
4070	2104	2105	2106	2107	2108	2109	2110	2111
4100	2112	2113	2114	2115	2116	2117	2118	2119
4110	2120	2121	2122	2123	2124	2125	2126	2127
4120	2128	2129	2130	2131	2132	2133	2134	2135
4130	2136	2137	2138	2139	2140	2141	2142	2143
4140	2144	2145	2146	2147	2148	2149	2150	2151
4150	2152	2153	2154	2155	2156	2157	2158	2159
4160	2160	2161	2162	2163	2164	2165	2166	2167
4170	2168	2169	2170	2171	2172	2173	2174	2175
4200	2176	2177	2178	2179	2180	2181	2182	2183
4210	2184	2185	2186	2187	2188	2189	2190	2191
4220	2192	2193	2194	2195	2196	2197	2198	2199
4230	2200	2201	2202	2203	2204	2205	2206	2207
4240	2208	2209	2210	2211	2212	2213	2214	2215
4250	2216	2217	2218	2219	2220	2221	2222	2223
4260	2224	2225	2226	2227	2228	2229	2230	2231
4270	2232	2233	2234	2235	2236	2237	2238	2239
4300	2240	2241	2242	2243	2244	2245	2246	2247
4310	2248	2249	2250	2251	2252	2253	2254	2255
4320	2256	2257	2258	2259	2260	2261	2262	2263
4330	2264	2265	2266	2267	2268	2269	2270	2271
4340	2272	2273	2274	2275	2276	2277	2278	2279
4350	2280	2281	2282	2283	2284	2285	2286	2287
4360	2288	2289	2290	2291	2292	2293	2294	2295
4370	2296	2297	2298	2299	2300	2301	2302	2303

	0	1	2	3	4	5	6	7
4400	2304	2305	2306	2307	2308	2309	2310	2311
4410	2312	2313	2314	2315	2316	2317	2318	2319
4420	2320	2321	2322	2323	2324	2325	2326	2327
4430	2328	2329	2330	2331	2332	2333	2334	2335
4440	2336	2337	2338	2339	2340	2341	2342	2343
4450	2344	2345	2346	2347	2348	2349	2350	2351
4460	2352	2353	2354	2355	2356	2357	2358	2359
4470	2360	2361	2362	2363	2364	2365	2366	2367
4500	2368	2369	2370	2371	2372	2373	2374	2375
4510	2376	2377	2378	2379	2380	2381	2382	2383
4520	2384	2385	2386	2387	2388	2389	2390	2391
4530	2392	2393	2394	2395	2396	2397	2398	2399
4540	2400	2401	2402	2403	2404	2405	2406	2407
4550	2408	2409	2410	2411	2412	2413	2414	2415
4560	2416	2417	2418	2419	2420	2421	2422	2423
4570	2424	2425	2426	2427	2428	2429	2430	2431
4600	2432	2433	2434	2435	2436	2437	2438	2439
4610	2440	2441	2442	2443	2444	2445	2446	2447
4620	2448	2449	2450	2451	2452	2453	2454	2455
4630	2456	2457	2458	2459	2460	2461	2462	2463
4640	2464	2465	2466	2467	2468	2469	2470	2471
4650	2472	2473	2474	2475	2476	2477	2478	2479
4660	2480	2481	2482	2483	2484	2485	2486	2487
4670	2488	2489	2490	2491	2492	2493	2494	2495
4700	2496	2497	2498	2499	2500	2501	2502	2503
4710	2504	2505	2506	2507	2508	2509	2510	2511
4720	2512	2513	2514	2515	2516	2517	2518	2519
4730	2520	2521	2522	2523	2524	2525	2526	2527
4740	2528	2529	2530	2531	2532	2533	2534	2535
4750	2536	2537	2538	2539	2540	2541	2542	2543
4760	2544	2545	2546	2547	2548	2549	2550	2551
4770	2552	2553	2554	2555	2556	2557	2558	2559

5000 to 5777 (Octal)	2560 to 3071 (Decimal)

	0	1	2	3	4	5	6	7
5000	2560	2561	2562	2563	2564	2565	2566	2567
5010	2568	2569	2570	2571	2572	2573	2574	2575
5020	2576	2577	2578	2579	2580	2581	2582	2583
5030	2584	2585	2586	2587	2588	2589	2590	2591
5040	2592	2593	2594	2595	2596	2597	2598	2599
5050	2600	2601	2602	2603	2604	2605	2606	2607
5060	2608	2609	2610	2611	2612	2613	2614	2615
5070	2616	2617	2618	2619	2620	2621	2622	2623
5100	2624	2625	2626	2627	2628	2629	2630	2631
5110	2632	2633	2634	2635	2636	2637	2638	2639
5120	2640	2641	2642	2643	2644	2645	2646	2647
5130	2648	2649	2650	2651	2652	2653	2654	2655
5140	2656	2657	2658	2659	2660	2661	2662	2663
5150	2664	2665	2666	2667	2668	2669	2670	2671
5160	2672	2673	2674	2675	2676	2677	2678	2679
5170	2680	2681	2682	2683	2684	2685	2686	2687
5200	2688	2689	2690	2691	2692	2693	2694	2695
5210	2696	2697	2698	2699	2700	2701	2702	2703
5220	2704	2705	2706	2707	2708	2709	2710	2711
5230	2712	2713	2714	2715	2716	2717	2718	2719
5240	2720	2721	2722	2723	2724	2725	2726	2727
5250	2728	2729	2730	2731	2732	2733	2734	2735
5260	2736	2737	2738	2739	2740	2741	2742	2743
5270	2744	2745	2746	2747	2748	2749	2750	2751
5300	2752	2753	2754	2755	2756	2757	2758	2759
5310	2760	2761	2762	2763	2764	2765	2766	2767
5320	2768	2769	2770	2771	2772	2773	2774	2775
5330	2776	2777	2778	2779	2780	2781	2782	2783
5340	2784	2785	2786	2787	2788	2789	2790	2791
5350	2792	2793	2794	2795	2796	2797	2798	2799
5360	2800	2801	2802	2803	2804	2805	2806	2807
5370	2808	2809	2810	2811	2812	2813	2814	2815

