RSX-11D SPEC

TO: RSX-11D Distribution

FROM: H, Krejci

DATE: 26 Jun 72 /

SUBJ: I/O OPERATIONS

DOC: 132-131-041-00

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

Unless specified otherwise, the terms "RSX" and "RSX-11" imply "RSX-11D".

INTRODUCTION

The RSX-11D I/O structure is intended to provide a flexible device- and function- independent I/O capability that can support standard PDP-11 peripherals as well as special purpose devices. It is expected that users will develop their own special purpose device handling software, and the RSX I/O structure has been designed to make implementation of I/O service as clean and straight-forward as possible (without jeopardizing system integrity or efficency).

Peripheral device support is NOT an integral part of the RSX executive. It is provided by "privileged Tasks" called I/O Handler Tasks, which may be developed or modified without an intimate knowlege of the executive code.

I/O requests are made to logical I/O units, and are mapped into physical device-unit references via a set of "device assignments". Each Task has its own set of assignments, and

they may be changed either by the Task in execution, or from the operator's console (TTY).

An I/O request is made by instructing the system (via System Directive) to queue an I/O request for an indicated LUN (Logical Unit Number). If the LUN is assigned to a physical unit, and if the Handler Task to support that unit is memory resident and initialized, the request is queued by priority (usually the requesting Task's priority) in a request list for the indicated physical unit.

The RSX executive does not attempt to interpret the the request, it only passes it to an I/O Handler Task per LUN assignment, and the disposition of the request is a function of the Handler Task (not the executive).

When an I/O request is queued for a Task, control is returned immediately (contingent upon Task priority, of course) to the requesting Task, and that Task always has the option of suspending execution until completion of an I/O request, or operating asynchronously.

I/O completion may be indicated in any of three optional forms. (1) An Event Flag may be specified to be set (accompanied by a declaration of a Significant Event) at I/O completion. Task execution may be suspended (using the WAITFOR Directive) until an indicated Event Flag, or logical combination of Event Flags, is set.[1] (2) An I/O status word may be specified to be set at completion of an operation. This word may be cleared before queueing a request, and then checked periodically. (3) a System Trap service routine may be included in a Task which will interrupt the Task's execution upon I/O completion.

DEVICE INDEPENDENCE

I/O requests are made to LOGICAL units, which are equivalenced to PHYSICAL device-units via a "Logical Unit Table" (LUT). Logical Units are represented by Logical Unit Numbers (LUNs), and each LUN is represented by an entry in a "Logical Unit Table" (LUT). Physical device-units are represented by entries in a table called the Physical Unit Directory (PUD).

---

[1] The WAITFOR Directive provides an "OR" combination, and a series of WAITFORs provides an "AND".

The logical/physical equivalences are made by "ASSIGNing LUNs to physical device-units,

When a LUN is assigned to a physical device-unit, the corasponding LUT entry (slot) is set to the address of the corresponding PUD entry,

When a LUN is deassigned (assigned to NONE), the corresponding LUT slot is zeroed,

Each Task has its own Logical Unit Table which is a part of the Task's disk image that is brought into memory whenever a Task is FIXED or whenever a non-FIXED Task is run, However, a Task's LUT is not within its virtual address space, When an I/O request is queued (via the QUEUE I/O Directive) the requesting task's LUT (in memory) is used to determine which physical device-unit is to perform the request, and contents of any other LUT is irrevelant,

There are four mechanisms in RSX-11D in which the selection of a PHYSICAL I/O device-unit can be altered, They are as follows,

INSTALL -- When a Task is INSTALLed (MCR or Batch) into a system, the number of LUNs and the assignment of each may be specified, Assignments to device-units without resident Handler Tasks ARE NOT flagged,

REASSIGN -- This MCR Function allows a Task's disk resident assignments to be changed, Assignments to device-units without resident Handler Tasks ARE flagged,

ASSIGN -- This Directive allows a Task to change its memory resident LUN assignments, Assignments to device-units without resident Handler Tasks ARE flagged,

REDIRECT -- This MCR Function allows ALL requests of an indicated PHYSICAL device-unit to be redirected to another PHYSICAL device-unit, This Function is intended to serve in case of peripheral failure, and does not provide normally useful device independence because the redirection is independent of Task or LUN,

A Task's LUN assignments may be made both before and during execution, Pre-execution assignments may be made from the MCR (or Batch) terminal, and run-time assignments may be made from the executing Task, However, except for the REDIRECT MCR Function, there is no means of externally changing the I/O deviceunits used by a Task under

execution.

I/O REQUESTS

I/O requests are made by Tasks using either the "QUEUE I/O PER TASK ASSIGNMENT DIRECTIVE", or the "QUEUE I/O PER MCR ASSIGNMENT DIRECTIVE". These directives are identical except for the LUN assignments used to map the logical unit into a physical unit. The former uses the Task's own LUT, and the latter (used to communicate with the console operator) uses the MCR Dispatch Task's LUT.

When a QUEUE I/O Directive is issued, a check is made to see if the I/O request can be queued[2]. If it cannot, the Directive Status word (requestor's virtual zero) is set negative to indicate rejection, and the negative value indicates the cause for rejection. If the QUEUE I/O Directive is accepted, a request node is formed and inserted into the the device-unit's request queue, the Handler Task, if idle, is triggered into service, and the requestor's Directive status word is set positive (+1) to indicate performance of the DIRECTIVE. An Event Flag and an I/O Status Block may be specified to be set upon completion (disposition) of the I/O request.

There is a separate I/O request queue for each physical device-unit. These queues are deques with their listheads in the PUD entry for the corresponding physical device-unit. I/O requests are queued by priority with the highest priority request at the front of the deque. Requests of equal priority are inserted in the order in which the requests are made.

