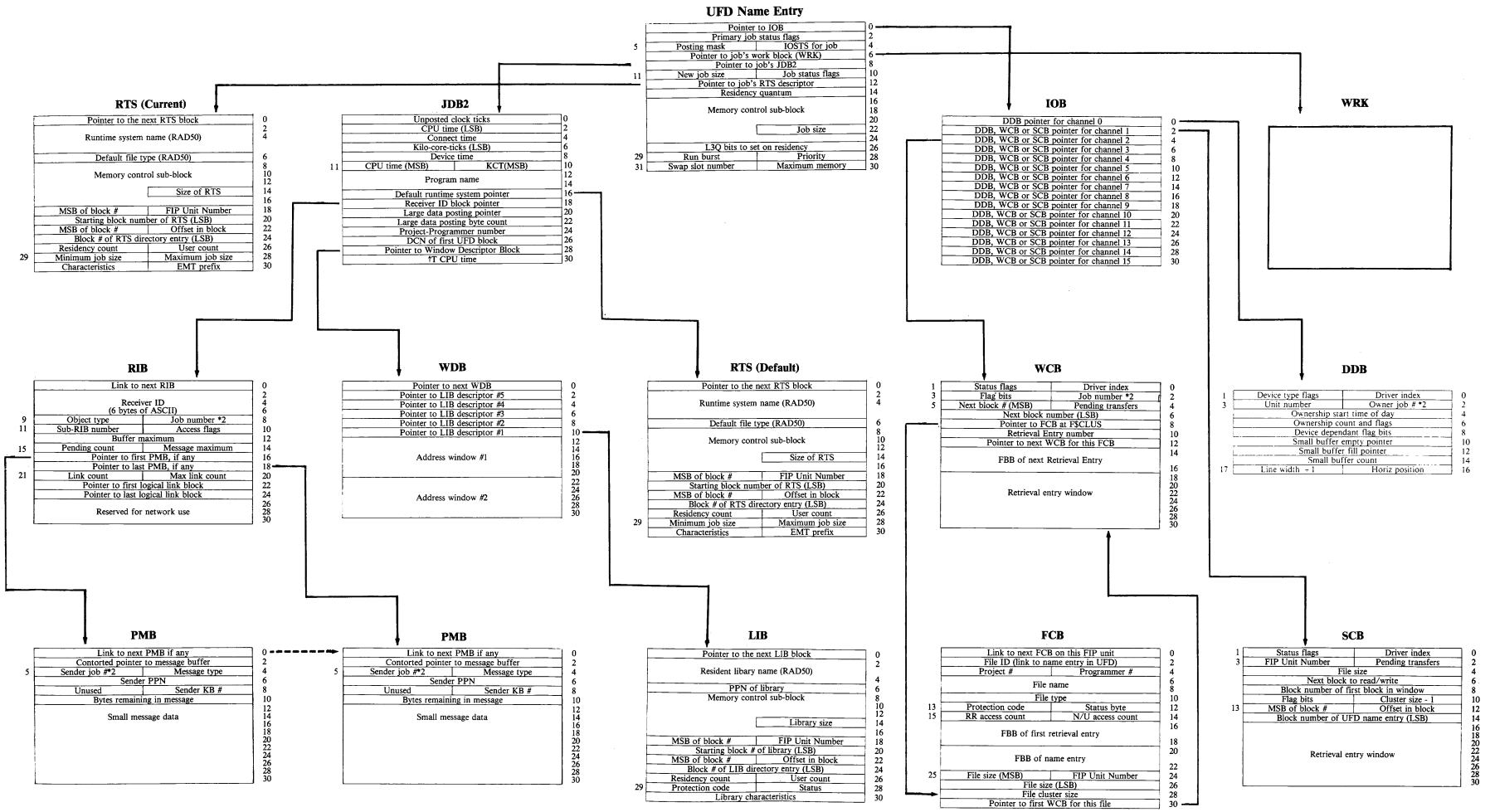
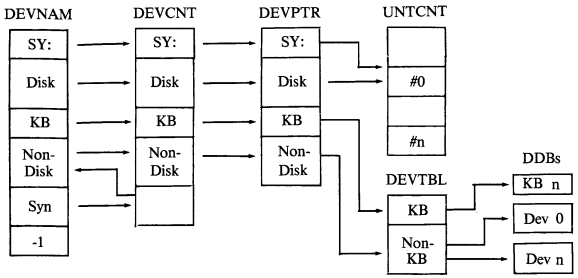
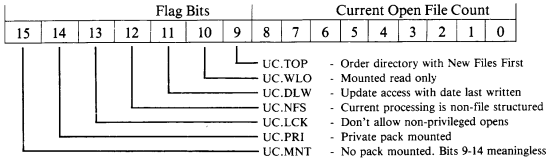
JOB CONTROL TABLES