	0	1	2	3	4	5	6	7
5400	2816	2817	2818	2819	2820	2821	2822	2823
5410	2824	2825	2826	2827	2828	2829	2830	2831
5420	2832	2833	2834	2835	2836	2837	2838	2839
5430	2840	2841	2842	2843	2844	2845	2846	2847
5440	2848	2849	2850	2851	2852	2853	2854	2855
5450	2856	2857	2858	2859	2860	2861	2862	2863
5460	2864	2865	2866	2867	2868	2869	2870	2871
5470	2872	2873	2874	2875	2876	2877	2878	2879
5500	2880	2881	2882	2883	2884	2885	2886	2887
5510	2888	2889	2890	2891	2892	2893	2894	2895
5520	2896	2897	2898	2899	2900	2901	2902	2903
5530	2904	2905	2906	2907	2908	2909	2910	2911
5540	2912	2913	2914	2915	2916	2917	2918	2919
5550	2920	2921	2922	2923	2924	2925	2926	2927
5560	2928	2929	2930	2931	2932	2933	2934	2935
5570	2936	2937	2938	2939	2940	2941	2942	2943
5600	2944	2945	2946	2947	2948	2949	2950	2951
5610	2952	2953	2954	2955	2956	2957	2958	2959
5620	2960	2961	2962	2963	2964	2965	2966	2967
5630	2968	2969	2970	2971	2972	2973	2974	2975
5640	2976	2977	2978	2979	2980	2981	2982	2983
5650	2984	2985	2986	2987	2988	2989	2990	2991
5660	2992	2993	2994	2995	2996	2997	2998	2999
5670	3000	3001	3002	3003	3004	3005	3006	3007
5700	3008	3009	3010	3011	3012	3013	3014	3015
5710	3016	3017	3018	3019	3020	3021	3022	3023
5720	3024	3025	3026	3027	3028	3029	3030	3031
5730	3032	3033	3034	3035	3036	3037	3038	3039
5740	3040	3041	3042	3043	3044	3045	3046	3047
5750	3048	3049	3050	3051	3052	3053	3054	3055
5760	3056	3057	3058	3059	3060	3061	3062	3063
5770	3064	3065	3066	3067	3068	3069	3070	3071

OCTAL-DECIMAL INTEGER CONVERSION TABLE (continued)

	0	1	2	3	4	5	6	7
6000	3072	3073	3074	3075	3076	3077	3078	3079
6010	3080	3081	3082	3083	3084	3085	3086	3087
6020	3088	3089	3090	3091	3092	3093	3094	3095
6030	3096	3097	3098	3099	3100	3101	3102	3103
6040	3104	3105	3106	3107	3108	3109	3110	3111
6050	3112	3113	3114	3115	3116	3117	3118	3119
6060	3120	3121	3122	3123	3124	3125	3126	3127
6070	3128	3129	3130	3131	3132	3133	3134	3135
6100	3136	3137	3138	3139	3140	3141	3142	3143
6110	3144	3145	3146	3147	3148	3149	3150	3151
6120	3152	3153	3154	3155	3156	3157	3158	3159
6130	3160	3161	3162	3163	3164	3165	3166	3167
6140	3168	3169	3170	3171	3172	3173	3174	3175
6150	3176	3177	3178	3179	3180	3181	3182	3183
6160	3184	3185	3186	3187	3188	3189	3190	3191
6170	3192	3193	3194	3195	3196	3197	3198	3199
6200	3200	3201	3202	3203	3204	3205	3206	3207
6210	3208	3209	3210	3211	3212	3213	3214	3215
6220	3216	3217	3218	3219	3220	3221	3222	3223
6230	3224	3225	3226	3227	3228	3229	3230	3231
6240	3232	3233	3234	3235	3236	3237	3238	3239
6250	3240	3241	3242	3243	3244	3245	3246	3247
6260	3248	3249	3250	3251	3252	3253	3254	3255
6270	3256	3257	3258	3259	3260	3261	3262	3263
6300	3264	3265	3266	3267	3268	3269	3270	3271
6310	3272	3273	3274	3275	3276	3277	3278	3279
6320	3280	3281	3282	3283	3284	3285	3286	3287
6330	3288	3289	3290	3291	3292	3293	3294	3295
6340	3296	3297	3298	3299	3300	3301	3302	3303
6350	3304	3305	3306	3307	3308	3309	3310	3311
6360	3312	3313	3314	3315	3316	3317	3318	3319
6370	3320	3321	3322	3323	3324	3325	3326	3327

