

Table of contents

5-	1	INITLN	-- Initialize a line
6-	1	NEWUSR	-- Start a new time-sharing job
7-	1	STOP	-- Stop program execution & enter KMON
8-	1	CLENUP	-- Do general cleanup when a job stops
9-	1	CANCPL	-- Cancel all pending completion routines
10-	1	LOGOFF	-- Log off a job
11-	1	TSXTX	-- Trap Handler
14-	1	FPTRPX	-- Floating point trap routine
15-	1	CLKRUN	-- Clock processing routine
16-	1	CLKDAT	-- update time of day and date
17-	1	CLKJOB	-- check time slice job status
18-	1	CLKO1S	-- 0.1 second clock processing
19-	1	CLKIOH	-- See if we need to cancel I/O hold timers
20-	1	CHKPRT	-- See if we need to print Professional screen
21-	1	WAKEUP	-- 0.5 second processing for sleeping users
22-	1	CKTWAT	-- Check on jobs doing .TWAIT waits
24-	1	CKMRKT	-- check mark-time requests
25-	1	CLKSCR	-- Execute completed system mark-time requests
26-	1	CLKPM	-- accumulate performance monitoring data
27-	1	CKSCHD	-- Check jobs and schedule
28-	1	CLKABD	-- Clock processing for autobaud logic
29-	1	TLCHK	-- Check Dial-up line status
30-	1	CLKPHN	-- Do timer driven checks of dial-up lines
31-	1	DLGDSS	-- Get data set status for DL11 line
32-	1	DLSDSS	-- Set data set status for DL11 line
33-	1	DLSBRK	-- Control break transmission for a DL11 line
34-	1	DLSSPD	-- Set transmission speed for DL11 line
35-	1	DZGDSS	-- Get data set status for DZ11 line
36-	1	DZSDSS	-- Set data set status for a DZ11 line
37-	1	DZSBRK	-- Control break transmission for a DZ11 line
38-	1	DZSSPD	-- Set transmission speed for a DZ11 line
39-	1	DHGDSS	-- Get data set status for a DH11 line
40-	1	DHSDSS	-- Set data set status for a DH11 line
41-	1	DHSSPD	-- Set transmit/receive speed for DH11 line
42-	1	DHSBRK	-- Control break transmission for a DH11 line
43-	1	VHGDSS	-- Get data set status for a DHV11 line
44-	1	VHSDSS	-- Set data set status for a DHV11 line
45-	1	VHSSPD	-- Set transmit/receive speed for a DHV11 line
46-	1	VHSBRK	-- Control break transmission for a DHV11 line
47-	1	DLCLOK	-- Timer driven routine for DL11 lines
48-	1	DZCLOK	-- Timer driven routine for DZ11 lines
49-	1	DHCLOK	-- Timer driven routine for DH11 lines
50-	1	SYSDIE	-- Fatal system halt
52-	1	EXCINI	-- Final system initialization
53-	1	INISPD	-- Initialize time-sharing line speeds
54-	1	INSINI	-- Initialize installed program table

```

1          .TITLE  TSEXC2  -- Misc. TSX-Plus Executive Routines
2          .ENABL  LC
3          .ENABL  AMA
4          .DSABL  GBL
5 000000   .CSECT  TSEXC2
6 000000   021440   TSEXC2: .RAD50  /EX2/
7          ;
8          ;
9          ;-----
10         ; TSEXC2 is a TSX-Plus virtual system overlay containing misc. routines.
11         ;
12         ; Copyright (c) 1980, 1981, 1982, 1983, 1984, 1985, 1986, 1987, 1988.
13         ; S&H Computer Systems, Inc.
14         ; Nashville, Tennessee USA
15         ;
16         ; All rights reserved
17         ;-----
18         ;
19         ; Global definitions
20         ;
21         .GLOBL LOGOFF, TRPCOM, TSXTX, FPTRPX, EXCINI, NSIP
22         .GLOBL SYSDIE, ABORT, STOP, NEWUSR, TRPBPT
23         .GLOBL INITLN, CLKRUN, DZSSPD, DLSBRK, DZSBRK
24         .GLOBL DLQDSS, DLSDSS, DZGDSS, DZSDSS, DHGDSS, DHSDDS
25         .GLOBL DLCLOK, DZCLOK, VHGDSS, VHSDDS, DLSSPD
26         .GLOBL DHSSPD, VHSSPD, DHSBRK, VHSBRK, DHCLOK, VHCLOK
27         ;
28         ; Global references
29         ;
30         .GLOBL R$SWPC, C. NUMQ, VUSPHN, $RDSAV, DOTRMP, $GEMAR, R$CFST, CFPSAV
31         .GLOBL LSW11, $PWKEY, AF$NPW, LN$SPAC, SUCF2, IB$SF2, IB$IJ, SPIJ
32         .GLOBL LTTCR, EMTBLK, AF$DUP, AF$IND, AF$UCL, AF$SET, LCXTBL
33         .GLOBL CINFLG, DIEARG, DIEMSG, DIEPC, DIESP, VSYDMP, DODUMP, SYSHL1
34         .GLOBL VPAR5, TRPAR5, VDMKTP, DMPOVL, DMPHND, DMPTXT, PLSINI
35         .GLOBL VSWPSL, SWPJOB, SWPPOS, SLTSIZ, CSHDEV, CSHDVN
36         .GLOBL $DETCH, LSW, $DILUP, LOTSPC, LOTSIZ, CDSSPD, $DHCDO
37         .GLOBL LSTPL, PVON, LNPRIM, LINSWT, LSECPT, MAXSEC, NUMON, NEDCLO
38         .GLOBL PO$SPV, PO$NAM, PO$SYS, PO$BYP, LPARNT, CL$EPS
39         .GLOBL PO$DBG, SYNAME, PO$MEM, PO$LOK, II$PRV, PO$NFR, PO$NFW
40         .GLOBL INSTBL, INSTBN, II$SZ, II$NAM, II$FLG, II$NPV, AF$PLK
41         .GLOBL AF$SCA, AF$NOW, AF$HIE, AF$NOI, AF$IOP, AF$MEM, AF$BYA
42         .GLOBL LNMAP, $DISCN, FORCEEX, LBASE, LNBLKS, VH$LCR, CLTOTL
43         .GLOBL LP$SPD, LP$7BT, LP$PAR, LP$ODD, SYSXIT, VDBFLG, CXBOWN
44         .GLOBL DZ$LEN, DZ$8BT, DZ$7BT, DZ$PAR, DZ$ODD, SP$LNK
45         .GLOBL HF$LEN, HF$8BT, HF$7BT, HF$PAR, HF$ODD, WINREL
46         .GLOBL VF$LEN, VF$8BT, VF$7BT, VF$PAR, VF$EVN, KMONCE, VMNUAO
47         .GLOBL $AUTO, S9600, $NABRS, SETSPD, LABTIM, PIDPTR, $CTRLD, CSHFIN
48         .GLOBL MXTYPE, CDX$DZ, INTMX1, VF$RIE, VF$TIE, VH$CSR, MH$SCR, FP$IOP
49         .GLOBL HF$TSB, MH$LPR, MH$BRK, VF$SC, VH$LPR, VF$BC, $NOUCR, JOBCCB
50         .GLOBL FRKINI, FQ$SZ, NUMFRK, FRKGEN, FREFRK, FQ$LNK, SCHED, NPCCB
51         .GLOBL LMEMIN, DEQ, LSW2, LSW4, LSW5, $1STLG, LSW6, $CARUP, TON, PMUSER
52         .GLOBL PMRUN, SS, CORUSR, EXEC, JCDB, LSW7, CANIOT, S150, LIOHLD, NSCP
53         .GLOBL UMODE, INTLVL, UTRPAD, CHKABT, UPMODE, UFPTRP, CW$FPU, SCPFHD
54         .GLOBL CONFIG, TRRDY, CTTSR, CTTBR, $DBGMD, $NOLF, LCXPAR, $RNMLK
55         .GLOBL FREIOQ, NUMIOQ, IOQ$SIZ, Q. LINK, USRINI, LSW9, $SUCF, SYPNCR
56         .GLOBL LSTMX, MXLNT, LSTPL, LMXNUM, MXLNT, CDCLOK, LCDTYP, LINSIZ
57         .GLOBL SNMSHD, NMSNMB, SB$SZ, SB$LNK, CLKINT, $CARMN, LINSPC, SP$SZ

```

```

58 . GLOBL $START, ILSW2, $DEAD, $PHONE, FSTD, LSTD, LSUCF, $TDEAD
59 . GLOBL INITFL, LSW3, PSW, INTPRI, SYSHLT, CHKUSP, MSGABT, CFACFL, $CFABT
60 . GLOBL $INKMN, ABRTAD, ABRTCD, STOP, LMXLN, OVRHC, MAPUSR, MONABT
61 . GLOBL $INIT, LQUAN, LJSW, LINBUF, LINNXT, LSTACT, LINPNT, LINCNT
62 . GLOBL LOTBUF, LOTNXT, LOTPNT, LACTIV, $LOFCF, CSHALC, NUMCCB, CCBHD
63 . GLOBL LCOL, LAFSIZ, LPROJ, LPROG, LSCCA, LBRKQ, LBRKCH, LCPUI
64 . GLOBL LCPULO, LCONTM, LINCUR, KPAR6, CURRDB, RPAR, RPDR, CXTRMN
65 . GLOBL VECBAS, MVWDS, ITRMTP, LTRMTP, MSGINI, VMAXMC, $SHICP, KILJOB
66 . GLOBL $CTRLS, $VNOTT, SPLINI, NSPLDV, LOKINI, VMXSF, LITIME, QNSPND
67 . GLOBL SETMAP, DFJMEM, MAXMEM, EMTCAD, EMTRAD, RCBBAS, RCBEND, PLSXIT
68 . GLOBL SPCPS, JSTKND, JSTK, EMTLEV, UERSEV, $DOOFF, $NOIN, ERRLOC
69 . GLOBL VSWPFL, KMNTOP, KMNBAS, SUTOP, UHIMEM, JSWLOC, R$CH17
70 . GLOBL KMNCHN, CSIARE, KMNSTKM, LQUAN, KMNSTR, $CTRLC, LSLEPH, LSLEPL
71 . GLOBL LSPND, $IDMAP, $MLOCK, LRDTIM, IOHALT, IOSTOP, RTSTOP
72 . GLOBL USRJOB, FREUSR, FRESPD, CANMKT, QFREE, LCMPL, CQ$LNK
73 . GLOBL CLSCDB, HF$TIE, HF$RIE, DLSTRT, DZSTRT, CQ$CP
74 . GLOBL LNSBLK, RCBBAS, RCBEND, FREMEM, VPLAS, TRNSTR, $HARD
75 . GLOBL CXPAG, KMNPGS, LIOCNT, VPRIDF, LBSPRI, LPRI, LSTHL
76 . GLOBL LHIPCT, QHIPRI, INVEC, INRECV, RSR, STPFLG, $VIRJB
77 . GLOBL RING, TRMRDY, MXRING, MXDTR, CARDET
78 . GLOBL MXCAR, RCVDON, $IITIM, RDINT, RDONE, MXCSR, RIE, NEDCDO, NEDCDI
79 . GLOBL $XCHAR, LOGCHR, LOGCR, LOGFLG, LF$IN, FPUUSE, $MAPOK, R$MFMV
80 . GLOBL SPSTAT, SS$RUN, SS$PRT, LPRG1, LPRG2, $NOABT
81 . GLOBL CURVC, SYSDAT, LSTATE, LSTSL, SWPCOT, DOSCHD, $INWT, $FPUX
82 . GLOBL LRTCHR, MRKTHD, CQ$LNK, $IOWT, PMBASE, PMTOP, PMNBPC, VPAR6
83 . GLOBL PMPAR, $SOTFN, $OTFN, $DEFER, $DODFR, $GCECO, $OTLO, NEDSOT
84 . GLOBL CLKRUN, TIKCNT, CLKCNT, LCPULO, LCPUI, TKIVAL, CQ$JOB
85 . GLOBL TK1CNT, TK5CNT, SYTIML, SYTIMH, DATIML, DATIMH, JM$LNK
86 . GLOBL $DHB1, $DHB2, LSW10, VF$RIE, HF$RIE
87 . GLOBL VONTM, VOFFTM, VTMOUT, VTMIN, VTMLC, LOFFTM, LCDTIM
88 . GLOBL VMXWIN, WININI, RDAR, RDDR
89 . GLOBL LMING, MINCTR, MINTIM, DTLX, UIOCNT, STRACT, $DEBUG
90 . GLOBL VPRIHI, VPRILO, VQUANO, VQUAN3, $SQ0, $SQ3, LCLUNT
91 . GLOBL TMTOTL, TMTOTH, TMIDL, TMIOH, INBSY, OUTBSY, DBGTRP
92 . GLOBL TMSWPL, TMSWPH, TMUSRL, TMUSRH, TMIOWL, TMIOWH
93 . GLOBL TMSWTL, TMSWTH, TMIDL, TMIDLH, SYSHLT, GETMEM, EMTCAS
94 . GLOBL S$HIP, S$RT, S$TMWT, S$TWFN, CQ$PRI, $OITIM
95 . GLOBL VQUAN1, VQUN1A, VQUAN2, QCPU, MBFFLG, UREGO, S$WSMB
96 . GLOBL LSLEPL, LSLEPH, ENQTL, QCOMPL, VQUN1B, MONFQH, VMXMON, JM$SZ
97 . GLOBL CQ$HOT, CQ$LOT, CQ$LNK, CQ$RNS, CSHINI, FRKPRI
98 . GLOBL PF$SYS, PF$IOW, PF$OVF, CLKPS, CLKPC, CURCP, SHRCB, SHRCN
99 . GLOBL PMFLGS, UMODE, LEMTPC, $INCOR, EM$DTL, CP$STD
100 . GLOBL LITIME, VINTIO, VQUN1C, QUNSIG, MS$BRK, TRBRK
101 . GLOBL LSW8, $SQ1, $SQ1A, $SQ1B, $SQ1C, $SQ2
102 . GLOBL CC$SZ, CC$LNK, CXTBAS, CXTWDS, PCCR2, PROFLG
103 . GLOBL CDGDS, CDSDSS, MS$DTR, MS$CAR, MS$RNG, PRODC, PLSOFF
104 . GLOBL CDX$VH, VH$CSR, VH$LSR, VH$LCR, $DBGK
105 . GLOBL VF$RNG, VF$DCD, VF$DTR, D. FLAG, D$RUN
106 . GLOBL MF$LIN, DM$CSR, DM$LSR, MF$RNG, MF$CAR, MF$DTR
107 . GLOBL FP$CK1, FRKGET, FORKQ, FQ$PRI, FQ$RTN
108 . GLOBL CDIRTN, CDIFLG, FP$CDI, CDORTN, CDIFLG, FP$CDO
109 . GLOBL TSR, MXBRK, LMXPRM, MXLPR, MXSBRK, FREEXT
110 . GLOBL OVRADD, O. ADR, O. PAR, NUMDEV, HANPAR, PNAME
111 . GLOBL $UDSPC, SR3FLG, USDSPC, SR3MMR

```

; Symbolic equates

115						
116	000015	;				
117	000012	CR	=	15		; Carriage-return
118	000007	LF	=	12		; Line-feed
		BELL	=	7		; Bell

```

1 ;-----
2 ; Macro calls
3 ;
4 ; .MCALL .READW, .PURGE
5 ;
6 ;-----
7 ; Macro definitions
8 ;
9 ; Macro to call a routine in another system overlay.
10 ;
11 ; .MACRO OCALL ENTADD
12 ; .IF B, ENTADD
13 ; .ERROR ;OCALL without entry address
14 ; .ENDC
15 ; CALL OVRHC ;call the low-core overlay handler
16 ; .WORD ENTADD
17 ; .ENDM OCALL
18 ;
19 ; Macro to disable interrupts
20 ;
21 ; .MACRO DISABL
22 ; BIS #340, @PSW
23 ; .ENDM DISABL
24 ;
25 ; Macro to enable interrrupts
26 ;
27 ; .MACRO ENABL
28 ; BIC INTPRI, @PSW
29 ; .ENDM ENABL
30 ;
31 ; Macro to print an error message when a system crash occurs.
32 ;
33 ; Arguments:
34 ; MSG = Name of error message to print.
35 ; ARG = (Optional) argument value to display with error message.
36 ;
37 ; .MACRO DIE MSG, ARG
38 ; MOV MSG, @DIEMSG
39 ; .IF NB, ARG
40 ; MOV ARG, @DIEARG
41 ; .ENDC
42 ; CALL @SYSHLT
43 ; .ENDM DIE
44 ;
45 ; Macro to define a system abort error message
46 ;
47 ; .MACRO SATXT NAME, TEST
48 ; .GLOBL EM$'NAME
49 EM$'NAME = . - DIEBAS
50 ; .ASCIZ \'NAME\'--\'TEST\'
51 ; .ENDM SATXT

```

```

1 ; -----
2 ; Fatal system abort error messages:
3 ;
4 ; .NLIST BEX
5 DIEBAS:
6 SATXT DTL,<Demonstration system time limit reached>
7 SATXT FRK,<No free FORK blocks>
8 SATXT JMO,<Jump occurred to location 0>
9 SATXT KRE,<KMON read error>
10 SATXT KTP,<Kernel mode trap>
11 SATXT LMF,<Job lock mem failure>
12 SATXT MID,<Need to increase value of MIONWB sysgen parameter>
13 SATXT MPR,<Memory parity error>
14 SATXT NQE,<Ran out of free I/O queue elements>
15 SATXT NSP,<No free swap command packets>
16 SATXT PFT,<Power-fail trap>
17 SATXT RIT,<Trap in real-time interrupt service routine>
18 SATXT SFO,<Job swap file overflow>
19 SATXT SIE,<Swap file I/O error>
20 SATXT SSE,<PLAS region swap file I/O error>
21 SATXT SJN,<Job # 0 at STOP>
22 SATXT UEI,<Interrupt occurred at unexpected location>
23 SATXT SOF,<Stack overflow>
24 ;
25 ; Other related text strings.
26 ;
27 001056 015 012 012 TXFSE: .ASCII <CR><LF><LF><BELL>/?TSX-F-Fatal system error at /<200>
28 001120 101 162 147 TXARG: .ASCII /Arg. value = /<200>
29 001136 000 TXNUL: .BYTE 0
30 001137 120 101 122 TXPAR5: .ASCII /PAR5 value = /<200>
31 001155 123 145 147 TXSEG: .ASCII /Seg. value = /<200>
32 001173 117 166 145 TXOID: .ASCII /Overlay: /<200>
33 001205 104 145 166 TXDEV: .ASCII /Device name: /<200>
34 001223 123 120 040 SPTXT: .ASCII /SP at time of crash = /<200>
35 ;
36 ;
37 ; Line select bits for a DH11 mux.
38 ;
39 001252 000001 000002 000004 DHLBIT: .WORD 1,2,4,10,20,40,100,200,400
40 001274 001000 002000 004000 .WORD 1000,2000,4000,10000,20000,40000,100000
41 ;
42 ; Line select bits for DZ11 mux.
43 ;
44 001312 001 002 004 MXLBIT: .BYTE 1,2,4,10,20,40,100,200
45 ;
46 ; Number of days in each month
47 ;
48 001322 037 034 037 MONDAY: .BYTE 31.,28.,31.,30.,31.,30.,31.,31.,30.,31.,30.,31.
49 ;
50 001336 000000 FORKIT: .WORD 0 ;Flag to create fork process
51 001340 0000000 PROTIM: .WORD PROODC ;Call PI driver after this many ticks
52 001342 177777 TIK01S: .WORD -1 ;# pending 0.1 second clock ticks
53 001344 063337 R5OPRO: .RAD50 /PRO/
54 001346 063344 R5OPRT: .RAD50 /PRT/
55 001350 073376 R5OSAV: .RAD50 /SAV/
56 ;
57 ; Table to convert normal TSX-Plus speed codes into DH11 speed codes

```

```
58  
59 001352      001      002      003 ; DHSPCT: .BYTE 1,2,3,4,5,7,10,11,12,0,13,0,14,0,15,16  
60 ;  
61 ; Table to convert normal TSX-Plus speed codes into DHV11 speed codes  
62 ;  
63 001372      000      001      002 VHSPCT: .BYTE 0,1,2,3,4,5,6,7,10,11,12,0,13,14,15,16  
64 .EVEN
```

```

1
2 ; -----
3 ; Table of programs that have automatic switch assignment when they
4 ; are started by TSKMON. The following flags may be specified following
5 ; the program name words:
6 ;
7 ;     AF$NOW = Allow non-wait .TTYIN operation.
8 ;     AF$HIE = Run program in high-efficiency mode.
9 ;     AF$NOI = Run program in non-interactive mode.
10 ;     AF$IOP = Map user PAR 7 to I/O page (requires operator priv.)
11 ;     AF$SCA = Enable single character activation.
12 ;     AF$MEM = Lock program in low memory.
13 ;     AF$PLK = RUN/LOCK program
14 ;     AF$DBG = RUN/DEBUG program
15 ;     AF$BYA = Bypass user ASSIGNS
16 ;     AF$TPO = Use transparent terminal output
17 ;     AF$DUP = Program is DUP
18 ;     AF$IND = Program is IND
19 ;     AF$UCL = Program is TSXUCL
20 ;     AF$SET = Program is SETUP
21 ;     AF$CCA = Suppress ctrl-C abort
22 ;     AF$NPW = No windows during program
23 ;
24 ; Each program entry must consist of two words containing the 6 character
25 ; program name followed by a word with the flags.
26 ;
27 001412 000000 NSIP = 0 ;No system installed programs yet
28 001412 016130 000000 SRFPRG: .RAD50 /DUP /
29 001416 000000G .WORD AF$DUP
30 001420 000000 .WORD 0
31 000001 NSIP = NSIP+1
32 001422 035164 000000 .RAD50 /IND /
33 001426 000000C .WORD AF$NOW!AF$IND
34 001430 000000 .WORD 0
35 000002 NSIP = NSIP+1 ;Count another system program
36 001432 042614 000000 .RAD50 /KED /
37 001436 000000C .WORD AF$SCA!AF$HIE!AF$NOW
38 001440 000000 .WORD 0
39 000003 NSIP = NSIP+1 ;Count another system program
40 001442 042640 000000 .RAD50 /KEX /
41 001446 000000C .WORD AF$SCA!AF$HIE!AF$NOW
42 001450 000000 .WORD 0
43 000004 NSIP = NSIP+1 ;Count another system program
44 001452 045130 000000 .RAD50 /K52 /
45 001456 000000C .WORD AF$SCA!AF$HIE!AF$NOW
46 001460 000000 .WORD 0
47 000005 NSIP = NSIP+1 ;Count another system program
48 001462 046537 057760 .RAD50 /LOGON /
49 001466 000000C .WORD AF$PLK!AF$BYA
50 001470 000001 000000C .WORD +1,PO$SPV!PO$NAM!PO$SYS!PO$BYP
51 001474 000000 .WORD 0
52 000006 NSIP = NSIP+1
53 001476 062074 012000 .RAD50 /PATCH /
54 001502 000000G .WORD AF$SCA
55 001504 000000 .WORD 0
56 000007 NSIP = NSIP+1 ;Count another system program
57 001506 073634 102700 .RAD50 /SETUP /

```



```

58 001512 000000C .WORD AF$IOP!AF$SET
59 001514 000000 .WORD 0
60 000010 NSIP = NSIP+1 ;Count another system program
61 001516 075273 051646 .RAD50 /SYSMON/
62 001522 000000 .WORD 0
63 001524 000001 000000G .WORD +1,PO$MEM
64 001530 000000 .WORD 0
65 000011 NSIP = NSIP+1 ;Count another system program
66 001532 076713 056700 .RAD50 /TECO /
67 001536 000000C .WORD AF$SCA!AF$NOW
68 001540 000000 .WORD 0
69 000012 NSIP = NSIP+1 ;Count another system program
70 001542 077721 055176 .RAD50 /TRANSF/
71 001546 000000C .WORD AF$SCA!AF$NOI!AF$NOW!AF$NPW
72 001550 000000 .WORD 0
73 000013 NSIP = NSIP+1
74 001552 077771 103150 .RAD50 /TSAUTH/
75 001556 000000G .WORD AF$BYA
76 001560 000000 .WORD 0
77 000014 NSIP = NSIP+1 ;Count another system program
78 001562 100020 101704 .RAD50 /TSXUCL/
79 001566 000000G .WORD AF$UCL
80 001570 000000 .WORD 0
81 000015 NSIP = NSIP+1
82 001572 106243 057710 .RAD50 /VTCOM /
83 001576 000000C .WORD AF$SCA!AF$NOW!AF$MEM
84 001600 000001 000000G .WORD +1,PO$LOK
85 001604 000000 .WORD 0
86 000016 NSIP = NSIP+1 ;Count another system program
87 001606 SRFEND: ;End of special program flag list

```

INITLN -- Initialize a line

```

1          .SBTTL  INITLN -- Initialize a line
2          ;-----
3          ; INITLN is called to initiate (logon) a line.
4          ; It initializes a number of line control tables and then places
5          ; the line in a high-priority execution state.
6          ; If the system is generated with job swapping turned off (SWAPFL=0)
7          ; a check is made to see if there is sufficient free memory available
8          ; for the job before it is initiated.  If there is not enough free memory
9          ; available, the job is not initiated.
10         ;
11         ; Inputs:
12         ; R1 = Number of line to be initiated
13         ; R0 = Pointer to I/O queue element with name of secondary
14         ;       start-up command file for the job (0=none).
15         ;
16         ; Outputs:
17         ; C-flag set if swapping is disabled and there is insufficient free
18         ; memory space available to start job.
19         ; The queue element with the secondary start-up command file name is
20         ; freed if the job cannot be started.  Otherwise it is freed after the
21         ; job is initialized.
22         ;
23 001606 010246  INITLN: MOV     R2,-(SP)
24 001610 010346          MOV     R3,-(SP)
25 001612 010003          MOV     R0,R3          ;Get pointer to Q element with CF name
26         ;
27         ; Never start a line that is being used as a CL unit
28         ;
29 001614 005761 000000G  TST     LCLUNT(R1)      ;Is this a CL line?
30 001620 002176          BGE     9%              ;Br if yes
31         ;
32         ; See if we are constrained by max # jobs allowed to be on
33         ;
34 001622 020127 000000G  CMP     R1,#LSTPL      ;Is this a primary line?
35 001626 101016          BHI     4%              ;Br if not, no limits on virtual or detached
36 001630 105737 000000G  TSTB   KMONCE         ;Is first job done initializing?
37 001634 001013          BNE     4%              ;Br if not, VMNUAD not set up yet
38 001636 113700 000000G  MOVB   VMNUAD,R0      ;Get max # jobs allowed to be on
39 001642 042700 177400  BIC     #^C377,R0      ;Mask out any sign extension
40 001646 012702 000123  MOV     #123,R2        ;Set for XOR
41 001652 074200          XOR     R2,R0          ;Decrypt max # jobs allowed on
42 001654 001403          BEQ     4%              ;Br if no limit on # logged on jobs
43 001656 123700 000000G  CMPB   NUMON,R0       ;Max # jobs already logged on?
44 001662 103155          BHIS   9%              ;Br if yes
45         ;
46         ; Initialize some line control tables
47         ;
48 001664 005061 000000G  4#:    CLR     LSW(R1)   ;Reset job status flags
49 001670 005061 000000G          CLR     LNBLKS(R1)   ;JOB HAS NO ASSIGNED MEMORY NOW
50 001674 005061 000000G          CLR     LIOCNT(R1)   ;JOB HAS NO ACTIVE I/O
51 001700 005061 000000G          CLR     LCMPL(R1)   ;JOB HAS NOT PENDING COMPLETION ROUTINES
52         ;
53         ; Determine how much memory the line needs to be initiated.
54         ;
55 001704 013700 000000G          MOV     DFJMEM,R0   ;GET DEFAULT # KB FOR JOB
56 001710 006300          ASL     R0            ;CVT TO # PAGES
57 001712 063700 000000G          ADD     CXTPAG,R0    ;ADD # PAGES NEEDED FOR JOB CONTEXT BLOCK

```

INITLN -- Initialize a line

```

58 001716 020037 000000G          CMP      R0,KMNPGS          ;COMPARE TO # PAGES NEEDED TO RUN KMON
59 001722 103002                   BHIS     1$                ;BR IF JOB SPACE > KMON SIZE
60 001724 013700 000000G          MOV      KMNPGS,R0         ;MUST HAVE AT LEAST ENOUGH MEMORY FOR KMON
61                                     ;
62                                     ;   If this is a non-swapping system, make sure enough free pages are
63                                     ;   available for this job.
64                                     ;
65 001730 010061 000000G          1$:     MOV      R0,LMEMIN(R1) ;SET # MEMORY PAGES NEEDED FOR JOB
66 001734 105737 000000G          TSTB    VSWPFL           ;IS THIS A SWAPPING SYSTEM?
67 001740 001010                   BNE     3$                ;BR IF YES
68 001742 010061 000000G          MOV      R0,LNBLKS(R1)   ;SET SIZE OF ROOT OF JOB REGION
69 001746 004737 000000G          CALL    GETMEM           ;TRY TO ALLOCATE MEMORY FOR THIS JOB
70 001752 103521                   BCS     9$                ;BR IF NOT ENOUGH MEMORY AVAILABLE
71 001754 052761 000000G 000000G  BIS      #INCR,LSW(R1)    ;SET FLAG SAYING JOB IS IN MEMORY
72                                     ;
73                                     ;   Initialize the line control tables for the job
74                                     ;
75 001762 016102 000000G          3$:     MOV      LINBUF(R1),R2 ;SET UP INFO ABOUT INPUT BUFFER
76 001766 010261 000000G          MOV      R2,LINNXT(R1)
77 001772 010261 000000G          MOV      R2,LSTACT(R1)
78 001776 010261 000000G          MOV      R2,LINPNT(R1)
79 002002 016161 000000G 000000G  MOV      LINSIZ(R1),LINSPC(R1)
80 002010 005061 000000G          CLR      LINCNT(R1)
81 002014 052761 000000G 000000G  BIS      #DILUP,LSW(R1) ;REMEMBER LINE IS ACTIVE
82 002022 052761 000000G 000000G  BIS      #NOIN,LSW3(R1) ;DON'T ACCEPT TT CHARS FOR LINE YET
83 002030 012761 000000G 000000G  MOV      #SUCF,LSW9(R1) ;Say we are in startup command file
84 002036 012761 000000G 000000G  MOV      #PWKEY,LSW11(R1);Initialize LSW11
85 002044 113761 000000G 000000G  MOVVB   VPRIDF,LBSPRI(R1);SET BASE PRIORITY FOR JOB
86 002052 113761 000000G 000000G  MOVVB   VPRIDF,LPRI(R1);SET CURRENT PRIORITY FOR JOB
87 002060 013761 000000G 000000G  MOV      VINTIO,LHIPCT(R1);INIT HIGH-PRIORITY HITS FOR JOB
88 002066 013761 000000G 000000G  MOV      VQUAN1,LITIME(R1);SET INTERACTIVE JOB TIME SLICE
89 002074 010361 000000G          MOV      R3,LPROJ(R1)    ;Store ptr to command file buffer in LPROJ
90                                     ;
91                                     ;   If this is a dial-up line and carrier is up, set flag that will
92                                     ;   cause us to log off the line if carrier is lost.
93                                     ;
94 002100 020127 000000G          CMP      R1,#LSTPL       ;Is this a primary line?
95 002104 101023                   BHI     2$                ;Br if not
96 002106 005061 000000G          CLR      LOFFTM(R1)      ;Don't have to time logoff time any more
97 002112 042761 000000G 000000G  BIC      #CARMN,LSW5(R1);Assume we do not need to monitor carrier
98 002120 032761 000000G 000000G  BIT      #PHONE,ILSW2(R1);Is this a dial-up line?
99 002126 001412                   BEQ     2$                ;Br if not
100 002130 013761 000000G 000000G  MOV      VTMLDC,LCDTIM(R1);Give local line this long to raise carrier
101 002136 032761 000000G 000000G  BIT      #CARUP,LSW3(R1);Is carrier up now?
102 002144 001403                   BEQ     2$                ;Br if not
103 002146 052761 000000G 000000G  BIS      #CARMN,LSW5(R1);Set flag saying to monitor carrier
104                                     ;
105                                     ;   Put job in high priority execution state
106                                     ;
107 002154 004737 000000G          2$:     CALL    QHIPRI           ;PUT JOB IN HIGH PRIORITY STATE
108                                     ;
109                                     ;   Count number of logged on jobs
110                                     ;
111 002160 105237 000000G          INCB    TOTON            ;Total number of logged on jobs
112 002164 020127 000000G          CMP      R1,#LSTPL       ;Is this a primary line?
113 002170 101003                   BHI     5$                ;Br if not
114 002172 105237 000000G          INCB    NUMON            ;Count another real line on