The QUEUE I/O Directives are indicated by Directive Identification Codes (DICs) "31" (for queue per Task's assignments) and "33" (for queue per MCR assignments), with a Directive Parameter Block (DPB) of the following format:

---

[2] There are several reasons why an I/O request cannot be queued: (1) the indicated LUN does not exist, (2) the LUN is not assigned to a physical unit, (3) a Handler Task to service the physical device is not resident, (4) the conditions for queuing specified were not met, or (5) a node for the request queue is not available.

Wd, 00 -- DIC (01 or 03) & [EFN],
Wd, 01 -- I/O Function code,
Wd, 02 -- LUN & [queuing conditions],
Wd, 03 -- [Priority] & Unused Byte,
Wd, 04 -- [Address of I/O Status Block],
Wd, 05 -- Parameter #1,
Wd, 06 -- Parameter #2,
Wd, 07 -- Parameter #3,
Wd, 10 -- Parameter #4,
Wd, 11 -- Parameter #5,
Wd, 12 -- Parameter #6,
Wd, 13 -- Parameter #7,

The QUEUE I/O Directives are described in the RSX-11D "DIRECTIVES SPEC",

I/O FUNCTION CODES

While the executive does not interpret Function Codes, it does recognize the low order three bits of ALL I/O Function Codes as the following "Function Atributes":

Bit-0 -- No Return function,
Bit-1 -- reserved,
Bit-2 -- reserved,

A "No Return" function is one in which no status or data is returned to the requestor, In these cases, a Task may queue requests and EXIT before their completion without invoking I/O Rundown,

The following is a list of I/O Function Codes recognized by common Handler Tasks, It is not a complete list of Function Codes (Any Handler Task may recognize any function code desired by its Implementer) but rather a list of codes used where devices independence is feasible and practiced,

000402 WRITE LOGICAL RECORD (LINE)
000401 PRINT FILE
001000 READ LOGICAL RECORD (LINE)
001001 READ WITHOUT ECHO (TTY)
001400 ATTACH UNIT TO TASK
002000 DETACH UNIT FROM TASK
002400 ALLOCATE DISK STORAGE
003000 DEALLOCATE DISK STORAGE
003400 TRANSFER IN
003401 LOAD TASK IMAGE **
004000 TRANSFER OUT
004001 RECORD TASK IMAGE **
004402 OPEN FILE FOR INPUT
005000 OPEN FILE FOR OUTPUT

005400 CLOSE FILE
006000 DELETE FILE
006400 RENAME FILE

** Executive functions performed by disk driver(s) only for executive,

I/O STATUS BLOCK

When I/O completion status is desired, the address of an I/O Status Block is included in the Queue I/O DPB, This Status Block consists of two words of the following format:

Wd, 00 -- Status Value & Unused Byte,
Wd, 01 -- Length of transfer (in bytes) for READ/WRITE functions, and device dependent in all other cases,

The following is a list of commonly returned I/O status values, All possible status values returned for a particular device, are described in the RSX Spec for the Handler Task that services it,

A positive value implies successful completion, and a negative value implies rejection of failure, The positive value returned is usually one (+1), however other positive values may be used, viz,, TTY Handler Task identifies CR & AM termination on input, and +U & +S termination on output,

-99 UNRECOGNIZED FUNCTION
-10 INVALID ADDRESS
-20 INVALID PARAMETER(S)
-30 UNIT ALREADY ATTACHED

QUEUING MECHANISM

When an I/O request is queued for a LUN, a request node is formed and inserted by priority in the request list for the device-unit to which the LUN is assigned, The format of the Request node is as follows:

Wd, 00 -- Forward linkage,
Wd, 01 -- Backward linkage,
Wd, 02 -- STD address (Task ID),
Wd, 03 -- ATL node adr of requestor,
Wd, 04 -- Priority & Unused Byte
Wd, 05 -- LUN & EFN,
Wd, 06 -- I/O Function Code,
Wd, 07 -- I/O Status Block address,

```
Wd, 10 -- Parameter #1,
Wd, 11 -- Parameter #2,
Wd, 12 -- Parameter #3,
Wd, 13 -- Parameter #4,
Wd, 14 -- Parameter #5,
Wd, 15 -- Parameter #6,
Wd, 16 -- Parameter #7,
```

After a request node is inserted in a device-unit's request list, the four following operations cause, or aid, the processing of the I/O request: (1) the Handler Task's Event Flag one is set, (2) an "I/O Requests Queued Counter" (for the unit) is incremented, (3) an "I/O Requests Pending Count" (for the requesting Task) is incremented,[3] and (4) a Significant Event is declared.

When an I/O Handler Task is idle, it issues a WAITFOR Directive with Event Flag one specified as an (usually, the) Event Flag whose setting should cause resumption of Handler Task execution. Thus, an idle Handler Task is triggered into service by the queing of an I/O request for any of the units it services.

The I/O Requests Queued Counter is a word in the PUD entry of the unit for which the request was queued. This count is incremented for every request that is queued for the unit, and is provided for Handler Task usage. Most Handler Tasks do not use it, but some special purpose and multi-unit Handler Tasks can operate more efficently with this facility. The count may be altered by the Handler Task at any time.

The "I/O Requests Pending Count" is a word in the ATL node of every active Task. This count is incremented and decremented to provide an indication of pending I/O requests. This is used to delay the freeing of Task's memory if it EXITs or is aborted with unsatisfied I/O requests.

Queuing an I/O request is a Significant Event because it is a possible cause for Task switching (resumption of Handler Task when it is of a higher priority than the requestor Task). Therefore, a Significant Event Declaration is made whenever an I/O request is queued.

---

[3] If a "No Return" function, Requests Pending count is NOT incremented.