	0	1	2	3	4	5	6	7
6400	3328	3329	3330	3331	3332	3333	3334	3335
6410	3336	3337	3338	3339	3340	3341	3342	3343
6420	3344	3345	3346	3347	3348	3349	3350	3351
6430	3352	3353	3354	3355	3356	3357	3358	3359
6440	3360	3361	3362	3363	3364	3365	3366	3367
6450	3368	3369	3370	3371	3372	3373	3374	3375
6460	3376	3377	3378	3379	3380	3381	3382	3383
6470	3384	3385	3386	3387	3388	3389	3390	3391
6500	3392	3393	3394	3395	3396	3397	3398	3399
6510	3400	3401	3402	3403	3404	3405	3406	3407
6520	3408	3409	3410	3411	3412	3413	3414	3415
6530	3416	3417	3418	3419	3420	3421	3422	3423
6540	3424	3425	3426	3427	3428	3429	3430	3431
6550	3432	3433	3434	3435	3436	3437	3438	3439
6560	3440	3441	3442	3443	3444	3445	3446	3447
6570	3448	3449	3450	3451	3452	3453	3454	3455
6600	3456	3457	3458	3459	3460	3461	3462	3463
6610	3464	3465	3466	3467	3468	3469	3470	3471
6620	3472	3473	3474	3475	3476	3477	3478	3479
6630	3480	3481	3482	3483	3484	3485	3486	3487
6640	3488	3489	3490	3491	3492	3493	3494	3495
6650	3496	3497	3498	3499	3500	3501	3502	3503
6660	3504	3505	3506	3507	3508	3509	3510	3511
6670	3512	3513	3514	3515	3516	3517	3518	3519
6700	3520	3521	3522	3523	3524	3525	3526	3527
6710	3528	3529	3530	3531	3532	3533	3534	3535
6720	3536	3537	3538	3539	3540	3541	3542	3543
6730	3544	3545	3546	3547	3548	3549	3550	3551
6740	3552	3553	3554	3555	3556	3557	3558	3559
6750	3560	3561	3562	3563	3564	3565	3566	3567
6760	3568	3569	3570	3571	3572	3573	3574	3575
6770	3576	3577	3578	3579	3580	3581	3582	3583

6000 to 6777 (Octal)	3072 to 3583 (Decimal)

Octal	Decimal
10000 -	4096
20000 -	8192
30000 -	12288
40000 -	16384
50000 -	20480
60000 -	24576
70000 -	28672

	0	1	2	3	4	5	6	7
7000	3584	3585	3586	3587	3588	3589	3590	3591
7010	3592	3593	3594	3595	3596	3597	3598	3599
7020	3600	3601	3602	3603	3604	3605	3606	3607
7030	3608	3609	3610	3611	3612	3613	3614	3615
7040	3616	3617	3618	3619	3620	3621	3622	3623
7050	3624	3625	3626	3627	3628	3629	3630	3631
7060	3632	3633	3634	3635	3636	3637	3638	3639
7070	3640	3641	3642	3643	3644	3645	3646	3647
7100	3648	3649	3650	3651	3652	3653	3654	3655
7110	3656	3657	3658	3659	3660	3661	3662	3663
7120	3664	3665	3666	3667	3668	3669	3670	3671
7130	3672	3673	3674	3675	3676	3677	3678	3679
7140	3680	3681	3682	3683	3684	3685	3686	3687
7150	3688	3689	3690	3691	3692	3693	3694	3695
7160	3696	3697	3698	3699	3700	3701	3702	3703
7170	3704	3705	3706	3707	3708	3709	3710	3711
7200	3712	3713	3714	3715	3716	3717	3718	3719
7210	3720	3721	3722	3723	3724	3725	3726	3727
7220	3728	3729	3730	3731	3732	3733	3734	3735
7230	3736	3737	3738	3739	3740	3741	3742	3743
7240	3744	3745	3746	3747	3748	3749	3750	3751
7250	3752	3753	3754	3755	3756	3757	3758	3759
7260	3760	3761	3762	3763	3764	3765	3766	3767
7270	3768	3769	3770	3771	3772	3773	3774	3775
7300	3776	3777	3778	3779	3780	3781	3782	3783
7310	3784	3785	3786	3787	3788	3789	3790	3791
7320	3792	3793	3794	3795	3796	3797	3798	3799
7330	3800	3801	3802	3803	3804	3805	3806	3807
7340	3808	3809	3810	3811	3812	3813	3814	3815
7350	3816	3817	3818	3819	3820	3821	3822	3823
7360	3824	3825	3826	3827	3828	3829	3830	3831
7370	3832	3833	3834	3835	3836	3837	3838	3839