JOB CONTROL TABLES



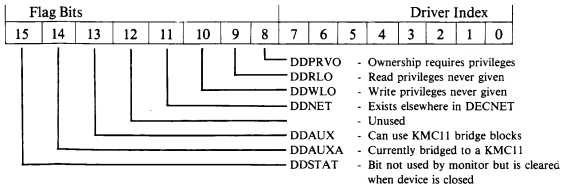
Entries in UNTCNT - Disk Unit Status Table



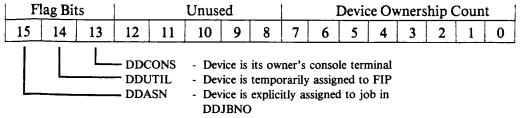
DDB

Symbol	Offset			Offset	Symbol
DDSTS	1	Device type flags	Driver index	0	DDIDX
DDUNT	3	Unit number	Owner job # *2	2	DDJBNO
		Ownership start time of day		4	DDTIME
		Ownership count and flags		6	DDCNT
		Device dependant flag bits		8	DDFLAG
		Small buffer empty pointer		10	DDBUFC
		Small buffer fill pointer		12	
		Small buffer count		14	
DDHORC	17	Line width +1	Horiz position	16	DDHORZ
		Driver specific data			

DDSTS - Device Characteristics Flags



DDCNT - Device Flags



FIXED MEMORY LOCATIONS

Octal	*Decimal*	*Symbol*	*Description*
44	36	IDATE	System startup date, as (year-1970)*1000 + (day of year).
46	38	ITIME	System startup time in minutes before midnight.
50	40		Start at this location to do a system reload.
52	42		Start at this location to do a crash dump, auto-restart.
54	44	HALT	HALT instruction for system crash.
56	46		Start at this location to do a system reload.
1000	512	DATE	Today's date, as (year-1970)*1000 + (day of year).
1002	514	TIME	Current time in minutes before midnight.
1004	516	TIMSEC	Seconds to next minute.
1005	517	TIMCLK	Ticks to next second.
1006	518	JOB	Job number of current job (times 2).
1007	519	NEXT	Job number (times 2) of the next job to run.
1010	520	JOBDA	Pointer to current job's Job Data Block (JDB).
1012	522	JOBF	Pointer to current job's flags (JDFLG).
1014	524	IOSTS	Pointer to current job's IOSTS (JDIOST).
1016	526	JOBWRK	Pointer to current job's Work Block (WRK).
1020	528	JOBJD2	Pointer to current job's Secondary Job Data Block (JDB2).
1022	530	JOBRTS	Pointer to current job's Runtime System Descriptor Block (RTS).
1024	532	CPUTIM	Pointer to current job's CPU time bucket (J2TICK).
1026	534	JOBWDB	Pointer to current job's Window Descriptor Block (WDB) at offset W.WIN1.
1040	544	MEMLST	Root of memory control list.
1100	576	DFTRTS	RTS Block for default runtime system.
1140	608		Tail of memory control list.
1200	640	ERLRIB	RIB for ERRCPY receiver.
1240	672	NULRTS	RTS Block for null runtime system.
1300	704	FIJOB	Job number (times 2) of job currently using FIP.
1301	705	FIPRIV	If non-zero, the current user of FIP is non-privileged.
1302	706	FIUSER	PPN of current job in FIP.
1304	708	FIJBDA	Pointer to the Job Data Block (JDB) for the current job in FIP.
1306	710	FIJBD2	Pointer to the Secondary Job Data Block (JDB2) for the current job in FIP.
1310	712	FIPSJN	Job number (times 2) of the system job currently using FIP.
2070	1080	SYSTAK	Top of the monitor's stack when the null job is not running. The monitor's stack is 256 words deep.
2600	1408	FISTAK	Top of FIP's stack. FIP's stack is 124 words deep.