```

```
115 002176 000403          BR      6$
116 002200 020127 000000G  5$:    CMP      R1,#LSTDL      ;Is this a virtual line?
117 002204 101402          BLOS   B$                    ;Br if not
118 002206 105237 000000G  6$:    INCB   PVON              ;Count # primary & virtual lines on
119                               ;
120                               ; Successful finish
121                               ;
122 002212 000241          8$:    CLC                      ;Signal successful return
123 002214 000407          BR      10$
124                               ;
125                               ; We were not able to start the job
126                               ; Free the I/O queue element used to pass name of start-up command file
127                               ;
128 002216 010146          9$:    MOV     R1,-(SP)          ;Save job number
129 002220 010301          MOV     R3,R1                ;Get pointer to Q element
130 002222 001402          BEQ    11$                  ;Br if no Q element to free
131 002224 004737 000000G  CALL   QFREE                ;Free the Q element
132 002230 012601          11$:   MOV     (SP)+,R1        ;Restore job number
133 002232 000261          SEC                      ;Signal failure on return
134                               ;
135                               ; Finished
136                               ;
137 002234 012603          10$:   MOV     (SP)+,R3
138 002236 012602          MOV     (SP)+,R2
139 002240 000207          RETURN
```

NEWUSR -- Start a new time-sharing job

```

1          .SBTTL  NEWUSR -- Start a new time-sharing job
2          ;-----
3          ; Do initialization for start-up of a new job.
4          ;
5 002242   NEWUSR:
6          ;
7          ; Initialize LSW tables.
8          ;
9 002242   052761 000000G 000000G   BIS    #$INIT,LSW(R1) ;SAY JOB INITIALIZATION IS BEING DONE
10 002250   005061 000000G           CLR    LQUAN(R1)      ;INITIALIZE JOB'S TIME QUANTUM
11 002254   005061 000000G           CLR    LJSW(R1)      ;JOB STATUS WORD
12 002260   005061 000000G           CLR    LSW7(R1)
13 002264   005061 000000G           CLR    LSW8(R1)
14 002270   005061 000000G           CLR    LNSBLK(R1)   ;NO SPACE FOR ANY PLAS REGIONS
15 002274   012761 000000G 000000G   MOV    #$INKMN,LSW4(R1);START OUT RUNNING KMON
16 002302   016102 000000G           MOV    LOTBUF(R1),R2 ;SET UP INFO ABOUT OUTPUT BUFFER
17 002306   010261 000000G           MOV    R2,LOTNXT(R1)
18 002312   010261 000000G           MOV    R2,LOTPNT(R1)
19 002316   016161 000000G 000000G   MOV    LOTSIZ(R1),LOTSPC(R1)
20 002324   005061 000000G           CLR    LACTIV(R1)   ;SAY NO ACTIVATION CHARS RECEIVED YET
21 002330   005061 000000G           CLR    LCOL(R1)
22 002334   005061 000000G           CLR    LAFSIZ(R1)   ;NO ACTIVATION FIELD SIZE
23 002340   005061 000000G           CLR    LPROG(R1)   ;OR PROGRAMMER NUMBER
24 002344   005061 000000G           CLR    LSCCA(R1)   ;NO .SCCA DONE YET
25 002350   005061 000000G           CLR    LBRKCG(R1)  ;NO BREAK KEY CONNECTION
26 002354   005061 000000G           CLR    LBRKCH(R1)  ;CLEAR BREAK CHARACTER
27 002360   005061 000000G           CLR    LTTCR(R1)   ;No TT activation completion routine
28 002364   005061 000000G           CLR    LCPUHI(R1)  ;CLEAR RUN-TIME ACCUMULATOR
29 002370   005061 000000G           CLR    LCPULO(R1)
30 002374   005061 000000G           CLR    LCONTM(R1)  ;CLEAR CONNTECT-TIME
31 002400   005061 000000G           CLR    LINCUR(R1)
32          ;
33          ; Map kernel mode PAR 6 to job context block.
34          ;
35 002404   016137 000000G 000000G   MOV    LCXPAR(R1),@#KPAR6;MAP KERNEL PAGE 6 TO CONTEXT BLOCK FOR JOB
36          ;
37          ; Zero the job's context block
38          ;
39 002412   012702 000000G           MOV    #CXTBAS,R2   ;Get address of base of context area
40 002416   013703 000000G           MOV    CXTWDS,R3    ;Get # words in context area
41 002422   006203           ASR    R3            ;Get # of doublewords
42 002424   005022           7$: CLR    (R2)+        ;Clear first word of pair
43 002426   005022           CLR    (R2)+        ;Clear second word of pair
44 002430   077303           SOB    R3,7$        ;Loop if more doublewords left to clear
45 002432   032737 000001 000000G   BIT    #1,CXTWDS    ;Is there an odd word at end to clear?
46 002440   001401           BEQ    B$           ;Br if not
47 002442   005012           CLR    (R2)         ;Clear last word of context block
48          ;
49          ; Initialize some cells in job context block
50          ;
51 002444   012737 177777 000000G 8$: MOV    #-1,EMTLEV   ;Say job is not executing an EMT
52 002452   012737 000000G 000000G   MOV    #EMTCAS,EMTCAD ;Say completion routine return stack is empty
53 002460   012737 123456 000000G   MOV    #123456,JSTKND ;Set value used to check for stack overflow
54          ;
55          ; Set up simulated RMON fixed offset table.
56          ;
57 002466   013702 000000G           MOV    CXTRMN,R2    ;Get addr of RMON in job context area

```

```

58 002472 012703 000000G      MOV      #VECBAS,R3      ;POINT TO SYSTEM CHANNEL BLOCK
59 002476 012700 000000G      MOV      #MVWDS,R0      ;GET # WORDS TO MOVE
60 002502 012322      4$:  MOV      (R3)+,(R2)+    ;SET UP JOB'S SIMULATED RMON IN CONTEXT BLK
61 002504 077002      SOB      R0,4$
62      ;
63      ; Zero the I/O count in the channel used to access system swap file.
64      ; (Count could be non-zero if swap was in progress when we copied RMON data)
65      ;
66 002506 013700 000000G      MOV      CXTRMN,R0      ;Get pointer to base of simulated RMON
67 002512 062700 000000G      ADD      #R$SWPC,R0     ;Point to swap file channel in context block
68 002516 105060 000000G      CLRB    C.NUMQ(R0)     ;Zero I/O count in swap file channel block
69      ;
70      ; Change instruction in .MFPS routine in simulated RMON to return a
71      ; value of 0 rather than trying to access the I/O page
72      ;
73 002522 013700 000000G      MOV      CXTRMN,R0      ;Get pointer to base of simulated RMON
74 002526 062700 000000G      ADD      #R$MFMV,R0     ;Point to MOV @#PSW,(SP) instruction
75 002532 012720 012716      MOV      #012716,(R0)+  ;Store MOV #0,(SP) instruction
76 002536 005010      CLR      (R0)          ;Store 0 value following instruction
77      ;
78      ; See if we need to set up the name of a secondary start-up command
79      ; file for the job or other info provided by initiating job.
80      ;
81 002540 016104 000000G      MOV      LPROJ(R1),R4    ;Get pointer to buffer with file name
82 002544 001421      BEQ      10$           ;Br if no secondary start-up file
83 002546 116437 000000G 000000G  MOVVB    IB$IJ(R4),SPIJ  ;Save index # of initiating job
84 002554 010402      MOV      R4,R2
85 002556 062702 000000G      ADD      #IB$SF2,R2     ;Point to start of name string
86 002562 012703 000000G      MOV      #SUCF2,R3     ;Point to cell in context block
87 002566 112223      11$:  MOVVB    (R2)+,(R3)+    ;Move name into context block
88 002570 001376      BNE      11$           ;Loop till asciz null reached
89 002572 010146      MOV      R1,-(SP)      ;Save job number
90 002574 010401      MOV      R4,R1        ;Get pointer to Q element used as buffer
91 002576 004737 000000G      CALL    QFREE          ;Free the Q element
92 002602 012601      MOV      (SP)+,R1     ;Restore job number
93 002604 005061 000000G      CLR      LPROJ(R1)     ;Say no project number
94      ;
95      ; Determine if this is a primary, virtual, or detached line
96      ;
97 002610 020127 000000G      10$:  CMP      R1,#LSTDL     ;REAL OR VIRTUAL LINE?
98 002614 003021      BGT      1$           ;BR IF VIRTUAL
99 002616 020127 000000G      CMP      R1,#LSTPL     ;REAL OR DETACHED LINE?
100 002622 003404      BLE      2$           ;BR IF REAL
101      ;
102      ; Initialization for detached jobs only.
103      ;
104 002624 052761 000000G 000000G  BIS      #$DETC,LSW(R1) ;REMEMBER THIS IS A DETACHED JOB
105 002632 000430      BR      3$
106      ;
107      ; Initialization for primary lines only.
108      ;
109 002634 016161 000000G 000000G  2$:  MOV      ILSW2(R1),LSW2(R1); INIT SOME LSW TABLES
110 002642 016161 000000G 000000G  MOV      ITRMTP(R1),LTRMTP(R1); SET DEFAULT TERMINAL TYPE
111 002650 042761 000000C 000000G  BIC      #$CTRLS!$CTRLD,LSW3(R1); Enable terminal output
112 002656 000416      BR      3$
113      ;
114      ; Initialization for virtual lines only.

```

NEWUSR -- Start a new time-sharing job

```

115          ; Copy some information from parent line.
116          ;
117 002660 016102 000000G 1$:      MOV      LNPRIM(R1),R2      ;GET PRIMARY LINE #
118 002664 016261 000000G 000000G      MOV      LSW2(R2),LSW2(R1)
119 002672 016261 000000G 000000G      MOV      LTRMTP(R2),LTRMTP(R1)
120 002700 016261 000000G 000000G      MOV      LPROJ(R2),LPROJ(R1)
121 002706 016261 000000G 000000G      MOV      LPROG(R2),LPROG(R1)
122          ;
123          ; See if we should start line with deferred echo
124          ;
125 002714 032761 000000G 000000G 3$:      BIT      #$DEFER,LSW2(R1); Is deferred echo wanted?
126 002722 001403          BEQ      6$              ;Br if not
127 002724 052761 000000C 000000G      BIS      <#$DODFR!$@CECD>,LSW3(R1) ;Start deferring now
128          ;
129          ; Set up mapping registers for job.
130          ;
131 002732 004737 000000G 6$:      CALL     SETMAP          ;SET UP MAPPING REGISTERS FOR THE JOB
132 002736 013700 000000G      MOV      DFJMEM,RO      ;GET DEFAULT # K-BYTES OF MEMORY FOR JOB
133 002742 072027 000012      ASH      #10.,RO        ;CONVERT TO ADDRESS
134 002746 010037 000000G      MOV      RO,MAXMEM      ;SET AS DEFAULT UPPER LIMIT FOR JOB
135          ;
136          ; Switch to stack in job's context area.
137          ;
138 002752 012706 000000G      MOV      #JSTK,SP      ;SWITCH TO CONTEXT-BLOCK STACK
139          ;
140          ; Enter code to load KMON.
141          ;
142 002756 000466          BR      LDKMON          ;GO LOAD KMON

```

STOP -- Stop program execution & enter KMON

```

1          .SBTTL  STOP  -- Stop program execution & enter KMON
2          ;-----
3          ; STOP is jumped to when the job wants to terminate its execution and
4          ; enter KMON. This is usually caused by .EXIT, .CHAIN or CTRL-C.
5          ;
6 002760 113701 000000G STOP:  MOV  CORUSR,R1      ;GET JOB # OF CURRENT JOB
7 002764 001007          BNE  1$              ;IF SHOULD NOT BE ZERO
8 002766          DIE  #EM$SJN,(SP)      ;DIE IF JOB # = 0
9 003004 012706 000000G 1$:  MOV  #JSTK,SP      ;RUN ON JOB'S CONTEXT-BLOCK STACK
10 003010          ENABL          ;MAKE SURE INTERRUPTS ARE ENABLED
11 003016 052761 000000G 000000G BIS  #NOUCR,LSW9(R1);Tell DDCMPL not to run user compl routines
12 003024 012737 177777 000000G MOV  #-1,EMTLEV      ;SAY JOB IS NOT EXECUTING AN EMT
13 003032 052737 000000G 000000G BIS  #UPMODE,@#PSW    ;MAKE SURE PREVIOUS-MODE = USER
14 003040 032761 000000G 000000G BIT  #INKMN,LSW4(R1); IS KMON RUNNING NOW?
15 003046 001010          BNE  2$              ;BR IF YES
16 003050 106537 000052          MFPD @#52          ;GET JOB'S ERROR CELLS
17 003054 000316          SWAB (SP)          ;PUT (53) IN LOW-ORDER BYTE
18 003056 112637 000000G          MOV  (SP)+,UERSEV    ;SAVE USER SPECIFIED ERROR SEVERITY LEVEL
19 003062 005046          CLR  -(SP)          ;NOW CLEAR USER ERROR SEVERITY
20 003064 106637 000052          MTPD @#52
21          ;
22          ; Do general cleanup on exiting job.
23          ;
24 003070 004737 003422' 2$:  CALL  CLENUP          ;CLEAN UP STATUS OF JOB
25 003074 042761 000000G 000000G BIC  #NOUCR,LSW9(R1);Reenable user completion routine processing
26          ;
27          ; See if we should force logoff of this job.
28          ;
29 003102 032761 000000G 000000G BIT  #DISCN,LSW(R1) ;DID A LINE DISCONNECT OCCUR?
30 003110 001411          BEQ  LDKMON          ;BR IF NOT
31 003112 052761 000000G 000000G BIS  #DOOFF,LSW(R1) ;FORCE LOGOFF
32 003120 052761 000000G 000000G BIS  #NOIN,LSW3(R1) ;IGNORE INPUT FROM LINE DURING LOGOFF
33 003126 042761 000000G 000000G BIC  #DISCN,LSW(R1) ;ACKNOWLEDGE DISCONNECT
34          ;
35          ; Read KMON into memory and enter it
36          ;
37 003134 052737 000000G 000000G LDKMON: BIS  #UPMODE,@#PSW    ;SET USER-PREVIOUS-MODE IN PS
38 003142 042761 173330 000000G BIC  #173330,LJSW(R1);CLEAN OUT SOME FLAGS IN JOB STATUS WORD
39 003150 016146 000000G          MOV  LJSW(R1),-(SP) ;GET CURRENT JOB STATUS WORD
40 003154 106537 000000G          MFPD @#ERRLOC      ;GET JOB'S ERROR CELLS
41 003160 052761 000000G 000000G BIS  #INKMN,LSW4(R1);SAY KMON IS RUNNING
42 003166 042761 000000G 000000G BIC  #MAPDK,LSW7(R1);SAY CONTEXT BLOCK MAPPING DATA IS INVALID
43 003174 042761 000000G 000000G BIC  #VIRJB,LSW9(R1);SAY THIS IS NOT A VIRTUAL JOB
44 003202 105737 000000G          TSTB VSWPFL      ;IS THIS A NON-SWAPPING SYSTEM?
45 003206 001406          BEQ  3$              ;BR IF YES -- DON'T CHANGE MEMORY ALLOCATION
46 003210 013700 000000G          MOV  KMNTOP,RO      ;GET ADDRESS ABOVE TOP OF KMON
47 003214 162700 000000G          SUB  #KMNBAS,RO      ;GET AMT OF MEMORY NEEDED FOR KMON
48 003220 004737 000000G          CALL  SUTOP          ;SET TOP OF MEMORY FOR JOB
49 003224 004737 000000G 3$:  CALL  SETMAP          ;MAKE SURE MEMORY MAPPING SET FOR KMON
50 003230 013737 000000G 000000G MOV  KMNTOP,UHIMEM ;SAY JOB CAN ACCESS UP TO TOP OF KMON
51 003236 012637 000000G          MOV  (SP)+,@#ERRLOC ;SET ERROR CELLS
52 003242 011637 000000G          MOV  (SP)+,@#JSWLOC ;SET JSW
53 003246 012661 000000G          MOV  (SP)+,LJSW(R1)
54          ;
55          ; Set up status in user channel # 17 to allow us to access kmon file.
56          ;
57 003252 012703 000017          MOV  #17,R3          ;USE USER CHANNEL # 17

```


STOP -- Stop program execution & enter KMON

```

58 003256          .PURGE R3          ; MAKE SURE CHANNEL 17 IS FREE
59 003266 013703 000000G      MOV      CXTRMN,R3          ; BASE OF JOB CHANNEL AREA
60 003272 062703 000000G      ADD      #R$CH17,R3       ; POINT TO AREA FOR CHANNEL # 17
61 003276 012700 000005      MOV      #5,R0            ; # WORDS TO MOVE
62 003302 012704 000000G      MOV      #KMNCHN,R4       ; POINT TO BLOCK WITH KMON SAVED STATUS
63 003306 012423          1$:      MOV      (R4)+,(R3)+       ; SET UP INFO IN USER CHANNEL 17 BLOCK
64 003310 077002          SOB      R0,1$
65
66          ; Now read Kmon into user's program space.
67
68 003312 013703 000000G      MOV      KMNTOP,R3        ; GET TOP OF MEMORY ADDRESS FOR KMON
69 003316 162703 000000G      SUB      #KMNBAS,R3      ; SUBTRACT BASE ADDRESS OF KMON
70 003322 000241          CLC
71 003324 006003          ROR      R3                ; CVT TO # WORDS TO READ
72 003326          .READW #CSIARE,#17,#0,R3,#32. ; READ IN KMON
73 003362 103005          BCC     2$                ; BR IF READ OK
74 003364          DIE     #EM$KRE          ; SYSTEM HALT IF KMON READ ERROR
75
76          ; Set up Kmon user-mode stack pointer.
77
78 003376 013746 000000G      2$:      MOV      KMNSTK,-(SP)    ; GET KMON STACK POINTER
79 003402 106606          MTPD     SP                ; SET IN USER-MODE SP
80
81          ; Give Kmon a full time-slice.
82
83 003404 005061 000000G      CLR     LQUAN(R1)        ; CLEAR JOB TIME QUANTUM
84
85          ; Use RTI to enter Kmon in user mode.
86
87 003410          MOV     #UMODE,-(SP)    ; USER-MODE PS
88 003414          MOV     KMNSTR,-(SP)    ; STARTING ADDRESS IN KMON
89 003420          RTI

```

CLENUP -- Do general cleanup when a job stops

```

1          .SBTTL  CLENUP -- Do general cleanup when a job stops
2          ;-----
3          ; CLENUP is called to do I/O rundown and other general cleanup
4          ; operations when a job stops.
5          ;
6          ; Inputs:
7          ;   R1 = Job index number
8          ;
9          003422 010146 CLENUP: MOV     R1,-(SP)
10         003424 010246          MOV     R2,-(SP)
11         003426 010446          MOV     R4,-(SP)
12         003430 010546          MOV     R5,-(SP)
13         003432 042761 000000C 000000G BIC     <#RDSAV!$GEMAR>,LSW11(R1);No longer reading in SAV file
14         003440 005061 000000G          CLR     LSLEPH(R1)      ;CLEAR .TWAIT SLEEP TIME
15         003444 005061 000000G          CLR     LSLEPL(R1)
16         003450 005061 000000G          CLR     LSPND(R1)      ;CLEAR .SPND COUNT FOR JOB
17         003454 042761 000000C 000000G BIC     <#$IOMAP!$MLOCK!$DBGMD!$NOLF>,LSW6(R1);CLEAN OUT LSW6
18         003462 042761 000000C 000000G BIC     <#$DBGBK!$NOABT!$RNMLK,LSW9(R1);Clear LSW9 flags
19         003470 005061 000000G          CLR     LRDTIM(R1)     ;CLEAR ANY TT-READ TIMEOUT
20         003474 005061 000000G          CLR     LSCCA(R1)     ;REMOVE CONTROL-C TRAP CONTROL
21         003500 005061 000000G          CLR     LBRKCH(R1)    ;CLEAR BREAK CHARACTER
22         003504 005037 000000G          CLR     EMTRAD
23         003510 005037 000000G          CLR     SPCPS
24         003514 012737 000000G 000000G MOV     #EMTCAS,EMTCAD ;Reset return stack for completion routines
25         003522 005037 000000G          CLR     D.FLAG      ;Clear debugger control flags
26         ;
27         ; If double control-C was typed, echo this to the log file
28         ;
29         003526 032761 000000G 000000G BIT     #$CTRLC,LSW(R1) ;Were we aborted by 2 ctrl-c's?
30         003534 001424          BEQ     12$          ;Br if not
31         003536 105037 000000G          CLRB    CINFLG      ;Reset chain-in-progress flag
32         003542 032737 000000G 000000G BIT     #LF$IN,LOGFLG ;Are we logging input characters?
33         003550 001413          BEQ     11$          ;Br if not
34         003552 012700 000003          MOV     #3,R0      ;Echo ^c
35         003556          OCALL   LOGCHR
36         003564          OCALL   LOGCHR      ; twice
37         003572          OCALL   LOGCR      ;Log Cr Lf
38         003600 042761 000000G 000000G 11$: BIC     #$CTRLC,LSW(R1) ;Clear control-C abort flag
39         ;
40         ; Abort all I/O for this job
41         ;
42         003606 105737 000000G          12$: TSTB    CINFLG      ;Are we doing a .CHAIN?
43         003612 001004          BNE     16$          ;Br if yes
44         003614 004737 000000G          CALL   CANIOT      ;Cancel any pending .TIMID requests for job
45         003620 004737 000000G          CALL   IOHALT      ;Abort pending I/O operations for this job
46         ;
47         ; Do any real-time associated cleanup
48         ;
49         003624          16$: OCALL   RTSTOP      ;DO REAL-TIME CLEANUP
50         ;
51         ; Free the USR
52         ;
53         003632 120137 000000G          4$:  CMPB    R1,USRJOB ;ARE WE HOLDING THE USR?
54         003636 001004          BNE     15$          ;BR IF NOT
55         003640          OCALL   FREUSR      ;RELEASE IT
56         003646 000771          BR      4$          ;WE MAY HAVE LOCKED IT MORE THAN ONCE
57         ;

```

CLENUP -- Do general cleanup when a job stops

```

58          ; Free job context block buffer
59          ;
60 003650 120137 000000G 15$:  CMPB    R1,CXBOWN    ;Are we holding context block buffer?
61 003654 001003          BNE     3$           ;Br if not
62 003656          DCALL   FREEXT    ;Free context block buffer
63          ;
64          ; Free the Special Device data base
65          ;
66 003664 3$:    DCALL   FRESPD    ;FREE SPECIAL DEVICE DATA BASE
67          ;
68          ; Cancel any pending mark-time requests for this job
69          ;
70 003672          DCALL   CANMKT    ;CANCEL ALL MARK-TIME REQUESTS FOR JOB
71          ;
72          ; Cancel any monitoring requests for this job
73          ;
74 003700          DCALL   MONABT    ;Cancel any monitoring requests
75          ;
76          ; Cancel any pending message requests for this job
77          ;
78 003706 005737 000000G   TST     VMAXMC    ;Is message system included in system?
79 003712 001403   BEQ     14$           ;Br if not
80 003714          DCALL   MSGABT    ;Cleanup message system
81          ;
82          ; Undo Break key connection.
83          ;
84 003722 016100 000000G 14$:  MOV     LBRKCQ(R1),R0 ;GET ADDRESS OF BREAK QUEUE ENTRY
85 003726 001407   BEQ     17$           ;BR IF NONE
86 003730 005061 000000G   CLR     LBRKCQ(R1) ;SAY NO BREAK KEY CONNECTION
87 003734 010146   MOV     R1,-(SP)    ;SAVE JOB INDEX NUMBER
88 003736 010001   MOV     R0,R1      ;GET QUEUE ENTRY ADDRESS FOR QFREE
89 003740 004737 000000G   CALL   QFREE      ;FREE THE QUEUE ENTRY
90 003744 012601   MOV     (SP)+,R1   ;GET BACK JOB NUMBER
91          ;
92          ; Undo TT input completion routine connection
93          ;
94 003746 016100 000000G 17$:  MOV     LTTTCR(R1),R0 ;GET ADDRESS OF QUEUE ENTRY
95 003752 001407   BEQ     1$           ;BR IF NONE
96 003754 005061 000000G   CLR     LTTTCR(R1) ;SAY NO COMPL ROUTINE
97 003760 010146   MOV     R1,-(SP)    ;SAVE JOB INDEX NUMBER
98 003762 010001   MOV     R0,R1      ;GET QUEUE ENTRY ADDRESS FOR QFREE
99 003764 004737 000000G   CALL   QFREE      ;FREE THE QUEUE ENTRY
100 003770 012601   MOV     (SP)+,R1   ;GET BACK JOB NUMBER
101          ;
102          ; Clean up all pending completion routine requests for this job
103          ;
104 003772 004737 004146' 1$:    CALL   CANCEL    ;Cancel all pending completion routines
105          ;
106          ; Close all shared files for this job.
107          ;
108 003776 013704 000000G 5$:    MOV     JCDB,R4    ;GET ADDRESS OF NEXT CDB FOR JOB
109 004002 001405   BEQ     6$           ;BR IF NO MORE
110 004004 013705 000002G   MOV     JCDB+2,R5   ;Get par pointer for CDB
111 004010 004777 000000G   CALL   @CLSCDB     ;CLOSE THE CDB
112 004014 000770   BR      5$         ;SEE IF THERE ARE MORE TO DO
113          ;
114          ; Release any PLAS regions created for the job

```

CLENUP -- Do general cleanup when a job stops

```

115
116 004016 005737 000000G      6$:   TST     VPLAS      ; IS PLAS SUPPORT GENNED INTO SYSTEM?
117 004022 001403              BEQ     13$          ; BR IF NOT
118 004024              OCALL   PLSXIT      ; FREE ANY PLAS REGIONS AND WINDOWS
119
120              ; Release trap control for job
121
122 004032 005037 000000G      13$:  CLR     UTRPAD      ; Undo .TRPSET
123 004036 005037 000000G      CLR     UFPTRP      ; Undo .SFPA
124 004042 105037 000000G      CLRB   FPUUSE      ; Tell system that FPU is not in use
125
126              ; Release any associated shared run-time systems
127
128 004046 005037 000000G      CLR     CURRDB      ; DISASSOCIATE RUN-TIME SYSTEM
129 004052 105037 000000G      CLRB   DOTRMP      ; Disable fast TRAP mapping
130 004056 005002              CLR     R2          ; INIT PAR INDEX
131 004060 005062 000000G      9$:   CLR     RPAR(R2) ; RESET PAR MAPPING INFO
132 004064 005062 000000G      CLR     RPDR(R2)
133 004070 005062 000000G      CLR     RDAR(R2)
134 004074 005062 000000G      CLR     RDDR(R2)
135 004100 062702 000002      ADD     #2,R2      ; ADVANCE INDEX
136 004104 020227 000016      CMP     R2,#2*7    ; DONE ALL?
137 004110 101763              BLOS   9$          ; LOOP IF MORE TO DO
138 004112 042761 000000G 000000G BIC     #$UDSPC,LSW11(R1) ; Clear flag saying user wants I/D-space
139 004120 105737 000000G      TSTB   SR3FLG     ; Does hardware support I/D space?
140 004124 001403              BEQ     10$        ; Br if not
141 004126 042737 000000G 000000G BIC     #USDSPC,SR3MMR ; Disable user D-space
142
143              ; Finished
144
145 004134 012605      10$:  MOV     (SP)+,R5
146 004136 012604      MOV     (SP)+,R4
147 004140 012602      MOV     (SP)+,R2
148 004142 012601      MOV     (SP)+,R1
149 004144 000207      RETURN