	0	1	2	3	4	5	6	7
7400	3840	3841	3842	3843	3844	3845	3846	3847
7410	3848	3849	3850	3851	3852	3853	3854	3855
7420	3856	3857	3858	3859	3860	3861	3862	3863
7430	3864	3865	3866	3867	3868	3869	3870	3871
7440	3872	3873	3874	3875	3876	3877	3878	3879
7450	3880	3881	3882	3883	3884	3885	3886	3887
7460	3888	3889	3890	3891	3892	3893	3894	3895
7470	3896	3897	3898	3899	3900	3901	3902	3903
7500	3904	3905	3906	3907	3908	3909	3910	3911
7510	3912	3913	3914	3915	3916	3917	3918	3919
7520	3920	3921	3922	3923	3924	3925	3926	3927
7530	3928	3929	3930	3931	3932	3933	3934	3935
7540	3936	3937	3938	3939	3940	3941	3942	3943
7550	3944	3945	3946	3947	3948	3949	3950	3951
7560	3952	3953	3954	3955	3956	3957	3958	3959
7570	3960	3961	3962	3963	3964	3965	3966	3967
7600	3968	3969	3970	3971	3972	3973	3974	3975
7610	3976	3977	3978	3979	3980	3981	3982	3983
7620	3984	3985	3986	3987	3988	3989	3990	3991
7630	3992	3993	3994	3995	3996	3997	3998	3999
7640	4000	4001	4002	4003	4004	4005	4006	4007
7650	4008	4009	4010	4011	4012	4013	4014	4015
7660	4016	4017	4018	4019	4020	4021	1022	4023
7670	4024	4025	4026	4027	4028	4029	4030	4031
7700	4032	4033	4034	4035	4036	4037	4038	4039
7710	4040	4041	4042	4043	4044	4045	4046	4047
7720	4048	4049	4050	4051	4052	4053	4054	4055
7730	4056	4057	4058	4059	4060	4061	4062	4063
7740	4064	4065	4066	4067	4068	4069	4070	4071
7750	4072	4073	4074	4075	4076	4077	4078	4079
7760	4080	4081	4082	4083	4084	4085	4086	4087
7770	4088	4089	4090	4091	4092	4093	4094	4095

7000 to 7777 (Octal)	3584 to 4095 (Decimal)

OCTAL-DECIMAL FRACTION CONVERSION TABLE

OCTAL	DEC.	OCTAL	DEC.	OCTAL	DEC.	OCTAL	DEC.
.000	.000000	.100	.125000	.200	.250000	.300	.375000
.001	.001953	.101	.126953	.201	.251953	.301	.376953
.002	.003906	.102	.128906	.202	.253906	.302	.378906
.003	.005859	.103	.130859	.203	.255859	.303	.380859
.004	.007812	.104	.132812	.204	.257812	.304	.382812
.005	.009765	.105	.134765	.205	.259765	.305	.384765
.006	.011718	.106	.136718	.206	.261718	.306	.386718
.007	.013671	.107	.138671	.207	.263671	.307	.388671
.010	.015625	.110	.140625	.210	.265625	.310	.390625
.011	.017578	.111	.142578	.211	.267578	.311	.392578
.012	.019531	.112	.144531	.212	.269531	.312	.394531
.013	.021484	.113	.146484	.213	.271484	.313	.396484
.014	.023437	.114	.148437	.214	.273437	.314	.398437
.015	.025390	.115	.150390	.215	.275390	.315	.400390
.016	.027343	.116	.152343	.216	.277343	.316	.402343
.017	.029296	.117	.154296	.217	.279296	.317	.404296
.020	.031250	.120	.156250	.220	.281250	.320	.406250
.021	.033203	.121	.158203	.221	.283203	.321	.408203
.022	.035156	.122	.160156	.222	.285156	.322	.410156
.023	.037109	.123	.162109	.223	.287109	.323	.412109
.024	.039062	.124	.164062	.224	.289062	.324	.414062
.025	.041015	.125	.166015	.225	.291015	.325	.416015
.026	.042968	.126	.167968	.226	.292968	.326	.417968
.027	.044921	.127	.169921	.227	.294921	.327	.419921
.030	.046875	.130	.171875	.230	.296875	.330	.421875
.031	.048828	.131	.173828	.231	.298828	.331	.423828
.032	.050781	.132	.175781	.232	.300781	.332	.425781
.033	.052734	.133	.177734	.233	.302734	.333	.427734
.034	.054687	.134	.179687	.234	.304687	.334	.429687
.035	.056640	.135	.181640	.235	.306640	.335	.431640
.036	.058593	.136	.183593	.236	.308593	.336	.433593
.037	.060546	.137	.185546	.237	.310546	.337	.435546
.040	.062500	.140	.187500	.240	.312500	.340	.437500
.041	.064453	.141	.189453	.241	.314453	.341	.439453
.042	.066406	.142	.191406	.242	.316406	.342	.441406
.043	.068359	.143	.193359	.243	.318359	.343	.443359
.044	.070312	.144	.195312	.244	.320312	.344	.445312
.045	.072265	.145	.197265	.245	.322265	.345	.447265
.046	.074218	.146	.199218	.246	.324218	.346	.449218
.047	.076171	.147	.201171	.247	.326171	.347	.451171
.050	.078125	.150	.203125	.250	.328125	.350	.453125
.051	.080078	.151	.205078	.251	.330078	.351	.455078
.052	.082031	.152	.207031	.252	.332031	.352	.457031
.053	.083984	.153	.208984	.253	.333984	.353	.458984
.054	.085937	.154	.210937	.254	.335937	.354	.460937
.055	.087890	.155	.212890	.255	.337890	.355	.462890
.056	.089843	.156	.214843	.256	.339843	.356	.464843
.057	.091796	.157	.216796	.257	.341796	.357	.466796
.060	.093750	.160	.218750	.260	.343750	.360	.468750
.061	.095703	.161	.220703	.261	.345703	.361	.470703
.062	.097656	.162	.222656	.262	.347656	.362	.472656
.063	.099609	.163	.224609	.263	.349609	.363	.474609
.064	.101562	.164	.226562	.264	.351562	.364	.476562
.065	.103515	.165	.228515	.265	.353515	.365	.478515
.066	.105468	.166	.230468	.266	.355468	.366	.480468
.067	.107421	.167	.232421	.267	.357421	.367	.482421
.070	.109375	.170	.234375	.270	.359375	.370	.484375
.071	.111328	.171	.236328	.271	.361328	.371	.486328
.072	.113281	.172	.238281	.272	.363281	.372	.488281
.073	.115234	.173	.240234	.273	.365234	.373	.490234
.074	.117187	.174	.242187	.274	.367187	.374	.492187
.075	.119140	.175	.244140	.275	.369140	.375	.494140
.076	.121093	.176	.246093	.276	.371093	.376	.496093
.077	.123046	.177	.248046	.277	.373046	.377	.498046