FIRQB

Symbol	*Oct*	*Dec*		*Dec*	*Oct*	*Symbol*
			0 | Returned status	0	0	FIRQB
FQFUN	3	3	Function code | Job number *2	2	2	FQJOB
FQSIZM	5	5	File size (MSB) | Channel number *2	4	4	FQFIL
			Project number | Programmer number	6	6	FQPPN
			File name (in RAD 50)	8	10	FQNAM1
				10	12	
			File type (in RAD 50)	12	14	FQEXT
			File size (LSB)	14	16	FQSIZ
			Buffer length (in bytes)	16	20	FQBUFL
			Open mode	18	22	FQMODE
			Status flags	20	24	FQFLAG
FQPROT	27	23	Protection code | Code "real" flag	22	26	
			Device name (in ASCII)	24	30	FQDEV
			Unit "real" flag | Device unit #	26	32	FQDEVN
			Cluster size	28	34	FQCLUS
			Number of entries in directory lookup	30	36	FQNENT

XRB

Symbol	*Oct*	*Dec*		*Dec*	*Oct*	*Symbol*
			Buffer length (in bytes)	0	0	XRLEN
			Transfer byte count	2	2	XRBC
			Buffer address	4	4	XRLOC
XRBLKM	7	7	Block # (MSB) | Channel # *2	6	6	XRCI
			Block number (LSB)	8	10	XRBLC
			Terminal input wait time	10	12	XRTIME
			Record modifier	12	14	XRMOD

COMMON MEMORY LOCATIONS

ENTRY POINTS (xxDVR PSECT)

ASN$xx - Assign
```
Input:   R0    Job number of assigner times 2.
         R1    Pointer to DDB for this unit.
Output:  All registers must be preserved.
Exit:    RETURN                    ;No error
or:      ERROR     code            ;If error in assignment
```

DEA$xx - Deassign
```
Input:   R1    Pointer to DDB for this unit.
Output:  All registers must be preserved.
Exit:    RETURN
```

OPN$xx - Open
```
Input:   R0    Unit number times two.
         R1    Pointer to DDB for this unit
         R4    Pointer to FIRQB (in WRK block). Default values have been loaded for
               FQFLAG and FQBUFL.
         R5    Pointer to job's IOB entry for this channel.
Output:  R0    Random
         R1    Must be preserved
         R4    Random
         R5    Must be preserved
Exit:    RETURN                    ;For successful open
or:      CALLX     RETDEV          ;For unsuccessful open
         ERROR     code
```

CLS$xx - Close
```
Input:   R0    Unit number times 2
         R1    Pointer to DDB
         R5    Pointer to DDB
         Z     Set: This is a real close
               Clear: This is a "reset" close
Output:  R0    Random
         R1    Random
         R5    Must be preserved
Exit:    RETURN
```

SER$xx - I/O Service
```
Input:   R0    Unit number times 2
         R1    Pointer to DDB for this unit
         R2    Function code: 2 (.READ) or 4 (.WRITE)
         R3    Pointer to XRB (contained in WRK block)
         R4    Calling job number times 2
         R5    Pointer to user's buffer (mapped through APR6)
         Z     Set: This is the first entry for this request
               Clear: This entry is for an IOREDO
         C     Set: This entry is for an IOREDO
               Clear: This is the first entry for this request
Output:  All registers random
         XRBC      Adjusted for the number of bytes transferred to or from the user's
                   buffer XRLOC Adjusted for the bytes transferred to or from the
                   user's buffer.
         XRBLK     0 for non-block structured devices. Next virtual block number for
                   block structured devices.
Exit:    JMPX      IOEXIT          ;I/O completed without error
or:      SETERR    code,@IOSTS     ;I/O completed with error
         JMPX      IOEXIT
or:      JMPX      IOREDO          ;Stall the job and then reenter
                                   ;SER$xx when the job is unstalled
```

SPC$xx - Special Service
```
Input:   R0    Unit number times 2
         R1    Pointer to DDB for this unit
         R2    Special function code
         R3    Pointer to XRB (in WRK block)
         R4    Calling job number times 2
         R5    Pointer to XRB (mapped through APR6)
Output:  All registers random
Exit:    JMPX      RTI3            ;If no error
or:      ERROR     code            ;If error
```

INT$xx - Interrupt Service
```
Input:   R0    Unit number times 2
         PR    Priority is device interrupt priority
Output:  All register random
Exit:    RETURN                    ;Normal return from interrupt
or:      CALLX   IOFINI,R5,JS.xx   ;Reschedule stalled job
         RETURN                    ;Return from interrupt
```

DEVICE DRIVERS

ENTRY POINTS (cont.)