```

CANCPL -- Cancel all pending completion routines

```

1          .SBTTL  CANCPL -- Cancel all pending completion routines
2          ;-----
3          ;  CANCPL is called during job exit cleanup to cancel any pending completion
4          ;  routines.  User-mode completion routines are removed from the pending
5          ;  list, system-mode completion routines are forced to be called.
6          ;
7          ;  Inputs:
8          ;  R1 = Job index number.
9          ;
10         004146  010546  CANCPL:  MOV      R5, -(SP)
11         ;
12         ;  Say we are not in a completion routine now
13         ;
14         004150  105037  000000G  1$:      CLRB     CURCP          ; Say not executing in completion rtn now
15         ;
16         ;  If any I/O is in progress for job, suspend execution while we
17         ;  wait for it to finish.
18         ;
19         004154          DISABL          ; ** Disable interrupts **
20         004162  126137  000000G  000000G  CMPB     LIOCNT(R1), NPCCB ; Is any I/O in progress for job?
21         004170  003404          BLE      2$          ; Br if not
22         004172  012700  000000G          MOV     #S$IOWT, R0      ; Suspend job till I/O completes
23         004176  004737  000000G          CALL    QNSPND         ; Suspend execution of job
24         ;
25         ;  See if there are any pending completion routines
26         ;
27         004202          2$:      ENABL          ; ** Enable interrupts **
28         004210  005761  000000G          TST     LCMP(LR1)       ; Is there a pending completion routine?
29         004214  001005          BNE     3$          ; Br if yes
30         004216  126137  000000G  000000G  CMPB     LIOCNT(R1), NPCCB ; Is all I/O finished too?
31         004224  003351          BGT     1$          ; If not, then wait till it is
32         004226  000403          BR      5$          ; All I/O is finished and no compl rtns pend
33         ;
34         ;  There is at least one pending completion routine.
35         ;  Now call SCHED to force pending system-mode routines to be run
36         ;
37         004230  004737  000000G  3$:      CALL    SCHED          ; Force completion routine execution
38         ;
39         ;  Go back and make sure all completion routines have been taken care of
40         ;
41         004234  000745          BR      1$          ; Loop till all completion routines finished
42         ;
43         ;  All I/O has terminated and there are no pending completion routines.
44         ;  Free any cache control blocks that were pending when job aborted.
45         ;
46         004236  005737  000000G  5$:      TST     JOBCCB          ; Any pending cache control blocks
47         004242  001406          BEQ     9$          ; Br if not
48         004244  013705  000000G  6$:      MOV     JOBCCB, R5      ; Get pointer to 1st pending control block
49         004250  001737          BEQ     1$          ; Br if none left pending
50         004252  004777  000000G          CALL    @CSHFIN       ; Free the cache control block
51         004256  000772          BR      6$          ; Loop to free any others
52         ;
53         ;  Finished
54         ;
55         004260  012605  9$:      MOV     (SP)+, R5
56         004262  000207          RETURN

```

LOGOFF -- Log off a job

```

1          SBTTL LOGOFF -- Log off a job
2          ;-----
3          ; LOGOFF is jumped to log off the current job.
4          ; All tables for the job are reset and then the scheduler is entered
5          ; to look for another job to run.
6          ;
7          ; Inputs:
8          ; R1 = Job number of job being logged off.
9          ;
10         004264 032761 000000G 000000G LOGOFF: BIT    #$DETCH,LSW(R1) ; IS THIS A DETACHED JOB LOGGING OFF?
11         004272 001402          BEQ    26$          ; Br if not
12         004274 000137 004754'          JMP    4$
13         ;
14         ; Wait for all TT output for the job to complete.
15         ;
16         004300 032761 000000G 000000G 26$: BIT    #$VNOTT,LSW(R1) ; IS JOB CONNECTED TO TERMINAL?
17         004306 001045          BNE    12$          ; BR IF NOT (DON'T WAIT FOR OUTPUT)
18         004310 013704          MOV    SYTIMH,R4      ; Get high-order time-of-day
19         004314 013705          MOV    SYTIML,R5      ; Get low-order time-of day
20         004320 062705 000454          ADD    #5.*60.,R5     ; Add time to allow for logoff message
21         004324 005504          ADC    R4            ; Propogate carry
22         004326 023704 000000G          10$: CMP    SYTIMH,R4      ; Have we waited long enough?
23         004332 101033          BHI    12$          ; Br if yes
24         004334 103403          BLO    16$          ; Br if not
25         004336 023705 000000G          CMP    SYTIML,R5     ; Compare low-order time
26         004342 103027          BHIS   12$          ; Br if have waited long enough
27         004344 116103 000000G          16$: MOVB   LNPRIM(R1),R3 ; GET PRIMARY LINE NUMBER
28         004350 032763 000000G 000000G BIT    #$DILUP,LSW(R3) ; IS PRIMARY LINE STILL LOGGED ON?
29         004356 001421          BEQ    12$          ; BR IF NOT
30         004360 042763 000000G 000000G BIC    #$CTRLS,LSW3(R3); CLEAR CTRL-S SUSPEND
31         004366 004777 000000G          CALL   @TRNSTR      ; MAKE SURE OUTPUT IS GOING
32         004372 026161 000000G 000000G CMP    LOTSPC(R1),LOTSIZ(R1); ANY CHARS LEFT TO TRANSMIT?
33         004400 001352          BNE    10$          ; WAIT FOR ALL OUTPUT TO COMPLETE
34         004402 032761 000000C 000000G BIT    #<#$DHF1!#$DHF2>,LSW10(R1); DH11 buffer being transmitted?
35         004410 001346          BNE    10$          ; Wait for DH11 to finish
36         004412 032761 000000G 000000G BIT    #$XCHAR,LSW3(R1); Wait for last char to go out
37         004420 001342          BNE    10$
38         ;
39         ; See if this is a primary or virtual line logging off.
40         ;
41         004422 020127 000000G          12$: CMP    R1,#LSTPL      ; PRIMARY OR VIRTUAL LINE?
42         004426 003466          BLE    1$          ; BR IF PRIMARY LINE
43         ;
44         ; Log off a virtual line.
45         ;
46         004430 105337 000000G          DECB   PVON          ; Count # primary & virtual lines on
47         004434 010102          MOV    R1,R2        ; Save virtual line number
48         004436 113703 000000G          MOVB   SPIJ,R3       ; Get # of process that started us
49         004442 001422          BEQ    22$          ; Br if unknown
50         004444 032763 000000G 000000G BIT    #$DISCN,LSW(R3) ; Is that process terminating now?
51         004452 001016          BNE    22$          ; Br if yes -- Switch to primary
52         004454 016100 000000G          MOV    LNPRIM(R1),R0 ; Get our primary process #
53         004460 020300          CMP    R3,R0        ; Switching back to primary process?
54         004462 001412          BEQ    22$          ; Br if yes
55         004464 016000 000000G          MOV    LSECPT(R0),R0 ; Get pointer to subprocess # table
56         004470 005005          CLR    R5           ; Init subprocess #
57         004472 005205          24$: INC   R5        ; Increment subprocess #

```

LOGOFF -- Log off a job

```

58 004474 020527 000000G      CMP      R5,#MAXSEC      ;Checked all subprocess entries?
59 004500 101003              BHI      22$             ;Br if yes -- Switch back to primary
60 004502 120320              CMPB     R3,(R0)+        ;Search for originating process in table
61 004504 001372              BNE      24$             ;Loop till found
62 004506 000407              BR       23$             ;Found subprocess to switch back to (R5=#)
63 004510 005005              22$:    CLR      R5             ;Say we are switching to primary process
64 004512 016103 000000G      MOV      LNPRIM(R1),R3   ;Get primary line number
65 004516 032763 000000G 000000G  BIT      #$DILUP,LSW(R3) ;Is the primary line still logged on?
66 004524 001423              BEQ      2$              ;Br if not
67 004526 016203 000000G      23$:    MOV      LNPRIM(R2),R3 ;Get primary process index
68 004532 120163 000000G      CMPB     R1,LNMAP(R3)   ;Are we running on line being logged off?
69 004536 001003              BNE      6$              ;Br if not -- Someone must have killed us
70 004540                      OCALL    LINSWT          ;Switch back to initiating process
71                               ;
72                               ; At this point R3 has the primary line number.
73                               ; R1 & R2 have the virtual line number.
74                               ; Remove the virtual line from the primary line's ownership table.
75                               ;
76 004546 005004              6$:    CLR      R4             ;CHECK 1ST VIRTUAL LINE ENTRY
77 004550 016305 000000G      MOV      LSECT(R3),R5   ;POINT TO TABLE OF VIRTUAL LINE #'S
78 004554 120225              5$:    CMPB     R2,(R5)+        ;IS THIS THE VIRTUAL LINE ENTRY?
79 004556 001405              BEQ      3$              ;BR IF YES
80 004560 005204              INC      R4             ;CHECK NEXT ENTRY
81 004562 020427 000000G      CMP      R4,#MAXSEC     ;CHECKED ALL?
82 004566 002772              BLT      5$              ;BR IF MORE TO CHECK
83 004570 000401              BR       2$              ;STRANGE -- COULDN'T FIND VIRTUAL LINE
84 004572 105045              3$:    CLRB     -(R5)        ;SAY VIRTUAL LINE NOT OWNED BY PRIMARY LINE
85 004574 005062 000000G      2$:    CLR      LNPRIM(R2)   ;REMOVE PRIMARY LINE NUMBER FOR VIRTUAL LINE
86 004600 010201              MOV      R2,R1          ;GET BACK VIRTUAL LINE NUMBER
87 004602 000464              BR       4$              ;
88                               ;
89                               ; Disconnect a primary line.
90                               ; This forces the disconnect of any associated virtual lines.
91                               ;
92 004604 105337 000000G      1$:    DECB     NUMON         ;# PRIMARY LINES ON
93 004610 105337 000000G      DECB     PVON           ;# PRIMARY & VIRTUAL LINES ON
94 004614 013700 000004G      MOV      EMTBLK+4,R0    ;Get time to drop DTR
95 004620 001002              BNE      21$            ;Br if time parameter specified
96 004622 013700 000000G      MOV      VOFFTM,R0      ;Get sysgen specified time
97 004626 010061 000000G      21$:    MOV      R0,LOFFTM(R1) ;Drop DTR after this much time
98 004632 016161 000000G 000000G  MOV      ILSW2(R1),LSW2(R1);Reset line status flags
99 004640 010103              MOV      R1,R3          ;SAVE PRIMARY LINE NUMBER
100 004642 010161 000000G      MOV      R1,LNMAP(R1)   ;REASSOCIATE TS LINE WITH PRIMARY LINE
101                               ;
102                               ; If this is an autobaud line, start time which will reset the line
103                               ; speed to 9600 baud after a short delay to get the logoff message out.
104                               ;
105 004646 032761 000000G 000000G  BIT      #$AUTO,ILSW2(R1);Is autobaud selected for this line?
106 004654 001406              BEQ      14$            ;Br if not
107 004656 012761 000012 000000G  MOV      #10,LABTIM(R1) ;Start autobaud timer for this line
108 004664 052761 000000G 000000G  BIS      #$NABRS,LSW9(R1);Set flag saying we need to reset the speed
109                               ;
110                               ; Reset all character translation for the line
111                               ;
112 004672 016104 000000G      14$:    MOV      LCXTBL(R1),R4 ;Get pointer to lines translation table
113 004676 001401              BEQ      25$            ;Br if no translation table
114 004700 005014              CLR      (R4)           ;Say no translation in effect

```

```

115 ;
116 ; Force disconnect of all associated virtual lines.
117 ;
118 004702 005004 25#: CLR R4 ; START WITH 1ST VIRTUAL LINE
119 004704 016305 000000G MOV LSECPT(R3),R5 ; GET ADDRESS OF VIRTUAL LINE TABLE
120 004710 020427 000000G 9#: CMP R4,#MAXSEC ; CHECKED ALL VIRTUAL LINE ENTRIES
121 004714 002016 BGE 7# ; BR IF YES
122 004716 111501 MOVB (R5),R1 ; GET VIRTUAL LINE NUMBER
123 004720 001411 BEQ 8# ; BR IF NO ASSOCIATED VIRTUAL LINE HERE
124 004722 032761 000000G 000000G BIT #LDFCF,LSW9(R1); IS JOB DOING LOGOFF PROCESSING NOW?
125 004730 001005 BNE 8# ; BR IF YES
126 004732 052761 000000G 000000G BIS #DISCN,LSW(R1) ; FORCE LOG OFF OF THIS VIRTUAL LINE
127 004740 004737 000000G CALL FORCEX ; FORCE ITS EXECUTION
128 004744 105025 8#: CLRB (R5)+ ; CLEAR ENTRY IN PRIMARY LINE'S TABLE
129 004746 005204 INC R4 ; CHECK NEXT ENTRY
130 004750 000757 BR 9#
131 004752 010301 7#: MOV R3,R1 ; GET BACK PRIMARY LINE NUMBER
132 ;
133 ; Release any display windows for job
134 ;
135 004754 005737 000000G 4#: TST VMXWIN ; Is window support included in system?
136 004760 001403 BEQ 20# ; Br if not
137 004762 OCALL WINREL ; Release windows for job
138 ;
139 ; Do cleanup of PLAS regions
140 ;
141 004770 005737 000000G 20#: TST VPLAS ; Is PLAS support included?
142 004774 001403 BEQ 17# ; Br if not
143 004776 OCALL PLSOFF ; Do PLAS cleanup
144 ;
145 ; Say we are no other job's parent
146 ;
147 005004 012702 000000G 17#: MOV #LSTSL,R2 ; Get # of last job
148 005010 026201 000000G 18#: CMP LPARNT(R2),R1 ; Are we that job's parent?
149 005014 001002 BNE 19# ; Br if not
150 005016 005062 000000G CLR LPARNT(R2) ; No longer its parent
151 005022 162702 000002 19#: SUB #2,R2 ; Check other jobs
152 005026 003370 BGT 18#
153 ;
154 ; Free all memory assigned to the job.
155 ;
156 005030 016102 000000G MOV LBASE(R1),R2 ; GET BASE PAGE # ASSIGNED TO JOB
157 005034 016100 000000G MOV LNBLKS(R1),RO ; GET # PAGES ASSIGNED TO JOB
158 005040 004737 000000G CALL FREMEM ; RELEASE THE MEMORY SPACE
159 005044 005061 000000G CLR LNBLKS(R1) ; SAY ALL MEMORY DEASSIGNED
160 005050 005061 000000G CLR LMEMIN(R1)
161 005054 005061 000000G CLR LBASE(R1)
162 ;
163 ; Now clear line tables.
164 ;
165 005060 004737 000000G CALL DEQ ; REMOVE JOB FROM RUN QUEUE ** DISABLE **
166 005064 005061 000000G CLR LSW(R1) ; CLEAR LINE STATUS TABLES
167 005070 005061 000000G CLR LSW4(R1)
168 005074 005061 000000G CLR LSW5(R1)
169 005100 042761 000000C 000000G BIC #^C<#1STLG>,LSW6(R1); CLEAR ALL BUT 1ST-LOGON FLAG
170 005106 005061 000000G CLR LSW7(R1)
171 005112 012700 000000C MOV #<$DEAD!$HARD!$CARUP!$XCHAR>,RO ; FLAGS TO PRESERVE IN LSW3

```


LOGOFF -- Log off a job

```
172 005116 005100          CDM      RO          ; MASK TO CLEAR ALL OTHERS
173 005120 040061 000000G  BIC      RO,LSW3(R1)
174 005124 005061 000000G  CLR      LPARNT(R1)      ; Say we have no parent job
175 005130 005061 000000G  CLR      LNTPAC(R1)      ; Say no user-defined activation characters
176 005134 105337 000000G  DECB    TOTON           ; TOTAL # JOBS
177 005140 120137 000000G  CMPB    R1,PMUSER        ; ARE WE DOING A PERFORMANCE ANALYSIS?
178 005144 001004          BNE      13$             ; BR IF NOT
179 005146 005037 000000G  CLR      PMUSER          ; SAY WE ARE DONE
180 005152 005037 000000G  CLR      PMRUN
181
182          ; Job is logged off.
183          ; Enter scheduler to find another one to run.
184
13$: 185 005156 012706 000000G  MOV      #SS,SP          ; SWITCH TO SYSTEM STACK
186 005162 105037 000000G  CLRB    CORUSR           ; NO USER RUNNING
187 005166          ENABL           ; ** ENABLE **
188 005174 105037 000000G  CLRB    MAPUSR           ; SAY MEMORY MAPPING NOT SET UP FOR ANY JOB
189 005200 000137 000000G  JMP      EXEC            ; GO LOOK FOR ANOTHER JOB TO RUN
```

TSXTX -- Trap Handler

```

1          .SBTTL  TSXTX  -- Trap Handler
2          ;-----
3          ; TSXTX is entered from the resident routines that catch traps to 4 and 10.
4          ;
5          ; Inputs:
6          ;   The following items are on the stack:
7          ;   0(SP) = Saved R5
8          ;   2(SP) = Saved R4
9          ;   4(SP) = Trap PC
10         ;   6(SP) = Trap PS
11         ;   R5   = Trap code (1==>Trap 4, 2==>Trap 10)
12         ;
13         ; See if trap occurred in user or kernel mode.
14         ;
15 005204 016604 000004 TSXTX:  MOV     4(SP),R4      ;GET ADDRESS OF TRAP
16 005210 032766 000000G 000006  BIT     #UMODE,6(SP)  ;DID TRAP OCCUR IN USER OR KERNEL MODE?
17 005216 001036          BNE     1$          ;BR IF TRAP IN USER MODE
18 005220 012705 000016  MOV     #16,R5        ;SET KERNEL-MODE-TRAP ERROR CODE
19 005224 105737 000000G  TSTB   CORUSR        ;IS A JOB RUNNING NOW?
20 005230 001416          BEQ     6$          ;IF NOT THEN TRAP MUST BE IN SYSTEM
21 005232 005737 000000G  TST    CURVC         ;IN REAL-TIME INTERRUPT SERVICE ROUTINE?
22 005236 001013          BNE     6$          ;BR IF YES
23 005240 105737 000000G  TSTB   FRKPRI        ;IN A FORK ROUTINE?
24 005244 001010          BNE     6$          ;BR IF YES
25 005246 105737 000000G  TSTB   INTLVL        ;ARE WE RUNNING AT INTERRUPT LEVEL?
26 005252 002005          BGE     6$          ;BR IF YES
27 005254 105737 000000G  TSTB   VDMKTP        ;SHOULD WE ALWAYS CRASH ON KERNEL TRAP?
28 005260 001002          BNE     6$          ;BR IF YES
29 005262 000137 005546'  JMP    TRPCOM        ;ALWAYS ABORT IF KERNEL MODE TRAP
30         ;
31         ; We had a trap within a critical system routine.
32         ; Cause a system crash.
33         ;
34 005266 010437 000000G 6$:   MOV     R4,DIEARG    ;Set address of trap location
35 005272 012737 000170 000000G  MOV     #EM$KTP,DIEMSG ;Set address of abort message
36 005300 012605          MOV     (SP)+,R5      ;Restore R5
37 005302 012604          MOV     (SP)+,R4      ;Restore R4
38 005304 062706 000004  ADD     #4,SP         ;Pop trap PC and PS
39 005310 004737 000000G  CALL   SYSHL1        ;Die without changing TRPAR5
40         ;
41         ; Trap in user mode.
42         ; See if user was executing a real-time interrupt service routine.
43         ;
44 005314 005737 000000G 1$:   TST     CURVC         ;ARE WE EXECUTING IN A REAL-TIME INT ROUTINE?
45 005320 001407          BEQ     7$          ;BR IF NOT
46 005322          DIE     #EM$RIT,R4  ;TRAP IN REAL-TIME INTERRUPT SERVICE ROUTINE
47         ;
48         ; See if a stack overflow occurred.
49         ;
50 005340 004737 000000G 7$:   CALL   CHKUSP        ;IS USER STACK POINTER OK?
51 005344 103002          BCC     2$          ;BR IF OK
52 005346 000137 005732'  JMP    ABORT         ;ABORT JOB
53         ;
54         ; See if user did a .TRPSET
55         ;
56 005352 005737 000000G 2$:   TST     UTRPAD        ;DID USER DO A .TRPSET?
57 005356 001010          BNE     3$          ;BR IF YES

```

TSXTX -- Trap Handler

```

58 005360 012704 000004      MOV      #4,R4          ;SET TRAP VECTOR ADDRESS TO 4
59 005364 020527 000001      CMP      R5,#1         ;DID WE GET TRAP TO 4 OR 10?
60 005370 001466             BEQ      TRPCOM        ;BR IF TRAP 4
61 005372 012704 000010      MOV      #10,R4       ;SET TRAP VECTOR ADDRESS TO 10
62 005376 000463             BR       TRPCOM        ;GO DO COMMON TRAP HANDLING
63                               ;
64                               ;   User did a .TRPSET
65                               ;
66 005400 004737 000000G     3$:      CALL     CHKABT          ;MAKE SURE JOB HASN'T BEEN ABORTED
67                               ;   Move trap PC & PS from kernel stack to user's stack.
68 005404 106506             MFPD     SP            ;GET USER'S SP
69 005406 012604             MOV      (SP)+,R4
70 005410 016646 000006     MOV      6(SP),-(SP)    ;PUSH TRAP PS ON USER'S STACK
71 005414 106644             MTPD     -(R4)
72 005416 016646 000004     MOV      4(SP),-(SP)    ;PUSH TRAP PC ON USER'S STACK
73 005422 106644             MTPD     -(R4)
74 005424 010446             MOV      R4,-(SP)      ;STORE UPDATED USER SP
75 005426 106606             MTPD     SP
76                               ;
77                               ;   Enter user's .trpset routine.
78                               ;
79 005430 012704 000000C     MOV      #UMODE!UPMODE,R4;GET USER-MODE PS
80 005434 020527 000002     CMP      R5,#2         ;WAS TRAP TO 4 OR 10?
81 005440 001001             BNE     4$            ;BR IF TRAP TO 4
82 005442 005204             INC      R4           ;SET C-FLAG IN PS TO SIGNAL TRAP TO 10
83 005444 010466 000006     4$:      MOV      R4,6(SP)    ;STORE NEW PS OVER TRAP PS
84 005450 013766 000000G 000004  MOV      UTRPAD,4(SP)   ;SET ADDRESS OF USER'S ROUTINE
85 005456 005037 000000G     CLR      UTRPAD        ;RESET .TRPSET
86 005462 012605             MOV      (SP)+,R5
87 005464 012604             MOV      (SP)+,R4
88 005466 000002             RTI                   ;ENTER USER'S TRAP ROUTINE

```

TSXTX -- Trap Handler

```

1          ; -----
2          ; TRPBPT is entered when a breakpoint (BPT) trap occurs to location 14
3          ; and the system debugger is not connected to the user's job.
4          ;
5          ; Inputs:
6          ; R4 = Job index number
7          ; Information that has been pushed on the stack:
8          ; PS-PC-R4
9          ;
10         TRPBPT:
11         ;
12         ; If breakpoint occurred in TSKMON, enter debugger
13         ;
14         005470 032764 000000G 000000G      BIT    $#INKMN,LSW4(R4); Is kmon running?
15         005476 001004                      BNE    1$          ; Br if yes
16         ;
17         ; See if user provided a PC in location 14
18         ;
19         005500 106537 000014                MFPD   @#14          ; Get contents of loc 14 from user's job
20         005504 005726                      TST    (SP)+        ; Did user provide a PC for trap?
21         005506 001011                      BNE    9$          ; Br if yes
22         ;
23         ; User did not provide a PC for trap.
24         ; Enter system debugger.
25         ;
26         005510 105737 000000G              1$:   TSTB   VDBFLG      ; Is system debugger included in system?
27         005514 001406                      BEQ    9$          ; Br if not
28         005516 052764 000000G 000000G      BIS    $#DBGBK,LSW9(R4); Set flag to force entry to debugger
29         005524 012604                      MOV    (SP)+,R4    ; Restore R4
30         005526 000137 000000G              JMP    SYSXIT      ; Exit through routine that will test flag
31         ;
32         ; User provided a PC for BPT trap, enter his routine.
33         ;
34         005532 010546                      9$:   MOV    R5,-(SP)
35         005534 012705 000012                MOV    #12,R5      ; Get error code
36         005540 012704 000014                MOV    #14,R4      ; Get trap location
37         005544 000400                      BR     TRPCOM      ; Enter trap processing routine

```

add 4

```

1
2 ; -----
3 ; General trap handling routine.
4 ;
5 ; Inputs:
6 ;   R4 = Address of trap vector.
7 ;   R5 = Error message number.
8 ;
9 ; If user provided a PC & PS in the user-mode trap vector, then we enter
10 ; his routine. Otherwise we abort the job.
11 005546 010146 TRPCOM: MOV     R1,-(SP)
12 ;
13 ; At this point the stack contains PS, PC, R4, R5, and R1
14 ;
15 005550 032766 000000G 000010      BIT     #UMODE,B.(SP) ;DID TRAP OCCUR IN USER OR KERNEL MODE?
16 005556 001005      BNE     4$          ;Br if in user mode
17 005560 032737 000000G 000000G      BIT     #D$RUN,D.FLAG ;Is the debugger program running now?
18 005566 001046      BNE     3$          ;Br if yes -- reenter the debugger
19 005570 000456      BR      2$          ;Trap occurred within the system
20 005572 004737 000000G      4$:    CALL    CHKUSP      ;SEE IF USER STACK POINTER IS OK
21 005576 103453      BCS     2$          ;BR IF INVALID STACK POINTER
22 005600 113701 000000G      MOVB   CORUSR,R1    ;GET CURRENT JOB NUMBER
23 005604 032761 000000G 000000G      BIT     #$INKMN,LSW4(R1);DID TRAP OCCUR IN KMON?
24 005612 001045      BNE     2$          ;ALWAYS ABORT IF YES
25 005614 106524      MFPD   (R4)+       ;GET PC FROM USER SPACE
26 005616 005726      TST    (SP)+       ;DID USER PROVIDE PC?
27 005620 001430      BEQ    1$          ;BR IF NOT -- ABORT THE JOB
28 ;
29 005622 004737 000000G      ; User supplied a PC.
30      CALL    CHKABT      ;MAKE SURE JOB HASN'T BEEN ABORTED
31 005626 106506      ; Move trap PC & PS from kernel stack to user stack.
32 005630 012605      MFPD   SP          ;GET USER'S SP
33 005632 016646 000010      MOV    (SP)+,R5
34 005636 106645      MOV    8.(SP),-(SP) ;GET PS FROM TRAP
35 005640 016646 000006      MTPD   -(R5)       ;PUSH ONTO USER'S STACK
36 005644 106645      MOV    6(SP),-(SP) ;GET PC FROM TRAP
37 005646 010546      MTPD   -(R5)       ;PUSH ONTO USER'S STACK
38 005650 106606      MOV    R5,-(SP)   ;UPDATE USER'S SP
39      MTPD   SP
40 ;
41 ; Enter user's trap routine
42 ;
43 005652 106514      MFPD   (R4)        ;GET PS FROM TRAP VECTOR
44 005654 052716 000000C      BIS    #UMODE!UPMODE,(SP);MAKE SURE USER-MODE IS SET
45 005660 012666 000010      MOV    (SP)+,B.(SP) ;STORE OVER TRAPPED PS
46 005664 106544      MFPD   -(R4)       ;GET PC FROM TRAP VECTOR
47 005672 012601 000006      MOV    (SP)+,6(SP) ;STORE OVER TRAPPED PC
48 005674 012605      MOV    (SP)+,R1
49 005676 012604      MOV    (SP)+,R5   ;RESTORE REGISTERS
50 005700 000002      MOV    (SP)+,R4
51      RTI          ;ENTER USER'S TRAP ROUTINE
52 ;
53 ; User did not specify a trap routine.
54 ; If program is running with the debugger, enter it.
55 005702 005004      1$:    CLR     R4          ;Set flag saying trap was not in debugger
56 005704 113701 000000G      3$:    MOVB   CORUSR,R1    ;Get job index number
57 005710 032761 000000G 000000G      BIT     #$DEBUG,LSW9(R1);Is program running with debugger?