OCTAL-DECIMAL FRACTION CONVERSION TABLE (continued)

OCTAL	DEC.	OCTAL	DEC.	OCTAL	DEC.	OCTAL	DEC.
.000000	.000000	.000100	.000244	.000200	.000488	.000300	.000732
.000001	.000003	.000101	.000247	.000201	.000492	.000301	.000736
.000002	.000007	.000102	.000251	.000202	.000495	.000302	.000740
.000003	.000011	.000103	.000255	.000203	.000499	.000303	.000743
.000004	.000015	.000104	.000259	.000204	.000503	.000304	.000747
.000005	.000019	.000105	.000263	.000205	.000507	.000305	.000751
.000006	.000022	.000106	.000267	.000206	.000511	.000306	.000755
.000007	.000026	.000107	.000270	.000207	.000514	.000307	.000759
.000010	.000030	.000110	.000274	.000210	.000518	.000310	.000762
.000011	.000034	.000111	.000278	.000211	.000522	.000311	.000766
.000012	.000038	.000112	.000282	.000212	.000526	.000312	.000770
.000013	.000041	.000113	.000286	.000213	.000530	.000313	.000774
.000014	.000045	.000114	.000289	.000214	.000534	.000314	.000778
.000015	.000049	.000115	.000293	.000215	.000537	.000315	.000782
.000016	.000053	.000116	.000297	.000216	.000541	.000316	.000785
.000017	.000057	.000117	.000301	.000217	.000545	.000317	.000789
.000020	.000061	.000120	.000305	.000220	.000549	.000320	.000793
.000021	.000064	.000121	.000308	.000221	.000553	.000321	.000797
.000022	.000068	.000122	.000312	.000222	.000556	.000322	.000801
.000023	.000072	.000123	.000316	.000223	.000560	.000323	.000805
.000024	.000076	.000124	.000320	.000224	.000564	.000324	.000808
.000025	.000080	.000125	.000324	.000225	.000568	.000325	.000812
.000026	.000083	.000126	.000328	.000226	.000572	.000326	.000816
.000027	.000087	.000127	.000331	.000227	.000576	.000327	.000820
.000030	.000091	.000130	.000335	.000230	.000579	.000330	.000823
.000031	.000095	.000131	.000339	.000231	.000583	.000331	.000827
.000032	.000099	.000132	.000343	.000232	.000587	.000332	.000831
.000033	.000102	.000133	.000347	.000233	.000591	.000333	.000835
.000034	.000106	.000134	.000350	.000234	.000595	.000334	.000839
.000035	.000110	.000135	.000354	.000235	.000598	.000335	.000843
.000036	.000114	.000136	.000358	.000236	.000602	.000336	.000846
.000037	.000118	.000137	.000362	.000237	.000606	.000337	.000850
.000040	.000122	.000140	.000366	.000240	.000610	.000340	.000854
.000041	.000125	.000141	.000370	.000241	.000614	.000341	.000858
.000042	.000129	.000142	.000373	.000242	.000617	.000342	.000862
.000043	.000133	.000143	.000377	.000243	.000621	.000343	.000865
.000044	.000137	.000144	.000381	.000244	.000625	.000344	.000869
.000045	.000141	.000145	.000385	.000245	.000629	.000345	.000873
.000046	.000144	.000146	.000389	.000246	.000633	.000346	.000877
.000047	.000148	.000147	.000392	.000247	.000637	.000347	.000881
.000050	.000152	.000150	.000396	.000250	.000640	.000350	.000885
.000051	.000156	.000151	.000400	.000251	.000644	.000351	.000888
.000052	.000160	.000152	.000404	.000252	.000648	.000352	.000892
.000053	.000164	.000153	.000408	.000253	.000652	.000353	.000896
.000054	.000167	.000154	.000411	.000254	.000656	.000354	.000900
.000055	.000171	.000155	.000415	.000255	.000659	.000355	.000904
.000056	.000175	.000156	.000419	.000256	.000663	.000356	.000907
.000057	.000179	.000157	.000423	.000257	.000667	.000357	.000911
.000060	.000183	.000160	.000427	.000260	.000671	.000360	.000915
.000061	.000186	.000161	.000431	.000261	.000675	.000361	.000919
.000062	.000190	.000162	.000434	.000262	.000679	.000362	.000923
.000063	.000194	.000163	.000438	.000263	.000682	.000363	.000926
.000064	.000198	.000164	.000442	.000264	.000686	.000364	.000930
.000065	.000202	.000165	.000446	.000265	.000690	.000365	.000934
.000066	.000205	.000166	.000450	.000266	.000694	.000366	.000938
.000067	.000209	.000167	.000453	.000267	.000698	.000367	.000942
.000070	.000213	.000170	.000457	.000270	.000701	.000370	.000946
.000071	.000217	.000171	.000461	.000271	.000705	.000371	.000949
.000072	.000221	.000172	.000465	.000272	.000709	.000372	.000953
.000073	.000225	.000173	.000469	.000273	.000713	.000373	.000957
.000074	.000228	.000174	.000473	.000274	.000717	.000374	.000961
.000075	.000232	.000175	.000476	.000275	.000720	.000375	.000965
.000076	.000236	.000176	.000480	.000276	.000724	.000376	.000968
.000077	.000240	.000177	.000484	.000277	.000728	.000377	.000972