TMO$xx - Timeout
```
Input:   R0    Unit number times 2
         R1    Pointer to DDB for this unit
         R3    Pointer to device CSR (loaded from CSRTBL)
         PR    Priority is PR3
Output:  All registers are random
Exit:    RETURN
```

ERL$xx - Error Logging
```
Input:   R1    Pointer to DDB for this unit
         R3    Pointer to CSR for this unit
Output:  All registers are random
Exit:    RETURN
```

SLP$xx - Sleep Check
```
Input:   R0    Unit number times 2
         R1    Pointer to DDB for this unit
         R4    Pointer to job's IOB entry for this channel
Output:  R0    Random
         R1    Random
         R4    Random
         C     Set: Don't let the job sleep
               Clear: Let the job sleep
Exit:    RETURN
```

UMR$xx - Unibus Mapping Register Available
```
Input:   R0    Pointer to base root of DSQ list for disk device drivers. Random for non-disk
               drivers.
         R3    Pointer to CSR for disk drivers. Random for non-disk drivers.
         PR    Priority is PR5
Output:  None
Exit:    RETURN
```

nnn$xx - Level Three Queue Reentry
```
Input:   All registers are random
         PR    Priority is PR3
         C     Clear
Output:  All registers are random
         PR    Must be preserved
Exit:    JMPX    RTI3
```

SYMBOLIC VALUES

STS.xx - DDB Status Byte
```
Purpose:  Define device status
Value:    <DDPRVO!DDRLO!DDWLO!DDNET!DDAUX!DDAUXA!DDSTAT>/400
File:     xxDVR.MAC
Scope:    Global
```

FLG.xx - Device Dependent Flags
```
Purpose:  Define device dependent flags for FLGTBL
Value:    FLGRND!FLGKB!FLGFRC!FLGMOD!FLGPOS!DDWLO!DDRLO!
          DDNFS+index
File:     xxDVR.MAC
Scope:    Global
```

SIZ.xx - Line Width
```
Purpose:  Define line width for the device
Value:    5*14.+1:  Line width does not apply
          width+1:  Line width is fixed
          0:        Line width is variable
File:     xxDVR.MAC
Scope:    Global
```

BUF.xx - I/O Buffer Size
```
Purpose:  Define default buffer size in BUFTBL
Value:    Buffer size, in bytes
File:     xxDVR.MAC
Scope:    Global
```

CNT.xx - Number of Units for Device
```
Purpose:  Define the number of units of device
Value:    Number of units for this device
File:     xxPRE.MAC
Scope:    Local
```

DDS.xx - DDB Size
```
Purpose:  Define the size of the DDB for this device
Value:    DDB size, in bytes
File:     xxPRE.MAC
Scope:    Global
```

SYMBOLIC VALUES (cont.)

CCC.xx - ↑C Flag
```
Purpose:  Signifies whether the device is ↑C interruptable
Value:    Non-zero: Device can be interrupted by a ↑C
File:     xxPRE.MAC
Scope:    Local
```

BFQ.xx - Buffer Quota
```
Purpose:  Define small buffer quota
Value:    Small buffer quota for first small buffer chain
File:     xxPRE.MAC
Scope:    Global
```

HOR.xx - Horizontal Line Width
```
Purpose:  Specify horizontal line width for device
Value:    Desired line width
File:     xxPRE.MAC
Scope:    Local
```

SLP.xx - Check Before Sleeping Flag
```
Purpose:  Specify whether the driver needs notification before honoring a conditional sleep
          request
Value:    Non-zero: Check with driver before sleeping
File:     xxPRE.MAC
Scope:    Local
```

UMR.xx - Notify Driver When UMR is Available
```
Purpose:  Specify that the driver contains a UMR$xx entry point and should be notified when
          a unibus mapping register becomes available
Value:    Non-zero: Notify driver if UMR is available
File:     xxPRE.MAC
Scope:    Local
```