```

```
58 005716 001403          BEQ      2$          ;Br if not
59 005720 012601          MOV      (SP)+,R1      ;Recover R1
60 005722 000137 000000G  JMP      DBGTRP       ;Enter debugger
61                          ;
62                          ; Abort the job
63                          ;
64 005726 016604 000006  2$:      MOV      6(SP),R4      ;GET PC WHERE TRAP OCCURED
65                          ;
66                          ; Enter at ABORT to abort the current job.
67                          ; Inputs:
68                          ; R4 = Address of aborted instruction.
69                          ; R5 = Abort error code.
70                          ;
71 005732 010437 000000G  ABORT:  MOV      R4,ABRTAD      ;SAVE ADDRESS OF ABORT
72 005736 110537 000000G      MOVB     R5,ABRTCD      ;SAVE ABORT ERROR CODE
73 005742 004737 002760'      CALL     STOP          ;TERMINATE THE JOB
```

```

1          .SBTTL  FPTRPX -- Floating point trap routine
2          ;-----
3          ; FPTRPX processed Floating Point Unit (FPU) exception interrupts.
4          ; This routine is jumped to when we are about to exit from an interrupt
5          ; back to user mode.
6          ; On entry, the current job index number is in R1.
7          ; The saved contents of R1 are on the top of the stack followed by the
8          ; interrupt PC and PS ready to do an RTI.
9          ;
10         005746 052737 000000G 000000G FPTRPX: BIS      #UPMODE,@#PSW ;Make sure previous mode = user
11         005754 042761 000000G 000000G        BIC      #$FPUEX,LSW(R1) ;Reset FPU exception flag for job
12         005762 012601                MOV      (SP)+,R1 ;Recover R1 contents
13         005764 023727 000000G 000001        CMP      UFPTRP,#1 ;Did user do a .SFPA?
14         005772 101004                BHI      1$ ;Br if yes
15         ;
16         ; User did not do a .SFPA, Abort the job.
17         ;
18         005774 012705 000005                MOV      #5,R5 ;Set abort code
19         006000 011604                MOV      (SP),R4 ;Get address of aborted instruction
20         006002 000753                BR       ABORT ;Abort the job
21         ;
22         ; User gets trap control.
23         ; Push trap PC & PS onto user's stack
24         ;
25         006004 004737 000000G                1$:      CALL     CHKUSP ;IS USER'S STACK POINTER OK?
26         006010 103002                BCC     2$ ;BR IF OK
27         006012 011604                MOV     (SP),R4 ;GET ADDRESS WHERE TRAP OCCURED
28         006014 000746                BR      ABORT ;ABORT THE JOB
29         006016 010346                2$:      MOV     R3,-(SP) ;GET A WORK REGISTER
30         006020 106506                MFPD   SP ;GET USER'S STACK POINTER
31         006022 012603                MOV     (SP)+,R3
32         006024 016646 000004                MOV     4(SP),-(SP) ;GET TRAP PS
33         006030 106643                MTPD   -(R3) ;PUSH ONTO USER'S STACK
34         006032 016646 000002                MOV     2(SP),-(SP) ;GET TRAP PC
35         006036 106643                MTPD   -(R3) ;PUSH ONTO USER'S STACK
36         ;
37         ; See if hardware has a FPU
38         ;
39         006040 032737 000000G 000000G                BIT     #CW$FPU,CONFIG ;DOES HARDWARE HAVE AN FPU UNIT?
40         006046 001407                BEQ     3$ ;BR IF NOT
41         006050 170346                STST   -(SP) ;GET FPU STATUS
42         006052 106663 177774                MTPD   -4(R3) ;MOVE FPU STATUS ONTO USER'S STACK
43         006056 106663 177776                MTPD   -2(R3)
44         006062 162703 000004                SUB     #4,R3 ;UPDATE USER'S SP
45         ;
46         ; Reset user's SP
47         ;
48         006066 010346                3$:      MOV     R3,-(SP) ;NEW USER SP
49         006070 106606                MTPD   SP ;RESET USER SP
50         006072 012603                MOV     (SP)+,R3 ;RESTORE WORK REGISTER
51         ;
52         ; Enter user's trap routine
53         ; Note, when user's trap routine does an RTI, it will transfer
54         ; control to the point we would have exited to if the FPU trap hadn't
55         ; have occurred.
56         ;
57         006074 013716 000000G                MOV     UFPTRP,(SP) ;SET PC FOR TRAP ROUTINE

```

58	006100	012766	000000C	000002	MOV	#UMODE!UPMODE, 2(SP); SET PS
59	006106	012737	000001	000000G	MOV	#1, UFPTRP ; RESET .SFPA TO AVOID REENTRENCY
60	006114	000002			RTI	; ENTER USER'S TRAP ROUTINE


```

1          .SBTTL  CLKRUN -- Clock processing routine
2          ;-----
3          ; CLKRUN is the clock interrupt service routine entered from TSEXC
4          ; running at fork level.
5          ;
6 006116  013703  000000G  CLKRUN:  MOV     TIKCNT,R3      ;GET # CLOCK TICKS THAT HAVE OCCURED
7 006122  005203                4$:      INC     R3              ;CONVERT TO ACTUAL NUMBER (STARTED AT -1)
8 006124  010337  000000G                MOV     R3,CLKCNT      ;ADVANCE ALL TIMERS BY THIS AMOUNT
9          ;
10         ; Keep track of time of day
11         ;
12 006130  004737  006432'          CALL    CLKDAT          ;ADVANCE TIME-OF-DAY AND DATE
13         ;
14         ; Keep track of time used by currently running job (if any)
15         ;
16 006134  113701  000000G                MOVVB   CORUSR,R1      ;GET INDEX # OF CURRENTLY RUNNING JOB
17 006140  001405                BEQ     3$              ;BR IF NO JOB RUNNING NOW
18 006142  063761  000000G 000000G          ADD     CLKCNT,LCPULD(R1);ACCUMULATE RUN-TIME FOR JOB
19 006150  005561  000000G                ADC     LCPUHI(R1)     ;PROPOGATE CARRY
20         ;
21         ; Check on .MRKT and .TIMIO requests
22         ;
23 006154  004737  010264'          3$:     CALL    CKMRKT          ;Check on .MRKT and .TIMIO requests
24         ;
25         ; Check on jobs doing .TWAIT's
26         ;
27 006160  004737  010164'          CALL    CKTWAT          ;Check on jobs doing .TWAIT's
28         ;
29         ; Check on job output buffer scheduling requests
30         ;
31 006164  005737  000000G                TST     NEDSOT         ;Output scheduling needed?
32 006170  001404                BEQ     8$              ;Br if not
33 006172  005037  000000G                CLR     NEDSOT         ;Say output scheduling done
34 006176  004737  011024'          CALL    CKSCHD         ;Do job scheduling for output buffer low
35         ;
36         ; See if we need to do performance monitoring.
37         ;
38 006202  005737  000000G          8$:     TST     PMRUN         ;IS PERFORMANCE MONITORING TO BE DONE?
39 006206  001402                BEQ     2$              ;BR IF NOT
40 006210  004737  010614'          CALL    CLKPM          ;DO PERFORMANCE MONITORING
41         ;
42         ; If we are running on a Professional, call the PI output service
43         ; routine every 20th of a second.
44         ;
45 006214  013700  000000G          2$:     MOV     PIDPTR,R0      ;Are we running on a Pro?
46 006220  001407                BEQ     6$              ;Br if not
47 006222  005337  001340'          DEC     PROTIM         ;Time to call PI driver?
48 006226  003004                BGT     6$              ;Br if not
49 006230  004710                CALL    (R0)           ;Call PI output driver
50 006232  012737  000000G 001340'          MOV     #PROODC,PROTIM ;Reset time counter
51         ;
52         ; Do clock driven processing of serial lines.
53         ; We do this as a lower priority fork request since this processing
54         ; could be lengthy.
55         ;
56 006240  005737  000000G          6$:     TST     NEDCDI         ;Input character processing needed?
57 006244  001417                BEQ     5$              ;Br if not

```

CLKRUN -- Clock processing routine

```

58 006246 105737 000000G          TSTB    CDIFLG          ;Are we still doing input char processing?
59 006252 001014                   BNE     5$             ;Br if yes
60 006254 105237 000000G          INCB    CDIFLG          ;Set flag saying processing is being done
61 006260 004737 000000G          CALL    FRKGET         ;Get a fork request block
62 006264 112764 000000G 000000G  MOVB    #FP$CDI,FQ$PRI(R4);Set low priority for fork request
63 006272 013764 000000G 000000G  MOV     CDIRTN,FQ$RTN(R4);Set address of routine to call
64 006300 004737 000000G          CALL    FORKQ         ;Queue the fork request
65 006304 005737 000000G          5$:    TST     NEDCDO         ;Output character processing needed?
66 006310 001417                   BEQ     7$             ;Br if not
67 006312 105737 000000G          TSTB    CDOFLG         ;Are we still doing output char processing?
68 006316 001014                   BNE     7$             ;Br if yes
69 006320 105237 000000G          INCB    CDOFLG         ;Say output processing being done
70 006324 004737 000000G          CALL    FRKGET         ;Get a fork request block
71 006330 112764 000000G 000000G  MOVB    #FP$CDO,FQ$PRI(R4);Set priority for fork request
72 006336 013764 000000G 000000G  MOV     CDORTN,FQ$RTN(R4);Set address of routine to call
73 006344 004737 000000G          CALL    FORKQ         ;Queue the fork request
74                                     ;
75                                     ; Processing done on 0.1 second frequency.
76                                     ; This is also done by queueing a lower priority fork request.
77                                     ;
78 006350 163737 000000G 000000G  7$:    SUB     CLKCNT,TK1CNT   ;Has 0.1 seconds of time passed?
79 006356 003020                   BGT     1$             ;Br if not
80 006360 063737 000000G 000000G  ADD     TK1VAL,TK1CNT   ;Reset 0.1 counter
81 006366 005237 001342'          INC     TIK01S         ;Say another 0.1 seconds has elapsed
82 006372 003012                   BGT     1$             ;Br if haven't finished last 0.1 sec routine
83 006374 004737 000000G          CALL    FRKGET         ;Get a fork request block
84 006400 112764 000000G 000000G  MOVB    #FP$CK1,FQ$PRI(R4);Set low priority for fork request
85 006406 012764 007172' 000000G  MOV     #CLK01S,FQ$RTN(R4);Set address of routine to be called
86 006414 004737 000000G          CALL    FORKQ         ;Queue the fork request
87                                     ;
88                                     ; Finished clock processing
89                                     ;
90 006420 163737 000000G 000000G  1$:    SUB     CLKCNT,TIKCNT   ;SUBTRACT # CLOCK TICKS ACCOUNTED FOR
91 006426 002233                   BGE     CLKRUN         ;BR IF WE NEED TO CYCLE AGAIN
92 006430 000207                   RETURN                ;FINISHED

```

CLKDAT -- update time of day and date

```

1          .SBTTL  CLKDAT -- update time of day and date
2          ;-----
3          ; CLKDAT is the timer subroutine called to keep track of the current
4          ; time-of-day and date.
5          ;
6          ; Inputs:
7          ;   CLKCNT = # clock ticks to account for.
8          ;
9          ; Outputs:
10         ;   SYTIML & SYTIMH = Updated time of day.
11         ;   SYSDAT = Updated date.
12         ;
13 006432  010146  CLKDAT: MOV      R1,-(SP)
14 006434  010246          MOV      R2,-(SP)
15 006436  010346          MOV      R3,-(SP)
16         ;
17         ; Advance system time counter.
18         ;
19 006440  063737  000000G 000000G      ADD      CLKCNT,SYTIML  ;ADD TO LOW-ORDER WORD
20 006446  005537  000000G          ADC      SYTIMH          ;PROPOGATE CARRY
21         ;
22         ; See if we need to do a date roll-over.
23         ;
24 006452  023737  000000G 000000G      CMP      SYTIMH,DATIMH  ;COMPARE HIGH-ORDER TIME VALUE
25 006460  103465          BLO      9$              ;BR IF NOT UP TO 24 HOURS YET
26 006462  023737  000000G 000000G      CMP      SYTIML,DATIML  ;COMPARE LOW-ORDER TIME
27 006470  103461          BLO      9$              ;BR IF NOT THERE YET
28         ;
29         ; Do a date roll-over.
30         ;
31 006472  163737  000000G 000000G      SUB      DATIML,SYTIML  ;RESET SYSTEM TIMER RELATIVE TO START OF DAY
32 006500  005037  000000G          CLR      SYTIMH
33 006504  013700  000000G          MOV      SYSDAT,R0     ;GET SYSTEM DATE VALUE
34 006510  001451          BEQ      9$              ;BR IF NO DATE WAS ENTERED
35 006512  010003          MOV      R0,R3         ;GET YEAR FIELD
36 006514  042703  177740          BIC      #^C<37>,R3
37 006520  072027  177773          ASH      #-5,R0       ;RIGHT JUSTIFY DAY #
38 006524  010001          MOV      R0,R1
39 006526  042700  177740          BIC      #^C<37>,R0   ;GET DAY # ONLY
40 006532  072127  177773          ASH      #-5,R1       ;RIGHT JUSTIFY MONTH VALUE
41 006536  042701  177740          BIC      #^C<37>,R1   ;GET MONTH VALUE ONLY
42 006542  005200          INC      R0           ;INCREMENT CURRENT DAY NUMBER
43 006544  116102  001321'        MOVVB   MONDAY-1(R1),R2 ;GET # DAYS IN CURRENT MONTH
44 006550  020127  000002        CMP      R1,#2        ;IS THIS FEBRUARY?
45 006554  001004          BNE      5$          ;BR IF NOT
46 006556  032703  000003        BIT      #3,R3        ;IS THIS A LEAP YEAR?
47 006562  001001          BNE      5$          ;BR IF NOT
48 006564  005202          INC      R2           ;SAY FEB HAS 29 DAYS
49 006566  020002          5$:  CMP      R0,R2        ;HAVE WE PASSED LAST DAY IN THIS MONTH?
50 006570  101411          BLOS    6$          ;BR IF NOT
51 006572  012700  000001        MOV      #1,R0        ;RESET DAY # TO 1
52 006576  005201          INC      R1           ;ADVANCE MONTH NUMBER
53 006600  020127  000014        CMP      R1,#12       ;DID WE JUST ADVANCE PAST DECEMBER?
54 006604  101403          BLOS    6$          ;BR IF NOT
55 006606  012701  000001        MOV      #1,R1        ;RESET MONTH TO JANUARY
56 006612  005203          INC      R3           ;ADVANCE YEAR NUMBER (HAPPY NEW YEAR)
57 006614  072027  000005        6$:  ASH      #5,R0     ;POSITION DAY # VALUE

```

CLKDAT -- update time of day and date

```
58 006620 050003          BIS      R0,R3          ;OR INTO YEAR WORD
59 006622 072127 000012   ASH      #10.,R1       ;POSITION MONTH #
60 006626 050103          BIS      R1,R3          ;COMBINE DATE VALUES
61 006630 010337 000000G   MOV      R3,SYSDAT     ;SAVE UPDATED DATE VALUE
62
63                       ; Finished
64                       ;
65 006634 012603          9#:    MOV      (SP)+,R3
66 006636 012602          MOV      (SP)+,R2
67 006640 012601          MOV      (SP)+,R1
68 006642 000207          RETURN
```

CLKJOB -- check time slice job status

```

1          .SBTTL  CLKJOB -- check time slice job status
2
3          ; -----
4          ; CLKJOB is the timer subroutine called every 0.1 seconds to check for
5          ; time-slice expiration of the currently running job.
6          ;
7          ;
8          ;
9          ;
10         ;
11         ;
12         ;
13         ;
14         ;
15         ;
16         ;
17         ;
18         ;
19         ;
20         ;
21         ;
22         ;
23         ;
24         ;
25         ;
26         ;
27         ;
28         ;
29         ;
30         ;
31         ;
32         ;
33         ;
34         ;
35         ;
36         ;
37         ;
38         ;
39         ;
40         ;
41         ;
42         ;
43         ;
44         ;
45         ;
46         ;
47         ;
48         ;
49         ;
50         ;
51         ;
52         ;
53         ;
54         ;
55         ;
56         ;
57         ;

```

6	006644	010146		CLKJOB: MOV	R1,-(SP)	
7	006646	010246			MOV	R2,-(SP)
11	006650	113701	000000G		MOV	CORUSR,R1 ;GET INDEX # FOR CURRENTLY RUNNING JOB
12	006654	001543			BEQ	3\$;BR IF NO JOB RUNNING NOW
16	006656	005261	000000G		INC	LQUAN(R1) ;Increment time quantum for job
21	006662	016100	000000G		MOV	LQUAN(R1),RO ;Get current time quantum for job
22	006666	116102	000000G		MOV	BPRI(R1),R2 ;Get job's execution priority
23	006672	120237	000000G		COMP	R2,VPRIHI ;Is this a high priority (real time) job?
24	006676	103412			BLO	10\$;Br if not
25	006700	020037	000000G		COMP	RO,VQUANO ;Have we exceeded QUANO time?
26	006704	101527			BLOS	3\$;Br if not
27	006706	005737	000000G		TST	VQUANO ;Are we doing time slicing for real-time jobs?
28	006712	001524			BEQ	3\$;Br if not
29	006714	004037	000000G		JSR	RO,QUNSIG ;Signal that QUANO expired
30	006720	000000G			.WORD	\$SGQ0
31	006722	000411			BR	11\$;Requeue the job at the tail of the list
32	006724	120237	000000G	10\$:	COMP	R2,VPRILO ;Is this a low priority job?
33	006730	101015			BHI	12\$;Br if not
34	006732	020037	000000G		COMP	RO,VQUAN3 ;Exceeded low priority quantum?
35	006736	101512			BLOS	3\$;Br if not
36	006740	004037	000000G		JSR	RO,QUNSIG ;Signal that QUAN3 has expired
37	006744	000000G			.WORD	\$SGQ3
42	006746	016100	000000G	11\$:	MOV	LSTATE(R1),RO ;Get job's current execution state
43	006752	004737	000000G		CALL	ENQTL ;Requeue job at tail of execution queue
44	006756	005061	000000G		CLR	LQUAN(R1) ;Reset job time quantum
45	006762	000500			BR	3\$
51	006764	005761	000000G	12\$:	TST	LITIME(R1) ;Is job interactive?
52	006770	001407			BEQ	2\$;Br if not
53	006772	005361	000000G		DEC	LITIME(R1) ;Reduce time remaining for job
54	006776	001004			BNE	2\$;Br if still interactive
55	007000	004037	000000G		JSR	RO,QUNSIG ;Signal that QUAN1 has expired
56	007004	000000G			.WORD	\$SGQ1 ;Check QUAN1 signal flag
57	007006	000464			BR	6\$;Now schedule job as compute bound

CLKJOB -- check time slice job status

```

58      ;
59      ;   Check for job quantum expiration.
60      ;
61 007010 016100 000000G 2$:   MOV     LQUAN(R1),RO
62      ;
63      ;   See if this job is currently running in a high-priority state.
64      ;
65 007014 026127 000000G 000000G   CMP     LSTATE(R1),#S##HIP;Is job in high-priority state now?
66 007022 101050          BHI     4$           ;Br if not
67      ;
68      ;   Don't do time-slicing for real-time jobs.
69      ;
70 007024 026127 000000G 000000G   CMP     LSTATE(R1),#S##RT;Is job in high-priority real-time state?
71 007032 101454          BLOS    3$           ;Br if yes -- skip time-quantum checking
72      ;
73      ;   Job is running in a high-priority state.
74      ;   See if job is interactive.
75      ;
76 007034 005761 000000G          TST     LITIME(R1)      ;Is this an interactive job?
77 007040 001007          BNE     5$           ;Br if yes
78 007042 020037 000000G          CMP     RO,VQUN1A      ;Time to requeue as compute bound?
79 007046 101446          BLOS    3$           ;Br if not
80 007050 004037 000000G          JSR     RO,QUNSIG      ;Signal that QUAN1A expired
81 007054 000000G          .WORD   $SQQ1A
82 007056 000440          BR      6$           ;Schedule as compute bound job
83      ;
84      ;   Job is "interactive"
85      ;
86 007060 026127 000000G 000000G 5$:   CMP     LSTATE(R1),#S##HICP;High priority interactive?
87 007066 103011          BHIS    9$           ;Br if not
88 007070 020037 000000G          CMP     RO,VQUN1C      ;Time to drop to lower level?
89 007074 101406          BLOS    9$           ;Br if not
90 007076 004037 000000G          JSR     RO,QUNSIG      ;Signal that QUAN1C expired
91 007102 000000G          .WORD   $SQQ1C
92 007104 012700 000000G          MOV     #S##HICP,RO    ;Drop to interactive computation state
93 007110 000410          BR      7$
94      ;
95      ;   If QUAN1B has expired, requeue job at tail of current queue
96      ;
97 007112 020037 000000G 9$:   CMP     RO,VQUN1B      ;Time to shuffle queue?
98 007116 101422          BLOS    3$           ;Br if not
99 007120 004037 000000G          JSR     RO,QUNSIG      ;Signal that QUAN1B expired
100 007124 000000G          .WORD   $SQQ1B
101      ;
102      ;   Requeue job at tail of current execution queue
103      ;
104 007126 016100 000000G          MOV     LSTATE(R1),RO  ;Get job's execution state
105 007132 004737 000000G 7$:   CALL    ENQTL          ;Requeue job at tail of execution queue
106 007136 005061 000000G          CLR     LQUAN(R1)     ;Give job a fresh time quantum
107 007142 000410          BR      3$
108      ;
109      ;   Job is not in high-priority state.
110      ;   Schedule jobs on quan2 basis.
111      ;
112 007144 020037 000000G 4$:   CMP     RO,VQUAN2      ;HAS JOB USED UP QUAN2 UNITS OF TIME?
113 007150 101405          BLOS    3$           ;BR IF NOT -- DON'T RESCHEDULE JOB YET
114 007152 004037 000000G          JSR     RO,QUNSIG      ;Signal that QUAN2 expired

```

```
115 007156 000000G          .WORD  $SGQ2
116                          ;
117                          ; Reschedule job in CPU-bound run state.
118                          ;
119 007160 004737 000000G  6$:   CALL   QCPU          ; RESCHEDULE JOB IN CPU-BOUND STATE
120                          ;
121                          ; Finished
122                          ;
123 007164 012602  3$:   MOV    (SP)+, R2
124 007166 012601      MOV    (SP)+, R1
125 007170 000207      RETURN
```

```

1          .SBTTL  CLK01S -- 0.1 second clock processing
2          ;-----
3          ; CLK01S is the timer called every 0.1 seconds to do processing
4          ; at this frequency.
5          ;
6 007172  010246  CLK01S: MOV      R2,-(SP)
7 007174  010346          MOV      R3,-(SP)
8          ;
9          ; Get # 0.1 second units that have elapsed since the last time we
10         ; were called.
11         ;
12 007176  013703  001342' 16$:  MOV      TIK01S,R3          ;Get tick counter
13 007202  005203          INC      R3              ;Actual time = counter + 1
14         ;
15         ; See if any jobs need to be restarted because they were waiting for
16         ; a free message buffer and one was freed.
17         ;
18 007204  005737  000000G          TST      MBFFLG          ;Were any message buffers freed?
19 007210  001406          BEQ      15$              ;Br if not
20 007212  005037  000000G          CLR      MBFFLG          ;Reset message-buffer-freed flag
21 007216  012700  000000G          MOV      #$WSMB,R0       ;Restart any jobs that are
22 007222  004737  000000G          CALL     UREGD           ; waiting for message buffers
23         ;
24         ; Decrement minimum core residency time for jobs in memory.
25         ;
26 007226  012702  000000G  15$:  MOV      #LSTSL,R2          ;GET # OF LAST JOB
27 007232  032762  000000G  000000G 13$:  BIT      #$INCOR,LSW(R2) ; IS JOB IN MEMORY NOW?
28 007240  001415          BEQ      14$              ;BR IF NOT
29 007242  005762  000000G          TST      LMINQ(R2)       ;HAS ITS MIN CORE TIME ALREADY EXPIRED?
30 007246  001412          BEQ      14$              ;BR IF YES -- DON'T MAKE IT GO NEGATIVE
31 007250  005362  000000G          DEC      LMINQ(R2)       ;DEC MIN CORE TIME REMAINING FOR JOB
32 007254  001007          BNE      14$              ;BR IF MIN CORE TIME DID NOT EXPIRE
33 007256  105737  000000G          TSTB     SWPCOT          ;DOES SCHEDULER WANT TO BE CALLED?
34 007262  001404          BEQ      14$              ;BR IF NOT
35 007264  105237  000000G          INCB     DOSCHD          ;REQUEST A JOB SCHEDULER CYCLE
36 007270  105037  000000G          CLRB     SWPCOT          ;CLEAR MIN-TIME SCHEDULER REQUEST
37 007274  162702  000002          14$:  SUB      #2,R2          ;CHECK NEXT LINE
38 007300  001354          BNE      13$
39         ;
40         ; Keep track of number of minutes of uptime for system.
41         ;
42 007302  160337  000000G          SUB      R3,MINCTR       ;HAS 1 MINUTE PASSED?
43 007306  003013          BGT      1$              ;BR IF NOT
44 007310  062737  001130  000000G          ADD      #600,MINCTR     ;RESET COUNTER
45 007316  005237  000000G          INC      MINTIM          ;COUNT # MINUTES OF SYSTEM UPTIME
46 007322  005737  000000G          TST      DTLX           ;IS THIS A DEMO VERSION OF THE SYSTEM?
47 007326  001403          BEQ      1$              ;BR IF NOT
48 007330  005337  000000G          DEC      DTLX           ;HAS DEMO TIME LIMIT EXPIRED?
49 007334  001537          BEQ      99$            ;BR IF DEMO TIME LIMIT REACHED
50         ;
51         ; Keep track of user/idle/swap-wait time
52         ;
53 007336  010302          1$:  MOV      R3,R2          ;Get timer ticks
54 007340  006302          ASL      R2              ;Count 2 time units per interval
55 007342  060237  000000G          ADD      R2,TMTOTL       ;COUNT TOTAL TIME
56 007346  005537  000000G          ADC      TMTOTH          ;PROPOGATE CARRY
57 007352  005737  000000G          TST      UIOCNT          ;IS ANY USER I/O IN PROGRESS NOW?

```


CLK01S -- 0.1 second clock processing

```

58 007356 001404          BEQ      7$          ;BR IF NOT
59 007360 060237 000000G  ADD      R2, TMIDL          ;COUNT TIME THAT USER I/O IS ACTIVE
60 007364 005537 000000G  ADC      TMIDLH
61 007370 105737 000000G  7$: TSTB   OUTBSY          ; IS OUTSWAP IN PROGRESS?
62 007374 001003          BNE      8$          ;BR IF YES
63 007376 105737 000000G  TSTB   INBSY          ; IS INSWAP IN PROGRESS?
64 007402 001404          BEQ      9$          ;BR IF NOT
65 007404 060237 000000G  8$: ADD      R2, TMSWPL          ;COUNT TIME SWAP IS IN PROGRESS
66 007410 005537 000000G  ADC      TMSWPH
67 007414 105737 000000G  9$: TSTB   CORUSR          ; IS A USER JOB IN EXECUTION NOW?
68 007420 001405          BEQ      2$          ;BR IF NOT
69 007422 060237 000000G  ADD      R2, TMUSRL          ;COUNT TIME FOR USER JOB EXECUTION
70 007426 005537 000000G  ADC      TMUSRH
71 007432 000437          BR       3$
72          ; No user is running.
73          ; See if we should count time as swap-wait, i/o-wait or idle.
74 007434 105737 000000G  2$: TSTB   OUTBSY          ; IS AN OUTSWAP IN PROGRESS?
75 007440 001003          BNE      4$          ;BR IF YES
76 007442 105737 000000G  TSTB   INBSY          ; IS AN INSWAP IN PROGRESS?
77 007446 001415          BEQ     10$          ;BR IF NOT
78          ; Swapping is in progress. See if user I/O is also going on.
79 007450 005737 000000G  4$: TST    UIOCNT          ; IS USER I/O IN PROGRESS?
80 007454 001405          BEQ     11$          ;BR IF NOT
81 007456 006202          ASR      R2          ;SPLIT TIME BETWEEN SWAP-WAIT AND I/O-WAIT
82 007460 060237 000000G  ADD      R2, TMIOWL          ;CHARGE TO I/O-WAIT
83 007464 005537 000000G  ADC      TMIOWH
84 007470 060237 000000G  11$: ADD     R2, TMSWTL          ;CHARGE TO SWAP-WAIT
85 007474 005537 000000G  ADC      TMSWTH
86 007500 000414          BR       3$
87          ; No swapping going on. See if user I/O is in progress.
88 007502 005737 000000G  10$: TST   UIOCNT          ; IS USER I/O IN PROGRESS?
89 007506 001405          BEQ     12$          ;BR IF NOT
90 007510 060237 000000G  ADD      R2, TMIOWL          ;CHARGE TO I/O-WAIT
91 007514 005537 000000G  ADC      TMIOWH
92 007520 000404          BR       3$
93          ; System is idle.
94 007522 060237 000000G  12$: ADD     R2, TMIDL          ;CHARGE TO IDLE TIME
95 007526 005537 000000G  ADC      TMIDLH
96          ;
97          ; Check for time-slice expiration of current job
98          ;
99 007532 004737 006644'  3$: CALL   CLKJOB          ;CHECK FOR TIME-SLICE EXPIRATION OF CUR JOB
100          ;
101          ; Check to see if we need to cancel I/O hold flag for any jobs
102          ;
103 007536 004737 007646'  CALL   CLKIOH          ;Check for I/O hold flags
104          ;
105          ; Check for processing needed for autobaud logic
106          ;
107 007542 004737 011116'  CALL   CLKABD          ;Check for autobaud timer processing
108          ;
109          ; Processing done with 0.5 second frequency.
110          ;
111 007546 160337 000000G  SUB      R3, TK5CNT          ;Has 0.5 seconds passed?
112 007552 003020          BGT      6$          ;BR IF NOT
113 007554 062737 000005 000000G  ADD      #5, TK5CNT          ;RESET TIMER
114 007562 004737 011210'  CALL   TLCHK          ;DO TIMED CHECKS ON TIMESHARING LINES

```

```
115 007566 004737 010102'      CALL  WAKEUP      ;SEE IF WE NEED TO WAKE UP SLEEPING JOBS
116 007572 004737 007746'      CALL  CHKPRT     ;See if we need to print professional screen
117 007576 005727 000000G      TST   #CLTOTL   ;Are there any CL lines?
118 007602 001404              BEQ   6$        ;Br if not
119 007604 005237 000000G      INC   NEDCDO    ;Say output character processing needed
120 007610 005237 000000G      INC   NEDCLO    ;Trigger CL clock-driven processing
121                          ;
122                          ; See if any more 0.1 second time units passed while we were working
123                          ;
124 007614 160337 001342'      6$:   SUB   R3,TIK01S ;Have any more 0.1 second intervals passed?
125 007620 002402              BLT   17$        ;Br if not
126 007622 000137 007176'      JMP   16$        ;Go back and process again
127                          ;
128                          ; Finished
129                          ;
130 007626 012603              17$:  MOV   (SP)+,R3
131 007630 012602              MOV   (SP)+,R2
132 007632 000207              RETURN
133                          ;
134                          ; Time limit has expired on demo version of TSX-Plus.
135                          ; Kill the system.
136                          ;
137 007634              99$:  DIE   #EM$DTL      ;SYSTEM CRASH -- DEMO TIME LIMIT REACHED
```

CLKIOH -- See if we need to cancel I/O hold timers

```

1          .SBTTL  CLKIOH -- See if we need to cancel I/O hold timers
2          ;-----
3          ; This routine is called every 0.1 second to see if we should cancel
4          ; the I/O hold timers for any jobs. The I/O hold timer is set when we
5          ; want to swap a job out of memory but the job has I/O in progress.
6          ; To avoid holding the I/O for a job forever, we release the I/O hold
7          ; after a certain period of time (IOHLTM) if a swap has not taken place.
8          ;
9 007646 010146 CLKIOH: MOV      R1, -(SP)
10         ;
11         ; Begin loop to check I/O hold time for each job
12         ;
13 007650 012701 000000G      MOV      #LSTSL,R1      ;Get index to last job
14         ;
15         ; See if I/O hold flag is set for this job
16         ;
17 007654 005761 000000G 1$:  TST      LIOHLD(R1)      ;Is I/O hold flag set for job?
18 007660 001425          BEQ      2$              ;Br if not
19         ;
20         ; Decrement the remaining I/O hold time
21         ;
22 007662 005361 000000G      DEC      LIOHLD(R1)      ;Less I/O hold time left
23 007666 003022          BGT      2$              ;Br if some time left
24         ;
25         ; We just cancelled the I/O hold time for this job.
26         ; If the job is in a wait state, restart it.
27         ;
28 007670 026127 000000G 000000G  CMP      LSTATE(R1),#S$IOWT ;Is job in I/O wait state?
29 007676 001003          BNE      3$              ;Br if not
30 007700 004737 000000G      CALL     FORCEX          ;Start the job running
31 007704 000413          BR       2$
32         ;
33         ; If the job has any pending completion routines, make sure the job
34         ; priority is at least as high as that of the 1st completion routine.
35         ; This is necessary since we held off user completion routines while
36         ; we were waiting for I/O to stop.
37         ;
38 007706 016100 000000G 3$:  MOV      LCMPL(R1),RO      ;Does job have any pending compl routines?
39 007712 001410          BEQ      2$              ;Br if not
40 007714 126160 000000G 000000G  CMPB     LSTATE(R1),CQ#RNS(RO);Is job priority high as cpl rtn prio?
41 007722 101404          BLOS     2$              ;Br if yes
42 007724 116000 000000G      MOVB     CQ#RNS(RO),RO      ;Get job state for compl routine
43 007730 004737 000000G      CALL     ENQTL          ;Change job state
44         ;
45         ; Process next job
46         ;
47 007734 162701 000002      2$:  SUB      #2,R1          ;Get index of next job
48 007740 003345          BGT      1$              ;Loop if more jobs to check
49         ;
50         ; Finished
51         ;
52 007742 012601          MOV      (SP)+,R1
53 007744 000207          RETURN