OCTAL-DECIMAL FRACTION CONVERSION TABLE (continued)

OCTAL	DEC.	OCTAL	DEC.	OCTAL	DEC.	OCTAL	DEC.
.000400	.000976	.000500	.001220	.000600	.001464	.000700	.001708
.000401	.000980	.000501	.001224	.000601	.001468	.000701	.001712
.000402	.000984	.000502	.001228	.000602	.001472	.000702	.001716
.000403	.000988	.000503	.001232	.000603	.001476	.000703	.001720
.000404	.000991	.000504	.001235	.000604	.001480	.000704	.001724
.000405	.000995	.000505	.001239	.000605	.001483	.000705	.001728
.000406	.000999	.000506	.001243	.000606	.001487	.000706	.001731
.000407	.001003	.000507	.001247	.000607	.001491	.000707	.001735
.000410	.001007	.000510	.001251	.000610	.001495	.000710	.001739
.000411	.001010	.000511	.001255	.000611	.001499	.000711	.001743
.000412	.001014	.000512	.001258	.000612	.001502	.000712	.001747
.000413	.001018	.000513	.001262	.000613	.001506	.000713	.001750
.000414	.001022	.000514	.001266	.000614	.001510	.000714	.001754
.000415	.001026	.000515	.001270	.000615	.001514	.000715	.001758
.000416	.001029	.000516	.001274	.000616	.001518	.000716	.001762
.000417	.001033	.000517	.001277	.000617	.001522	.000717	.001766
.000420	.001037	.000520	.001281	.000620	.001525	.000720	.001770
.000421	.001041	.000521	.001285	.000621	.001529	.000721	.001773
.000422	.001045	.000522	.001289	.000622	.001533	.000722	.001777
.000423	.001049	.000523	.001293	.000623	.001537	.000723	.001781
.000424	.001052	.000524	.001296	.000624	.001541	.000724	.001785
.000425	.001056	.000525	.001300	.000625	.001544	.000725	.001789
.000426	.001060	.000526	.001304	.000626	.001548	.000726	.001792
.000427	.001064	.000527	.001308	.000627	.001552	.000727	.001796
.000430	.001068	.000530	.001312	.000630	.001556	.000730	.001800
.000431	.001071	.000531	.001316	.000631	.001560	.000731	.001804
.000432	.001075	.000532	.001319	.000632	.001564	.000732	.001808
.000433	.001079	.000533	.001323	.000633	.001567	.000733	.001811
.000434	.001083	.000534	.001327	.000634	.001571	.000734	.001815
.000435	.001087	.000535	.001331	.000635	.001575	.000735	.001819
.000436	.001091	.000536	.001335	.000636	.001579	.000736	.001823
.000437	.001094	.000537	.001338	.000637	.001583	.000737	.001827
.000440	.001098	.000540	.001342	.000640	.001586	.000740	.001831
.000441	.001102	.000541	.001346	.000641	.001590	.000741	.001834
.000442	.001106	.000542	.001350	.000642	.001594	.000742	.001838
.000443	.001110	.000543	.001354	.000643	.001598	.000743	.001842
.000444	.001113	.000544	.001358	.000644	.001602	.000744	.001846
.000445	.001117	.000545	.001361	.000645	.001605	.000745	.001850
.000446	.001121	.000546	.001365	.000646	.001609	.000746	.001853
.000447	.001125	.000547	.001369	.000647	.001613	.000747	.001857
.000450	.001129	.000550	.001373	.000650	.001617	.000750	.001861
.000451	.001132	.000551	.001377	.000651	.001621	.000751	.001865
.000452	.001136	.000552	.001380	.000652	.001625	.000752	.001869
.000453	.001140	.000553	.001384	.000653	.001628	.000753	.001873
.000454	.001144	.000554	.001388	.000654	.001632	.000754	.001876
.000455	.001148	.000555	.001392	.000655	.001636	.000755	.001880
.000456	.001152	.000556	.001396	.000656	.001640	.000756	.001884
.000457	.001155	.000557	.001399	.000657	.001644	.000757	.001888
.000460	.001159	.000560	.001403	.000660	.001647	.000760	.001892
.000461	.001163	.000561	.001407	.000661	.001651	.000761	.001895
.000462	.001167	.000562	.001411	.000662	.001655	.000762	.001899
.000463	.001171	.000563	.001415	.000663	.001659	.000763	.001903
.000464	.001174	.000564	.001419	000664	.001663	.000764	.001907
.000465	.001178	.000565	.001422	.000665	.001667	.000765	.001911
.000466	.001182	.000566	.001426	.000666	.001670	.000766	.001914
.000467	.001186	.000567	.001430	000667	.001674	.000767	.001918
.000470	.001190	.000570	.001434	.000670	.001678	.000770	.001922
.000471	.001194	.000571	.001438	.000671	.001682	.000771	.001926
.000472	.001197	.000572	.001441	.000672	.001686	.000772	.001930
.000473	.001201	.000573	.001445	.000673	.001689	.000773	.001934
.000474	.001205	.000574	.001449	.000674	.001693	.000774	.001937
.000475	.001209	.000575	.001453	.000675	.001697	.000775	.001941
.000476	.001213	.000576	.001457	.000676	.001701	.000776	.001945
.000477	.001216	.000577	.001461	.000677	.001705	.000777	.001949