ALT.xx - Alternate Device Name
```
Purpose:  Alternate device name
File:     TBL.MAC
Scope:    Global
```

TIM.xx - Timeout Clock Setting
```
Purpose:  Specify the amount of time to wait on a device operation request before aborting
File:     TBL.MAC
Scope:    Global
```

IDX.xx - Driver Index
```
Purpose:  Defines the driver index
File:     TBL.MAC
Scope:    Global
```

JS.xx - JBWAIT/JBSTAT Status Bit
```
Purpose:  Bit to set in JBWAIT and JBSTAT
File:     TBL.MAC
Scope:    Global
```

CSR.xx - Pointer to CSR Address
```
Purpose:  Point to the entry in CSRTBL for this device
File:     TBL.MAC
Scope:    Global
```

DEV.xx - Pointer to DDB for Unit n
```
Purpose:  Entry in DEVTBL where the DDB pointers for this device start.
File:     TBL.MAC
Scope:    Global
```

xxDDDB - Address of Unit 0 DDB
```
Purpose:  Point to first DDB for this device
File:     TBL.MAC
Scope:    Global
```

LOG$xx - Enter Error Logging
```
Purpose:  Initiate error logging
File:     TBL.MAC
Scope:    Global
```

WAIT2T - Reenter After Two Clock Ticks
```
Purpose:  Reenter at L3Q entry point after two clock ticks
File:     Not applicable
Scope:    Global
```

DEVICE DRIVERS

ENTRY POINTS

SYMBOLIC VALUES

FREBUF - Check Small Buffer Availability

Call:	CALLX	FREBUF,R5,BFQ.xx
Input:	BFQ.xx	Initial small buffer quota
	R1	Pointer to the DDB for this unit
Output:	R4	Random
	C-bit	Set: Insufficient buffers are available. Clear: Sufficient small buffers are available.

STORE - Store Character in Small Buffer

Call:	CALLX	STORE,R5,ddbufc+FP
Input:	ddbufc	Offset within DDB to Small Buffer Control Block
	R1	Pointer to DDB.
	R2	Byte to store in small buffer.
Output:	R4	Random
	C-bit	Set: Store failed due to small buffer shortage Clear: Store succeeded

FETCH - Get Character from Small Buffer

Call:	CALLX	FETCH,R5,ddbufc+EP
Input:	ddbufc	Offset within DDB to Small Buffer Control Block
	R1	Pointer to DDB
Output:	R2	Character fetched (if successful)
	R4	Random
	C-bit	Set: Fetch failed (no more data in small buffers) Clear: Fetch succeeded

CLRBUF - Return All Small Buffers

Call:	CALLX	CLRBUF,R5,ddbufc+EP
Input:	ddbufc	Offset within DDB to Small Buffer Control Block
	R1	Pointer to DDB
Output:	R4	Random

BUFFER - Allocate a Large Buffer

Call:	CALLX	BUFFER,R5,start
Input:	start	Global symbol corresponding to the buffer pool to start searching in for free space. The global symbols can be MONPOL, LRGPOL and EXTPOL
	R1	Requested buffer size (in bytes)
	R2	Number of buffers to leave in the monitor's pool (MONPOL)
Output:	R1	Size of buffer allocated.
	R4	Pointer to the buffer allocated.
	C-bit	Set: The request failed due to lack of buffers. Clear: The request succeeded.

RETCHN - Return All Large Buffers in a Chain.

Call:	CALLX	RETCHN
Input:	R4	Address of first buffer in the chain.
Output:	R3	Random
	R4	Random

INTSAV - Enter Interrupt Service Routine

Call:	CALLX	INTSAV,R5
	+	1200
	+	INT$xx
Input:	1200	
	INT$xx	Address of the interrupt service routine within the driver.
Output:	All registers preserved.	

INTSVX - Enter Secondary Interrupt Service Routine

Call:	CALLX	INTSVX,R5
	+	xxDAP5
	+	entry
Input:	xxDAP5	This global symbol points to the APR5 base value specified in the INTSAV subroutine call.
	entry	This word specifies the entry point within the interrupt service routine to transfer to.
Output:	All registers preserved.	