HANDLER TASKS

Under RSX-11D, I/O is supported by "privileged tasks" called I/O Handler Tasks,[4] These Tasks are called privileged because they have access to (1) the PDP-11 External Page, (2) the executive's lists and tables, and (3) routines whose misuse could interfere with normal system operation, Privileged Tasks are trusted not to destroy the system under which they run,

I/O Handler Tasks consists of two sections: (1) A "Task level", and (2) an "interrupt service routine", The Task level portion of an I/O Handler Task runs as a normal Task, with its own context, and in a software priority multiprogramming environment, This part of a Handler Task generally interfaces with the executive (viz, dequeues requests) and performs the bulk of the I/O service, The interrupt service routine part of a Handler Task runs per hardware priority (asynchronous to the software priority multiprogramming system) in response to a peripheral device interrupt, Interrupt service routines run in a position independent environment, and generally do as little as possible,

Handler Task names are dictated by convention so that their memory residency may be conveniently controlled, The name of an I/O Handler Task always consists of a two character symbolic peripheral name followed by four dots (periods), viz,, The Handler Task that supports DT0 thru DT9 is called "DT....",

All Handler Tasks (at least all that contain interrupt service routines) should not be declared "checkpointable" when they are INSTALLed into a system, Also, most Handler Tasks use their initialization code for Stack storage space, and therefore should be declared "Not Fixable" when INSTALLed,

Handler Task residency is controlled by the following MCR Functions,

> LOAD -- This MCR Function allows an operator to cause I/O Handler Tasks to become ready to service I/O requests, Handler Tasks are indicated by specifying symbolic peripheral names (viz,, DT,LP,CD), A partition and/or priority may also

---

[4] The system disk driver is a part of the executive assembly, but appears as an I/O Handler Task,

be specified. If a Handler Task cannot be loaded, a rejection message is output.

UNLOAD -- This MCR Function allows an operator to cause an I/O Handler Task to cease to service requests when its request queue(s) are empty. This, of course, frees up memory.

When a Handler Task is loaded, it initializes itself and instructs the system (WAITFOR Directive) to suspend its execution until an I/O request is queued for the Handler Task.

I/O Handler Tasks are supplimented by two sets of memory resident re-entrant subroutines: (1) a Handler Task library, and (2) the System Subroutines.

The Handler Library is created (or not created) at system configuration (SGEN), and provides routines that are common to the more sofisticated I/O service, viz., file structure blocking/unblocking, access methods, etc. The Handler Library must be re-entrant, but not necessarily position-independent.

The System Subroutines are a part of the executive assembly, and always exist. These subroutines provide basic functions, most of which are common to all Handler Tasks, viz., dequeue an I/O request node, return a node to the pool of available list elements, etc. The use of System Subroutines by Handler Tasks is described later.

When a "privileged Task" is INSTALLed into a system, the ASR contents for all but the Task code are determined and recorded with the Task's disk image. The contents of the ASR used for Task code is set by the executive when the Task is loaded (because the real address space in which it is loaded is not fixed). The following is a description of the virtual address space allocations (ASR usage) for I/O Handler Task level code.

Virtual locations 000000-017777 (ASR0) are used for Task level code. Note, this code is limited to 4K.

Virtual locations 020000-077777 (ASRs 1,2,3) are used for execution of Handler Library routines. Note, the Handler Library is limited to 12K.

Virtual locations 100000-157777 (ASRs 4,5,6) are used to access the executive's tables, lists, and System Subroutines. Note -- the system lists, tables, pool, and System Subroutines cannot exceede 12K.

Virtual locations 160000-177777 (ASR7) are used to access the PDP-11 External Page.

While interrupt service routines are physically a part of I/O Handler Tasks (usually a part of the same assembly), they are executed in Kernel mode, and under the kernel's ASR3. Thus, interrupt service routines are written as position-independent code (or written to run in virtual address space 060000-077777).

The kernel address space 100000-177777 (ASRs 4,5,6,7) and an I/O Handler Task's (user space) 100000-177777 are coincidently mapped. I.e., The executive's tables, lists, & System Subroutines, and the external page are available to interrupt service routines (kernel mode) as well as to a Handler's Task level code (user mode).

SAMPLE HANDLER TASK

The following PDP-11 program is a sample I/O Handler Task to support a single unit device called "LP" which prints directly from a requestor's memory. The purpose of this sample program is to illustrate Handler Task construction, and does not represent optimal code or a functional Handler Task. All code unique to the device (not presently an existent peripheral) has been omitted, and in many cases registers are redundantly loaded.

NOTE -- This sample program is included for illustration purposes only. It is expected that changes (perhaps radical) will be made as Handler Task implimentation experience is gained. This sample will be replaced by an actual listing as soon as possible.