GLOSSARY

AFT(ATTACH FLAG TABLE)

A table corresponding to the Logical Unit Table (LUT) with 2 word entries for each LUT slot. Whenever a Logical Unit Number (LUN) is attached to a Task, the Task name is set in the corresponding AFT slot. Whenever a LUN and Device-unit are both attached to a Task, the Device attach flag in the Physical Device List points to the appropriate AFT slot.

ATL(ACTIVE TASK LIST)

A priority ordered list of Active Tasks used to drive the system. The ATL is a deque consisting of one node for each Active Task in the system.

CAL INSTRUCTION

A PDP-15 Hardware Instruction used to request Executive routines. All System Directives issue CAL instructions to the Executive when making their requests.

CAL PARAMETER BLOCK

A block consisting of one or more words of contiguous core used to store parameters when issuing System Directives. The System Directive is implemented as a CAL Instruction with the address of the CAL Parameter Block as its operand.

CLOCK QUEUE

The Clock Queue is a deque consisting of one node for each item to be done at some time in the future. These items are: scheduling of Tasks (SCHEDULE, RUN, and SYNC Directives), rescheduling of Tasks (Clock interrupt service routine), and setting of Event Variables after elapsed time periods (MARK Directive). The nodes are linked in the order in which they come due.

COMMON BLOCK, INTERNAL

An area of contiguous core memory within a partition, available only to the Task in the partition during its residency.

COMMON BLOCK, SYSTEM

An area of contiguous core memory, defined at System Configuration time, where data can be stored and referenced by all Tasks. A SYSTEM COMMON BLOCK is referenced by using a COMMON name matching a SYSTEM COMMON BLOCK name and declaring that COMMON as SYSTEM COMMON to the Task Builder.

CONSOLE TELETYPE

The control Teletype of the RSX System where MCR Function requests may be issued by the operator.

CONTROL TABLE

A 3-word table used when requesting or relinquishing disk space or when issuing disk GET and PUT Directives.

CORE RESIDENT TASK

A Task which has been fixed-in-core.

DEFAULT PRIORITY

A priority given to a Task during Task Building or Task Installation that is used when a priority is not specified and the Task's execution is requested or scheduled.

DEQUE

A double ended queue consisting of a listhead and list elements (nodes), circularly linked by both forward and backward pointers. Deques, or link lists, are used, rather than tables, to store system information.

DIRECTIVES

Instructions to the RSX Executive (implemented with the use of CAL Instructions), to perform indicated operations.

DISK RESIDENT TASK

A Task which normally resides on the disk and is brought into a core partition when requested.

EVENT VARIABLE

A word or variable used to determine the status of a Directive. The Event Variable is set to indicate successful completion, rejection, status, or a request still pending. An Event Variable address of zero indicates that no Event Variable is specified.

EXECUTIVE

The heart of the real-time operating system. It coordinates all activities in the system including Task scheduling, I/O supervision, resource allocation, and interactive operator communication.

I/O HANDLER TASK

A Task in the RSX System which contains an interrupt service routine. I/O Handler Tasks are requested whenever they are assigned to a LUN.

LISTHEAD

A two-word core block with forward and backward pointers pointing to the next and previous list node or to itself if empty. The listhead is a reference point in a circularly linked list.

LINKED LIST

A deque consisting of nodes and listhead used to store system information. An empty list consists of only a listhead.

LUN(LOGICAL UNIT NUMBER)

Logical Unit Numbers are used to represent Logical I/O Device Units

rather than Physical Units. Each Logical Unit Number is represented by an entry in the Logical Unit Table.

LUT(LOGICAL UNIT TABLE)

A block of contiguous core with a one-word entry, or slot, for each Logical Unit Number. When a LUN is assigned to a Physical Device Unit, the corresponding LUT slot contains the address of the appropriate Physical Device List node.

