

Table of contents

3-	1	Tables of commands and options
4-	1	RUN command
10-	1	CHKIND -- Force IND to run from boot device

```

1          .TITLE  TSKM2C -- The RUN command
2          .ENABL  IC
3          .DSABL  CBL
4 000000   .CSECT  TSKM2C
5 000000   TSKM2C:
6          ;
7          ; TSKM2C is the portion of TSKMON that contains the code
8          ; to implement the RUN command.
9          ;
10         ; Copyright 1986.
11         ; S&M Computer Systems, Inc.
12         ; Nashville, Tennessee
13         ;
14         ;
15         ; Macro calls
16         ;
17         .MCALL  .LOOKUP, .READW, .CLOSE
18         ; Global definitions
19         ;
20         .GLOBL  CMDRSY, CMDRUN, DORUN, RUNNAM, PLOAD, KDOCIN
21         ;
22         ; Global references
23         ;
24         .GLOBL  CMDOFF, NOPRG, FPRINT, SYNAME, INDSAV, NOIND
25         .GLOBL  $DEBUG, $HITY, $INDRN, $NOINT, PO$DBG
26         .GLOBL  $NOWTT, $PRGLK, $RNIOF, $RNMLK, $TRNSP, $UCLRN
27         .GLOBL  AF$BYA, AF$CCA, AF$DBG, AF$DUP, AF$HIE, TRMSTR
28         .GLOBL  AF$IND, AF$IOP, AF$MEM, AF$NOI, AF$NOW, LSW4
29         .GLOBL  AF$PLK, AF$SCA, AF$SET, AF$TPO, AF$UCL, TRMSTR
30         .GLOBL  $CFOPN, CFBUF, CFSQEZ, PUSHCF, ACRFN, TOOLNG
31         .GLOBL  II$FLG, II$NPV, II$PRV, $QUIET
32         .GLOBL  DKSAV, SYSAV, CORUSR, EM$NPD, AF$NPWFILNAM
33         .GLOBL  VDBFLG, CINDAT, LOGASN, $STSNG, LSW6
34         .GLOBL  RUNARG, RUNCHN, RUNFLG, RUNDEV, INSSRC, PVNPW
35         .GLOBL  RUNHD, RUNEMT, JS$RUN, IIBUF
36         .GLOBL  LSW7, LITIME, PO$MEM, LSW2, LSW2S
37         .GLOBL  PRIVCO, EM$MPV, EM$NAL, PO$LCK
38         .GLOBL  CCSRV, ABRTCF, PO$DBG, EM$NAD
39         .GLOBL  $NOWIN, LSW11, $DUPRN, IN$ACT, INDSTA
40         .GLOBL  IN$CNT, IN$ACT, UCLBLK, EM$NUC
41         .GLOBL  $SETRN, LSW9, RDERM, CINFLG, CVTTAB
42         .GLOBL  RSTPRV, LJSW, NEWJSW, R5OVIR, VIMAGE, MAXMEM
43         .GLOBL  ODTBAS, VSWPFLMXADR, PRGSIZ, PRGTOP
44         .GLOBL  USRSTK, NOSTRT, USTART, LPRG1, LPRG2, LMONHD
45         .GLOBL  SMONHD, GENMON, OVRCOR, BADSAV, LDNAM, LDLEN
46         .GLOBL  NOCIN, SKP$PC, SEARCH, INVOPT, FKILL
47         .GLOBL  XAREA, FILNAM, LSW5, $SCCA, BLKO, BLKWDS
48         .GLOBL  LSW3, $CHACT, RDCMD, MXADR, FIXPRV, AF$CF
49         ;
50 000000 000000  RUNOPS: .WORD 0 ; AF$xxx flags from RUN switches

```

```

1      ; -----
2      ; Macro to cause a fatal error message to be printed.
3      ;
4      ;     .MACRO FERR MSG
5      ;     MOV R5, -(SP)
6      ;     MOV MSG, R5
7      ;     CALL FPRINT
8      ;     MOV (SP)+, R5
9      ;     .ENDM FERR
10     ;
11     ; -----
12     ; Macro to print a fatal error message, clean up
13     ; and then jump to RDCMD.
14     ;
15     ;     .MACRO FABORT MSG
16     ;     MOV MSG, R5
17     ;     JMP FKILL
18     ;     .ENDM FABORT
19     ;
20     ; -----
21     ; Macro to print a warning message
22     ;
23     ;     .MACRO FWARN MSG
24     ;     MOV R5, -(SP)
25     ;     MOV MSG, R5
26     ;     CALL PRTWRN
27     ;     MOV (SP)+, R5
28     ;     .ENDM FWARN
29     ;
30     ; -----
31     ; Macro to start a standard option table.
32     ; Name = 1 to 4 character table name.
33     ; NA = Number of arguments per table entry.
34     ;
35     ;     .MACRO TBLDEF NAME, NA
36     ;     NARGS = NA
37     ;     .CSECT CMDV2C
38     ;     NAME /HD: .WORD 2*NA
39     ;     .ENDM TBLDEF
40     ;
41     ; -----
42     ; Macro to enter an option text name and a set of parameters
43     ; into the currently open table.
44     ; STRNG = Ascii name
45     ; A,B,C = Set of option parameters to store in table with name.
46     ;
47     ;     .MACRO CMODEF STRNG, A, B, C
48     ;     .CSECT NAME2C
49     ;     L =
50     ;     .ASCIZ /STRNG/
51     ;     .CSECT CMDV2C
52     ;     .WORD L ; POINTER TO NAME STRING
53     ;     .WORD A
54     ;     .IIF GE, <NARGS-2> .WORD B
55     ;     .IIF GE, <NARGS-3> .WORD C
56     ;     .ENDM CMODEF
57     ;