The program is used as an example for the remainder of this Spec.

```
; GENERAL REGISTER DEFINITIONS
;
R0=%0
R1=%1
R2=%2
R3=%3
R4=%4
R5=%5
SP=%6
PC=%7
```

```
;
; GLOBAL (EXEC) SYMBOL REFERENCES
;
 .GLOBL   ,,CINT ;CONNECT INTERRUPT
 .GLOBL   ,,DINT ;DISCONNECT INTERRUPT
 .GLOBL   ,,DSUT ;DECLARE & SET
 .GLOBL   ,,CLEF ;CLEAR EVENT FLAG(S) 1-16
 .GLOBL   ,,STEF ;SET EVENT FLAG(S) 1-16
 .GLOBL   ,,DQRQ ;DE-QUEUE AN I/O REQUEST
 .GLOBL   ,,ATUN ;ATTACH UNIT
 .GLOBL   ,,DTUN ;DETACH UNIT
 .GLOBL   ,,IODN ;I/O DONE
 .GLOBL   ,,RNTP ;RTN NODE TO POOL
 .GLOBL   ,,FLSH ;FLUSH QUEUED I/O REQUESTS
 .GLOBL   ,,RNDN ;RUN-DOWN DONE
;
 .GLOBL   ,,INTX ;INTERRUPT SERVICE EXIT
 .GLOBL   ,,RDID ;ID OF TASK BEING RUNDOWN
;
; I/O REQUEST NODE ENTRY DEFINITIONS
;
R.TD=04  ;SYSTEM TASK DIRECTORY ENTRY ADR (REQUESTOR ID)
R.RP=06  ;REQUEST PRIORITY
R.LU=10  ;LOGICAL UNIT NUMBER
R.EF=11  ;EVENT FLAG NUMBER
R.FC=12  ;I/O FUNCTION CODE
R.SA=14  ;ADDRESS (VIRTUAL) OF REQUESTOR I/O STATUS WORD
R.PA=16  ;TEN-BYTE REQEUST PARAMETER BLOCK
;
; DIRECTIVE PARAMETER BLOCKS (DPB'S) USED FOR INITIALIZATION
;
DPB1:    101,    ;SET-SYSTEM-TRAP DIC
         12      ;PWR RECOVERY TRAP ID
         PWRUP   ;SERVICE ROUTINE ENTRY
;
DPB2:    101,    ;SET-SYSTEM-TRAP DIC
         11      ;I/O RUNDOWN TRAP ID
         IORUN   ;SERVICE ROUTINE ENTRY
;
; VAIRABLES USED FOR INITIALIZATION
;
         0       ;DEVICE
         0       ;   DEPENDENT
         0       ;      VARIABLES
;
; START -- HANDLER TASK ENTRY,  HANDLER IS INITIALIZED,
; AND THE INITIALIZATION CODE IS THEN USED FOR STACK
; STORAGE.
;
; POWER-RECOVERY & I/O-RUNDOWN SYSTEM TRAPS ARE CONNECTED TO
; THE SYSTEM VIA "SET SYSTEM TRAP" DIRECTIVES.
;
START:   MOV     #DPB1,-(SP)
```

```
        EMT     377
        MOV     #DPB2,-(SP)
        EMT     377
;
; AN INTERRUPT SERVICE ROUTINE IS CONNECTED TO TRAP LOCATION
; 240, AND THE BASE OF THE SERVICE ROUTINE'S ADDRESS SPACE
; IS SPECIFIED AS THE HANDLER TASK'S VIRTUAL ZERO.
;
; IF THE SERVICE ROUTINE CANNOT BE CONNECTED (ANOTHER
; INTERRUPT SERVICE ROUTINE IS CONNECTED), THE HANDLER
; TASK EXITS.
;
        MOV     #240,R0     ;TRAP ADR TO R0
        MOV     #INTENT,R1  ;ENTRY POINT TO R1
        CLR     R2          ;BASE OF ADR SPACE TO R2
        JSR     PC,',,CINT  ;CONNECT
        BVC     EXIT        ;EXIT IF NOT CONNECTABLE
;
; THE HANDLER TASK IS DECLARED RESIDENT, AND ITS UNIT
; IDENTIFICATION TABLE (UIT) IS INITIALIZED.
;
; IF NO UNITS FOR THE DEVICE EXIST, THE HANDLER TASK
; DISCONNECTS & EXITS.
;
        MOV     #UIT,R0     ;UNIT ID TBL ADR TO R0
        MOV     #1,R1       ;NUMBER OF UNITS TO R1
        MOV     #"LP,R2     ;DEVICE NAME TO R2
        JSR     PC,',,DSUT  ;DECLARE & SET
        BVC     DAEXIT
;
; INITIALIZATION PECULIAR TO THE PERIPHERIAL DEVICE
; BEING SUPPORTED IS NOT SHOWN.
;
        NOP     ;DEVICE
        NOP     ;   DEPENDENT
        NOP     ;       INITIALIZATION
;
; INITIALIZATION COMPLETED -- INSTRUCTIONS AND DATA
; PRECEDING THIS LOCATION ARE NO LONGER NEEDED.  THE
; HANDLER TASK'S STACK IS EXTENDED TO UTILITIZE
; THIS STORAGE.
;
        MOV     PC,SP
;
; HANDLER TASK IS NOW IDLE AND READY TO DE-QUEUE REQUESTS
;
; THE HANDLER TASK'S EXECUTION IS SUSPENDED UNTIL (OR
; UNLESS) ITS EVENT FLAG ONE IS SET.
;
; THE TASK WILL REMAIN SUSPENDED UNTIL AN I/O REQUEST IS
; QUEUED FOR ONE OF THE UNITS SERVICED BY THE HANDLER TASK
; (A UNIT IDENTIFIED IN THE UIT).
;
```

```
;
; IF A SYSTEM TRAP OCCURS WHILE THE TASK IS WAITING FOR AN
; I/O REQUEST TO BE QUEUED, THE TRAP SERVICE ROUTINE WILL
; BE EXECUTED, BUT THE TASK WILL REMAIN SUSPENDED.
;
IDLE:
;
        MOV     #WF1,-(SP) ;WAITFOR E,F, #1
        EMT     377
;
; TASK EXECUTION HAS RESUMED BECAUSE ITS EVENT FLAG ONE HAS
; BEEN SET.
;
; THE EVENT FLAG IS CLEARED, AND AN ATTEMPT TO DE-QUEUE
; A REQUEST FROM UNIT-0 (THERE IS ONLY ONE UNIT) IS MADE.
;
; IF A REQUEST IS DE-QUEUED, IT IS PROCESSED.  IF A
; REQUEST IS NOT DE-QUEUED, THE HANDLER BECOMES IDLE.
;
DEQ:
;
        MOV     UIT+0,R0    ;PUD ENTRY ADR TO R0
        MOV     #1,R1       ;FLAG IND TO R1
        JSR     PC,',,CLEF  ;CLEAR EF #1
;
        MOV     UIT+0,R0    ;UNIT-0 PUD ENTRY ADR TO R0
        MOV     #RNA,R1     ;RNA BUF ADR TO R1
        JSR     PC,',,DQRQ  ;DE-QUEUE ATTAMPT
        BVC     IDLE        ;IDLE IF NO DE-QUEUE
;
; CONTROL IS DISPATCHED TO APPROPRIATE SERVICE CODE PER
; I/O FUNCTION CODE, WITH THE REQUEST NODE ADDRESS IN R1.
;
; IF THE REQUESTED FUNCTION CODE IS NOT RECOGNIZED, AN
; I/O STATUS VALUE OF -99 IS RETURNED.
;
        MOV     R,FC(R1),R2 ;SET I/O FUNCTION CODE IN R2
        CMP     R2,#2400    ;WRITE REQUEST?
        BEQ     WRTREQ      ;YES -- WRITE LINE
        CMP     R2,#1400    ;NO -- ATTACH REQUEST?
        BEQ     ATTREQ      ;YES -- ATTACH IF DETACHED
        CMP     R2,#2000    ;NO -- DETACH REQUEST?
        BEQ     DETREQ      ;YES -- DETACH IF ATTACHED
        CMP     R2,#-177771;NO -- HANDLER EXIT REQUEST?
        BEQ     EXTREQ      ;YES -- FINISH SERVICE & EXIT
        MOV     #-99,,R0    ;NO -- UN-RECOGNIZED FUNCTION
        JMP     UNSUC
;
; WRITE REQUEST -- THE STARTING ADDRESS AND LENGTH
; OF THE LINE TO BE WRITTEN ARE VALIDATED.  IF OKAY,
; THE LINE IS PRINTED.  IF NOT, AN I/O STATUS VALUE
; OF -10 IS RETURNED.
;