DEVICE DRIVERS

MONITOR SUBROUTINES

IOFINC - I/O Conditionally Finished

Call:	CALLX	IOFINC,R5,<FBQ.xx-count,JS.xx>
Input:	BFQ.xx	Initial small buffer quota.
	count	Minimum number of small buffers available in quota to allow IOFINI.
	JS.xx	JBSTAT bit to set to make the job urnnable.
	R1	Points to the DDB for this unit.
Output:	R4	Job number times 2

IOREDO - Stall for I/O Redo

Call:	JMPX	IOREDO
Input:	None	
Output:	None.	This is an exit routine.

RTI3 - Exit and Check Level Three Queue

Call:	JMPX	RTI3
Input:	None	
Output:	None.	This is an exit routine.

RETDEV - Return an Open Device

Call:	CALLX	RETDEV
Input:	R1	Pointer to DDB for this unit
	R5	Pointer to job's IOB entry for this channel
Output:	R2	Random

ERLDVR - Enter Error Log Entry

Call:	CALLX	ERLDVR,R5
	.BYTE	errcod
	.BYTE	ddbsiz
	.BYTE	offset,regcnt
	...	
	.BYTE	0,0
Input:	errcod	Error code to enter in error log
	ddbsiz	Size of DDB
	offset	Offset from CSR to first register to log
	regcnt	Number of registers to log
	R1	Pointer to DDB
	R3	Pointer to CSR for this device
Output:	R0	Pointer to timeout flag in error log table
	R1	Random
	R2	Pointer to terminator in error log table
	R3	Random
	R5	Pointer to error logging control table

QUEUER - Enter Item in Queue

Call:	CALLX	QUEUER,R5,root
Input:	root	Address of location that contains a pointer to the first element in the queue.
	R4	Pointer to the link word of the item to enter into the queue.
Output:	R4	Preserved
	C-bit	Set: This item is the first in the queue Clear: This item is not the first in the queue

FNDJOB - Force a Job Into Memory

Call:	CALLX	FNDJOB
Input:	R0	Job number times 2 of job to find
	R3	L3Q bits to set on residency if non-resident
	R5	Pointer to user's buffer (unmapped)
Output:	R0	Random
	R2	Physical location of job, bits <16.21>
	R3	Physical location of job, bits <0:15>
	R5	Pointer to user's buffer (mapped by APR6)
	C-bit	Set: The job was non-resident and is coming in. Clear: The job is resident.
Exit:	Non-skip	A fatal error occurred when loading the job image, or its associated RTS or LIBs.
	Skip	The job image and its associated RTS and LIBs are OK.

UNLOCK - Unlock a Job's Memory

Call:	CALLX	UNLOCK
Input:	R0	Job number times 2
Output:	R2	Random

CLRRSQ - Clear Residency Quantum

Call:	CALLX	CLRRSQ
Input:	R1	Pointer to Job Data Block (JDB) for job
Output:	None	

DMPJOB - Dump the Current Job

Call:	CALLX	DMPJOB
Input:	None	
Output:	R0	Random
	R1	Random
	R2	Random
	R5	Random
	APR6	Random

MAPBUF - Map a Buffer

Call:	CALLX	MAPBUF
Input:	R4	Contorted address
Output:	R3	Mapped address
	APR6	Altered if address is in non-monitor buffer

GETUMR - Get a Unibus Mapping Register

Call:	CALLX	@GETUMR
Input:	R0	Negative of word count to transfer
	R1	MSB of 22-bit bus address
	R2	LSB of 22-bit bus address
	R4	Pointer to DDB or DSQ
Output:	R1	Most significant two bits of unibus address for use in DMA transfer. Random if request failed.
	R2	Least significant 18 bits of unibus address for use in DMA transfer. Random if request failed.
	C-bit	Clear: UMR was allocated. Set: Insufficient UMRs were available. None were allocated.

RHMADR - Initialize Unibus Mapping for RH70 or RH11

Call:	CALLX	@RHMADR
Input:	R0	Negative of word count to transfer
	R1	MSB of 22-bit bus address
	R2	LSB of 22-bit bus address
	R3	Address of RH70 CSR
	R4	Pointer to DDB or DSQ
Output:	Extended bus address loaded into RH70 or RH11	
	C-bit	Clear: UMR was allocated. Set: Insufficient UMRs were available. None were allocated.