```

CHKPRT -- See if we need to print Professional screen

```

1          .SBTTL  CHKPRT -- See if we need to print Professional screen
2          ;-----
3          ; CHKPRT is called to see if the PI handler has requested that the
4          ; contents of the professional screen be printed. If so, an asynchronous
5          ; completion routine in the PROPRT program is triggered.
6          ;
7 007746 010246 CHKPRT: MOV      R2,-(SP)
8 007750 010446      MOV      R4,-(SP)
9          ;
10         ; Return immediately if we are not running on a professional
11         ;
12 007752 105737 000000G      TSTB     PROFLG      ;Are we running on a Professional?
13 007756 001446      BEQ      9$          ;Br if not
14         ;
15         ; See if the PROPRT program is running
16         ;
17 007760 012702 000000G      MOV      #LSTSL,R2      ;Get index to last line
18 007764 026237 000000G 001344' 1$:  CMP      LPRG1(R2),R5OPRO;1st 3 chars of name = "PRO"?
19 007772 001004      BNE      2$          ;Br if not
20 007774 026237 000000G 001346'      CMP      LPRG2(R2),R5OPRT;2nd 3 chars of name = "PRT"?
21 010002 001404      BEQ      3$          ;Br if found program
22 010004 162702 000002      2$:  SUB      #2,R2          ;More lines to check?
23 010010 003365      BGT      1$          ;Loop if yes
24 010012 000425      BR       4$          ;PROPRT program is not running
25         ;
26         ; The PROPRT program is running. See if it has scheduled a
27         ; completion routine.
28         ;
29 010014 005762 000000G 3$:  TST      LBRKCQ(R2)      ;Did it specify a completion routine?
30 010020 001422      BEQ      4$          ;Br if not
31 010022 052737 000000G 000000G      BIS      #SS$RUN,SPSTAT ;Set flag saying spooler is running
32         ;
33         ; See if PI handler requested that screen be printed
34         ;
35 010030 032737 000000G 000000G      BIT      #SS$PRT,SPSTAT ;Did PI request that screen be printed?
36 010036 001416      BEQ      9$          ;Br if not
37         ;
38         ; Trigger completion routine in PROPRT
39         ;
40 010040 016204 000000G      MOV      LBRKCQ(R2),R4 ;Get address of completion queue element
41 010044 001404      BEQ      5$          ;
42 010046 005062 000000G      CLR      LBRKCQ(R2)      ;Say completion @ element used up
43 010052 004737 000000G      CALL     QCOMPL          ;Queue completion routine for the job
44 010056 042737 000000C 000000G 5$:  BIC      #SS$PRT!SS$RUN,SPSTAT ;Clear print-screen flag
45 010064 000403      BR       9$          ;
46         ;
47         ; The PROPRT program is not running
48         ;
49 010066 042737 000000C 000000G 4$:  BIC      #SS$RUN!SS$PRT,SPSTAT ;Say program not running
50         ;
51         ; Finished
52         ;
53 010074 012604 9$:  MOV      (SP)+,R4
54 010076 012602      MOV      (SP)+,R2
55 010100 000207      RETURN

```

WAKEUP -- 0.5 second processing for sleeping users

```

1          .SBTTL  WAKEUP -- 0.5 second processing for sleeping users
2          ;-----
3          ; Timer routine called ever 0.5 seconds to see if sleeping users
4          ; need to be awaken.
5          ;
6 010102 010146          WAKEUP: MOV      R1, -(SP)
7 010104 012701 000000G      MOV      #LSTSL, R1          ; GET INDEX TO LAST LINE
8          ;
9          ; Check for jobs that need to have TT reads timed out
10         ;
11 010110 026127 000000G 000000G 1$:  CMP      LSTATE(R1), #S$INWT; IS JOB WAITING FOR TT INPUT?
12 010116 001015          BNE      2$          ; BR IF NOT
13 010120 005761 000000G      TST      LRDTIM(R1)          ; DOES JOB HAVE A TT READ TIME VALUE SPECIFIED?
14 010124 001412          BEQ      2$          ; BR IF NOT
15 010126 005361 000000G      DEC      LRDTIM(R1)          ; HAS TIME EXPIRED?
16 010132 001007          BNE      2$          ; BR IF NOT
17 010134 010546          MOV      R5, -(SP)
18 010136 016105 000000G      MOV      LRTCHR(R1), R5          ; GET TIME-OUT ACTIVATION CHARACTER
19 010142          OCALL  STRACT          ; STORE ACTIVATION CHARACTER
20 010150 012605          MOV      (SP)+, R5
21         ;
22         ; Check next line
23         ;
24 010152 162701 000002      2$:  SUB      #2, R1          ; GET NEXT LINE INDEX
25 010156 001354          BNE      1$          ; BR IF THERE IS ANOTHER LINE TO CHECK
26         ;
27         ; Finished
28         ;
29 010160 012601          MOV      (SP)+, R1
30 010162 000207          RETURN

```

```

1          .SBTTL  CKTWAT -- Check on jobs doing .TWAIT waits
2          ;-----
3          ; CKTWAT is called every clock tick to see if any jobs doing .TWAIT waits
4          ; need to be restarted.
5          ;
6 010164 010146  CKTWAT: MOV      R1,-(SP)
7          ;
8          ; Check for jobs doing timed waits (.twait)
9          ;
10 010166 012701 000000G      MOV      #LSTSL,R1      ;GET HIGHEST JOB INDEX NUMBER
11 010172 026127 000000G 000000G 4$:  CMP      LSTATE(R1),#S$TMWT; IS THIS JOB DOING A TIMED WAIT?
12 010200 001024          BNE      5$          ;BR IF NOT
13 010202 163761 000000G 000000G      SUB      CLKCNT,LSLEPL(R1);DEC SLEEP TIME
14 010210 005661 000000G      SBC      LSLEPH(R1)      ;PROPOGATE CARRY
15 010214 002404          BLT      6$          ;BR IF COUNT WENT NEGATIVE
16 010216 001015          BNE      5$          ;BR IF GREATER THAN ZERO
17 010220 005761 000000G      TST      LSLEPL(R1)      ;CHECK LOW-ORDER VALUE
18 010224 001012          BNE      5$          ;BR IF NOT ZERO
19          ;
20          ; Timed wait is completed.
21          ; Put job in high priority execution state.
22          ;
23 010226 005761 000000G 6$:  TST      LITIME(R1)      ; IS THIS AN INTERACTIVE JOB?
24 010232 001403          BEQ      11$          ;BR IF NOT
25 010234 012700 000000G      MOV      #S$HICP,RO      ;PUT JOB IN INTERACTIVE CPU STATE
26 010240 000402          BR      10$
27 010242 012700 000000G 11$:  MOV      #S$TWFN,RO      ;PUT JOB IN NORMAL HIGH-PRIID EXECUTION STATE
28 010246 004737 000000G 10$:  CALL     ENQTL
29 010252 162701 000002 5$:  SUB      #2,R1          ;MORE JOBS TO CHECK?
30 010256 001345          BNE      4$          ;BR IF YES
31          ;
32          ; Finished
33          ;
34 010260 012601          MOV      (SP)+,R1
35 010262 000207          RETURN

```

CKMRKT -- check mark-time requests

```

1          .SBTTL  CKMRKT -- check mark-time requests
2          ;-----
3          ; CKMRKT is called every clock tick to see if any mark-time requests have
4          ; reached their specified time to be triggered.
5          ;
6 010264 010146 CKMRKT: MOV      R1,-(SP)
7 010266 010246      MOV      R2,-(SP)
8 010270 010446      MOV      R4,-(SP)
9          ;
10         ; Check for pending mark-time requests
11         ;
12 010272 005037 001336'      CLR      FORKIT          ;Clear fork request flag
13 010276 012702 000000C      MOV      #MRKTHD-CQ$LNK,R2;FAKE POINTER TO QUEUE HEAD
14 010302          DISABL          ;** Disable interrupts **
15 010310 016204 000000G 1$:  MOV      CQ$LNK(R2),R4    ;;;GET ADDRESS OF NEXT ELEMENT IN LIST
16 010314 001442          BEQ      8$              ;;;BR IF END OF LIST REACHED
17         ;
18         ; Subtract time that has past from specified mark-time interval.
19         ;
20 010316 163764 000000G 000000G      SUB      CLKCNT,CQ$LOT(R4);;;SUBTRACT FROM LOW-ORDER TIME VALUE
21 010324 005664 000000G          SBC      CQ$HOT(R4)      ;;;PROPOGATE BORROW TO HIGH-ORDER VALUE
22 010330 002406          BLT      3$              ;;;BR IF TIME WENT NEGATIVE
23 010332 001003          BNE      2$              ;;;BR IF TIME STILL POSITIVE
24 010334 005764 000000G          TST      CQ$LOT(R4)      ;;;CHECK LOW-ORDER VALUE
25 010340 001402          BEQ      3$              ;;;Br if zero (time has elapsed)
26 010342 010402 2$:  MOV      R4,R2          ;;;Chain forward to next entry in list
27 010344 000761          BR       1$
28         ;
29         ; This mark-time request has expired.
30         ; Remove the mark-time request entry from the waiting list.
31         ;
32 010346 016462 000000G 000000G 3$:  MOV      CQ$LNK(R4),CQ$LNK(R2);;;Remove from pending mark-time chain
33         ;
34         ; Put request on list of pending requests and schedule a lower-priority
35         ; fork routine to actually execute the completion routine.
36         ;
37 010354 005064 000000G          CLR      CQ$LNK(R4)      ;;;Clear forward link in completed element
38 010360 013700 000000G          MOV      SYPNCR,R0      ;;;Get address of 1st pending compl request
39 010364 001005          BNE      9$              ;;;Br if there are pending requests
40         ;
41         ; First entry of completion requires a fork process to be executed.
42         ;
43 010366 010437 000000G 5$:  MOV      R4,SYPNCR      ;;;Set us as 1st pending compl routine
44 010372 010437 001336'      MOV      R4,FORKIT      ;;;Set fork request flag
45 010376 000744          BR       1$              ;;;Go check for more finished requests
46         ;
47         ; Other completion entries exist so add current completion to list tail.
48         ;
49 010400 005760 000000G 9$:  TST      CQ$LNK(R0)      ;;;Is there another pending request?
50 010404 001403          BEQ      6$              ;;;Br if not
51 010406 016000 000000G          MOV      CQ$LNK(R0),R0    ;;;Chain forward to next pending request
52 010412 000772          BR       9$              ;;;Follow list to end
53 010414 010460 000000G 6$:  MOV      R4,CQ$LNK(R0)    ;;;Add our entry to end of list
54 010420 000733          BR       1$              ;;;Check for more completed requests
55         ;
56         ; Finished. Create fork process if needed.
57         ;

```

```
58 010422          B$: ENABL          ;** Enable interrupts **
59 010430 005737 001336' TST FORKIT      ;Check fork request flag
60 010434 001412          BEQ 10$        ;Br if fork is not needed
61 010436 004737 000000G CALL FRKGET     ;Get a free fork request block
62 010442 112764 000000G 000000G MOVB #FP#IOF,FQ#PRI(R4);Set fork priority
63 010450 012764 010472' 000000G MOV #CLKSCR,FQ#RTN(R4);Set address of routine to execute
64 010456 004737 000000G CALL FORKQ      ;Queue the fork request
65 010462 012604          10$: MOV (SP)+,R4
66 010464 012602          MOV (SP)+,R2
67 010466 012601          MOV (SP)+,R1
68 010470 000207          RETURN
```



```

1          .SBTTL  CLKSCR -- Execute completed system mark-time requests
2          ;-----
3          ; This routine is at a lower-priority clock-driven fork priority
4          ; to process all completed mark-time completion requests for system
5          ; routines.
6          ;
7 010472 010146 CLKSCR: MOV      R1,-(SP)
8 010474 010446      MOV      R4,-(SP)
9          ;
10         ; Unlink next completed entry from pending list
11         ;
12 010476      1$:      DISABL          ;** Disable interrupts **
13 010504 013704 000000G      MOV      SYPNCR,R4      ;;;Get address of next completion block
14 010510 001433      BEQ      9$              ;;;Br if no more pending
15 010512 016437 000000G 000000G      MOV      CQ$LNK(R4),SYPNCR ;;;Unlink block from list
16 010520      ENABL          ;** Enable interrupts **
17         ;
18         ; See if this mark-time request is for a user job or the system.
19         ;
20 010526 112764 000000G 000000G      MOVVB   #CP$STD,CQ$CP(R4);Set completion routine class priority
21 010534 116401 000000G      MOVVB   CQ$JOB(R4),R1 ;Get index # of job that did the .MRKT
22 010540 001414      BEQ      4$              ;Br if timer request came from the system
23         ;
24         ; Timer request is for a user job.
25         ; Call QCOMPL to queue the completion routine for the user job.
26         ;
27 010542 012700 000000G      MOV      #S$TWFN,RO      ;Get compl prio for non-interactive jobs
28 010546 005761 000000G      TST      LITIME(R1)      ;Is this job interactive?
29 010552 001402      BEQ      2$              ;Br if not
30 010554 012700 000000G      MOV      #S$HICP,RO      ;Get compl prio for interactive jobs
31 010560 110064 000000G      2$:      MOVVB   RO,CQ$RNS(R4) ;Set execution state for compl routine
32 010564 116164 000000G 000000G      MOVVB   LPRI(R1),CQ$PRI(R4);Set execution priority value
33         ;
34         ; Process this completion request
35         ;
36 010572 004737 000000G      4$:      CALL     QCOMPL          ;Process the completed request
37         ;
38         ; Go back and see if there are more pending requests
39         ;
40 010576 000737      BR       1$
41         ;
42         ; Finished all pending requests
43         ;
44 010600      9$:      ENABL
45 010606 012604      MOV      (SP)+,R4
46 010610 012601      MOV      (SP)+,R1
47 010612 000207      RETURN

```

CLKPM -- accumulate performance monitoring data

```

1          .SBTTL  CLKPM  -- accumulate performance monitoring data
2          ;-----
3          ; CLKPM is called to accumulate performance monitoring information.
4          ;
5          ; Inputs:
6          ;   CLKCNT = Number of clock ticks to charge to job.
7          ;   CLKPC  = PC when clock interrupt occurred.
8          ;   CLKPS  = PS when clock interrupt occurred.
9          ;   PMUSER = Job number of user who is doing performance analysis.
10         ;   PMBASE = Base address of region being monitored.
11         ;   PMTOP  = Top address of region being monitored.
12         ;   PMFLGS = PF$ control flags
13         ;   LEMTPC(Job) = PC when last EMT was executed for job.
14         ;
15         ; Outputs:
16         ;   Appropriate cell in performance counter table is incremented.
17         ;
18 010614 010146 CLKPM:  MOV    R1,-(SP)
19 010616 010246      MOV    R2,-(SP)
20 010620 010346      MOV    R3,-(SP)
21         ;
22         ; See if we are monitoring system execution or user job execution.
23         ;
24 010622 032737 000000G 000000G      BIT    #PF$SYS,PMFLGS ;MONITORING SYSTEM OR USER JOB?
25 010630 001407      BEQ    4$      ;BR IF MONITORING USER JOB
26         ; We are monitoring the system.
27 010632 032737 000000G 000000G      BIT    #UMODE,CLKPS ;DID INTERRUPT OCCUR IN KERNEL MODE?
28 010640 001065      BNE    9$      ;BR IF NOT -- USER WAS RUNNING
29 010642 013703 000000G      MOV    CLKPC,R3 ;GET ADDRESS WHERE INTERRUPT OCCURED
30 010646 000432      BR     3$      ;GO COUNT HIT
31         ;
32         ; We are monitoring user job execution.
33         ; Determine if we should count a hit against running job.
34         ;
35 010650 013701 000000G      4$:  MOV    PMUSER,R1 ;GET # OF JOB BEING MONITORED
36 010654 032761 000000G 000000G      BIT    #KMON,LSW4(R1);IS KMON RUNNING?
37 010662 001054      BNE    9$      ;DON'T MONITOR KMON
38 010664 120137 000000G      CMPB   R1,CORUSR ;IS JOB BEING MONITORED THE CURRENT JOB?
39 010670 001007      BNE    1$      ;BR IF NOT
40         ; Monitored job is running now.
41         ; See if interrupt occurred in user or kernel mode.
42 010672 032737 000000G 000000G      BIT    #UMODE,CLKPS ;DID INTERRUPT OCCUR IN USER OR KERNEL MODE?
43 010700 001413      BEQ    2$      ;BR IF IN KERNEL MODE
44         ; Job was in user mode so use interrupted PC.
45 010702 013703 000000G      MOV    CLKPC,R3 ;COUNT HIT AGAINST THIS PC
46 010706 000412      BR     3$
47         ; Monitored job is not now running.
48         ; See if we should charge I/O wait time to job.
49 010710 032737 000000G 000000G 1$:  BIT    #PF$IOW,PMFLGS ;SHOULD WE CHARGE FOR I/O WAIT TIME?
50 010716 001436      BEQ    9$      ;BR IF NOT
51         ; See if monitored job is waiting for I/O.
52 010720 026127 000000G 000000G      CMP    LSTATE(R1),#S$IOWT;IS MONITORED JOB WAITING FOR I/O?
53 010726 001032      BNE    9$      ;BR IF NOT
54         ; Use last EMT address as cell to charge hit to.
55 010730 016103 000000G      2$:  MOV    LEMTPC(R1),R3 ;GET ADDRESS OF LAST EMT DONE BY JOB
56         ;
57         ; At this point we have in R3 the PC that we are to charge this time to.

```

CLKPM -- accumulate performance monitoring data

```

58          ; See if the PC is in the region being monitored.
59          ;
60 010734 020337 000000G 3$:    CMP    R3, PMBASE    ; IS IT BELOW BASE OF REGION?
61 010740 103425          BLO    9$          ; BR IF YES
62 010742 020337 000000G    CMP    R3, PMTOP    ; IS IT ABOVE TOP OF REGION?
63 010746 103022          BHIS   9$          ; BR IF YES
64          ; PC is in region being monitored.
65 010750 163703 000000G    SUB    PMBASE, R3    ; SUBTRACT BASE TO GET OFFSET
66 010754 005002          CLR    R2          ; SET FOR DIVIDE
67 010756 071237 000000G    DIV    PMNBPC, R2   ; DIVIDE BY # BYTES PER CELL
68 010762 006302          ASL    R2          ; CONVERT CELL # TO BYTE #
69 010764 062702 000000G    ADD    #VPAR6, R2   ; ADD VIRTUAL ADDRESS OF PAR6 REGION
70 010770 013737 000000G 000000G  MOV    PMPAR, @#KPAR6 ; MAP PAR6 TO PM DATA AREA
71 010776 063712 000000G    ADD    CLKCNT, (R2) ; ADD TIME TO COUNTER FOR THIS CELL
72 011002 103004          BCC    9$          ; BR IF NO OVERFLOW OF CELL
73 011004 005312          DEC    (R2)         ; SET COUNTER VALUE BACK TO -1
74 011006 052737 000000G 000000G  BIS    #PF$OVF, PMFLGS ; REMEMBER THAN AN OVERFLOW OCCURED
75          ;
76          ; Finished
77          ;
78 011014 012603 9$:    MOV    (SP)+, R3
79 011016 012602    MOV    (SP)+, R2
80 011020 012601    MOV    (SP)+, R1
81 011022 000207    RETURN

```

CKSCHD -- Check jobs and schedule

```

1          .SBTTL  CKSCHD -- Check jobs and schedule
2          ;-----
3          ; CKSCHD will check all the jobs and schedule those that have been flagged
4          ; as needing a priority boost because of output buffer empty or low.
5          ;
6          ; Inputs:
7          ;     LSW7 - job scheduling flag
8          ;
9 011024 010046  CKSCHD: MOV     RO,-(SP)          ;Save registers
10 011026 010146  MOV     R1,-(SP)
11 011030 012701 000000G  MOV     #LSTSL,R1          ;Obtain index to the last line
12          ;
13          ; Check all jobs to see if any need a priority boost for terminal buffer
14          ; empty or low.
15          ;
16 011034 032761 000000G 000000G 1$: BIT     ##SOTFN,LSW7(R1)      ;Check for scheduling flag enable
17 011042 001417  BEQ     10$
18 011044 042761 000000G 000000G  BIC     ##SOTFN,LSW7(R1)      ;Reset job scheduling flag
19 011052 012700 000000G  MOV     #S$OTFN,R0          ;Get output-buffer empty state
20 011056 005761 000000G  TST     LITIME(R1)         ;Is this an interactive job?
21 011062 001002  BNE     2$
22 011064 012700 000000G  MOV     #S$OTLO,R0        ;If not interactive then use lower pri
23 011070 026100 000000G 2$: CMP     LSTATE(R1),R0      ;Is job already in this prio or better
24 011074 101402  BLOS   10$
25 011076 004737 000000G  CALL    ENQTL             ;Queue job at tail of execution list
26          ;
27          ; Check the next line.
28          ;
29 011102 162701 000002 10$: SUB     #2,R1          ;Check the next job
30 011106 003352  BGT     1$
31 011110 012601  MOV     (SP)+,R1         ;Continue until all job tested
32 011112 012600  MOV     (SP)+,R0
33 011114 000207  RETURN

```

CLKABD -- Clock processing for autobaud logic

```

1          .SBTTL  CLKABD -- Clock processing for autobaud logic
2          ;-----
3          ; CLKABD is called on a 1/10 second basis to do clock driven processing
4          ; related to autobaud logic.
5          ;
6 011116  010146  CLKABD: MOV      R1, -(SP)
7          ;
8          ; Begin loop to check each line
9          ;
10 011120  012701  000000G      MOV      #LSTPL, R1      ;Get index # of last line
11          ;
12          ; See if this line has autobaud control
13          ;
14 011124  032761  000000G 000000G 1$: BIT      #$AUTO, ILSW2(R1); Does this line have autobaud control?
15 011132  001421          BEQ      2$      ;Br if not
16          ;
17          ; Decrement autobaud timer for this line
18          ;
19 011134  005761  000000G      TST      LABTIM(R1)      ;Is the autobaud timer active?
20 011140  001416          BEQ      2$      ;Br if not
21 011142  005361  000000G      DEC      LABTIM(R1)      ;Decrement timer
22 011146  001013          BNE      2$      ;Br if timer did not time out
23          ;
24          ; Timer timed out for this line.
25          ; See if we need to reset the line speed.
26          ;
27 011150  032761  000000G 000000G      BIT      #$NABRS, LSW9(R1); Do we need to reset the line speed?
28 011156  001407          BEQ      2$      ;Br if not
29 011160  042761  000000G 000000G      BIC      #$NABRS, LSW9(R1); Clear the flag
30 011166  012700  000000G      MOV      #S9600, R0      ;Reset line speed to 9600 baud
31 011172  004737  000000G      CALL     SETSPD      ;Reset speed
32          ;
33          ; Check next line
34          ;
35 011176  162701  000002      2$: SUB      #2, R1      ;More lines to do?
36 011202  003350          BGT      1$      ;Loop if yes
37          ;
38          ; Finished
39          ;
40 011204  012601          MOV      (SP)+, R1
41 011206  000207          RETURN

```

TLCHK -- Check Dial-up line status

```

1          .SBTTL  TLCHK  -- Check Dial-up line status
2          ;-----
3          ; TLCHK is called from the clock interrupt routine every 0.5 seconds
4          ; to see if dial-up lines need to be answered or hung up.
5          ;
6 011210   010146   TLCHK:  MOV     R1, -(SP)
7 011212   010246           MOV     R2, -(SP)
8          ;
9          ; Begin loop to check each physical line
10         ;
11 011214   012701   000000G      MOV     #LSTHL, R1      ; Index to last real line
12         ;
13         ; See if this line is installed
14         ;
15 011220   032761   000000G 000000G 1$:  BIT     #$DEAD, LSW3(R1) ; Is this line installed?
16 011226   001024           BNE     2$             ; Br if not
17 011230   032761   000000G 000000G      BIT     #$HARD, LSW3(R1) ; Is this line connected to hardware?
18 011236   001420           BEQ     2$             ; Br if not
19         ;
20         ; Call processing routine based on type of communications device
21         ;
22 011240   016102   000000G      MOV     LCDTYP(R1), R2 ; Get comm device index number
23 011244   004772   000000G      CALL    @CDCLOK(R2)    ; Call processing routine for this line
24         ;
25         ; If this is a dial-up line, check on line ringing, lost carrier, etc.
26         ;
27 011250   005761   000000G      TST     LCLUNT(R1)    ; Is this line in use as a CL unit?
28 011254   002011           BGE     2$             ; Br if this is a CL line
29 011256   032761   000000G 000000G      BIT     #$PHONE, ILSW2(R1); Is this a dial-up line?
30 011264   001405           BEQ     2$             ; Br if not
31 011266   020127   000000G      CMP     R1, #LSTPL    ; Is this a time-sharing or CL line?
32 011272   101002           BHI     2$             ; Don't do phone checks for CL lines
33 011274   004737   011314'      CALL    CLKPHN        ; Check phone line
34         ;
35         ; See if there are more lines to be checked
36         ;
37 011300   162701   000002      2$:    SUB     #2, R1      ; Get index number for next line
38 011304   001345           BNE     1$             ; Loop if more lines to check
39         ;
40         ; Finished
41         ;
42 011306   012602           MOV     (SP)+, R2
43 011310   012601           MOV     (SP)+, R1
44 011312   000207           RETURN

```

CLKPHN -- Do timer driven checks of dial-up lines

```

1          .SBTTL  CLKPHN -- Do timer driven checks of dial-up lines
2          ;-----
3          ; CLKPHN is called periodically to perform checks on dial-up lines.
4          ; Checks are made to see if the phone is ringing or if carrier
5          ; has been detected or lost.
6          ;
7          ; Inputs:
8          ; R1 = Physical line index number.
9          ;
10         011314  010246  CLKPHN: MOV      R2, -(SP)
11         ;
12         ; Call device-dependent routine to get the data set status for this line
13         ;
14         011316  016102  000000G  MOV      LCDTYP(R1),R2  ;Get comm device type code index
15         011322  004772  000000G  CALL     @CDGDSS(R2)   ;Get data set status
16         ;
17         ; At this point, the generic modem status (MS$xxx flags) is in R0.
18         ; See if the phone is ringing
19         ;
20         011326  105737  000000G  TSTB    STPFLG         ;Is system being stopped?
21         011332  001020          BNE     5$             ;Br if yes
22         011334  032700  000000G  BIT     #MS$RNG,R0     ;Is line ringing?
23         011340  001415          BEQ     5$             ;Br if not
24         011342  032700  000000G  BIT     #MS$DTR,R0     ;Have we enabled answer?
25         011346  001012          BNE     5$             ;Br if yes
26         ;
27         ; Time to answer ringing phone
28         ;
29         011350  052700  000000G  BIS     #MS$DTR,R0     ;Enable answer
30         011354  004772  000000G  CALL     @CDSOSS(R2)   ;Set data set status
31         011360  013761  000000G  000000G  MOV     VTMIN, LCDTIM(R1); Start carrier-down timer
32         011366  013761  000000G  000000G  MOV     VONTM, LOFFTM(R1); Drop DTR if not logged on by this time
33         ;
34         ; Check status of carrier on dial-up lines.
35         ;
36         011374  032700  000000G  5$:     BIT     #MS$CAR,R0  ;Is carrier up or down?
37         011400  001054          BNE     16$            ;Br if up
38         ;
39         ; Carrier is down
40         ;
41         011402  005361  000000G  11$:    DEC     LCDTIM(R1)  ;Has it been down very long?
42         011406  003057          BGT     8$             ;Br if not
43         011410  042761  000000G  000000G  BIC     ##$CARUP,LSW3(R1); Remember that we have lost carrier
44         011416  032761  000000G  000000G  BIT     ##$DILUP,LSW(R1); Is line active?
45         011424  001407          BEQ     1$             ;Br if not
46         011426  105737  000000G  TSTB    VUSPHN         ;Should we always mon. carrier for line?
47         011432  001004          BNE     1$             ;Br if so
48         011434  032761  000000G  000000G  BIT     ##$CARMN,LSW5(R1); Are we monitoring carrier for this line?
49         011442  001436          BEQ     17$            ;Br if not
50         011444  052761  000000G  000000G  1$:     BIS     ##$NOIN,LSW3(R1); Ignore tt input from the line
51         011452  032761  000000G  000000G  BIT     ##$DILUP,LSW(R1); Is line active?
52         011460  001415          BEQ     21$            ;Br if not
53         011462  032761  000000G  000000G  BIT     #<$DISCN+$DOOFF>,LSW(R1); Are we logging off line now?
54         011470  001026          BNE     8$             ;Br if yes -- that takes care of it
55         011472  032761  000000G  000000G  BIT     ##$LOFCF,LSW9(R1); Are we doing logoff command file now?
56         011500  001022          BNE     8$             ;Br if yes
57         011502  010100          MOV     R1,R0         ;Get index of job being aborted

```

CLKPHN -- Do timer driven checks of dial-up lines

```
58 011504          DCALL  KILJOB          ;Kill the job
59 011512 000415    BR      8$
60 011514 042700 000000G 21$:  BIC      #MS$DTR,RO      ;Drop data terminal ready (hang up)
61 011520 004772 000000G      CALL    @CSDSS(R2)      ;Set data set status
62 011524 005061 000000G      CLR     LOFFTM(R1)      ;Clear logoff timer
63 011530 000403    BR      17$
64                ;
65                ; Carrier is up
66                ;
67 011532 052761 000000G 000000G 16$:  BIS     #$CARUP,LSW3(R1);Remember carrier is up
68                ;
69                ; Reset lost-carrier timer
70                ;
71 011540 013761 000000G 000000G 17$:  MOV     VTMOU,LCDTIM(R1);Reset carrier-lost timer
72                ;
73                ; If jobs on dial-up lines remain in a logged off state for more than
74                ; a specified interval, drop DTR to hang up on them.
75                ;
76 011546 005761 000000G 8$:   TST     LOFFTM(R1)      ;Do we need to check time for this job?
77 011552 001407    BEQ     6$              ;Br if not
78 011554 005361 000000G      DEC     LOFFTM(R1)      ;Is it time to drop DTR for this line?
79 011560 003004    BGT     6$              ;Br if not
80 011562 042700 000000G      BIC     #MS$DTR,RO      ;Drop data terminal ready (hang up)
81 011566 004772 000000G      CALL    @CSDSS(R2)      ;Set data set status
82                ;
83                ; Finished
84                ;
85 011572 012602    6$:   MOV     (SP)+,R2
86 011574 000207    RETURN
```