WRTREQ: NOP     ;VALIDATION YET TO
```

```
          NOP     ;BE DETERMINED
;
; EVENT FLAG TWO IS CLEARED, THE PRINT OPERATION IS
; STARTED, AND EXECUTION IS SUSPENDED UNTIL EVENT
; FLAG TWO IS SET (BY INTERRUPT SERVICE ROUTINE).
;
PF1:      ;REFERENCE POINT FOR PWR FAIL RECOVERY
;
          MOV     UIT+0,R0    ;CLEAR INTERRUPT FLAG.
          MOV     #2,R1       ;FLAG IND TO R1
          JSR     PC,',,CLEF  ;CLEAR EF #2
;
          NOP     ;DEVICE
          NOP     ;   DEPENDENT
          NOP     ;       CODE TO
          NOP     ;           START I/O
;
          MOV     #WF2,-(SP) ;WAITFOR E,F, #2
          EMT     377
;
PF2:      ;REFERENCE POINT FOR PWR FAIL RECOVERY
;
; REQUEST IS FINISHED BY SETTING THE REQUESTOR'S
; I/O STATUS WORD (IF SPECIFIED), AND EVENT FLAG
; (IF INDICATED).
;
          MOV     DEVSTS,R3   ;SUCCESSFUL COMPLETION?
          BPL     SUC         ;YES -- RETURN STATUS = +1
          BR      UNSUC       ;NO -- RETURN DEVICE STATUS
;
; ATTREQ -- UNIT IS ATTACHED TO REQUESTING TASK (UNLESS
; IT IS ALREADY ATTACHED TO IT).
;
ATTREQ:   MOV     UIT+0,R3    ;UNIT-0 PUD ENTRY ADR TO R3
          JSR     PC,',,ATUN  ;ATTACH UNIT
          BVS     SUC         ;IF SUCCESSFUL, RETUN STS=+1,
          MOV     #-30.,,R3   ;OTHERWISE, RETURN STS=-30.
          BR      UNSUC
;
; DETREQ -- UNIT IS DETACHED FROM REQUESTING TASK (UNLESS
; IT IS NOT ATTACHED).
;
DETREQ:   MOV     UIT+0,R3    ;UNIT-0 PUD ENTRY ADR TO R3
          JSR     PC,',,DTUN  ;DETACH UNIT
          BVS     SUC         ;IF SUCCESSFUL, RETURN STS=+1,
          MOV     #-30.,,R3   ;OTHERWISE, RETURN STS=-30.
          BR      UNSUC
;
; EXTREQ -- HANDLER TASK EXIT REQUEST -- CLEANUP & EXIT
;
EXTREQ:   NOP     ;DEVICE
          NOP     ;   DEPENDENT
          NOP     ;       CODE TO
```

```
        NOP     ;       TERMINATE USAGE
;
; DECREMENT REQUESTS PENDING COUNT AND RETURN REQUEST NODE
;
        MOV     RNA,R1      ;REQUEST NODE ADR TO R1
        CLR     R2          ;NO DECREMENT ADJ
        MOV     #1,R3       ;STATUS TO R3
        JSR     PC,',,IODN  ;I/O DONE
;
        MOV     RNA,R1      ;REQUEST NODE ADR TO R1
        JSR     PC,',,RNTP  ;RTN NODE TO POOL
;
DAEXIT:                     ;DISCONNECT & EXIT
;
        MOV     #240,R0     ;TRAP ADR TO R0
        JSR     PC,',,DINT  ;DISCONNECT INTERRUPT
;
EXIT:                       ;EXIT HANDLER TASK
;
        MOV     51,,-(SP)   ;EXIT DIRECTIVE
        EMT     377
;
; SUC -- FINISH SUCCESSFUL REQUEST, I/O STATUS IS SET TO +1
;
SUC:    MOV     #+1,R3      ;SET I/O STATUS IN R3
;
; UNSUC -- FINISH UNSUCCESSFUL REQUEST, R3 CONTAINS STATUS.
;
UNSUC:
;
        MOV     RNA,R1      ;REQUEST NODE ADR TO R1
        CLR     R2          ;NO CNTS P/O ADJ
        JSR     PC,',,IODN  ;I/O DN
;
        MOV     RNA,R1      ;REQUEST NODE ADR TO R1
        JSR     PC,',,RNTP  ;RTN NODE TO POOL
;
; 'RNA' IS CLEARED TO INDICATE "NO REQUEST DE-QUEUED"  TO
; THE I/O RUNDOWN TRAP SERVICE ROUTINE.
;
; IF AN I/O RUNDOWN SYSTEM TRAP OCCURRED WHILE THE
; PRESENT REQUEST WAS DE-QUEUED, INDICATE I/O RUN-
; DOWN COMPLETE BY RESUMING THE RUNDOWN TASK.
;
        CLR     RNA
;
        TST     RDFLAG
        BEQ     DEQ
        JSR     PC,',,RNDN
        BR      DEQ
;
; POWER RECOVERY SYSTEM TRAP -- IF A WRITE WAS STARTED
; AND NOT FINISHED, IT MUST BE REDONE.
```

```
;
PWRUP:  CMP     @SP,#PF1        ;'PF1' ;LE, PC ;LE, 'PF2' ?
        BLT     PWREX           ;NO -- EXIT SYS TRAP
        CMP     @SP,#PF2                    ;
        BGT     PWREX           ;NO -- EXIT SYS TRAP
        MOV     #PF1,@SP        ;YES -- CAUSE I/O TO BE REDONE
;
        MOV     R0,-(SP)
        MOV     R1,-(SP)
        MOV     UIT+0,R0
        MOV     #2,R1
        JSR     PC,',,STEF      ;SET E,F; #2
        MOV     (SP)+,R1
        MOV     (SP)+,R0
;
PWREX:
;
        MOV     #XST,-(SP) ;EXIT SYSTEM TRAP ROUTINE
        EMT     377
;
; I/O RUNDOWN SYSTEM TRAP
;
IORUN:  MOV     R0,-(SP)
        MOV     #UIT+0,R0
        JSR     PC,',,FLSH  ;FLUSH QUEUED REQUESTS
;
        TST     RNA
        BEQ     IORR
        MOV     RNA,R0
        CMP     R',TD(R0),,,RDID
        BNE     IORR
        INC     RDFLAG
        BR      IORX
;
IORR:   JSR     PC,',,RNDN
;
IORX:
;
        MOV     (SP)+,R0
        MOV     #XST,-(SP)
        EMT     377
;
; TASK DIRECTIVE PARAMETER BLOCKS
;
WF1:    ,BYTE   41.,,0
        000001
;
WF2:    ,BYTE   41.,,0
        000002
;
XST:    103.
;
; TASK'S VARIABLES
```

```
;
RNA:    0       ;REQUEST NODE ADDRESS
RDFLAG: 0       ;I/O RUNDOWN FLAG
;
; HANDLER TASK'S UNIT IENTIFICATION TABLE (ONLY ONE UNIT)
;
UIT:    0       ;PUD ENTRY ADR FOR UNIT-0
;
; INTERRUPT SERVICE ROUTINE -- POSITION INDEPENDENT ROUTINE
; (RUNS UNDER KERNEL ASR3, WHICH IS SET AT, OR AS CLOSELY
; BELOW AS POSSIBLE, THE BASE OF THE INTERRUPT SERVICE
; ROUTINE'S ADDRESS SPACE AS SPECIFIED WHEN CONNECTED).
;
INTENT: NOP     ;DEVICE
        NOP     ;   DEPENDENT
        NOP     ;       INTERRUPT
        NOP     ;           SERVICE CODE
;
; SET INTERRUPT FLAG
;
        MOV     R0,-(SP)
        MOV     UIT+0,R0
        MOV     #2,R1
        JSR     PC,',,STEF  ;SET E,F', #2
        MOV     (SP)+,R0
;
; EXIT INTERRUPT SERVICE ROUTINE
;
        JMP     @',,INTX
;
DEVSTS: 0       ;HARDWARE DEVICE STATUS
;
        .END    START
```