RETUMR - Return a Unibus Mapping Register

Call:	CALLX	@RETUMR
Input:	R4	Pointer to DDB or DSQ
Output:	None	

IOFINI - I/O Finished

Call:	CALLX	IOFINI,R5,JS.xx
Input:	JS.xx	JBSTAT bit to set to make the job runnale.
Output:	R4	Preserved

MONITOR ODT COMMANDS

v/	Open location v as a word and display its contents.
v\	Open location v as a byte and display its contents.
v'	Open location v as a byte and display as an ASCII character.
v''	Open location v as a word and display as 2 ASCII characters.
v%	Open location v as a word and display as 3 RAD50 characters.
<CR>	Close currently open location.
<LF>	Close current location and open following one.
↑	Close current location and open preceding one.
_	Close current location and open (PC + contents of current loc.)
@	Close current location and open (contents of current loc.)
>	Close current location and open (PC + low byte of current loc.)
<	Close current location and reopen last explicitly opened loc.
v=	Print v using print format specified in the $= register.
v;B	Set a breakpoint at location v and let ODT pick breakpoint #.
v;nB	Set breakpoint n at location v.
B	Remove all breakpoints.
nB	Remove breakpoint number n.
K	Print address of current location as offset from closest relocation register <> to current location.
nK	Print address of current location as an offset from relocation register n.
v;nK	Print address v as an offset from relocation register n.
F	Fill memory locations between $L and $H with contents of $A.
nF	Fill memory locations between $L and $H with n.
G	Start execution at location in register $7.
nG	Start execution at location n.
vO	Print PC-relative offset and branch displacement from the currently open location to location v.
v1;v2O	Print PC-relative offset and branch displacement from address v1 to address v2.
P	Proceed from the current breakpoint.
nP	Proceed through the current breakpoint n times before stopping
v;nR	Set relocation register n to value v.
v;R	Set relocation register 0 to value v.
R	Clear all relocation registers.
nR	Clear relocation register n.
n,v	Add contents of relocation register n to v to get new value.
S	Execute one instruction and stop.
nS	Execute n instructions before stopping.
W	Print locations between $L and $H equal to $A using mask $M.
N	Print locations between $L and $H not equal to $A using mask $M
E	Print locations between $L and $H with effective address of $A using mask $M.
L	Print all locations between $L and $H.
n(..)	Define action routine n.
nA	Execute action routine n.
n1;n2D	Execute action routine n1 when breakpoint n2 is entered.
nD	Remove the action routine associated with breakpoint n.
D	Remove action routines from all breakpoints.
n)	Display action routine n.
v[..]	Execute ODT commands within [ ] if expression v is non-zero.
1;v[..]	Execute ODT commands within [ ] if expression v is zero.
vT	Use the device located at location v in the I/O page as the console terminal.
<RO>	A rubout will abort any command currently in process.

ODT CONTROL REGISTERS

$P	ODT priority. (range 0-7 or set to 377_8)
$S	Processor status word (PSW).
$A	Argument in search command. (see the W, N and E commands)
$M	Mask in search command. (see the W, N and E commands)
$L	Starting address of range of locations in memory.
$H	Ending address of range of locations in memory.
$C	Constant register
$Q	Last value displayed by ODT.
$F	Print format for addresses. (if 0, relative to relocation register if non-0, absolute value)
$=	Print format for numeric values (if 0, octal. If -, signed decimal. If +, unsigned decimal)
$E	Value of APR6 at the time ODT was entered.

DEVICE DRIVERS

SYSTEM MACROS

.BR - Branch to Following Location

Format:	.BR	loc
Arguments:	loc	Location to branch to

.CALLR - Call Following Routine and Return

Format:	.CALLR	loc
Arguments:	loc	Location to call

PUSH - Push a Value on the Stack

Format:	PUSH	value
or:	PUSH	<list>
Arguments:	value	Value to be pushed on the stack
	list	List of values to be pushed on the stack

POP - Pop a Value from the Stack

Format:	POP	destin
or:	POP	<list>
Arguments:	destin	Destination to pop top of stack into
	list	List of destinations to pop top of stack into

.ASSUME - Verify Assumption

Format:	.ASSUME	arg1 cond arg2
Arguments:	arg1	First value
	arg2	Second value
	cond	Assumed relation between arg1 and arg2

REGSAV - Save Registers R0-R5

Format:	REGSAV	[inline]
Arguments:	inline	"INLINE" specifies that code should be generated inline.