```

58  
59  
60  
61  
62  
63  
64  
65

```
-----  
; Macro to end a set of table entries.  
;  
    .MACRO  TBLEND)  
    .CSECT  CMDV2C  
    .WORD   0  
    .CSECT  TSKM2C  
    .ENDM   TBLEND
```

1  
2  
3  
4  
5 000002  
6 000002  
7 000006  
8 000012  
9 000016  
10 000022  
11 000026  
12 000032  
13 000036  
14 000042  
15 000046  
16 000052  
17 000056  
18 000062

.SBTTTL Tables of commands and options

-----  
; Define option switches for the RUN command

; TBLDEF RUN, 1  
; CMDDEF B\*YPASN, RNRYA  
; CMDDEF D\*EBUG, RNDK  
; CMDDEF H\*IGH, RNSHD  
; CMDDEF I\*OPAGE, RNSIOP  
; CMDDEF L\*OCK, RNSLK  
; CMDDEF M\*EMLOCK, RNSMLK  
; CMDDEF NONI\*NTERACTIVE, RNSNOI  
; CMDDEF NOW\*INDOWING, RNNPW  
; CMDDEF O\*DT, RNDR  
; CMDDEF S\*INGLECHAR, RNSCHR  
; CMDDEF SC\*CA, RNSCCA  
; CMDDEF T\*RANSPARENT, RNSTPO  
; TBLEND

```

1          .SBTTL  RUN command
2          ;-----
3          ; Process the 'R' (RUN) command.
4          ;
5          ; Entry point for 'RUN' command
6          ;
7 000002  012705  0000000  CMDRUN: MOV      #DKSAV,R5      ;SET DEFAULT DEV AND EXT
8 000006  000402                BR      RUNPRS
9          ;
10         ; Entry point for 'R' command
11         ;
12 000010  012705  0000000  CMDRSY: MOV      #SYSAV,R5      ;SET DEFAULT DEV AND EXT
13         ;
14         ; Begin scanning command line
15         ;
16 000014  004767  0000000  RUNPRS: CALL    CVTTAB          ;CONVERT TABS TO SPACES IN COMMAND LINE
17 000020  005067  177754      CLR      RUNOPS          ;Clear option flag word
18         ;
19         ; See if any switches were specified with run command
20         ;
21 000024  004767  0000000  RUNSW:  CALL    SKPSPC          ;Skip over any spaces
22 000030  121327  000057      CMPB    (R3),#'/          ;ANY SWITCH?
23 000034  001100                BNE     RUNNAM          ;BR IF NOT
24 000036  005203                INC     R3              ;POINT PAST SLASH
25 000040  012704  0000000  MOV     #RUNHD,R4          ;POINT TO RUN SWITCH TABLE
26 000044  004767  0000000  CALL    SEARCH          ;ACCRUE AND LOOKUP SWITCH
27 000050  103004                BCC    I$              ;BR IF SWITCH OK
28 000052                FABORT  #INVOPT         ;BAD SWITCH
29 000062  116701  0000000  I$:    MOVB    CORUSR,R1     ;PUT JOB INDEX NUMBER IN R1
30 000066  000134                JMP     @(R4)+          ;ENTER SWITCH PROCESSING ROUTINE
31         ;
32         ; Process the /LOCK switch
33         ;
34 000070  052767  0000000 177702  RNSLK: BIS     #AF$PLK,RUNOPS  ;Remember to lock program to line
35 000076  000752                BR     RUNSW          ;GO CHECK FOR OTHER SWITCHES
36         ;
37         ; Process the /MEMLOCK switch
38         ;
39 000100  052767  0000000 177672  RNSMLK: BIS    #AF$MEM,RUNOPS  ;Remember to lock program in memory
40 000106  000746                BR     RUNSW          ;Go check for more switches
41         ;
42         ; Process the /SINGLECHAR switch.
43         ;
44 000110  052767  0000000 177662  RNSCHR: BIS    #<AF$SCA!AF$NOW>,RUNOPS ;Enable single char activ & non-wait
45 000116  000742                BR     RUNSW
46         ;
47         ; Process the /TRANSPARENT switch.
48         ;
49 000120  052767  0000000 177652  RNSTPD: BIS    #AF$TPD,RUNOPS  ;Say to run with transparent output
50 000126  000736                BR     RUNSW
51         ;
52         ; Process the /BYPASS (bypass assign) option
53         ;
54 000130  052767  0000000 177642  RNBYA:  BIS    #AF$BYA,RUNOPS  ;Bypass assign commands
55 000136  000732                BR     RUNSW
56         ;
57         ; Process the /HIGH (high-efficiency) switch.