HANDLER TASK INITIALIZATION

Handler Task initialization consists of connecting to System Traps and hardware interrupt(s), determining which device-units exist (often only one) and how each is identified, and declaring the Handler Task resident and able to de-queue I/O requests.

I/O Handler Tasks normally use two System Traps, power recovery, and I/O rundown. These System Traps provide a means of interrupting a Handler Task's normal operation whenever power is restored (after a power failure), and

whenever a Task EXITs or Is aborted with I/O requests pending. The System Trap service routines are connected to the system (as are all System Traps) using the SET SYSTEM TRAP Directive.

Most Handler Tasks require only one Interrupt service routine, however, as many as are desired may exist within a Handler Task. Connecting an interrupt service routine to a hardware interrupt is instructing the system (1) to transfer control to an indicated service routine whenever an interrupt occurs via an indicated interrupt trap address, and (2) where the base of the interrupt service routine's address space is to be set.

A System Subroutine to connect to an interrupt is called as follows:

```
R0 -- Interrupt trap address,
R1 -- Entry point of service routine,
R2 -- Base of interrupt service address space,
R3 -- Bits 0-3 prescribe the states of Condition Codes
C, V, Z, & N at entry to interrupt service routine,

JSR  PC,,;CINT
```

When the connect is successful, CC-V (Condition Code "V") is SET upon subroutine return; if unsuccessful, CC-V is CLEAR upon return.