REGSCR - Save Registers Co-Routine

Format:	REGSCR	
Arguments:	None	

REGRES - Restore Registers

Format:	REGRES	
Arguments:	None	

DEVICE - Define Device Driver Information

Format:	DEVICE	name,[alt],[<13qbit>]
Arguments:	name	Two character device name
	alt	Synonym for device name (optional)
	l3qbit	List of L3Q bit names to define for this device (optional)

BUFFER - Allocate/Deallocate Small Buffers

Format:	BUFFER	GETSML,clear,leave
or:	MOV	smlptr,R4
	BUFFER	RETSML
or:	MOV	lrgptr,R4
	BUFFER	RETURN
Arguments:	clear	Number of bytes in the small buffer to preset to 0 before returning from GETSML.
	leave	Minimum number of small buffers to leave in the monitor pool.
	smlptr	Address of small buffer to return.
	lrgptr	Address of large buffer header for buffer to return.

GETUSR - Get Byte from User Buffer

Format:	GETUSR	
Arguments:	R5	Pointer to byte in user buffer to be retrieved
	R2	Receives byte from user buffer

PUTUSR - Store Byte in User Buffer

Format:	PUTUSR	
Arguments:	R2	Character to store in user buffer
	R5	Pointer to location in user buffer to receive character

SETERR - Post Error Code

Format:	SETERR	errcod,destin[,WORD]
Arguments:	errcod	RSTS/E error code to post to the destination.
	destin	Location to receive error code.
	WORD	The error code should be moved as a word.

SYSTEM MACROS (cont.)

ERROR - Post Error and Exit

Format:	ERROR	errcod
Arguments:	errcod	RSTS/E error code to post to IOSTS

L3QSET - Set Bit in L3Q

Format:	L3QSET	bit1[,bit2]
Arguments:	bit1:	Bit to set in L3QUE or L3QUE2
	bit2:	Optional bit to set in L3QUE

MAP - Access Memory Management Registers

Format:	MAP	(Source),[OFFSET = offset,]APR = aprval,(CODE)[,PIC]
		(PUSH) (DATA)
		(POP)
Arguments:	source	Value to load into selected APR. If PUSH is used, the current value of the specified APR is pushed on the SP stack. If POP is used, the top of the SP stack is popped into the specified APR.
	offset	Constant offset to add to the value specified in "source".
	aprval	APR number
	CODE	The I-space register should be used on processors that support I and D space.
	DATA	The D-space register should be used on processors that support I and D space.
	PIC	Generate position independant code.

SPL - Set Processer Priority

Format:	SPL	prlvl
Arguments:	prlvl	Desired priority level

SPLC - Set Program Status Word

Format:	SPLC	prlvl
Arguments:	prlvl	Desired priority level

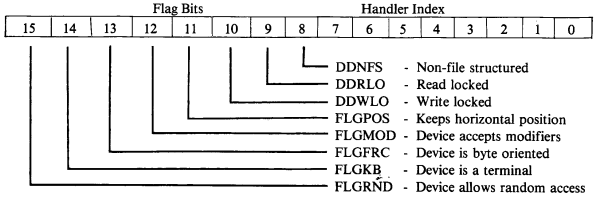
CRASH - Crash the System

Format:	CRASH	
Arguments:	None	

INTERRUPT ROUTINE (xxDINT PSECT)

```
        DEFORG  xxDINT
        ;Interrupt vector for unit 0
xxDINT: CALLX   INTSAV,R5       ;Map and enter interrupt service
          +     1200            ;PAR5 value
          +     INT$xx          ;Interrupt service routine address
        ;Repeat the following for each additional interrupt vector
        ;  xxnINT is the symbol for the location to vector
        ;  to for unit "n" of the device. INTxxn is the
        ;  address of the associated interrupt service routine.
xxnINT: CALLX   INTSVX,R5       ;Map and enter interrupt service
          +     xxDINT+4        ;Pointer to PAR5 value
          +     INTxxn          ;Interrupt service routine address
        ORG     xxDVR
```

Entries in FLGTBL - (FLg.xx) Device Dependent Flags



DEVICE DRIVERS

ODT

SYSTEMS MACROS

INTERRUPT SERVICE

DEVICE FLAGS