```

```
58 ;
59 000140 052767 0000000 177632 RNSHI: BIS #AF$HIE,RUNOPS ;Enable high-efficiency mode
60 000146 000726 BR RUNSW
61 ;
62 ; Process the /IOPAGE switch
63 ;
64 000150 052767 0000000 177622 RNSIOP: BIS #AF$IOP,RUNOPS ;Program is to be run with I/O page access
65 000156 000722 BR RUNSW
66 ;
67 ; Process the /NONINTERACTIVE switch
68 ;
69 000160 052767 0000000 177612 RNSNOI: BIS #AF$NOI,RUNOPS ;Run program in non-interactive mode
70 000166 000716 BR RUNSW
71 ;
72 ; Process the /DEBUG switch
73 ;
74 000170 105767 0000000 RNDB: TSTB VDBFLG ;Was program debugger genned into the system?
75 000174 001004 BNE 1$ ;Br if yes
76 000176 FABORT #EM$NPD ;No program debugger available
77 000206 052767 0000000 177564 1$: BIS #AF$DBG,RUNOPS ;Remember to run program with debugger
78 000214 000703 BR RUNSW
79 ;
80 ; Process the /SCCA switch
81 ;
82 000216 052767 0000000 177554 RNSCCA: BIS #AF$CCA,RUNOPS ;Remember to suppress control-C aborts
83 000224 000677 BR RUNSW
84 ;
85 ; Process the /NOWINDOW switch
86 ;
87 000226 052767 0000000 177544 RNNPW: BIS #AF$NPW,RUNOPS ;Remember to suspend windowing
88 000234 000673 BR RUNSW
```

```

1      ;
2      ; Accrue name of program to run
3      ;
4 000236 004767 0000000  RUNNAM: CALL   ACRFN           ; ACCRUE THE PROGRAM FILE NAME
5 000242 103002          BCC     1$           ; BR IF GOT NAME OK
6 000244 000167 001242   JMP     CLSEXT        ; BR IF ERROR IN GETTING FILE NAME
7 000250 004767 002226   1$:   CALL   CHKIND        ; SEE IF WE ARE RUNNING IND
8 000254          . LOOKUP #XAREA, #RUNCHN, #FILNAM ; TRY TO OPEN FILE
9 000274 103020          BCC     DORUNX        ; BRANCH IF OPEN OK
10 000276 032767 0000000 177474 BIT     #AF$PLK, RUNOPS ; Was program supposed to be locked?
11 000304 001402          BEQ     2$           ; NO, GO AHEAD
12 000306 000167 0000000   JMP     CMDOFF        ; LOCKED FILE NOT FOUND, LOG OFF
13 000312          2$:   FERR    #NOPRO        ; PRINT ERROR MESSAGE
14 000326 000167 001160   JMP     CLSEXT        ; GO CLEANUP AND GET NEXT COMMAND
15 000332 005067 177442   DORUN: CLR    RUNOPS    ; NO RUN OPTION FLAGS
16 000336 116701 0000000  DORUNX: MOVB   CORUSR, R1 ; GET LINE INDEX #
17 000342 016161 0000000 0000000   MOV    LSW2(R1), LSW2S(R1); SAVE VALUE OF LSW2
18      ;
19      ; See if an argument was specified with run command
20      ;
21 000350 012767 000001 0000100   MOV    #1, CINDAT+10 ; Set 1 to go in 510 chain area
22 000356 005067 0000120   CLR    CINDAT+12     ; Set null at run argument string
23 000362 105267 0000000   INCB   RUNARG        ; Set flag saying RUN argument string present
24 000366 112300          1$:   MOVB   (R3)+, R0    ; ANY ARGUMENT FIELD PRESENT?
25 000370 001502          BEQ    RNDARG        ; BR IF NOT
26 000372 120027 000040   CMPB   R0, #'        ; IGNORE SPACES
27 000376 001770          BEQ    1$           ;
28      ;
29      ; There is an argument field.
30      ; Move argument string to chain area.
31      ;
32 000400 005303          DEC    R3            ; Point to start of string
33 000402 010302          MOV    R3, R2        ; Get pointer to argument string
34 000404 012704 0000120   MOV    #CINDAT+12, R4 ; Point to where chain data for 512 starts
35 000410 020427 0002700   7$:   CMP    R4, #CINDAT+270 ; Have we hit end of chain data area?
36 000414 103002          BHIS   8$           ; Don't go past end of chain data area
37 000416 112224          MOVB   (R2)+, (R4)+  ; Copy argument to chain data area
38 000420 001373          BNE   7$           ; Loop till end reached
39 000422 162704 0000120   8$:   SUB    #CINDAT+12, R4 ; Get # chars in string (including null)
40 000426 010467 0000100   MOV    R4, CINDAT+10 ; It will be passed in 510
41      ;
42      ; Set argument up to be read like it was part of a command file.
43      ; If we have pending commands in the command file buffer, compress
44      ; them to make room for new commands.
45      ; If input is coming from a real command file, we do not need to
46      ; compress since we will reread buffer when we hit end of new command.
47      ;
48 000432 032761 0000000 0000000   BIT    #CFOPN, LSW4(R1); Is a command file open?
49 000440 001403          BEQ    11$          ; Br if not
50 000442 012705 0010000   MOV    #CFBUF+512, R5 ; Say entire command file buffer is free
51 000446 000403          BR    12$          ;
52 000450 004767 0000000   11$:  CALL   CFSQEZ        ; Compress info in current command file buffer
53 000454 010005          MOV    R0, R5        ; Save address of end of free space in buffer
54      ;
55      ; Push current command file and start new one
56      ;
57 000456 004767 0000000   12$:  CALL   PUSHCF        ; START NEW COMMAND FILE