In most cases, when an interrupt service routine cannot be connected, the Handler Task cannot run, and it simply EXITs. However, the Handler Task can be coded to do whatever is appropriate.

In the sample program, the interrupt service routine references the "Unit Identification Table" (UIT), but no other part of the Task outside of the interrupt service routine. Hence, the base of the interrupt service routine could have been set as high as 'UIT'. It is set at the Task's virtual zero to show that: "Unless a Handler Task is larger than 4K words, interrupt service routine references do not restrict Handler Task layout".

When control is transferred to an interrupt service routine, ASR3 is set as close to the specified "base of interrupt service address space" without excluding it, i.e., virtual location 060000 exists at a 32-word bound at or below the location indicated in R2.

All Handler Tasks provide space for a system-set "Unit Identification Table". This table consists of one-word entries for each device-unit that the Handler Task can service. Words representing non-existent units (no PUD

entry) are set to zero. Words representing existing units are set to the coresponding PUD entry addresses. These addresses are normally used only as unit identifiers when making requests of System Subroutines; however, they also provide an access to a unit's PUD entry, which is useful in some special cases.

A System Subroutine to initialize this table and declare the Handler Task resident and ready to de-queue requests is called as follows:

```
R0 <- Device name (two ASCII characters).
R1 <- Maximum number of units (table size).

JSR  PC,,,DSUT
```

If at least one PUD entry for the specified device is found, CC-V is SET upon subroutine return; if the device name is not found in the PUD, CC-V is CLEAR upon return. The Handler Task is flagged resident in the PUD entry for each unit identified to the Handler Task.

IDLE STATE FOR HANDLER TASKS

When a Handler Task is idle, it suspends its execution until an I/O request is queued for a unit supported by it. This is done by issuing a WAITFOR Directive. The Handler Task's Event Flag one is set whenever a request is queued for one of its units. Normally the WAITFOR DIR indicates Event Flag Range 1-16, and flag number one, however, in some cases it may be desirable (and is possible) to wait for more than one Event Flag's setting.

If a Power Recovery or a I/O Rundown System Trap occurs while a handler Task is idle, the Task will remain suspended unless the System Trap service routine caused its resumption.

I/O REQUEST PROCESSING

When an Idle Handler Task's execution is resumed (as a result of its Event Flag one being set) it normally clears that Flag before attempting to de-queue a request. [5] however, in some special cases it is desirable to clear the Queue Flag (Event Flag one) at other times.

The CLEAR EVENT FLAG Directive could be used, however, since the Privileged Task has access to the system's lists, a System Subroutine imposes less overhead (and can clear more

than one flag). A Subroutine to clear any of a Handler Task's Event Flags 1-16 is called as follows:

```
RØ <- PUD entry address (from UIT),
R1 <- Flags indicator,

JSR  PC,,CLEF
```

R1 bits Ø-15 represent Event Flags 1-16 respectively.

I/O requests for an indicated unit are de-queued by using a subroutine to attempt to de-queue a request node. This subroutine is called as follows:

```
RØ <- PUD Entry address (from UIT),
R1 <- Address of buffer for "RNA",

JSR  PC,,DQRQ
```

If this subroutine is called and the request list for indicated unit is empty, a request node is not de-queued (of course). Also, even when a request list is not empty, it is possible to NOT be able to de-queue a request because (1) the unit is ATTACHed and no requests for the attaching Task are in the list, and/or (2) requests in the list have been made by a Task that is checkpointed.

When a request is de-queued, the Request Node Address (RNA) is stored in the buffer indicated in R1, a "Requests In Process Count" is incremented,[6] and CC-V is SET upon subroutine return. When a request is NOT de-queued, CC-V is CLEAR upon subroutine return.

When a request node is de-queued, its address is set in the calling program's buffer (per R1) with interrupts inhibited. This is done so that a non-zero RNA buffer can be used as a "request de-queued" flag.

---

[5] To avoid the race condition that exists when a second request is queued just after a failure to de-queue has caused the Handler Task to become idle again.

---

[6] If "No Return" function, Requests In Progress Count is NOT incremented.

The Requests In Process Count is a byte in the ATL node of every active Task. This count is incremented whenever a request for the Task is de-queued, and decremented whenever a request for the Task is completed,[7] It provides an indication of requests being processed, and is used to delay the recording (swapping-out) of a checkpointed Task until I/O in process has been completed.

Normally after de-queuing a request, the I/O Function Code is examined, and control is transferred to a routine to perform the indicated function. If the function is not recognized by the Handler Task, a status of -99 (by convention) is returned.

If an ATTACH request is de-queued, and the unit is not already attached (to the requesting Task), the unit is flagged for the exclusive use of the attaching Task. This causes the De-queue Request System Subroutine (,,DQRQ) to only de-queue requests for that Task.

A System Subroutine to attach a unit is called as follows:

```
R0 <- PUD entry address (from UIT),
R1 <- Request node address.

JSR  PC,,,ATUN
```

When a unit is attached, CC-V is SET upon subroutine return. When a unit is NOT attached, CC-V is CLEAR upon return.

If a DETACH request is de-queued, and the unit is attached to the requesting Task, the ATTACH is nullified. This causes the De-queue Request System Subroutine (,,DQRQ) to de-queue from the top (high priority) of the unit's request list.

A system Subroutine to detach from a unit is called as follows:

```
R0 <- PUD entry address (from UIT),
R1 <- Request node address.

JSR  PC,,,DTUN
```

When a unit is detached, CC-V is SET upon subroutine return. When a unit is NOT detached, CC-V is CLEAR upon return.