DLGDSS -- Get data set status for DL11 line

```

1          .SBTTL  DLGDSS -- Get data set status for DL11 line
2          ;-----
3          ; DLGDSS is called to get the data set status for a DL11 line.
4          ;
5          ; Inputs:
6          ;   R1 = Physical line index number.
7          ;
8          ; Outputs:
9          ;   R0 = Generic data set status flags (MS$xxx)
10         ;
11 011576 010346 DLGDSS: MOV     R3, -(SP)
12 011600 005000      CLR     R0          ;Form result in R0
13         ;
14         ; Get contents of DL11 receiver status register
15         ;
16 011602 017103 000000G      MOV     @RSR(R1),R3    ;Get receiver status register contents
17         ;
18         ; See if line is ringing
19         ;
20 011606 032703 000000G      BIT     #RING,R3      ;Is phone ringing?
21 011612 001402      BEQ     1$          ;Br if not
22 011614 052700 000000G      BIS     #MS$RNG,R0    ;Set ring flag
23         ;
24         ; See if carrier is up or down
25         ;
26 011620 032703 000000G  1$:  BIT     #CARDET,R3      ;Is carrier up or down?
27 011624 001402      BEQ     2$          ;Br if down
28 011626 052700 000000G      BIS     #MS$CAR,R0    ;Set carrier-up flag
29         ;
30         ; See if Data Terminal Ready is asserted
31         ;
32 011632 032703 000000G  2$:  BIT     #TRMRDY,R3      ;Is DTR asserted?
33 011636 001402      BEQ     3$          ;Br if not
34 011640 052700 000000G      BIS     #MS$DTR,R0    ;Set DTR flag
35         ;
36         ; Finished
37         ;
38 011644 012603  3$:  MOV     (SP)+,R3
39 011646 000207      RETURN

```

DLSDSS -- Set data set status for DL11 line

```

1          .SBTTL  DLSDSS -- Set data set status for DL11 line
2          ;-----
3          ; DLSDSS is called to set data set control status for a DL11 line.
4          ;
5          ; Inputs:
6          ;   R1 = Physical line index number.
7          ;   R0 = Control flags (MS#DTR)
8          ;
9 011650   DLSDSS:
10         ;
11         ; See if we should set or drop DTR
12         ;
13 011650   032700   000000G          BIT      #MS#DTR,R0      ;Set or drop DTR?
14 011654   001404          BEQ      1$              ;Br to drop DTR
15         ;
16         ; Set DTR
17         ;
18 011656   052771   000000G 000000G          BIS      #TRMRDY,@RSR(R1);Set Data Terminal Ready
19 011664   000403          BR       9$
20         ;
21         ; Drop DTR
22         ;
23 011666   042771   000000G 000000G 1$:     BIC      #TRMRDY,@RSR(R1);Drop DTR
24         ;
25         ; Finished
26         ;
27 011674   000207          9$:     RETURN

```

DLSBRK -- Control break transmission for a DL11 line

```

1          .SBTTL  DLSBRK -- Control break transmission for a DL11 line
2          ;-----
3          ; DLSBRK is called to start or stop sending a break character to a DL11 line.
4          ;
5          ; Inputs:
6          ;   R1 = Physical line number.
7          ;   R0 = Break control flag (MS$BRK)
8          ;
9 011676   DLSBRK:
10         ;
11         ; See if we are to start or stop transmitting a break
12         ;
13 011676   032700   0000000   BIT      #MS$BRK,R0      ;Start or stop break?
14 011702   001404           BEQ      1$              ;Br if stop
15         ;
16         ; Start transmitting a break
17         ;
18 011704   052771   0000000 0000000   BIS      #TRBRK,@TSR(R1) ;Start transmitting a break
19 011712   000403           BR       9$
20         ;
21         ; Stop transmitting a break
22         ;
23 011714   042771   0000000 0000000 1$:   BIC      #TRBRK,@TSR(R1) ;Stop transmitting a break
24         ;
25         ; Finished
26         ;
27 011722   000207           9$:   RETURN

```

DLSSPD -- Set transmission speed for DL11 line

```

1          .SBTTL  DLSSPD -- Set transmission speed for DL11 line
2          ;-----
3          ; DLSSPD is called to set the transmission speed for a DL11 line.
4          ;
5          ; Inputs:
6          ;   R0 = Speed code.
7          ;   R1 = Physical line index number.
8          ;
9 011724 010246 DLSSPD: MOV      R2, -(SP)
10         ;
11         ; Set speed in DL11 control register
12         ;
13 011726 110061 000001G      MOVB   R0, LMXPRM+1(R1) ; Store new code flags for line
14 011732 010002              MOV     R0, R2          ; Get speed code
15 011734 072227 000014      ASH    #12, R2          ; Position the speed code
16 011740 052702 004000      BIS    #004000, R2      ; Set programmable-baud-rate-enable bit
17 011744              DISABL              ; ** Disable interrupts **
18 011752 017146 000000G      MOV     @TSR(R1), -(SP) ; Get current transmitter status
19 011756 042716 170000      BIC    #170000, (SP) ; Clear the baud rate field
20 011762 050216              BIS    R2, (SP)          ; Set new baud rate value
21 011764 012671 000000G      MOV     (SP)+, @TSR(R1) ; Store new value for transmitter
22 011770              ENABL              ; ** Enable interrupts **
23         ;
24         ; Finished
25         ;
26 011776 012602              MOV     (SP)+, R2
27 012000 000207              RETURN

```

DZGDSS -- Get data set status for DZ11 line

```

1          .SBTTL  DZGDSS -- Get data set status for DZ11 line
2          ;-----
3          ; DZGDSS is called to get the data set status for a DZ11 line
4          ;
5          ; Inputs:
6          ;   R1 = Physical line index number.
7          ;
8          ; Outputs:
9          ;   R0 = Generic data set status flags (MS$xxx)
10         ;
11 012002  010246  DZGDSS: MOV      R2, -(SP)
12 012004  010346          MOV      R3, -(SP)
13 012006  005000          CLR      R0          ;Build result in R0
14         ;
15         ; Get DZ11 index number
16         ;
17 012010  016102  000000G      MOV      LMXNUM(R1),R2  ;Get DZ11 number
18 012014  016103  000000G      MOV      LMXLN(R1),R3  ;Get # of line within DZ11 group (0-7)
19 012020  116303  001312'      MOVVB   MXLBIT(R3),R3  ;Get line select bit for DZ11 registers
20         ;
21         ; See if line is ringing
22         ;
23 012024  130372  000000G      BITB    R3,@MXRING(R2) ;Is this line ringing?
24 012030  001402          BEQ     1$          ;Br if not
25 012032  052700  000000G      BIS     #MS$RNG,R0   ;Set ring flag
26         ;
27         ; See if carrier is up
28         ;
29 012036  130372  000000G  1$:    BITB    R3,@MXCAR(R2) ;Is carrier up or down?
30 012042  001402          BEQ     2$          ;Br if down
31 012044  052700  000000G      BIS     #MS$CAR,R0   ;Set carrier-up flag
32         ;
33         ; See if Data Terminal Ready is asserted
34         ;
35 012050  130372  000000G  2$:    BITB    R3,@MXDTR(R2) ;Is DTR asserted?
36 012054  001402          BEQ     3$          ;Br if not
37 012056  052700  000000G      BIS     #MS$DTR,R0   ;Set DTR flag
38         ;
39         ; Finished
40         ;
41 012062  012603  3$:    MOV     (SP)+,R3
42 012064  012602          MOV     (SP)+,R2
43 012066  000207          RETURN

```

DZSDSS -- Set data set status for a DZ11 line

```

1          .SBTTL  DZSDSS -- Set data set status for a DZ11 line
2          ;-----
3          ; DZSDSS is called to set data set status for a DZ11 line.
4          ;
5          ; Inputs:
6          ;   R1 = Physical line number.
7          ;   R0 = Data set status flags (MS#DTR).
8          ;
9 012070 010246 DZSDSS: MOV     R2,-(SP)
10 012072 010346      MOV     R3,-(SP)
11          ;
12          ; Get DZ11 index number
13          ;
14 012074 016102 000000G      MOV     LMXNUM(R1),R2 ;Get DZ11 number
15 012100 016103 000000G      MOV     LMXLN(R1),R3 ;Get line # within DZ11 (0-7)
16 012104 116303 001312'      MOVB    MXLBIT(R3),R3 ;Get line select bit
17          ;
18          ; See if we should set or drop Data Terminal Ready
19          ;
20 012110 032700 000000G      BIT     #MS#DTR,R0 ;Set or drop DTR?
21 012114 001003              BNE     1$ ;Br if set DTR
22          ;
23          ; Drop DTR
24          ;
25 012116 140372 000000G      BICB    R3,@MXDTR(R2) ;Clear DTR flag for our line
26 012122 000402              BR     9$
27          ;
28          ; Set DTR
29          ;
30 012124 150372 000000G      1$:    BISB    R3,@MXDTR(R2) ;Set DTR flag for our line
31          ;
32          ; Finished
33          ;
34 012130 012603      9$:    MOV     (SP)+,R3
35 012132 012602      MOV     (SP)+,R2
36 012134 000207      RETURN

```

DZSBRK -- Control break transmission for a DZ11 line

```

1                                     .SBTTL  DZSBRK -- Control break transmission for a DZ11 line
2                                     ;-----
3                                     ; DZSBRK is called to start or stop transmitting a break character
4                                     ; to a DZ11 line.
5                                     ;
6                                     ; Inputs:
7                                     ; R0 = Break control flag (MS$BRK)
8                                     ; R1 = Physical line index number.
9                                     ;
10 012136 010246 DZSBRK: MOV      R2, -(SP)
11 012140 010346      MOV      R3, -(SP)
12 012142 010446      MOV      R4, -(SP)
13                                     ;
14                                     ; Get DZ11 index number
15                                     ;
16 012144 016102 000000G      MOV      LMXNUM(R1),R2 ;Get DZ11 number
17 012150 016103 000000G      MOV      LMXLN(R1),R3 ;Get line # within DZ11 (0-7)
18 012154 116303 001312'      MOVVB   MXLBIT(R3),R3 ;Get line select bit
19                                     ;
20                                     ; We keep a "shadow" copy of the break register in memory since we
21                                     ; cannot read the status of the hardware break register.
22                                     ;
23 012160 116204 000000G      MOVVB   MXSBRK(R2),R4 ;Get contents of shadow register
24 012164 140304      BICB    R3,R4 ;Assume we want to stop sending break
25 012166 032700 000000G      BIT     #MS$BRK,R0 ;Do we want to start sending break?
26 012172 001401      BEQ     1$ ;Br if not
27 012174 150304      BISB    R3,R4 ;Set break flag for the line
28                                     ;
29                                     ; Set new break control flags in hardware register and shadow register
30                                     ;
31 012176 110472 000000G 1$: MOVVB   R4,@MXBRK(R2) ;Set status in hardware register
32 012202 110462 000000G      MOVVB   R4,MXSBRK(R2) ;Update shadow register
33                                     ;
34                                     ; Finished
35                                     ;
36 012206 012604 9$: MOV     (SP)+,R4
37 012210 012603      MOV     (SP)+,R3
38 012212 012602      MOV     (SP)+,R2
39 012214 000207      RETURN

```

DZSSPD -- Set transmission speed for a DZ11 line

```

1                                     .SBTTL  DZSSPD -- Set transmission speed for a DZ11 line
2                                     ;-----
3                                     ; DZSSPD is called to set the transmit/receive speed for a DZ11 line.
4                                     ;
5                                     ; Inputs:
6                                     ;   RO = Speed code.
7                                     ;   R1 = Physical line index number.
8                                     ;
9 012216 010346 DZSSPD: MOV      R3, -(SP)
10                                     ;
11                                     ; Build line parameter register value
12                                     ;
13 012220 110061 0000010 MOVB   RO, LMXPRM+1(R1) ; Save new parameter flags
14 012224 010003 MOV     RO, R3          ; Get speed code
15 012226 042703 000000C BIC    ^C<LP#SPD>, R3 ; Clear all but speed code
16 012232 000303 SWAB   R3              ; Position speed code to match LPR field
17 012234 032700 0000000 BIT    #LP#7BT, RO      ; Are 7 bit characters wanted?
18 012240 001003 BNE    1$              ; Br if yes
19 012242 052703 0000000 BIS    #DZ#8BT, R3      ; Select 8 bit characters
20 012246 000402 BR     2$              ;
21 012250 052703 0000000 1$: BIS   #DZ#7BT, R3      ; Select 7 bit characters
22 012254 032700 0000000 2$: BIT   #LP#PAR, RO      ; Is parity wanted?
23 012260 001407 BEQ    3$              ; Br if not
24 012262 052703 0000000 BIS    #DZ#PAR, R3      ; Enable parity
25 012266 032700 0000000 BIT    #LP#ODD, RO      ; Is odd parity wanted?
26 012272 001402 BEQ    3$              ; Br if not
27 012274 052703 0000000 BIS    #DZ#ODD, R3     ; Select odd parity
28                                     ;
29                                     ; Store LPR value for line
30                                     ;
31 012300 052703 010000 3$: BIS   #10000, R3      ; Set receiver-on flag
32 012304 056103 0000000 BIS    LMXLN(R1), R3     ; Get line within mux (0-7)
33 012310 016100 0000000 MOV    LMXNUM(R1), RO    ; Get DZ11 number
34 012314 010370 0000000 MOV    R3, @MXLPR(RO)   ; Set speed for the line
35                                     ;
36                                     ; Finished
37                                     ;
38 012320 012603 MOV    (SP)+, R3
39 012322 000207 RETURN

```


DHGDSS -- Get data set status for a DH11 line

```

1          .SBTTL  DHGDSS -- Get data set status for a DH11 line
2          ;-----
3          ; DHGDSS is called to get the data set status for a DH11 line.
4          ;
5          ; Inputs:
6          ;   R1 = Physical line number
7          ;
8          ; Outputs:
9          ;   R0 = Generic modem status flags (MS$xxx)
10         ;
11 012324 010246 DHGDSS: MOV     R2, -(SP)
12 012326 010346      MOV     R3, -(SP)
13 012330 005000      CLR     R0          ;Build result in R0
14         ;
15         ; Get DH11 index number
16         ;
17 012332 016102 000000G MOV     LMXNUM(R1),R2 ;Get DH11 index number
18 012336 016103 000000G MOV     LMXLN(R1),R3 ;Get line within DH11 (0-15)
19         ;
20         ; Get modem status
21         ;
22 012342      DISABL          ;;; ** Disable interrupts **
23 012350 042772 000000G 000000G BIC     #MF$LIN,@DM$CSR(R2) ;;; Clear DM11 line select field
24 012356 050372 000000G      BIS     R3,@DM$CSR(R2) ;;; Select line
25 012362 017203 000000G      MOV     @DM$LSR(R2),R3 ;;; Get line status value
26 012366      ENABL          ;;; ** Enable interrupts **
27         ;
28         ; See if phone is ringing
29         ;
30 012374 032703 000000G      BIT     #MF$RNG,R3 ;Is the phone ringing?
31 012400 001402      BEQ     1$ ;Br if not
32 012402 052700 000000G      BIS     #MS$RNG,R0 ;Set ring flag
33         ;
34         ; See if carrier is detected
35         ;
36 012406 032703 000000G 1$: BIT     #MF$CAR,R3 ;Is carrier detected?
37 012412 001402      BEQ     2$ ;Br if not
38 012414 052700 000000G      BIS     #MS$CAR,R0 ;Set carrier flag
39         ;
40         ; See if Data Terminal Ready is asserted
41         ;
42 012420 032703 000000G 2$: BIT     #MF$DTR,R3 ;Is DTR asserted?
43 012424 001402      BEQ     3$ ;Br if not
44 012426 052700 000000G      BIS     #MS$DTR,R0 ;Set DTR flag
45         ;
46         ; Finished
47         ;
48 012432 012603 3$: MOV     (SP)+,R3
49 012434 012602      MOV     (SP)+,R2
50 012436 000207      RETURN

```

```

1                                     .SBTTL  DHSDSS -- Set data set status for a DH11 line
2                                     ;-----
3                                     ; DHSDSS is called to set the data set status for a DH11 line.
4                                     ;
5                                     ; Inputs:
6                                     ; R1 = Physical line index number.
7                                     ; R0 = Data set status flags (MS%DTR)
8                                     ;
9 012440 010246 DHSDSS: MOV      R2,-(SP)
10 012442 010346      MOV      R3,-(SP)
11                                     ;
12                                     ; Get modem index number and select our line
13                                     ;
14 012444 016102 000000G      MOV      LMXNUM(R1),R2 ;Get DH11 index number
15 012450 016103 000000G      MOV      LMXLN(R1),R3 ;Get line # within DH11 (0-15)
16 012454      DISABL      ;;;** Disable interrupts **
17 012462 042772 000000G 000000G BIC      #MF$LIN,@DM$CSR(R2) ;;;Clear DM11 line # field
18 012470 050372 000000G      BIS      R3,@DM$CSR(R2) ;;;Select our line
19                                     ;
20                                     ; See if we should set or drop Data Terminal Ready
21                                     ;
22 012474 032700 000000G      BIT      #MS%DTR,R0 ;;;Set or drop DTR?
23 012500 001004      BNE      1$ ;;;Br to set DTR
24 012502 042772 000000G 000000G BIC      #MF%DTR,@DM$LSR(R2) ;;;Drop DTR
25 012510 000403      BR      9$
26 012512 052772 000000G 000000G 1$: BIS      #MF%DTR,@DM$LSR(R2);;;Set DTR
27                                     ;
28                                     ; Finished
29                                     ;
30 012520      9$: ENABL      ;;; Enable interrupts **
31 012526 012603      MOV      (SP)+,R3
32 012530 012602      MOV      (SP)+,R2
33 012532 000207      RETURN

```

DHSSPD -- Set transmit/receive speed for DH11 line

```

1          .SBTTTL  DHSSPD -- Set transmit/receive speed for DH11 line
2          ;-----
3          ; DHSSPD is called to set the transmit/receive speed for a DH11 line.
4          ; The parity and character length parameters are also set.
5          ;
6          ; Inputs:
7          ;   R0 = Speed, length, and parity codes.
8          ;   R1 = Physical line index number.
9          ;
10         DHSSPD: MOV      R2, -(SP)
11         MOV      R3, -(SP)
12         MOV      R4, -(SP)
13         ;
14         ; Update the LMXPRM table for this line
15         ;
16         MOV      R0, LMXPRM+1(R1) ; Store new codes for line
17         ;
18         ; Convert TSX-Plus speed code into DH11 speed code
19         ;
20         MOV      R0, R3           ; Get speed code
21         BIC      #^C<LP$SPD>, R3 ; Clear all but speed code
22         MOV      R0, DHSPCT(R3), R3 ; Convert to DH11 speed code
23         ;
24         ; Get DH11 index number
25         ;
26         MOV      R2, LMXNUM(R1) ; Get DH11 index number
27         MOV      R4, LMXLN(R1) ; Get # of line within mux group
28         BIS      #HF$RIE, R4    ; Set receiver interrupt enable flag
29         ;
30         ; Build value to use for line parameter register
31         ;
32         ASH      #6, R3         ; Position speed code for receive speed
33         MOV      R3, -(SP)     ; Save positioned receive speed
34         ASH      #4, R3         ; Position code for transmit speed
35         BIS      (SP)+, R3      ; Combine transmit and receive speed codes
36         BIT      #LP$7BT, R0   ; 7 bit characters wanted?
37         BNE      1$           ; Br if yes
38         BIS      #HF$8BT, R3   ; Select 8 bit characters
39         BR       2$           ;
40         1$: BIS      #HF$7BT, R3 ; Select 7 bit characters
41         2$: BIT      #LP$PAR, R0 ; Parity wanted?
42         BEQ      3$           ; Br if not
43         BIS      #HF$PAR, R3   ; Enable parity
44         BIT      #LP$ODD, R0   ; Odd parity wanted?
45         BEQ      3$           ; Br if not
46         BIS      #HF$ODD, R3   ; Select odd parity
47         ;
48         ; Select LPR register for line being set and store the LPR value
49         ;
50         3$: DISABL                ; ** Disable interrupts **
51         MOV      R4, @MH$SCR(R2) ; Select our mux line
52         MOV      R3, @MH$LPR(R2) ; Store the LPR value for this line
53         ENABL                ; ** Enable interrupts **
54         ;
55         ; Finished
56         ;
57         MOV      (SP)+, R4

```

58	012702	012603	MOV	(SP)+,R3
59	012704	012602	MOV	(SP)+,R2
60	012706	000207	RETURN	

DHSBRK -- Control break transmission for a DH11 line

```

1          .SBTTL DHSBRK -- Control break transmission for a DH11 line
2          ;-----
3          ; DHSBRK is called to start or stop transmitting a break to a DH11 line.
4          ;
5          ; Inputs:
6          ;   R0 = Break control flag (MS$BRK)
7          ;   R1 = Line index number
8          ;
9 012710   010246 DHSBRK: MOV     R2,-(SP)
10 012712   010346      MOV     R3,-(SP)
11          ;
12          ; Get DH11 index number and line select flag
13          ;
14 012714   016102   000000G      MOV     LMXNUM(R1),R2 ;Get DH11 index number
15 012720   016103   000000G      MOV     LMXLN(R1),R3 ;Get line within DH11 (0-15)
16 012724   006303           ASL     R3           ;Convert line # to word table index
17 012726   016303   001252'      MOV     DHLBIT(R3),R3 ;Get flag bit corresponding to line #
18          ;
19          ; See if we should start or stop sending a break
20          ;
21 012732   032700   000000G      BIT     #MS$BRK,R0   ;Start or stop sending break?
22 012736   001403           BEQ     1$           ;Br if stop
23          ;
24          ; Start sending a break to this line
25          ;
26 012740   050372   000000G      BIS     R3,@MH$BRK(R2) ;Set break flag for our line
27 012744   000402           BR      9$
28          ;
29          ; Stop sending a break to this line
30          ;
31 012746   040372   000000G 1$: BIC     R3,@MH$BRK(R2) ;Stop sending a break to this line
32          ;
33          ; Finished
34          ;
35 012752   012603 9$:  MOV     (SP)+,R3
36 012754   012602      MOV     (SP)+,R2
37 012756   000207      RETURN

```

VHGDSS -- Get data set status for a DHV11 line

```

1          .SBTTL  VHGDSS -- Get data set status for a DHV11 line
2          ;-----
3          ; VHGDSS is called to get the data set status for a DHV11 line.
4          ;
5          ; Inputs:
6          ;   R1 = Physical line index number.
7          ;
8          ; Outputs:
9          ;   R0 = Generic modem status flags (MS$xxx)
10         ;
11 012760 010246 VHGDSS: MOV     R2, -(SP)
12 012762 010346      MOV     R3, -(SP)
13 012764 010446      MOV     R4, -(SP)
14 012766 005000      CLR     R0          ; Form result in R0
15         ;
16         ; Get DHV11 index number and line number
17         ;
18 012770 016102 000000G MOV     LMXNUM(R1), R2  ; Get mux index number
19 012774 016103 000000G MOV     LMXLN(R1), R3  ; Get line # within mux group
20 013000 052703 000000G BIS     #VF$RIE, R3    ; Set receiver interrupt enable flag
21         ;
22         ; Get modem status
23         ;
24 013004          DISABL          ; ** Disable interrupts **
25 013012 110372 000000G MOVB   R3, @VH$CSR(R2) ; Select our line in mux
26 013016 017204 000000G MOV     @VH$LCR(R2), R4 ; Get line control register
27 013022 017203 000000G MOV     @VH$LSR(R2), R3 ; Get current line status
28 013026          ENABL          ; ** Enable interrupts
29         ;
30         ; See if line is ringing
31         ;
32 013034 032703 000000G BIT     #VF$RNG, R3    ; Is the line ringing?
33 013040 001402          BEQ     1$          ; Br if not
34 013042 052700 000000G BIS     #MS$RNG, R0    ; Set ringing flag
35         ;
36         ; See if carrier is up
37         ;
38 013046 032703 000000G 1$: BIT     #VF$DCD, R3    ; Is carrier detected?
39 013052 001402          BEQ     2$          ; Br if not
40 013054 052700 000000G BIS     #MS$CAR, R0    ; Set carrier flag
41         ;
42         ; See if Data Terminal Ready is asserted
43         ;
44 013060 032704 000000G 2$: BIT     #VF$DTR, R4    ; Is Data Terminal Ready asserted?
45 013064 001402          BEQ     3$          ; Br if not
46 013066 052700 000000G BIS     #MS$DTR, R0    ; Set DTR flag
47         ;
48         ; Finished
49         ;
50 013072 012604 3$: MOV     (SP)+, R4
51 013074 012603      MOV     (SP)+, R3
52 013076 012602      MOV     (SP)+, R2
53 013100 000207      RETURN

```

VHSDSS -- Set data set status for a DHV11 line

```

1          .SBTTL  VHSDSS -- Set data set status for a DHV11 line
2          ;-----
3          ; VHSDSS is called to set the data set status for a DHV11 line.
4          ;
5          ; Inputs:
6          ; R1 = Physical line index number.
7          ; R0 = Data set status flags (MS#DTR).
8          ;
9 013102   010246   VHSDSS: MOV      R2,-(SP)
10 013104   010346           MOV      R3,-(SP)
11          ;
12          ; Get DHV11 mux index number and line number
13          ;
14 013106   016102   000000G   MOV      LMXNUM(R1),R2   ;Get mux index number
15 013112   016103   000000G   MOV      LMXLN(R1),R3   ;Get # of line within mux group
16 013116   052703   000000G   BIS      #VF#RIE,R3    ;Set receiver interrupt enable flag
17          ;
18          ; Set or drop the Data Terminal Ready flag
19          ;
20 013122           DISABL                    ;;; ** Disable interrupts **
21 013130   110372   000000G   MOVVB   R3,@VH#CSR(R2)   ;;; Select our line in mux
22 013134   032700   000000G   BIT     #MS#DTR,R0    ;;; Set or drop DTR?
23 013140   001004           BNE     1$              ;;; Br if want to set DTR
24 013142   042772   000000G 000000G   BIC    #VF#DTR,@VH#LCR(R2);;; Clear DTR bit
25 013150   000403           BR     2$
26 013152   052772   000000G 000000G 1$:   BIS    #VF#DTR,@VH#LCR(R2);;; Set DTR bit
27          ;
28          ; Finished
29          ;
30 013160           2$:   ENABL                    ;;; Enable interrupts **
31 013166   012603           MOV    (SP)+,R3
32 013170   012602           MOV    (SP)+,R2
33 013172   000207           RETURN

```

VHSSPD -- Set transmit/receive speed for a DHV11 line

```

1          .SBTTL  VHSSPD -- Set transmit/receive speed for a DHV11 line
2          ;-----
3          ; Set the transmit/receive speed for a DHV11 line.
4          ;
5          ; Inputs:
6          ;   RO = Speed code.
7          ;   R1 = Line index number
8          ;
9 013174 010246 VHSSPD: MOV      R2,-(SP)
10 013176 010346      MOV      R3,-(SP)
11 013200 010446      MOV      R4,-(SP)
12          ;
13          ; Update the LMXPRM table for this line
14          ;
15 013202 110061 000001G      MOVB     RO,LMXPRM+1(R1) ;Store flags for line
16          ;
17          ; Convert TSX-Plus speed code into DHV11 speed code
18          ;
19 013206 010003      MOV      RO,R3          ;Get flags
20 013210 042703 000000C      BIC      #^C<LP$SPD>,R3 ;Clear all but speed flags
21 013214 116303 001372'      MOVB     VHSPCT(R3),R3 ;Convert to DHV11 speed code
22          ;
23          ; Get DHV11 index number
24          ;
25 013220 016102 000000G      MOV      LMXNUM(R1),R2 ;Get DHV11 index number
26 013224 116104 000000G      MOVB     LMXLN(R1),R4 ;Get line # within mux group
27 013230 052704 000000G      BIS      #VF$RIE,R4 ;Set receiver interrupt enable flag
28          ;
29          ; Construct line parameter value for this line
30          ;
31 013234 000303      SWAB     R3          ;Position speed for receive speed
32 013236 010346      MOV      R3,-(SP)
33 013240 072327 0000004      ASH     #4,R3          ;Position speed for transmit speed
34 013244 052603      BIS      (SP)+,R3 ;Combine receive and transmit speeds
35 013246 032700 000000G      BIT      #LP$7BT,R0 ;7 bit characters wanted
36 013252 001003      BNE     2$          ;Br if yes
37 013254 052703 000000G      BIS      #VF$8BT,R3 ;Set 8 bit characters
38 013260 000402      BR      3$          ;
39 013262 052703 000000G 2$:  BIS      #VF$7BT,R3 ;Set 7 bit characters
40 013266 032700 000000G 3$:  BIT      #LP$PAR,R0 ;Parity wanted?
41 013272 001407      BEQ     1$          ;Br if not
42 013274 052703 000000G      BIS      #VF$PAR,R3 ;Enable parity
43 013300 032700 000000G      BIT      #LP$ODD,R0 ;Odd parity wanted?
44 013304 001002      BNE     1$          ;Br if yes
45 013306 052703 000000G      BIS      #VF$EVN,R3 ;Select even parity
46          ;
47          ; Select our line and store the parameter value
48          ;
49 013312 1#:  DISABL ;;;** Disable interrupts **
50 013320 110472 000000G      MOVB     R4,@VH$CSR(R2) ;;;Select our mux line
51 013324 010372 000000G      MOV      R3,@VH$LPR(R2) ;;;Set LPR value for this line
52 013330      ENABL ;;;** Enable interrupts **
53          ;
54          ; Finished
55          ;
56 013336 012604      MOV      (SP)+,R4
57 013340 012603      MOV      (SP)+,R3