```



```

58 000462 012704 0000000      MOV      #CFBUF,R4      ;POINT TO COMMAND FILE BUFFER
59                               ;
60                               ; See if arg string contains a space which separates input file
61                               ; specs from output file specs.
62                               ;
63 000466 010302      MOV      R3,R2      ;POINT TO START OF ARG STRING
64 000470 112200      2$:      MOVVB   (R2)+,R0      ;GET CHAR FROM STRING
65 000472 001416      BEQ      3$          ;NO SPACE IN STRING
66 000474 120027 000040      CNPB   R0,#'        ;SEARCH FOR SPACE
67 000500 001373      BNE     2$          ;
68                               ;
69                               ; String contains a space. Move part of string following space
70                               ; to front of command file buffer and insert trailing '='.
71                               ;
72 000502 032761 0000000 0000000      BIT     #$INDRN,LSW5(R1); Is IND being started?
73 000510 001007      BNE     3$          ;Br if yes -- Don't reverse parameters
74 000512 105062 177777      CLRB   -1(R2)       ;MAKE 1ST PART BE ASCIZ
75 000516 112224      4$:      MOVVB   (R2)+,(R4)+      ;MOVE STRING TO COMMAND FILE BUFFER
76 000520 001376      BNE     4$          ;LOOP TILL DONE
77 000522 112764 000075 177777      MOVVB   #'=-,-1(R4)   ;PUT IN SEPARATING =
78                               ;
79                               ; Now move 1st part of string to command file buffer
80                               ;
81 000530 112324      3$:      MOVVB   (R3)+,(R4)+      ;
82 000532 001376      BNE     3$          ;
83                               ;
84                               ; Put in CR-LF-^C to terminate the line
85                               ;
86 000534 012703 0000000      MOV     #TRMSTR,R3   ;POINT TO STRING OF CONTROL CHARS
87 000540 005304      DEC     R4          ;STORE OVER TRAILING NULL
88 000542 112324      5$:      MOVVB   (R3)+,(R4)+      ;PUT IN TERMINATING CHARS
89 000544 001376      BNE     5$          ;
90                               ;
91                               ; Pad rest of buffer with nulls
92                               ;
93 000546 020405      6$:      CMP     R4,R5          ;Reached end of area to fill?
94 000550 001407      BEQ     13$         ;Br if yes
95 000552 103002      BHIS   14$         ;Br if buffer overflow
96 000554 105024      CLRB   (R4)+       ;Null fill the buffer
97 000556 000773      BR     6$          ;
98 000560      14$:      FABORT  #TOOLNG      ;Buffer overflow
99                               ;
100                               ; Set flags to suppress listing of argument string being read.
101                               ;
102 000570 052761 0000000 0000000      13$:     BIS     #$QUIET,LSW4(R1); SUPPRESS LISTING
103                               ;
104                               ; Program file is open.
105                               ;
106 000576 012704 0000000      RNDARG: MOV    #FILNAM,R4      ;POINT TO CELL WITH FILE NAME
107 000602 016700 177172      MOV    RUNOPS,R0      ;Get run option flags for PLOAD
108 000606 000400      BR     PLOAD        ;Load and start execution of program