---

[7] Unless "No Return" function.

Many I/O requests require a transfer either to or from a requestor's memory. The range of these transfers must be validated in order to maintain system integrity. The following three subroutines aid in this operation.

A System Subroutine to validate a transfer and setup an 18-bit starting address (for a peripheral controller) is called as follows:

```
R2 <-- First word address (user virtual),
R3 <-- Transfer length (in bytes),

JSR  PC,,,VXFR
```

If the transfer is invalid, CC-V is CLEAR upon subroutine return. If the transfer is valid, CC-V is SET upon return, and the 18-bit starting address is in R4 & R5. The low order 16 bits are in R5, and the high order two bits are in bits 5 & 4 of R4 with all other R4 bits cleared.

System Subroutines to validate a transfer, and if valid perform the transfer, are called as follows:

```
R2 <-- First word address (user virtual),
R3 <-- Transfer length (in words),
R4 <-- Memory buffer address,

JSR  PC,,,BLXI      For transfer IN,
       or
JSR  PC,,,BLXO      For transfer OUT,
```

If the transfer is performed, CC-V is SET upon subroutine return. If the transfer is NOT performed, CC-V is CLEAR upon return.

When an I/O operation is completed, an Event Flag and an I/O Status Word may be set (if indicated in request node), the requesting Task's I/O Pending Count is usually decremented, and its Requests In Progress Count is decremented.[8]

A System Subroutine to finish an I/O request is called as follows:

```
R1 <-- Request node address,
R2 <-- Adjustment to unity decrement,
R3 <-- I/O Status Block Wd, 00,
R4 <-- I/O Status Block Wd, 01,
```

---

[8] If "No Return" function, NEITHER count is altered.

```
JSR  PC,,,IODN
```

If an I/O Status Block address was specified in the request node, the status block (in the requestor's memory) is set to the contents of R3 & R4;

If an Event Flag number was specified in the request node, that event flag is set and a Significant Event is declared;

The I/O Requests Pending Count for the requesting Task is decremented and then modified by adding the contents of R2 to it, R2 is normally zero, however in some cases the decrementing may be adjusted (viz., R2=+1 for FILE OPEN and R2=-1 for FILE CLOSE). The ATTACH & DETACH subroutines modify a Task's I/O pending count so that a request is considered pending while a unit is attached to that Task;

When an I/O request node is no longer needed by a Handler Task, it is returned to the pool of available list elements. A System Subroutine to return a node to the pool is called as follows:

```
R1 <- Request node address;

JSR  PC,,,RNTP
```

When an I/O operation is started whose termination is signaled by a hardware interrupt, the Handler Task may use the WAITFOR INTERRUPT Directive to suspend its execution until the Interrupt (or appropriate series of interrupts) has occurred. Normally the Handler Task's Event Flag two (2) is used as an "interrupt flag". This flag is normally cleared before starting an operation and set by an interrupt service routine to signal completion;

A Subroutine to clear any of a Handler Task's Event Flags 1-16 is called as follows:

```
R0 <- PUD entry address (from UIT);
R1 <- Flags indicator;

JSR  PC,,,CLEF
```

R1 bits 0-15 represent Event Flags 1-16 respectively;

A Subroutine to set any of a Handler Task's Event Flags 1-16 is called as follows:

```
R0 <- PUD entry address (from UIT);
R1 <- Flags indicator;

JSR  PC,,,STEF
```

R1 bits 0-15 represent Event Flags 1-16 respectively.

HANDLER TASK EXIT

The UNLOAD MCR Function is used to cause a a Handler Task to exit. This is done by queuing a low priority request to the unit represented by the device's first PUD entry, and inhibiting further queuing by declaring the Handler Task non-resident (in each PUD entry). The I/O function code for an exit request is 177771 (octal).

In most single-unit devices, servicing an exit request consists of finishing the request (..IODN & ..RNTP), disconnecting from hardware interrupts, and EXITing. More complex Handler Tasks require additional code to process all queued requests before EXITing.

While a Handler Task is in the process of EXITing, it cannot be reloaded (the LOAD MCR Function will find it active).

I/O RUNDOWN

When a Task EXITs or is aborted with I/O requests pending, the Task is considered active (ATL node and memory still exist) but not runable, and a request to "rundown" its I/O is made (via SEND & REQUEST Directive) to a Task called "..IO.." .

The I/O Rundown Tasks requests Handler Tasks to flush queued requests and either finish or abort any requests in process for an indicated Task, by (1) clearing its Event Flag one, (2) placing the Task's STL node address in the SCOM word ..RDID, (3) causing an I/O Rundown System Trap for a particular Handler Task, and (4) suspending its execution until its Event Flag one is set.

When the Handler Task completes its I/O rundown service, it resumes the execution of the I/O rundown Task by setting its Event Flag one.

A System Subroutine to set Event Flag one of a Task called "..IO.." is called as follows:

```
JSR PC,..RNDN
```

The I/O Rundown Task continues this process until either the Requests Pending Count for the Task being rundown is decremented to zero, or all devices have been scanned (in which case something is wrong, likely a bad handler).

A System Subroutine to flush all requests from a device request list for the Task on which I/O is being rundown (per SCOM) is called as follows:

```
R0 <- PUD entry address (from UIT);

JSR  PC,,;FLSH
```

POWER-FAILURE RECOVERY

When the system recovers from a power failure, a Power Recovery System Trap is generated for all Tasks that are setup to service the trap.

I/O Handler Tasks are coded to do what ever is necessary to recover. In some cases, this is simply repeating a request If a request was being processed, viz., Magtape read. In other cases, recovery is determined by the degree of completion at power failure, viz., If power fails during a Magtape write, the Handler Task must determine whether anything was written and conditionally backspace before re-writing.