```


58 013342 012602
59 013344 000207

MOV (SP)+,R2
RETURN

VHSBRK -- Control break transmission for a DHV11 line

```

1          .SBTTL  VHSBRK -- Control break transmission for a DHV11 line
2          ;-----
3          ; Start or stop transmitting a break to a DHV11 line.
4          ;
5          ; Inputs:
6          ;   R0 = Break control flag (MS#BRK)
7          ;   R1 = Line index number
8          ;
9 013346 010246 VHSBRK: MOV     R2,-(SP)
10 013350 010346      MOV     R3,-(SP)
11          ;
12          ; Get mux index number and line number
13          ;
14 013352 016102 000000G      MOV     LMXNUM(R1),R2 ;Get mux index number
15 013356 016103 000000G      MOV     LMXLN(R1),R3 ;Get # of line within mux
16 013362 052703 000000G      BIS     #VF$RIE,R3 ;Set receiver interrupt enable flag
17          ;
18          ; Set or drop the break control flag within the line control register
19          ;
20 013366      DISABL                    ;** Disable interrupts **
21 013374 110372 000000G      MOVB   R3,@VH$CSR(R2) ;;Select our mux line
22 013400 032700 000000G      BIT     #MS$BRK,R0 ;;Start or stop break?
23 013404 001404      BEQ     1$ ;;Br if stop
24 013406 052772 000000G 000000G      BIS     #VF$BC,@VH$LCR(R2);;Set the break flag for the line
25 013414 000403      BR     2$
26 013416 042772 000000G 000000G 1$:  BIC     #VF$BC,@VH$LCR(R2);;Clear the break flag for the line
27          ;
28          ; Finished
29          ;
30 013424      2$:  ENABL                    ;** Enable interrupts **
31 013432 012603      MOV     (SP)+,R3
32 013434 012602      MOV     (SP)+,R2
33 013436 000207      RETURN

```

DLCLOCK -- Timer driven routine for DL11 lines

```

1          .SBTTL  DLCLOCK -- Timer driven routine for DL11 lines
2          ;-----
3          ; DLCLOCK is a timer-driven routine called periodically to check on the
4          ; status of a DL11 line.
5          ;
6          ; Inputs:
7          ; R1 = Physical line index number
8          ;
9          013440 DLCLOCK:
10         ;
11         ; See if this line has been stolen by some special device handler
12         ;
13         013440 027127 000000G 000000G          CMP      @INVEC(R1),#INRECV;HAS LX HANDLER STOLEN INTERRUPT VECTOR
14         013446 101050                          BHI      20$                ;BR IF YES
15         ;
16         ; Check for lost input interrupts
17         ;
18         013450 032771 000000G 000000G          BIT      #RCVDON,@RSR(R1);IS AN INPUT CHAR PENDING NOW?
19         013456 001424                          BEQ      1$                ;BR IF NOT
20         013460 032761 000000G 000000G          BIT      #IITIM,LSW5(R1);HAVE WE STARTED TIMER YET?
21         013466 001415                          BEQ      15$               ;BR IF NOT
22         013470                          DISABL                          ;** DISABLE ** (NEEDED FOR FLAKEY DL11'S)
23         013476 042771 000000G 000000G          BIC      #RDINT,@RSR(R1) ;WE SEEM TO HAVE LOST AN INTERRUPT
24         013504 052771 000000G 000000G          BIS      #RDINT,@RSR(R1) ;TRY TO FORCE AN INTERRUPT
25         013512                          ENABL                          ;** ENABLE **
26         013520 000403                          BR       1$
27         013522 052761 000000G 000000G 15$:    BIS      #IITIM,LSW5(R1);START INPUT INTERRUPT TIMER
28         ;
29         ; Check for lost output interrupts
30         ;
31         013530 032761 000000G 000000G 1$:    BIT      #OITIM,LSW5(R1);Have we started timer interval?
32         013536 001411                          BEQ      13$               ;Br if not
33         013540 032761 000000G 000000G          BIT      #XCHAR,LSW3(R1);Are we still waiting for interrupt?
34         013546 001410                          BEQ      20$               ;Br if not
35         013550 042761 000000G 000000G          BIC      #XCHAR,LSW3(R1);Say wait is over
36         013556 004737 000000G                      CALL     DLSTRT           ;Try to start transmitter
37         013562 052761 000000G 000000G 13$:    BIS      #OITIM,LSW5(R1);Start timed interval
38         ;
39         ; Finished
40         ;
41         013570 000207 20$:    RETURN

```

DZCLOCK -- Timer driven routine for DZ11 lines

```

1                                     .SBTTL  DZCLOCK -- Timer driven routine for DZ11 lines
2                                     ;-----
3                                     ; DZCLOCK is called periodically from the clock routine to check on the
4                                     ; status of DZ11 lines.
5                                     ;
6                                     ; Inputs:
7                                     ; R1 = Physical line index number
8                                     ;
9 013572 010246 DZCLOCK: MOV      R2, -(SP)
10 013574 010346      MOV      R3, -(SP)
11                                     ;
12                                     ; Get DZ11 mux index number and number of line within mux
13                                     ;
14 013576 016102 000000G      MOV      LMXNUM(R1), R2 ;Get DZ11 mux index number
15 013602 016103 000000G      MOV      LMXLN(R1), R3 ;Get # of line within mux (0-7)
16 013606 116303 001312'      MOVB    MXLBIT(R3), R3 ;Get line select bit for mux register
17                                     ;
18                                     ; Check for lost input interrupts
19                                     ;
20 013612 032772 000000G 000000G      BIT      #RDONE, @MXCSR(R2); Does DZ11 have a pending input character?
21 013620 001415      BEQ      3$ ; Br if not
22 013622 032761 000000G 000000G      BIT      #IITIM, LSW5(R1); Have we started input interrupt timer?
23 013630 001406      BEQ      1$ ; Br if not
24 013632 042772 000000G 000000G      BIC      #RIE, @MXCSR(R2); Drop receiver interrupt enable
25 013640 052772 000000G 000000G      BIS      #RIE, @MXCSR(R2); and raise it again to try to force interrupt
26 013646 052761 000000G 000000G 1$:   BIS      #IITIM, LSW5(R1); Start input interrupt timer
27                                     ;
28                                     ; Check for lost output interrupts
29                                     ;
30 013654 032761 000000G 000000G 3$:   BIT      #OITIM, LSW5(R1); Have we started timer interval?
31 013662 001415      BEQ      13$ ; Br if not
32 013664 032761 000000G 000000G      BIT      #XCHAR, LSW3(R1); Are we still waiting for interrupt?
33 013672 001414      BEQ      9$ ; Br if not
34 013674 032761 000000G 000000G      BIT      #CTRLS, LSW3(R1); Is output suspended due to ctrl-S?
35 013702 001010      BNE      9$ ; Br if yes
36 013704 042761 000000G 000000G      BIC      #XCHAR, LSW3(R1); Say wait is over
37 013712 004737 000000G      CALL    DZSTRT ; Try to start transmitter
38 013716 052761 000000G 000000G 13$:  BIS      #OITIM, LSW5(R1); Start timed interval
39                                     ;
40                                     ; Finished
41                                     ;
42 013724 012603 9$:      MOV      (SP)+, R3
43 013726 012602      MOV      (SP)+, R2
44 013730 000207      RETURN

```

DHCLOCK -- Timer driven routine for DH11 lines

```

1          .SBTTL  DHCLOCK -- Timer driven routine for DH11 lines
2          ;-----
3          ; DHCLOCK is called every 0.5 seconds from the clock driven routine to
4          ; do checking for DH11 and DHV11 lines
5          ;
6          ; Inputs:
7          ; R1 = Physical line index number.
8          ;
9 013732   VHCLOCK:
10 013732  DHCLOCK:
11          ;
12          ; Set flag which requests clock driven output processing for the line
13          ;
14 013732  052761  000000G 000000G      BIS    #$DHCDO,LSW10(R1)      ;Request clock-driven output
15 013740  005237  000000G      INC    NEDCDO                ;Say clock processing needed
16          ;
17          ; Finished
18          ;
19 013744  000207      RETURN

```

SYSDIE -- Fatal system halt

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15 013746 010046
16 013750 010146
17 013752 010246
18 013754 010346
19 013756 010446
20 013760 010546
21
22
23
24 013762 005037 000000G
25 013766 005037 000000G
26
27
28
29 013772 012701 001056'
30 013776 004737 014330'
31
32
33
34 014002 013701 000000G
35 014006 004737 014376'
36
37
38
39 014012 013701 000000G
40 014016 062701 000002'
41 014022 004737 014330'
42
43
44
45 014026 012701 001120'
46 014032 004737 014330'
47 014036 013701 000000G
48 014042 004737 014376'
49
50
51
52
53 014046 013701 000000G
54 014052 001422
55
56
57

```

.SBTTL SYSDIE -- Fatal system halt
-----
; SYSDIE is entered from the system SYSHLT routine when a system
; crash is occurring because a DIE macro was executed.
;
; Inputs:
;   DIEMSG = Address of error message to print.
;   DIEARG = Argument value to print with error message.
;   DIEPC  = Address of call to SYSHLT.
;   DIESP  = Stack pointer at time of crash.
;   TRPAR5 = Kernel PAR5 contents at time of crash.
;
; Save all registers on the stack.
;
SYSDIE: MOV     R0, -(SP)
        MOV     R1, -(SP)
        MOV     R2, -(SP)
        MOV     R3, -(SP)
        MOV     R4, -(SP)
        MOV     R5, -(SP)
;
; Initialize some cells that are used to pass information to TSDUMP
;
        CLR     DMPOVL      ; Overlay name
        CLR     DMPHND      ; Handler name
;
; Print message heading.
;
        MOV     #TXFSE, R1   ; "FATAL SYSTEM ERROR..."
        CALL    HLTprt      ; PRINT IT
;
; Print abort location.
;
        MOV     DIEPC, R1    ; GET ADDRESS OF CALL TO SYSHLT
        CALL    HLTOct      ; PRINT OCTAL VALUE
;
; Print error message.
;
        MOV     DIEMSG, R1   ; GET ADDRESS OF ERROR MESSAGE
        ADD     #DIEBAS, R1
        CALL    HLTprt      ; PRINT IT
;
; Print argument value.
;
        MOV     #TXARG, R1   ; "ARGUMENT VALUE = "
        CALL    HLTprt      ; PRINT HEADING
        MOV     DIEARG, R1   ; GET ARGUMENT VALUE
        CALL    HLTOct      ; PRINT OCTAL VALUE
;
; If the argument value is in the par 5 range, it is probably in
; a system overlay.
;
        MOV     TRPAR5, R1   ; Get the KPAR5 value
        BEQ    21$          ; Br if zero - print value
;
; Check for base address of system overlay region.
;

```

SYSDIE -- Fatal system halt

```

58 014054 013700 000000G      MOV      OVRADD,R0      ;Find address of the overlay table
59 014060 026001 000000G      1$:      CMP      O.PAR(R0),R1  ;Check PAR5 with mapped overlay address
60 014064 001426                BEQ      2$            ;Br if values match
61 014066 062700 000006      ADD      #6,R0         ;Find the next overlay region
62 014072 021027 004537      CMP      (R0),#4537    ;Check for end of table, a <JSR R5,$OVRH>
63 014076 001370                BNE      1$            ;Br to check next table entry
64
65                          ; Check for base of loaded handler region.
66
67 014100 013700 000000G      MOV      NUMDEV,R0     ;Get highest byte index for loaded devices
68 014104 026001 000000G      11$:     CMP      HANPAR(R0),R1 ;Check PAR5 with mapped handler address
69 014110 001446                BEQ      12$          ;Br if values match
70 014112 162700 000002      SUB      #2,R0         ;Offset to next device handler
71 014116 002372                BGE      11$          ;Br to check next handler entry
72
73                          ; Print kernel PAR5 contents.
74
75 014120 012701 001137'      21$:     MOV      #TXPAR5,R1    ;"PAR5 VALUE = "
76 014124 004737 014330'      CALL     HLTprt        ;PRINT HEADING
77 014130 013701 000000G      MOV      TRPAR5,R1    ;GET ARGUMENT VALUE
78 014134 004737 014376'      CALL     HLTOCT        ;PRINT OCTAL VALUE
79 014140 000445                BR       3$            ;Go halt the system
80
81                          ; PAR5 address located in mapped system region - report segment ID.
82
83 014142 016004 000000G      2$:      MOV      O.ADR(R0),R4  ;Get the rad50 overlay identifier
84 014146 163700 000000G      SUB      OVRADD,R0    ;Sub the base address of the overlay table
85 014152 010003                MOV      R0,R3         ;Move to low-order address
86 014154 005002                CLR      R2            ;Clear high-order address
87 014156 071227 000006      DIV      #6,R2        ;Divide by 6 (# bytes/ overlay table entry)
88 014162 005202                INC      R2            ;Normalize base to one
89 014164 012701 001155'      MOV      #TXSEG,R1    ;"Seg. value ="
90 014170 004737 014330'      CALL     HLTprt        ;Print heading
91 014174 010201                MOV      R2,R1         ;Get the segment number
92 014176 004737 014376'      CALL     HLTOCT        ;Print the octal segment number
93 014202 012701 001173'      MOV      #TXOID,R1    ;"Overlay: "
94 014206 004737 014330'      CALL     HLTprt        ;Print heading
95 014212 010401                MOV      R4,R1         ;Restore the overlay identifier
96 014214 010137 000000G      MOV      R1,DMPQVL    ;Pass overlay name to TSDUMP
97 014220 004737 014454'      CALL     HLTRAD        ;Print the rad50 overlay identifier
98 014224 000413                BR       3$            ;Go halt the system
99
100                          ; PAR5 address located in loaded device handler - report device name.
101
102 014226 016002 000000G      12$:     MOV      PNAME(R0),R2  ;Get the RAD50 device name
103 014232 012701 001205'      MOV      #TXDEV,R1    ;"Device name : "
104 014236 004737 014330'      CALL     HLTprt        ;Print heading
105 014242 010201                MOV      R2,R1         ;Get the device name
106 014244 010137 000000G      MOV      R1,DMPHND    ;Pass device name to TSDUMP
107 014250 004737 014454'      CALL     HLTRAD        ;Print the rad50 device name
108
109                          ; Print stack pointer at time of crash
110
111 014254 012701 001223'      3$:      MOV      #SPTXT,R1    ;Point to message heading
112 014260 004737 014330'      CALL     HLTprt        ;Print the heading
113 014264 013701 000000G      MOV      DIESP,R1     ;Get SP at time of crash
114 014270 004737 014376'      CALL     HLTOCT        ;Print it

```

SYSDIE -- Fatal system halt

```
115 ;
116 ; See if we should call the system crash dump module or halt the system
117 ;
118 014274 105737 000000G          TSTB    VSYDMP          ;Should we do a system dump?
119 014300 001001                BNE     4$              ;Br if yes
120 014302 000000                HALT                       ;Halt the system
121 ;
122 ; Enter system overlay to produce a crash dump
123 ;
124 014304 013701 000000G          4$:    MOV     DIEMSG,R1          ;Get address of error message text
125 014310 062701 000002'          ADD     #DIEBAS,R1
126 014314 012702 000000G          MOV     #DMPTXT,R2        ;Point to area where we pass message
127 014320 112122                5$:    MOVB   (R1)+,(R2)+    ;Store message text
128 014322 001376                BNE     5$              ;Loop till all of message moved
129 014324 000137 000000G          JMP     DODUMP           ;Enter dump routine
```



```

1
2 ; -----
3 ; HLTprt is called to print an ASCIZ string on the console terminal.
4 ;
5 ; Inputs:
6 ; R1 = Address of ASCIZ string to print.
7 ;
8 HLTprt: MOV R1, -(SP)
9 1$: MOV (R1)+, R0 ; GET NEXT CHAR FROM TEXT STRING
10 BEQ 2$ ; BR IF HIT END OF STRING
11 CMPB RO, #200 ; END WITHOUT CR-LF?
12 BEQ 3$ ; BR IF YES
13 CALL HLTCHR ; SEND CHAR TO TERMINAL
14 BR 1$ ; GO GET NEXT CHAR
15 ; Print CR-Lf.
16 2$: MOV #CR, R0 ; PRINT CR
17 CALL HLTCHR
18 MOV #LF, R0 ; PRINT LF
19 CALL HLTCHR
20 ; Finished
21 3$: MOV (SP)+, R1
22 RETURN
23 ; -----
24 ; HLTDOCT is called to convert a binary value to an octal character string
25 ; and print that string on the console terminal.
26 ;
27 ; Inputs:
28 ; R1 = Binary value to be converted and printed.
29 ;
30 HLTDOCT: MOV R1, -(SP)
31 MOV R2, -(SP)
32 MOV #6, R2 ; GET # OF OCTAL DIGITS IN RESULT STRING
33 CLR R0 ; SET FOR SHIFT
34 ASHC #1, R0 ; SHIFT 1ST BIT INTO R0
35 BR 2$ ; ENTER CONVERSION LOOP
36 1$: CLR R0 ; SET FOR SHIFT
37 ASHC #3, R0 ; SHIFT AN OCTAL DIGIT INTO R0
38 2$: ADD #'0, R0 ; CONVERT BINARY VALUE TO ASCII CHARACTER
39 CALL HLTCHR ; PRINT A CHARACTER
40 SOB R2, 1$ ; LOOP TO PRINT REST OF VALUE
41 MOV #TXNUL, R1 ; NOW PRINT CR-LF
42 CALL HLTprt
43 MOV (SP)+, R2
44 MOV (SP)+, R1
45 RETURN
46 ; -----
47 ; HLTRAD is called to convert a RAD50 value to an ascii character string
48 ; and print that string on the console terminal.
49 ;
50 ; Inputs:
51 ; R1 = RAD50 value to be converted and printed.
52 ;
53 HLTRAD: MOV R1, -(SP)
54 MOV R2, -(SP)
55 CLR R0 ; Clear high order
56 DIV #50*50, R0 ; Divide for 1st byte
57

```

SYSDIE -- Fatal system halt

```

58 014466 116000 014542'      MOVB    R50CHR(R0),R0      ;Get output character
59 014472 004737 014612'      CALL    HLTCHR           ;Print a character
60 014476 005000                CLR     RO               ;Clear high order
61 014500 071027 000050        DIV     #50,R0          ;Divide for 2nd byte
62 014504 116000 014542'      MOVB    R50CHR(R0),R0      ;Get output character
63 014510 004737 014612'      CALL    HLTCHR           ;Print a character
64 014514 116100 014542'      MOVB    R50CHR(R1),R0      ;Get output character
65 014520 004737 014612'      CALL    HLTCHR           ;Print a character
66 014524 012701 001136'      MOV     #TXNUL,R1        ;Now print cr-lf
67 014530 004737 014330'      CALL    HLTprt          ;
68 014534 012602                MOV     (SP)+,R2
69 014536 012601                MOV     (SP)+,R1
70 014540 000207                RETURN
71
72                                .EVEN
73
74
75 014542      040      101      102  R50CHR: .ASCII / ABCDEFGHIJKLMNOPQRSTUVWXYZ#. 0123456789/
76
77
78                                .EVEN
79
80                                ;-----
81                                ; HLTCHR is called to print a single character on the console terminal
82                                ; during a system crash.
83                                ;
84                                ; Inputs:
85                                ; RO = Character to print.
86                                ;
87 014612 032737 000000G 000000G HLTCHR: BIT     #TRRDY,@#CTTSR ; IS TERMINAL READY FOR ANOTHER CHARACTER?
88 014620 001774                BEQ     HLTCHR           ; BR IF NOT
89 014622 110037 000000G        MOVB    RO,@#CTTBR       ; SEND CHARACTER TO CONSOLE TERMINAL
90 014626 000207                RETURN

```



```

58 015014 005021          CLR      (R1)+      ;Zero the area
59 015016 000773          BR       36$
60
61          ; Initialize free list of swap command packets
62
63 015020 013701 000000G 35$:     MOV      SCPFHD,R1      ;Point to area where packets are
64 015024 001413          BEQ      37$            ;Br if nothing to initialize
65 015026 012702 177777G          MOV      #NSCP-1,R2     ;Get # packets -1
66 015032 010103          38$:     MOV      R1,R3        ;Get pointer to current packet
67 015034 062703 000000G          ADD      #SP$$SZ,R3    ;Get pointer to next packet
68 015040 010361 000000G          MOV      R3,SP$LNK(R1) ;Make our packet point to next
69 015044 010301          MOV      R3,R1         ;Get pointer to next packet
70 015046 077207          SOB      R2,38$        ;Loop till all packets but last linked in
71 015050 005061 000000G          CLR      SP$LNK(R1)    ;Say last packet is end of list
72
73          ; Initialize the free chain of monitor control blocks
74
75 015054 013701 000000G 37$:     MOV      MONFQH,R1     ;Get base of area for control blocks
76 015060 013702 000000G          MOV      VMXMON,R2     ;Get # monitor blocks
77 015064 001413          BEQ      28$            ;Br if none wanted
78 015066 005302          DEC      R2            ;Get one less than # wanted
79 015070 001407          BEQ      42$            ;Br if only one wanted
80 015072 010103          29$:     MOV      R1,R3        ;Get address of current block
81 015074 062703 000000G          ADD      #JM$$SZ,R3    ;Get address of next control block
82 015100 010361 000000G          MOV      R3,JM$LNK(R1) ;Make current block point to next
83 015104 010301          MOV      R3,R1         ;Get address of next block
84 015106 077207          SOB      R2,29$        ;Br if more to allocate
85 015110 005061 000000G 42$:     CLR      JM$LNK(R1)   ;Zero last link
86
87          ; Initialize the tables that keep track of free space in job swap file
88
89 015114 013700 000000G 28$:     MOV      VSWPSL,R0     ;Get # slots in swap file
90 015120 001412          BEQ      33$            ;Br if no swap file
91 015122 013702 000000G          MOV      SWPPOS,R2     ;Point to table that has starting blk #'s
92 015126 013703 000000G          MOV      SWPJOB,R3     ;Point to table that has job #'s
93 015132 005004          CLR      R4            ;1st slot is at block 0
94 015134 010422          32$:     MOV      R4,(R2)+      ;Set block # for this slot
95 015136 005023          CLR      (R3)+         ;Say no job using this slot now
96 015140 063704 000000G          ADD      SLTSIZ,R4     ;Add # blocks used by a slot
97 015144 077005          SOB      R0,32$        ;Loop till all slots initialized
98
99          ; Initialize vector for each multiplexor that is used to map from
100         ; the Mux line number to the TSX-Plus logical line number
101
102 015146 012701 000000G 33$:     MOV      #LSTMX,R1     ;Get index # of last mux
103 015152 001470          BEQ      27$            ;Branch if no mux's to initialize
104 015154 016103 000000G 15$:     MOV      MXLNT(R1),R3   ;GET ADDRESS OF MUX MAPPING TABLE
105 015160 012700 000020          MOV      #16.,R0       ;ZERO 16 BYTES IN TABLE
106 015164 105023          17$:     CLR      (R3)+         ;
107 015166 077002          SOB      R0,17$        ;
108 015170 162701 000002          SUB      #2,R1          ;More mux tables to init?
109 015174 003367          BGT      15$            ;Loop if yes
110 015176 012701 000000G 16$:     MOV      #LSTHL,R1     ;GET INDEX # OF LAST PHYSICAL LINE
111 015202 032761 000000G 000000G 19$:  BIT      #$HARD,LSW3(R1) ;Is this line connected to hardware?
112 015210 001410          BEQ      18$            ;Br if not
113 015212 016102 000000G          MOV      LMXNUM(R1),R2 ;IS THIS LINE CONNECTED TO A MUX?
114 015216 001405          BEQ      18$            ;BR IF NOT

```

EXCINI -- Final system initialization

```

115 015220 016202 000000G      MOV      MXLNT(R2),R2      ;GET ADDRESS OF MAP VECTOR FOR THIS MUX
116 015224 066102 000000G      ADD      LMXLN(R1),R2      ;ADD OFFSET WITHIN VECTOR
117 015230 110112                MOVVB    R1,(R2)          ;SET TSX LINE # WITHIN MAP VECTOR
118 015232 162701 000002      18$:    SUB      #2,R1          ;GET INDEX # OF NEXT LINE
119 015236 003361                BGT      19$             ;LOOP IF MORE LINES
120                                ;
121                                ; Enable interrupts for multiplexors
122                                ;
123 015240 105737 000000G      TSTB    PROFLG           ;Are we running on a Pro?
124 015244 001033                BNE      27$             ;Br if yes -- Don't have any mux's on pro
125 015246 012701 000000G      MOV      #LSTMX,R1        ;Get index # of last mux
126 015252 005761 000000G      23$:    TST      MXCSR(R1)    ;Is this mux installed?
127 015256 001423                BEQ      25$             ;Br if not
128 015260 016100 000000G      MOV      MXTYPE(R1),RO    ;Get mux type code
129 015264 020027 000000G      CMP      RO,#CDX$DZ       ;Is this a DZ11?
130 015270 001004                BNE      24$             ;Br if not
131 015272 012771 000000G 000000G  MOV      #INTMX1,@MXCSR(R1);Enable DZ11 interrupts
132 015300 000412                BR       25$
133 015302 020027 000000G      24$:    CMP      RO,#CDX$VH    ;Is this a DHV11?
134 015306 001004                BNE      26$             ;Br if not
135 015310 052771 000000C 000000G  BIS      #<VF$TIE!VF$RIE>,@VH$CSR(R1);Enable DHV11 interrupts
136 015316 000403                BR       25$
137 015320 052771 000000C 000000G  26$:    BIS      #<HF$TIE!HF$RIE>,@MH$SCR(R1);Enable DH11 interrupts
138 015326 162701 000002      25$:    SUB      #2,R1          ;More mux's to initialize?
139 015332 003347                BGT      23$             ;Loop if yes
140                                ;
141                                ; Initialize system message buffer free list.
142                                ;
143 015334 013703 000000G      27$:    MOV      SNMSHD,R3      ;GET ADDRESS OF 1ST MESSAGE BUFFER
144 015340 012704 177777G      MOV      #<NMSNMB-1>,R4    ;GET # MESSAGE BUFFERS -1
145 015344 010302 7$:    MOV      R3,R2          ;GET ADDRESS OF MESSAGE BUFFER
146 015346 062702 000000G      ADD      #SB#$SZ,R2        ;POINT TO FOLLOWING BUFFER
147 015352 010263 000000G      MOV      R2,SB$LNK(R3)    ;CHAIN TOGETHER THE ENTRIES
148 015356 010203                MOV      R2,R3            ;MOVE ON TO NEXT ENTRY
149 015360 077407                SOB      R4,7$           ;DO ALL BUT LAST
150 015362 005063 000000G      CLR      SB$LNK(R3)       ;SET FORWARD LINK FOR LAST ENTRY TO ZERO
151                                ;
152                                ; Initialize INSTALLED program table
153                                ;
154 015366 004737 016034'      CALL     INSINI           ;Initialize installed program table
155                                ;
156                                ; Initialize CL table information
157                                ;
158 015372 012704 000000C      MOV      #2*<CLTOTL-1>,R4;Get index to last CL unit
159 015376 002407                BLT      39$             ;Br if there are no CL units
160 015400 016400 000000G      41$:    MOV      CL$EPS(R4),RO  ;Get pointer to EOF string buffer
161 015404 001401                BEQ      45$             ;Br if no string buffer
162 015406 105010                CLRB    (RO)            ;Say no EOF string
163 015410 162704 000002      45$:    SUB      #2,R4          ;Get next index
164 015414 002371                BGE      41$             ;Loop if more CL units
165                                ;
166                                ; Initialize the file management tables
167                                ;
168 015416      39$:    OCALL    USRINI           ;Call TSUSR initialization routine
169                                ;
170                                ; Initialize spool buffer list
171                                ;

```

EXCINI -- Final system initialization

```

172 015424 105737 000000G          TSTB   NSPLDV          ;Are there any spooled devices?
173 015430 001403          BEQ    11$             ;Br if not
174 015432          DCALL  SPLINI          ;Initialize the spooling system
175          ;
176          ; Initialize the record locking system
177          ;
178 015440 005737 000000G 11$:   TST    VMXSF          ;Is record locking support wanted?
179 015444 001402          BEQ    12$             ;Br if not
180 015446 004777 000000G          CALL   @LOKINI        ;Initialize the shared file system
181          ;
182          ; Initialize the message communication system
183          ;
184 015452 005737 000000G 12$:   TST    VMAXMC        ;Is message communication support wanted?
185 015456 001403          BEQ    13$             ;Br if not
186 015460          DCALL  MSGINI          ;Initialize the message system
187          ;
188          ; Initialize the data caching facility
189          ;
190 015466 005737 000000G 13$:   TST    CSHALC        ;Is data caching wanted?
191 015472 001402          BEQ    43$             ;Br if not
192 015474 004777 000000G          CALL   @CSHINI        ;Initialize data caching facility
193          ;
194          ; Initialize the PLAS system
195          ;
196 015500 005737 000000G 43$:   TST    VPLAS          ;Is PLAS support included in system?
197 015504 001403          BEQ    44$             ;Br if not
198 015506          DCALL  PLSINI          ;Do PLAS initialization
199          ;
200          ; Initialize the display window management system
201          ;
202 015514 005737 000000G 44$:   TST    VMXWIN        ;Is window support wanted?
203 015520 001403          BEQ    34$             ;Br if not
204 015522          DCALL  WININI          ;Initialize window system
205          ;
206          ; Connect clock interrupt to clock interrupt routine
207          ;
208 015530 012737 000000G 000100 34$:   MOV    #CLKINT,@#100    ;Set up clock interrupt vector
209 015536 105737 000000G          TSTB   PROFLG          ;Is this a PRO?
210 015542 001410          BEQ    22$             ;Br if not
211 015544 012737 000000G 000230  MOV    #CLKINT,@#230    ;380 clock interrupt vector
212 015552 012737 000340 000232  MOV    #340,@#232
213 015560 005737 000000G          TST    @#PCCR2        ;Access CSR2 to start clock interrupts
214          ;
215          ; Initialize time-sharing line parameters and speeds
216          ;
217 015564 004737 015714' 22$:   CALL   INISPD          ;Initialize time-sharing line speeds
218          ;
219          ; Start lines that specified $START when genned.
220          ;
221 015570 012701 000002          MOV    #2,R1          ;INDEX # OF 1ST LINE
222 015574 032761 000000G 000000G 2$:   BIT    ##$START,ILSW2(R1); DOES THIS LINE WANT AUTO STARTUP?
223 015602 001413          BEQ    3$              ;BR IF NOT
224 015604 032761 000000G 000000G  BIT    ##$DEAD,LSW3(R1); IS THIS LINE INSTALLED?
225 015612 001007          BNE    3$              ;BR IF NOT
226 015614 032761 000000G 000000G  BIT    ##$PHONE,ILSW2(R1); IS THIS A DIAL-UP LINE?
227 015622 001003          BNE    3$              ;BR IF IT IS (NO AUTO STARTUP THEN)
228 015624 005000          CLR    R0              ;No secondary start-up command file