```

```
1 ;-----  
2 ; Load a SAV file into memory and start its execution.  
3 ;  
4 ; Inputs:  
5 ; RUNCHN channel is now open to program file.  
6 ; R0 = AF$xxx run option flags.  
7 ; R1 = Job index number.  
8 ; R4 = Address of buffer with RAD50 program name.  
9 ;  
10 000610 010067 177164 PLOAD: MOV R0,RUNOPS ;Save initial run option flags  
11 ;  
12 ; Set name of running program  
13 ;  
14 000614 012705 0000000 MOV #RUNDEV,R5 ;Save program name here  
15 000620 012700 0000004 MOV #4,R0 ;Save 4 words  
16 000624 012425 1$: MOV (R4)+,(R5)+ ;Copy program file spec  
17 000626 077002 SDR R0,1$  
18 000630 012705 0000000 MOV #RUNDEV,R5 ;Point to file spec  
19 000634 004767 0000000 CALL LOGASN ;Do any logical assignment  
20 000640 012704 0000000 MOV #RUNDEV,R4 ;Point to file spec
```

```

1          ; See if we need to invoke some automatic run switches for this program.
2          ;
3          ; See if we should default to single character activation
4          ;
5 000644 032761 0000000 0000000 RUNASW: BIT    ##STSNG,LSW6(R1);Is single char activation the default?
6 000652 001403          BEQ      7$          ;Br if not
7 000654 052767 0000000 177116      BIS      #AF$SCA,RUNOPS ;Remember to enable single char activation
8          ;
9          ; See if the program has been installed
10         ;
11 000662 012700 0000000          7$:      MOV      #RUNDEV,R0      ;Point to file spec for program
12 000666 004767 0000000          CALL     INSSRC        ;See if this program has been installed
13 000672 103420          BCS      6$          ;Br if not
14         ;
15         ; Apply privilege flags specified for installed program
16         ;
17 000674 012702 0000000          MOV      #<CPVNPW-1>*2,R2 ;Get index to last privilege word
18 000700 056262 0000000 0000000 4$:      BIS      II$PRV+IIBUF(R2),PRIVCO(R2);Set flags specified for program
19 000706 046262 0000000 0000000      BIC      II$NPV+IIBUF(R2),PRIVCO(R2);Clear flags specified for program
20 000714 162702 0000002          SUB      #2,R2          ;Get index to next word
21 000720 002367          BGE      4$          ;Loop if more to do
22 000722 004767 0000000          CALL     FIXPRV        ;Do privilege setup
23 000726 056767 0000000 177044      BIS      II$FLG+IIBUF,RUNOPS ;Combine run option switches for program
24         ;
25         ; Or in attribute flags for enclosing command file
26         ;
27 000734 056767 0000000 177036 6$:      BIS      AFCF,RUNOPS      ;Combine attributes from command file
28         ;
29         ; Set some automatic switches for this program
30         ;
31 000742 016702 177032          MOV      RUNOPS,R2      ;Get attribute flags for program
32 000746 032702 0000000          BIT      #AF$SCA,R2      ;Single character activation wanted?
33 000752 001403          BEQ      3$          ;Br if not
34 000754 052761 0000000 0000000      BIS      ##CHACT,LSW5(R1);Enable single character activation
35 000762 032702 0000000          3$:      BIT      #AF$NOW,R2      ;Non wait TT input wanted?
36 000766 001403          BEQ      15$         ;Br if not
37 000770 052761 0000000 0000000      BIS      ##NOWTT,LSW5(R1);Enable non-wait TT input
38 000776 032702 0000000          15$:     BIT      #AF$HIE,R2      ;Is high efficiency mode wanted?
39 001002 001403          BEQ      16$         ;Br if not
40 001004 052761 0000000 0000000      BIS      ##HITTY,LSW4(R1);Enable high-efficiency mode
41 001012 032702 0000000          16$:     BIT      #AF$TPO,R2      ;Transparent output wanted?
42 001016 001403          BEQ      23$         ;Br if not
43 001020 052761 0000000 0000000      BIS      ##TRNSP,LSW3(R1);Enable transparent output
44 001026 032702 0000000          23$:     BIT      #AF$NOI,R2      ;Is non-interactive execution wanted?
45 001032 001406          BEQ      1$          ;Br if not
46 001034 052761 0000000 0000000      BIS      ##NOINT,LSW7(R1);Set non-interactive execution flag
47 001042 012761 0000002 0000000      MOV      #2,LITIME(R1) ;Set very short interactive time
48 001050 032702 0000000          1$:      BIT      #AF$IOP,R2      ;Map to I/O page?
49 001054 001417          BEQ      17$         ;Br if not
50 001056 032767 0000000 0000000      BIT      #PO$MEM,PRIVCO ;Are we authorized to access I/O page?
51 001064 001010          BNE      20$         ;Br if yes
52 001066          FERR     #EM$MPV      ;Not authorized to run this program
53 001102 000167 000404          JMP      CLSEXT
54 001106 052761 0000000 0000000 20$:     BIS      ##RNIDP,LSW9(R1);Set flag to cause mapping to I/O page
55 001114 032702 0000000          17$:     BIT      #AF$MEM,R2      ;Lock program in memory?
56 001120 001417          BEQ      18$         ;Br if not
57 001122 032767 0000000 0000000      BIT      #PO$LCK,PRIVCO ;Are we authorized to do this?