MCR(MONITOR CONSOLE ROUTINE)

The MCR allows the user to communicate on-line with the system from the console teleprinter. The MCR consists of the Resident MCR Task, which accepts user's commands, and the MCR Functions, which actually carry out the indicated requests.

NODES

The list elements of a deque. All nodes (of dynamic lists) consist of the listhead followed by eight words of data (list elements).

PARTITION

An area of contiguous core memory, defined at System Configuration time, from which Tasks are executed.

PARTITION BLOCK

An abnormal node (34_8 words) generated by the System Configurator to serve three functions. (1) It contains partition description information to assure that a Task being installed into the system has been built for an existing partition; (2) It provides core for an Event Variable and disk (DSKGET) control table necessary to load Tasks into partitions; and (3) it provides for saving a Task's environment when it is interrupted by the Executive.

PBDL(PARTITION BLOCKS DESCRIPTION LIST)

Partition Blocks generated by the System Configurator are linked together into a deque called the PBDL.

PDVL(PHYSICAL DEVICE LIST)

A deque constructed by the System Configurator used to describe the devices and units in the system. When a logical I/O unit is assigned to a physical unit, the address of the node describing the device and unit is set in a LUT entry corresponding to the LUN.

POOL(POOL OF EMPTY NODES)

Empty ten-word nodes for use in any deque. The Pool is generated by the System Configurator from core area that has not been specified for other use.

SCDL(SYSTEM COMMON BLOCK DEFINITIONS LIST)

A deque consisting of nodes which contain a record of the descriptions of each System COMMON Block.

SIGNIFICANT EVENT

An event which results in the scanning of the Active Task List. The following events are considered "Significant Events": (1) I/O queuing; (2) normal I/O request completion (dependent upon I/O Handler Task); (3) A Task request; (4) a scheduled SCHEDULE, RUN, or SYNC coming due; (5) a Mark time expiration; (6) a Task resumption (RESUME Directive); and (7) a Task EXIT.

STL(SYSTEM TASK LIST)

A directory of all Tasks in the System.

SYSTEM CONFIGURATOR

A Task which allows the user to tailor the RSX System to best fit his requirements.

TKB

The Task Builder program used to build executable Tasks from relocatable binary files.

TRIGGER EVENT VARIABLE

An Event Variable referenced within a PDVL node. The Trigger Event Variable is used to stimulate a dormant I/O Handler Task.

HOW TO OBTAIN SOFTWARE INFORMATION

Announcements for new and revised software, as well as programming notes, software problems, and documentation corrections are published by Software Information Service in the following newsletters.

Digital Software News for the PDP-8 & PDP-12
Digital Software News for the PDP-11
Digital Software News for the PDP-9/15 Family

These newsletters contain information applicable to software available from Digital's Program Library, Articles in Digital Software News update the cumulative Software Performance Summary which is contained in each basic kit of system software for new computers. To assure that the monthly Digital Software News is sent to the appropriate software contact at your installation, please check with the Software Specialist or Sales Engineer at your nearest Digital office.

Questions or problems concerning Digital's Software should be reported to the Software Specialist. In cases where no Software Specialist is available, please send a Software Performance Report form with details of the problem to:

Software Information Service
Digital Equipment Corporation
146 Main Street, Bldg. 3-5
Maynard, Massachusetts 01754

These forms which are provided in the software kit should be fully filled out and accompanied by teletype output as well as listings or tapes of the user program to facilitate a complete investigation. An answer will be sent to the individual and appropriate topics of general interest will be printed in the newsletter.

Orders for new and revised software and manuals, additional Software Performance Report forms, and software price lists should be directed to the nearest Digital Field office or representative. U.S.A. customers may order directly from the Program Library in Maynard. When ordering, include the code number and a brief description of the software requested.

Digital Equipment Computer Users Society (DECUS) maintains a user library and publishes a catalog of programs as well as the DECUSCOPE magazine for its members and non-members who request it. For further information please write to:

DECUS
Digital Equipment Corporation
146 Main Street, Bldg. 3-5
Maynard, Massachusetts 01754

RSX-15
Reference Manual
DEC-15-GRQA-D

READER'S COMMENTS

Digital Equipment Corporation maintains a continuous effort to improve the quality and usefulness of its publications. To do this effectively we need user feedback -- your critical evaluation of this manual.

Please comment on this manual's completeness, accuracy, organization, usability and readability.

Did you find errors in this manual? If so, specify by page.

How can this manual be improved?

Other comments?

Please state your position. _____ Date: _____

Name: _____ Organization: _____

Street: _____ Department: _____

City: _____ State: _____ Zip or Country _____

— — — — — — — — — — — — — — — — — Fold Here — — — — — — — — — — — — — — — — — —

— — — — — — — — — — — — — Do Not Tear - Fold Here and Staple — — — — — — — — — — —

FIRST CLASS
PERMIT NO. 33
MAYNARD, MASS.

BUSINESS REPLY MAIL
NO POSTAGE STAMP NECESSARY IF MAILED IN THE UNITED STATES

Postage will be paid by.

digital

Digital Equipment Corporation
Software Information Services
146 Main Street, Bldg. 3-5
Maynard, Massachusetts 01754

Digital Equipment Corporation
Maynard, Massachusetts

digital

printed in U.S.A.