```

EXCINI -- Final system initialization

```
229 015626 004737 001606'          CALL  INITLN          ; INITIATE THE LINE
230 015632 062701 000002          3$:  ADD   #2,R1          ; ADVANCE JOB #
231 015636 020127 000000G        CMP   R1,#LSTPL        ; MORE TO CHECK?
232 015642 101754                BLOS  2$              ; BR IF YES
233                                ;
234                                ; Start any detached jobs
235                                ;
236 015644 012701 000000G        MOV   #FSTD,L,R1       ; # OF FIRST DETACHED JOB
237 015650 020127 000000G        6$:  CMP   R1,#LSTD,L        ; DONE ALL DETACHED JOBS?
238 015654 101013                BHI   4$              ; BR IF YES
239 015656 016102 000000G        MOV   LSUCF(R1),R2    ; DOES THIS JOB HAVE A START-UP COMMAND FILE?
240 015662 001405                BEQ   5$              ; BR IF NOT
241 015664 105712                TSTB  (R2)            ; IS COMMAND FILE NAME NULL?
242 015666 001403                BEQ   5$              ; BR IF YES
243 015670 005000                CLR   R0              ; No secondary start-up command file
244 015672 004737 001606'          CALL  INITLN          ; INITIATE THE LINE
245 015676 062701 000002          5$:  ADD   #2,R1          ; CHECK NEXT LINE
246 015702 000762                BR    6$              ;
247 015704          4$:          ;
248                                ;
249                                ; Finished system initialization
250 015704 105037 000000G        CLRB  INITFL          ; SAY SYSTEM INITIALIZATION IS FINISHED
251                                ;
252                                ; Enter job scheduler to wait for first job to run
253                                ;
254 015710 000137 000000G        JMP   EXEC            ; ENTER JOB SCHEDULER
```

INISPD -- Initialize time-sharing line speeds

```

1          .SBTTL  INISPD -- Initialize time-sharing line speeds
2          ;-----
3          ; INISPD is called to initialize the transmit/receive speeds for
4          ; time-sharing lines.
5          ;
6 015714 010146 INISPD: MOV      R1,-(SP)
7 015716 010246          MOV      R2,-(SP)
8          ;
9          ; Begin loop to set each line
10         ;
11 015720 012701 000000G          MOV      #LSTHL,R1          ;Get index to last hardware line
12         ;
13         ; Skip this line if it is dead or not connected to hardware
14         ;
15 015724 032761 000000G 000000G 1$: BIT      #$HARD,LSW3(R1) ;Is this line connected to hardware?
16 015732 001432          BEQ      2$          ;Br if not
17 015734 032761 000000G 000000G          BIT      #$DEAD,LSW3(R1) ;Is this line installed?
18 015742 001026          BNE      2$          ;Br if not
19         ;
20         ; Set the speed of this line
21         ;
22 015744 116100 000001G          MOVB     LMXPRM+1(R1),R0 ;Get speed parameters
23         ;
24         ; Initialize speed to 9600 baud if autobaud was specified for line
25         ;
26 015750 032761 000000G 000000G          BIT      #$AUTO,ILSW2(R1);Is autobaud wanted for this line?
27 015756 001402          BEQ      3$          ;Br if not
28 015760 012700 000000G          MOV      #S9600,R0          ;Set speed to 9600
29 015764 016102 000000G          3$: MOV      LCDTYP(R1),R2          ;Get device type code for this line
30 015770 004772 000000G          CALL     @CDSSPD(R2)          ;Call hardware-dependent routine to set speed
31         ;
32         ; Convert $TDEAD lines (deaded with TSXMOD) to $DEAD lines
33         ;
34 015774 020127 000000G          CMP      R1,#LSTPL          ;Is this a time-sharing line?
35 016000 101007          BHI      2$          ;Skip if not (skip sub, det & io lines)
36 016002 032761 000000G 000000G          BIT      #$TDEAD,LSW11(R1) ;Do we want this line to be dead?
37 016010 001403          BEQ      2$          ;Br if not
38 016012 052761 000000G 000000G          BIS      #$DEAD,LSW3(R1) ;Flag line as dead
39         ;
40         ; See if there are more lines
41         ;
42 016020 162701 000002          2$: SUB      #2,R1          ;Are there more lines to do?
43 016024 003337          BGT      1$          ;Br if yes
44         ;
45         ; Finished
46         ;
47 016026 012602          MOV      (SP)+,R2
48 016030 012601          MOV      (SP)+,R1
49 016032 000207          RETURN

```


INSINI -- Initialize installed program table

```

1          .SBTTL  INSINI -- Initialize installed program table
2          ;-----
3          ; Initialize the installed program table.
4          ;
5 016034 010246  INSINI: MOV      R2,-(SP)
6 016036 010346          MOV      R3,-(SP)
7          ;
8          ; Initially, zero the entire table
9          ;
10 016040 013702 000000G          MOV      INSTBL,R2          ;Point to start of table
11 016044 005022          1$:    CLR      (R2)+          ;Zero the table
12 016046 020237 000000G          CMP      R2,INSTBN          ;Reached end of table?
13 016052 103774          BLO      1$          ;Loop if not
14          ;
15          ; Now install certain system programs
16          ;
17 016054 013702 000000G          MOV      INSTBL,R2          ;Point to 1st table entry
18 016060 012703 001412'          MOV      #SRFPRG,R3          ;Point to table with info about sys programs
19          ;
20          ; Set file spec for program
21          ;
22 016064 013762 000000G 000000G 2$:    MOV      SYNAME,II$NAM(R2); Set SY as device name
23 016072 012362 000002G          MOV      (R3)+,II$NAM+2(R2); Set 1st 3 chars of program name
24 016076 012362 000004G          MOV      (R3)+,II$NAM+4(R2); Set 2nd 3 chars of program name
25 016102 013762 001350' 000006G          MOV      R5OSAV,II$NAM+6(R2); Set SAV as file extension
26          ;
27          ; Set run attribute flags
28          ;
29 016110 012362 000000G          MOV      (R3)+,II$FLG(R2); Set run attribute flags
30          ;
31          ; Set privileges for program
32          ;
33 016114 052762 000000G 000000G          BIS      #PO$DBG,II$NPV(R2); Set NODEBUG privilege flag
34 016122 012704 000000G          3$:    MOV      #II$PRV,R4          ;Assume we will set some privileges
35 016126 012300          MOV      (R3)+,R0          ;Are there any privilege flags?
36 016130 001412          BEQ      4$          ;Br if not
37 016132 002003          BGE      5$          ;Br if we are to set privileges
38 016134 005400          NEG      R0          ;Get positive offset
39 016136 012704 000000G          MOV      #II$NPV,R4          ;Point to reset-privilege vector
40 016142 005300          5$:    DEC      R0          ;Convert to offset
41 016144 006300          ASL      R0          ;Convert to word offset
42 016146 060004          ADD      R0,R4          ;Point to word in vector to change
43 016150 060204          ADD      R2,R4          ;Add address of install table entry
44 016152 012314          MOV      (R3)+,(R4)          ;Set bits in install table entry
45 016154 000762          BR      3$          ;Go see if more privileges for program
46          ;
47          ; See if there are more programs to install
48          ;
49 016156 062702 000000G          4$:    ADD      #II$SZ,R2          ;Point to next install table entry
50 016162 020327 001606'          CMP      R3,#SRFEND          ;Installed all system programs?
51 016166 103736          BLO      2$          ;Loop if not
52          ;
53          ; Finished
54          ;
55 016170 012603          MOV      (SP)+,R3
56 016172 012602          MOV      (SP)+,R2
57 016174 000207          RETURN

```

TSEXC2 -- Misc. TSX-Plus Execu MACRO V05.05 Wednesday 18-Jan-89 15:26 Page 54-1
INSINI -- Initialize installed program table

58
59 000001 .END
Errors detected: 0

*** Assembler statistics

Work file reads: 0
Work file writes: 0
Size of work file: 9760 Words (39 Pages)
Size of core pool: 18176 Words (71 Pages)
Operating system: RT-11

Elapsed time: 00:01:31.51
,LP:TSEXC2=DK:TSEXC2/C/N:SYM

\$1STLG	1-51	10-169					
\$AUTO	1-47	10-105	28-14	53-26			
\$CARMN	1-57	5-97	5-103	30-48			
\$CARUP	1-51	5-101	10-171	30-43	30-67		
\$CFABT	1-59						
\$CTRLC	1-70	8-29	8-38				
\$CTRLO	1-47	6-111					
\$CTRLS	1-66	6-111	10-30	48-34			
\$DBGBK	1-104	8-18	12-28				
\$DBGMD	1-54	8-17					
\$DEAD	1-58	10-171	29-15	52-224	53-17	53-38	
\$DEBUG	1-89	13-57					
\$DEFER	1-83	6-125					
\$DETCH	1-36	6-104	10-10				
\$DHBF1	1-86	10-34					
\$DHBF2	1-86	10-34					
\$DHCDO	1-36	49-14					
\$DILUP	1-36	5-81	10-28	10-65	30-44	30-51	
\$DISCN	1-42	7-29	7-33	10-50	10-126	30-53	
\$DODFR	1-83	6-127					
\$DOOFF	1-68	7-31	30-53				
\$FPUEX	1-81	14-11					
\$GCECO	1-83	6-127					
\$GEMAR	1-30	8-13					
\$HARD	1-74	10-171	29-17	52-111	53-15		
\$IITIM	1-78	47-20	47-27	48-22	48-26		
\$INCOR	1-99	5-71	18-27				
\$INIT	1-61	6-9					
\$INKMN	1-60	6-15	7-14	7-41	12-14	13-23	26-36
\$IOMAP	1-71	8-17					
\$LOFCF	1-62	10-124	30-55				
\$MAPOK	1-79	7-42					
\$MLOCK	1-71	8-17					
\$NABRS	1-47	10-108	28-27	28-29			
\$NOABT	1-80	8-18					
\$NOIN	1-68	5-82	7-32	30-50			
\$NOLF	1-54	8-17					
\$NOUCR	1-49	7-11	7-25				
\$OITIM	1-94	47-31	47-37	48-30	48-38		
\$PHONE	1-58	5-98	29-29	52-226			
\$PWKEY	1-31	5-84					
\$RDSAV	1-30	8-13					
\$RNMLK	1-54	8-18					
\$SGQ0	1-90	17-30					
\$SGQ1	1-101	17-56					
\$SGQ1A	1-101	17-81					
\$SGQ1B	1-101	17-100					
\$SGQ1C	1-101	17-91					
\$SGQ2	1-101	17-115					
\$SGQ3	1-90	17-37					
\$SOTFN	1-83	27-16	27-18				
\$START	1-58	52-222					
\$SUCF	1-55	5-83					
\$TDEAD	1-58	53-36					
\$UDSPC	1-111	8-138					
\$VIRJB	1-76	7-43					

\$VNOTT	1-66	10-16							
\$XCHAR	1-79	10-36	10-171	47-33	47-35	48-32	48-36		
...V1	7-58	7-72	7-72						
...V2	7-58	7-58#	7-72	7-72	7-72#	7-72#			
ABORT	1-22	11-52	13-71#	14-20	14-28				
ABRTAD	1-60	13-71*							
ABRTCD	1-60	13-72*							
AF\$BYA	1-41	4-49	4-75						
AF\$DUP	1-32	4-29							
AF\$HIE	1-41	4-37	4-41	4-45					
AF\$IND	1-32	4-33							
AF\$IOP	1-41	4-58							
AF\$MEM	1-41	4-83							
AF\$NOI	1-41	4-71							
AF\$NOW	1-41	4-33	4-37	4-41	4-45	4-67	4-71	4-83	
AF\$NPW	1-31	4-71							
AF\$PLK	1-40	4-49							
AF\$SCA	1-41	4-37	4-41	4-45	4-54	4-67	4-71	4-83	
AF\$SET	1-32	4-58							
AF\$UCL	1-32	4-79							
BELL	1-118#	3-27							
C. NUMQ	1-30	6-68*							
CANCPPL	8-104	9-10#							
CANIOT	1-52	8-44							
CANMKT	1-72	8-70							
CARDET	1-77	31-26							
CC\$\$\$SZ	1-102	52-40							
CC\$LNK	1-102	52-41*	52-44*						
CCBHD	1-62	52-38							
CDCLOK	1-56	29-23							
CDGDSS	1-103	30-15							
CDIFLG	1-108	15-58	15-60*						
CDIRTN	1-108	15-63							
CDOFLG	1-108	15-67	15-69*						
CDORTN	1-108	15-72							
CDSDDS	1-103	30-30	30-61	30-81					
CDSSPD	1-36	53-30							
CDX\$DZ	1-48	52-129							
CDX\$VH	1-104	52-133							
CFACFL	1-59								
CFPSAV	1-30								
CHKABT	1-53	11-66	13-29						
CHKPRT	18-116	20-7#							
CHKUSP	1-59	11-50	13-20	14-25					
CINFLG	1-33	8-31*	8-42						
CKMRKT	15-23	24-6#							
CKSCHD	15-34	27-9#							
CKTWAT	15-27	22-6#							
CL\$EPS	1-38	52-160							
CLENUP	7-24	8-9#							
CLK01S	15-85	18-6#							
CLKABD	18-107	28-6#							
CLKCNT	1-84	15-8*	15-18	15-78	15-90	16-19	22-13	24-20	26-71
CLKDAT	15-12	16-13#							
CLKINT	1-57	52-208	52-211						
CLKIOH	18-103	19-9#							

DIESP	1-33	50-113				
DLCLOK	1-25	47-9#				
DLGDSS	1-24	31-11#				
DLSBRK	1-23	33-9#				
DLSDSS	1-24	32-9#				
DLSSPD	1-25	34-9#				
DLSTRT	1-73	47-36				
DM\$CSR	1-106	39-23*	39-24*	40-17*	40-18*	
DM\$LSR	1-106	39-25	40-24*	40-26*		
DMPHND	1-34	50-25*	50-106*			
DMPOVL	1-34	50-24*	50-96*			
DMPXT	1-34	50-126				
DODUMP	1-33	50-129				
DOSCHD	1-81	18-35*				
DOTRMP	1-30	8-129*				
DTLX	1-89	18-46	18-48*			
DZ\$7BT	1-44	38-21				
DZ\$8BT	1-44	38-19				
DZ\$LEN	1-44					
DZ\$ODD	1-44	38-27				
DZ\$PAR	1-44	38-24				
DZCLOK	1-25	48-9#				
DZGDSS	1-24	35-11#				
DZSBRK	1-23	37-10#				
DZSDSS	1-24	36-9#				
DZSSPD	1-23	38-9#				
DZSTRT	1-73	48-37				
EM\$DTL	1-99	3-6	3-6#	18-137		
EM\$FRK	3-7	3-7#				
EM\$JMO	3-8	3-8#				
EM\$KRE	3-9	3-9#	7-74			
EM\$KTP	3-10	3-10#	11-35			
EM\$LMF	3-11	3-11#				
EM\$MIO	3-12	3-12#				
EM\$MPR	3-13	3-13#				
EM\$NQE	3-14	3-14#				
EM\$NSP	3-15	3-15#				
EM\$PFT	3-16	3-16#				
EM\$RIT	3-17	3-17#	11-46			
EM\$SFO	3-18	3-18#				
EM\$SIE	3-19	3-19#				
EM\$SJM	3-21	3-21#	7-8			
EM\$SOF	3-23	3-23#				
EM\$SSE	3-20	3-20#				
EM\$UEI	3-22	3-22#				
EMTBLK	1-32	10-94				
EMTCAD	1-67	6-52*	8-24*			
EMTCAS	1-93	6-52	8-24			
EMTLEV	1-68	6-51*	7-12*			
EMTRAD	1-67	8-22*				
ENQTL	1-96	17-43	17-105	19-43	22-28	27-25
ERRLOC	1-68	7-40	7-51*			
EXCINI	1-21	52-7#				
EXEC	1-52	10-189	52-254			
FORCEX	1-42	10-127	19-30			
FORKIT	3-50#	24-12*	24-44*	24-59		

INVEC	1-76	47-13							
IDHALT	1-71	8-45							
IDQSIZ	1-55	52-14							
IDSTOP	1-71								
ITRMTP	1-65	6-110							
JCDB	1-52	8-108	8-110						
JM\$SZ	1-96	52-81							
JM\$LNK	1-85	52-82*	52-85*						
JOBCCB	1-49	9-46	9-48						
JSTK	1-68	6-138	7-9						
JSTKND	1-68	6-53*							
JSWLOC	1-69	7-52*							
KILJOB	1-65	30-58							
KMNBAS	1-69	7-47	7-69						
KMNCHN	1-70	7-62							
KMNPGS	1-75	5-58	5-60						
KMNSTK	1-70	7-78							
KMNSTR	1-70	7-88							
KMNTOP	1-69	7-46	7-50	7-68					
KMONCE	1-46	5-36							
KPAR6	1-64	6-35*	26-70*						
LABTIM	1-47	10-107*	28-19	28-21*					
LACTIV	1-62	6-20*							
LAFSIZ	1-63	6-22*							
LBASE	1-42	10-156	10-161*						
LBRKCH	1-63	6-26*	8-21*						
LBRKCQ	1-63	6-25*	8-84	8-86*	20-29	20-40	20-42*		
LBSPRI	1-75	5-85*							
LCDTIM	1-87	5-100*	30-31*	30-41*	30-71*				
LCDTYP	1-56	29-22	30-14	53-29					
LCLUNT	1-90	5-29	29-27						
LCMPL	1-72	5-51*	9-28	19-38					
LCOL	1-63	6-21*							
LCONTM	1-64	6-30*							
LCPUHI	1-63	1-84	6-28*	15-19*					
LCPULO	1-64	1-84	6-29*	15-18*					
LCXPAR	1-54	6-35							
LCXTBL	1-32	10-112							
LDKMON	6-142	7-30	7-37#						
LEMTPC	1-99	26-55							
LF	1-117#	3-27	3-27	51-17					
LF\$IN	1-79	8-32							
LHIPCT	1-76	5-87*							
LINBUF	1-61	5-75							
LINCNT	1-61	5-80*							
LINCUR	1-64	6-31*							
LINNXT	1-61	5-76*							
LINPNT	1-61	5-78*							
LINSIZ	1-56	5-79							
LINSPC	1-57	5-79*							
LINSWT	1-37	10-70							
LIOCNT	1-75	5-50*	9-20	9-30					
LIOHLD	1-52	19-17	19-22*						
LITIME	1-66	1-100	5-88*	17-51	17-53*	17-76	22-23	25-28	27-20
LJSW	1-61	6-11*	7-38*	7-39	7-53*				
LMEMIN	1-51	5-65*	10-160*						

LMING	1-89	18-29	18-31*									
LMXLN	1-60	35-18	36-15	37-17	38-32	39-18	40-15	41-27	42-15	43-19	44-15	45-26
	46-15	48-15	52-116									
LMXNUM	1-56	35-17	36-14	37-16	38-33	39-17	40-14	41-26	42-14	43-18	44-14	45-25
	46-14	48-14	52-113									
LMXPRM	1-109	34-13*	38-13*	41-16*	45-15*	53-22						
LNBLKS	1-42	5-49*	5-68*	10-157	10-159*							
LNMAP	1-42	10-68	10-100*									
LNPRIM	1-37	6-117	10-27	10-52	10-64	10-67	10-85*					
LNSBLK	1-74	6-14*										
LNSPAC	1-31	10-175*										
LOFFTM	1-87	5-96*	10-97*	30-32*	30-62*	30-76	30-78*					
LOGCHR	1-79	8-35	8-36									
LOGCR	1-79	8-37										
LOGFLG	1-79	8-32										
LOGOFF	1-21	10-10#										
LOKINI	1-66	52-180										
LOTBUF	1-62	6-16										
LOTNXT	1-62	6-17*										
LOTPNT	1-62	6-18*										
LOTSIZ	1-36	6-19	10-32									
LOTSPC	1-36	6-19*	10-32									
LP#7BT	1-43	38-17	41-36	45-35								
LP#ODD	1-43	38-25	41-44	45-43								
LP#PAR	1-43	38-22	41-41	45-40								
LP#SPD	1-43	38-15	41-21	45-20								
LPARNT	1-38	10-148	10-150*	10-174*								
LPRG1	1-80	20-18										
LPRG2	1-80	20-20										
LPRI	1-75	5-86*	17-22	25-32								
LPROG	1-63	6-23*	6-121	6-121*								
LPROJ	1-63	5-89*	6-81	6-93*	6-120	6-120*						
LQUAN	1-61	1-70	6-10*	7-83*	17-16*	17-21	17-44*	17-61	17-106*			
LRDTIM	1-71	8-19*	21-13	21-15*								
LRTCHR	1-82	21-18										
LSCCA	1-63	6-24*	8-20*									
LSECPT	1-37	10-55	10-77	10-119								
LSLEPH	1-70	1-96	8-14*	22-14*								
LSLEPL	1-70	1-96	8-15*	22-13*	22-17							
LSPND	1-71	8-16*										
LSTACT	1-61	5-77*										
LSTATE	1-81	17-42	17-65	17-70	17-86	17-104	19-28	19-40	21-11	22-11	26-52	27-23
LSTDL	1-58	5-116	6-97	52-237								
LSTHL	1-75	29-11	52-110	53-11								
LSTMX	1-56	52-102	52-125									
LSTPL	1-37	1-56	5-34	5-94	5-112	6-99	10-41	28-10	29-31	52-231	53-34	
LSTSL	1-81	10-147	18-26	19-13	20-17	21-7	22-10	27-11				
LSUCF	1-58	52-239										
LSW	1-36	5-48*	5-71*	5-81*	6-9*	6-104*	7-29	7-31*	7-33*	8-29	8-38*	10-10
	10-16	10-28	10-50	10-65	10-126*	10-166*	14-11*	18-27	30-44	30-51	30-53	
LSW10	1-86	10-34	49-14*									
LSW11	1-31	5-84*	8-13*	8-138*	53-36							
LSW2	1-51	6-109*	6-118	6-118*	6-125	10-98*						
LSW3	1-59	5-82*	5-101	6-111*	6-127*	7-32*	10-30*	10-36	10-173*	29-15	29-17	30-43*
	30-50*	30-67*	47-33	47-35*	48-32	48-34	48-36*	52-111	52-224	53-15	53-17	53-38*
LSW4	1-51	6-15*	7-14	7-41*	10-167*	12-14	13-23	26-36				

R50PRT	3-54#	20-20					
R50SAV	3-55#	54-25					
RCBBAS	1-67	1-74					
RCBEND	1-67	1-74					
RCVDON	1-78	47-18					
RDAR	1-88	8-133*					
RDDR	1-88	8-134*					
RDINT	1-78	47-23	47-24				
RDONE	1-78	48-20					
RIE	1-78	48-24	48-25				
RING	1-77	31-20					
RPAR	1-64	8-131*					
RPDR	1-64	8-132*					
RSR	1-76	31-16	32-18*	32-23*	47-18	47-23*	47-24*
RTSTOP	1-71	8-49					
S\$\$HIP	1-94	17-65					
S\$\$RT	1-94	17-70					
S\$HICP	1-65	17-86	17-92	22-25	25-30		
S\$INWT	1-81	21-11					
S\$IOWT	1-82	9-22	19-28	26-52			
S\$OTFN	1-83	27-19					
S\$OTLO	1-83	27-22					
S\$TMWT	1-94	22-11					
S\$TWFN	1-94	22-27	25-27				
S\$WSMB	1-95	18-21					
S150	1-52						
S9600	1-47	28-30	53-28				
SB\$\$SZ	1-57	52-146					
SB\$LNK	1-57	52-147*	52-150*				
SCHED	1-50	9-37					
SCPFHD	1-53	52-63					
SEMAP	1-67	6-131	7-49				
SETSPD	1-47	28-31					
SHRRCB	1-98	52-55					
SHRRCN	1-98	52-56					
SLTSIZ	1-35	52-96					
SNMSHD	1-57	52-143					
SP\$\$SZ	1-57	52-67					
SP\$LNK	1-44	52-68*	52-71*				
SPCPS	1-68	8-23*					
SPIJ	1-31	6-83*	10-48				
SPLINI	1-66	52-174					
SPSTAT	1-80	20-31*	20-35	20-44*	20-49*		
SPTXT	3-34#	50-111					
SR3FLG	1-111	8-139					
SR3MMR	1-111	8-141*					
SRFEND	4-87#	54-50					
SRFPRG	4-27#	54-18					
SS	1-52	10-185					
SS\$PRT	1-80	20-35	20-44	20-49			
SS\$RUN	1-80	20-31	20-44	20-49			
STOP	1-22	1-60	7-6#	13-73			
STPFLG	1-76	30-20					
STRACT	1-89	21-19					
SUCF2	1-31	6-86					
SUTOP	1-69	7-48					

Cross reference table (CREF V05.05)

SWPCOT	1-81	18-33	18-36*						
SWPJOB	1-35	52-92							
SWPPOS	1-35	52-91							
SYNAME	1-39	54-22							
SYPNCR	1-55	24-38	24-43*	25-13	25-15*				
SYSDAT	1-81	16-33	16-61*						
SYSDIE	1-22	50-15#							
SYSHL1	1-33	11-39							
SYSHLT	1-59	1-93	7-8	7-74	11-46	18-137			
SYSXIT	1-43	12-30							
SYTIMH	1-85	10-18	10-22	16-20*	16-24	16-32*			
SYTIML	1-85	10-19	10-25	16-19*	16-26	16-31*			
TIK01S	3-52#	15-81*	18-12	18-124*					
TIKCNT	1-84	15-6	15-90*						
TK1CNT	1-85	15-78*	15-80*						
TK1VAL	1-84	15-80							
TK5CNT	1-85	18-111*	18-113*						
TLCHK	18-114	29-6#							
TMIDLH	1-93	18-95*							
TMIDLL	1-93	18-94*							
TMIOH	1-91	18-60*							
TMIOL	1-91	18-59*							
TMIOWH	1-92	18-83*	18-91*						
TMIOWL	1-92	18-82*	18-90*						
TMSWPH	1-92	18-66*							
TMSWPL	1-92	18-65*							
TMSWTH	1-93	18-85*							
TMSWTL	1-93	18-84*							
TMTOTH	1-91	18-56*							
TMTOTL	1-91	18-55*							
TMUSRH	1-92	18-70*							
TMUSRL	1-92	18-69*							
TOTON	1-51	5-111*	10-176*						
TRBRK	1-100	33-18	33-23						
TRMRDY	1-77	31-32	32-18	32-23					
TRNSTR	1-74	10-31							
TRPAR5	1-34	50-53	50-77						
TRPBPT	1-22	12-10#							
TRPCOM	1-21	11-29	11-60	11-62	12-37	13-11#			
TRRDY	1-54	51-87							
TSEXC2	1-6#								
TSR	1-109	33-18*	33-23*	34-18	34-21*				
TSXTX	1-21	11-15#							
TXARG	3-28#	50-45							
TXDEV	3-33#	50-103							
TXFSE	3-27#	50-29							
TXNUL	3-29#	51-41	51-66						
TXOID	3-32#	50-93							
TXPAR5	3-30#	50-75							
TXSEG	3-31#	50-89							
UERSEV	1-68	7-18*							
UFPTRP	1-53	8-123*	14-13	14-57	14-59*				
UHIMEM	1-69	7-50*							
UIOCNT	1-89	18-57	18-79	18-88					
UMODE	1-53	1-99	7-87	11-16	11-79	13-15	13-43	14-58	26-27
UPMODE	1-53	7-13	7-37	11-79	13-43	14-10	14-58		26-42

UREGO	1-95	18-22					
USDSPC	1-111	8-141					
USRINI	1-55	52-168					
USRJOB	1-72	8-53					
UTRPAD	1-53	8-122*	11-56	11-84	11-85*		
VDBFLG	1-43	12-26					
VDMKTP	1-34	11-27					
VECBAS	1-65	6-58					
VF\$7BT	1-46	45-39					
VF\$8BT	1-46	45-37					
VF\$BC	1-49	46-24	46-26				
VF\$DCD	1-105	43-38					
VF\$DTR	1-105	43-44	44-24	44-26			
VF\$EVN	1-46	45-45					
VF\$LEN	1-46						
VF\$PAR	1-46	45-42					
VF\$RIE	1-48	1-86	43-20	44-16	45-27	46-16	52-135
VF\$RNG	1-105	43-32					
VF\$SC	1-49						
VF\$TIE	1-48	52-135					
VH\$CSR	1-48	1-104	43-25*	44-21*	45-50*	46-21*	52-135*
VH\$LCR	1-42	1-104	43-26	44-24*	44-26*	46-24*	46-26*
VH\$LPR	1-49	45-51*					
VH\$LSR	1-104	43-27					
VHCLOK	1-26	49-9#					
VHGDSS	1-25	43-11#					
VHSBRK	1-26	46-9#					
VHSDSS	1-25	44-9#					
VHSPCT	3-63#	45-21					
VHSSPD	1-26	45-9#					
VINTIO	1-100	5-87					
VMAXMC	1-65	8-78	52-184				
VMNUAO	1-46	5-38					
VMXMON	1-96	52-76					
VMXSF	1-66	52-178					
VMXWIN	1-88	10-135	52-202				
VOFFTM	1-87	10-96					
VONTM	1-87	30-32					
VPAR5	1-34						
VPAR6	1-82	26-69					
VPLAS	1-74	8-116	10-141	52-196			
VPRIDF	1-75	5-85	5-86				
VPRIHI	1-90	17-23					
VPRILO	1-90	17-32					
VQUANO	1-90	17-25	17-27				
VQUAN1	1-95	5-88					
VQUAN2	1-95	17-112					
VQUAN3	1-90	17-34					
VQUN1A	1-95	17-78					
VQUN1B	1-96	17-97					
VQUN1C	1-100	17-88					
VSWPFL	1-69	5-66	7-44				
VSWPSL	1-35	52-89					
VSYDMP	1-33	50-118					
VTMIN	1-87	30-31					
VTML0C	1-87	5-100					

VTMOUT	1-87	30-71
VUSPHN	1-30	30-46
WAKEUP	18-115	21-6#
WININI	1-88	52-204
WINREL	1-45	10-137

... CM1	7-72												
... CM2	7-72	7-72	7-72	7-72									
... CM3	7-58												
... CM5	7-72												
... CM7	7-72												
. PURGE	2-4#	7-58											
. READW	2-4#	7-72											
DIE	2-37#	7-8	7-74	11-46	18-137								
DISABL	2-21#	9-19	24-14	25-12	34-17	39-22	40-16	41-50	43-24	44-20	45-49	46-20	
	47-22												
ENABL	2-27#	7-10	9-27	10-187	24-58	25-16	25-44	34-22	39-26	40-30	41-53	43-28	
	44-30	45-52	46-30	47-25									
OCALL	2-11#	8-35	8-36	8-37	8-49	8-55	8-62	8-66	8-70	8-74	8-80	8-118	
	10-70	10-137	10-143	21-19	30-58	52-168	52-174	52-186	52-198	52-204			
SATXT	2-47#	3-6	3-7	3-8	3-9	3-10	3-11	3-12	3-13	3-14	3-15	3-16	
	3-17	3-18	3-19	3-20	3-21	3-22	3-23						