```

```

58 001130 001010          BNE      21$          ;Br if yes
59 001132                FERR     #EM$NAL        ;Not authorized to lock program in memory
60 001146 000167 000340    JMP      CLSEXT
61 001152 052761 0000000 0000000 21$:    BIS     ##RNMLK,LSW9(R1);Set flag to cause program to be locked
62 001160 032702 0000000          18$:    BIT     #AF$PLK,R2      ;Want to lock program to line?
63 001164 001407                BEQ     22$          ;Br if not
64 001166 052761 0000000 0000000        BIS     ##PRGLK,LSW5(R1);Remember program is locked to line
65 001174 004767 0000000          CALL    CCSPRV        ;Copy current to set privileges
66 001200 004767 0000000          CALL    ABRTCF        ;Close any open command files
67 001204 032702 0000000          22$:    BIT     #AF$DBG,R2      ;Want to run program with debugger?
68 001210 001417                BEQ     24$          ;Br if not
69 001212 032767 0000000 0000000        BIT     #PO$DBG,PRIVCO ;Are we privileged to use the debugger?
70 001220 001010          BNE      19$          ;Br if yes
71 001222                FERR     #EM$NAD        ;Can't use debugger
72 001236 000167 000250    JMP      CLSEXT
73 001242 052761 0000000 0000000 19$:    BIS     ##DEBUG,LSW9(R1);Enable the debugger
74 001250 032702 0000000          24$:    BIT     #AF$CCA,R2      ;Disable control-C abort?
75 001254 001403                BEQ     32$          ;Br if not
76 001256 052761 0000000 0000000        BIS     ##SCCA,LSW5(R1);Disable control-C abort
77 001264 032702 0000000          32$:    BIT     #AF$NPW,R2      ;Suspend process windowing?
78 001270 001403                BEQ     30$          ;Br if not
79 001272 052761 0000000 0000000        BIS     ##NOWIN,LSW11(R1);Suspend process windowing
80                                ;
81                                ; Check for startup of special programs
82                                ;
83 001300 032702 0000000          30$:    BIT     #AF$DUP,R2      ;Is DUP being started?
84 001304 001403                BEQ     25$          ;Br if not
85 001306 052761 0000000 0000000        BIS     ##DUPRN,LSW6(R1);Remember DUP is running
86 001314 032702 0000000          25$:    BIT     #AF$IND,R2      ;Is IND being started?
87 001320 001423                BEQ     26$          ;Br if not
88 001322 132767 0000000 0000000        BITB    #IN$ACT,INDSTA ;Is IND already active?
89 001330 001414                BEQ     27$          ;Br if not
90 001332 132767 0000000 0000000        BITB    #IN$CNT,INDSTA ;Are we continuing its execution?
91 001340 001010          BNE      27$          ;Br if yes
92 001342                FERR     #INDACT        ;Error -- IND is already active
93 001356 000167 000130    JMP      CLSEXT
94 001362 052761 0000000 0000000 27$:    BIS     ##INDRN,LSW5(R1);Remember IND is running
95 001370 032702 0000000          26$:    BIT     #AF$UCL,R2      ;Is TSXUCL being started?
96 001374 001412                BEQ     28$          ;Br if not
97 001376 005767 0000000          TST     UCLBLK        ;Are we supporting user-defined commands?
98 001402 001004          BNE      31$          ;Br if yes
99 001404                FABORT   #EM$NUC        ;Not supporting user-defined commands
100 001414 052761 0000000 0000000 31$:    BIS     ##UCLRN,LSW7(R1);Remember TSXUCL is running
101 001422 032702 0000000          28$:    BIT     #AF$SET,R2      ;Is SETUP being started?
102 001426 001403                BEQ     2$           ;Br if not
103 001430 052761 0000000 0000000        BIS     ##SETRN,LSW9(R1);Remember SETUP is running
104 001436                2$:

```

```

1
2 ; Read in block zero of file being started.
3 ;
4 001436 RNREAD: .READW #XAREA,#RUNCHN,#BLKO,#BLKWDS,#0
5 001474 103045 BCC SVINBG ;BRANCH IF READ OK
6 001476 FERR #RDERM
7 001512 CLSEXT: .CLOSE #RUNCHN ;CLOSE SAV FILE
8 001520 105067 000000G CLRFB CINFLG ;RESET CHAIN FLAG
9 001524 105067 000000G CLRFB RUNARG ;No argument string with RUN command
10 001530 116701 000000G MOVFB CORUSR,R1 ;GET CURRENT JOB INDEX NUMBER
11 001534 042761 000000G 000000G BIC #STRNSP,LSW3(R1)
12 001542 042761 000000G 000000G BIC #HITTY,LSW4(R1);CLEAN OUT MORE RUN FLAGS
13 001550 042761 000000C 000000G BIC #<$INDRN!$CHACT!$NDWTT>,LSW5(R1);CLEAN OUT FLAGS
14 001556 042761 000000G 000000G BIC #DUPRN,LSW6(R1)
15 001564 042761 000000C 000000G BIC #UCLRN!$NOINT,LSW7(R1)
16 001572 042761 000000C 000000G BIC #DEBUG!$RN1OP!$RNMLK!$SETRN,LSW9(R1)
17 001600 004767 000000G CALL RSTPRV ;Reset job privilege flags
18 001604 000167 000000G JMP RDCMD
19 ;
20 ; Examine information in block 0 of sav file.
21 ;
22 001610 116701 000000G SVINBG: MOVFB CORUSR,R1 ;GET JOB INDEX NUMBER
23 001614 012702 000000G MOV #BLKO,R2 ;POINT TO BLOCK 0 DATA WE JUST READ IN
24 ; Set up new JSW for job being started.
25 001620 016203 000044 MOV 44(R2),R3 ;GET JSW SPECIFIED IN SAV FILE IMAGE
26 001624 042703 100007 BIC #^C<077770>,R3 ;CLEAR SOME FLAGS
27 001630 016104 000000G MOV LJSW(R1),R4 ;GET CURRENT JSW
28 001634 042704 075570 BIC #075570,R4 ;CLEAR SOME FLAGS IN IT
29 001640 050304 BIS R3,R4 ;MERGE FLAGS
30 001642 010467 000000G MOV R4,NEWJSW ;THIS WILL BE JSW FOR JOB WE ARE STARTING
31 001646 026267 000000 000000G CMP 0(R2),R5OVIR ;WAS JOB LINKED WITH /V SWITCH?
32 001654 001410 BEQ 23$ ;IF YES THEN SET VIRTUAL-MODE FLAG
33 001656 026227 000056 000034 CMP 56(R2),#28. ;DOES JOB NEED MORE THAN 28KW?
34 001664 101004 BHI 23$ ;BR IF YES
35 001666 026227 000050 160000 CMP 50(R2),#160000 ;CHECK ADDRESS ABOVE TOP OF PROGRAM ROOT
36 001674 101403 BLOS 27$ ;BR IF DOES NOT NEED MORE THAN 28KW
37 001676 052767 000000G 000000G 23$: BIS #VIMAGE,NEWJSW ;FORCE VIRTUAL-JOB MODE IF > 28KW NEEDED
38 ; Set up information about the max amount of memory job will be allowed
39 ; to use. This is constrained by three things:
40 ; 1. The amount of memory specified by the MEMORY command.
41 ; 2. 56Kb if this is not a virtual job, (64Kb if it is a virtual job).
42 001704 016705 000000G 27$: MOV MAXMEM,R5 ;Get amt of memory specified by MEMORY command
43 001710 032767 000000G 000000G BIT #VIMAGE,NEWJSW ;Is this a virtual job?
44 001716 001005 BNE 17$ ;Br if this is a virtual job
45 001720 020527 160000 CMP R5,#160000 ;Constrain to 56Kb if not virtual job
46 001724 101402 BLOS 17$
47 001726 012705 160000 MOV #160000,R5
48 001732 010567 000000G 17$: MOV R5,ODTBAS ;This is base address of debugger
49 ; If stack is allocated above top of program then say top of program
50 ; is equal to top of stack.
51 001736 026262 000042 000050 CMP 42(R2),50(R2) ;IS STACK ALLOCATED ABOVE PROGRAM TOP?
52 001744 101403 BLOS 16$ ;BR IF NOT
53 001746 016262 000042 000050 MOV 42(R2),50(R2) ;SAY TOP OF PROGRAM = STACK TOP
54 ; See if amount of memory to allocate for program was specified
55 ; in SAV file.
56 001754 016703 000000G 16$: MOV MAXMEM,R3 ;GET DEFAULT TOP OF MEMORY ADDRESS
57 001760 105767 000000G TSTB VSWPFL ;IS THIS A NON-SWAPPING SYSTEM?

```

```

58 001764 001442          BEQ      13$          ; BR IF YES
59 001766 032767 0000000 0000000  BIT      #VIMAGE,NEWJSW ; IS THIS A VIRTUAL IMAGE?
60 001774 001005          BNE      26$          ; BR IF YES
61 001776 020327 160000    CMP      R3,#160000    ; CONSTRAIN TO 56KB IF NOT VIRTUAL
62 002002 101402          BLOS    26$
63 002004 012703 160000    MOV      #160000,R3
64 002010 016200 000056    26$:    MOV      56(R2),R0     ; WAS MEMORY ALLOCATION FOR PROGRAM SPECIFIED?
65 002014 001406          BEQ      7$           ; BR IF NOT
66 002016 072027 000013    ASH     #11.,R0       ; CONVERT # KW TO BYTE ADDRESS
67 002022 001002          BNE     21$          ; BR IF NOT 64KB
68 002024 012700 177774    MOV      #177774,R0   ; SET 64KB ADDRESS TOP
69 002030 010003          21$:    MOV      R0,R3        ; SET AS ADDRESS OF TOP OF PROGRAM SPACE
70 002032 016200 000050    7$:    MOV      50(R2),R0   ; GET ADDRESS OF TOP OF STATIC PROGRAM AREA
71 002036 062700 000002    ADD     #2,R0
72 002042 001002          BNE     22$          ; BR IF DIDN'T OVERFLOW 64KB
73 002044 012700 177774    MOV      #177774,R0   ; SET SIZE FOR 64KB
74 002050 020300          22$:    CMP      R3,R0        ; IS ALLOCATED SIZE AT LEAST THIS BIG?
75 002052 103001          BHI     8$           ; BR IF YES
76 002054 010003          MOV     R0,R3        ; USE ENOUGH SPACE FOR STATIC PROGRAM SIZE
77 002056 016705 0000000    8$:    MOV      MXJADR,R5    ; GET MAX ADDRESS JOB CAN POSSIBLY USE
78 002062 010367 0000000    MOV     R3,ODTBAS    ; START ODT ABOVE TOP OF PROGRAM AREA
79 002066 020305          CMP     R3,R5        ; ROOM FOR PROGRAM TO RUN?
80 002070 101123          BHI     TOOBIG       ; BR IF NOT ENOUGH MEMORY
81 002072 032767 0000000 0000000 13$:    BIT      #VIMAGE,NEWJSW ; IS THIS A VIRTUAL JOB?
82 002100 001005          BNE     25$          ; BR IF YES
83 002102 020327 160000    CMP     R3,#160000   ; IS NOT, THEN CONSTRAIN SIZE TO 56KB
84 002106 101402          BLOS    25$
85 002110 012703 160000    MOV     #160000,R3
86 002114 010367 0000000    25$:    MOV     R3,PRGSIZ    ; ADDRESS ABOVE TOP OF PROGRAM AREA
87                                     ; Check top of program address.
88 002120 016203 000050    MOV     50(R2),R3    ; GET TOP-OF-PROGRAM ADDRESS FOR SAV FILE
89 002124 020327 001000    CMP     R3,#1000    ; MAKE SURE IT'S NOT TOO SMALL
90 002130 103512          BLO     ILLSAV       ; BR IF INVALID
91 002132 010367 0000000    MOV     R3,PRGTOP    ; SAVE TOP ADDRESS OF PROGRAM
92 002136 016700 0000000    MOV     ODTBAS,R0   ; GET TOP ADDRESS WE MAY GO UP TO
93 002142 020300          CMP     R3,R0        ; IS PROGRAM TOO BIG?
94 002144 103075          BHI     TOOBIG       ; BR IF YES
95                                     ; Check program's stack address.
96 002146 016203 000042    MOV     42(R2),R3    ; GET SPECIFIED STARTING STACK ADDRESS
97 002152 001407          BEQ     2$           ; BR IF NO INITIAL STACK ADDRESS SPECIFIED
98 002154 020367 0000000    CMP     R3,PRGTOP    ; CAN'T BE ABOVE TOP OF PROGRAM
99 002160 101076          BHI     ILLSAV
100 002162 032703 000001    BIT     #1,R3        ; CAN'T BE ODD
101 002166 001073          BNE     ILLSAV
102 002170 000402          BR      3$
103 002172 012703 001000    2$:    MOV     #1000,R3   ; DEFAULT TO 1000 FOR STACK IF NONE SPECIFIED
104 002176 010367 0000000    3$:    MOV     R3,USRSTK   ; SET AS STARTUP STACK ADDRESS
105                                     ; Check program starting address.
106 002202 016203 000040    MOV     40(R2),R3    ; GET PROGRAM START ADDRESS
107 002206 020327 000400    CMP     R3,#400     ; MAKE SURE IT'S VALID
108 002212 103406          BLO     4$           ; BR IF TOO LOW
109 002214 032703 000001    BIT     #1,R3        ; MAKE SURE IT'S NOT ODD
110 002220 001003          BNE     4$
111 002222 020367 0000000    CMP     R3,PRGTOP    ; MAKE SURE IT IS IN MEMORY RANGE FOR PROGRAM
112 002226 101410          BLOS    5$
113 002230          4$:    FERR    #NOSTRT     ; INVALID START ADDRESS
114 002244 000167 177242    JMP     CLSEXT

```

```

115 002250 010367 0000000 5$:      MOV      R3, USTART      ;SAVE PROGRAM STARTING ADDRESS
116
117
118
119 002254 012704 0000000          MOV      #RUNDEV, R4      ;POINT TO PROGRAM NAME STORAGE CELLS
120 002260 016461 0000002 0000000  MOV      2(R4), LPRG1(R1) ;SAVE NAME OF RUNNING PROGRAM
121 002266 016461 0000004 0000000  MOV      4(R4), LPRG2(R1)
122
123
124
125 002274 016767 175500 0000000  MOV      RUNOPS, RUNFLO  ;Set status flags for running program
126
127
128
129
130 002302 005761 0000000          TST      LMONHD(R1)      ;Is our job being monitored?
131 002306 001003          BNE      29$             ;Br if it is
132 002310 005767 0000000          TST      SMONHD         ;Anyone monitoring all jobs?
133 002314 001406          BEQ      28$             ;Br if not
134 002316 012700 0000000 29$:   MOV      #GENMON, R0     ;Point to EMT argument block
135 002322 012760 0000000 0000002  MOV      #JS$RUN, 2(R0)  ;Set status code
136 002330 104375          EMT      375             ;Broadcast status message
137
138
139
140 002332 012700 0000000 28$:   MOV      #RUNEMT, R0     ;EMT TO START PROGRAM EXECUTION
141 002336 104375          EMT      375             ;START SAV FILE
142
143
144
145 002340          TOOBIG: FERR      #OVR CUR
146 002354 000406          BR       CLEJMP
147
148
149
150 002356          ILLSAV: FERR      #BADSAV
151 002372 000167 177114  CLEJMP: JMP      CLSEXT

```

```
1 ;-----  
2 ; PROCESS A .CHAIN REQUEST  
3 ;  
4 002376 012704 000000G KDDCIN: MOV #CINDAT,R4 ;POINT TO CELL WITH PROGRAM NAME  
5 ;  
6 ; If chain is being done to SY:LD.SYS, simply do a SET LD CLEAN  
7 ;  
8 002402 010405 MOV R4,R5 ;POINT TO PROGRAM NAME  
9 002404 012703 000000G MOV #LDNAM,R3 ;POINT TO "SY:LD.SYS"  
10 002410 012700 000004 MOV #4,R0 ;COMPARE 4 WORDS  
11 002414 022325 3$: CMP (R3)+,(R5)+ ;COMPARE PROGRAM NAMES  
12 002416 001004 BNE 2$ ;BR IF NOT CHAINING TO LD HANDLER  
13 002420 077003 SOB R0,3$  
14 002422 004767 000000G CALL LDCLEN ;DO "SET LD CLEAN" OPERATION  
15 002426 000421 BR 4$ ;DON'T DO THE CHAIN  
16 ;  
17 ; Look up SAV file we are chaining to  
18 ;  
19 002430 2$: .LOOKUP #XAREA,#RUNCHN,R4  
20 002446 103403 BCS 1$ ;BRANCH IF ERROR  
21 002450 005000 CLR R0 ;Say no RUN options  
22 002452 000167 176132 JMP PLOAD ;LOAD AND RUN PROGRAM  
23 002456 1$: FERR #NOPRG  
24 002472 105067 000000G 4$: CLRB CINFLG  
25 002476 000167 000000G JMP NDCIN  
26 ;
```



CHKIND -- Force IND to run from boot device

```

1          .SBTTL  CHKIND -- Force IND to run from boot device
2          ;-----
3          ;
4          ; Inputs: FILNAM has dev: filspc.ext of program about to be run
5          ;
6          ; Outputs: If program is IND, dev: in FILNAM changed to boot device
7          ;                   and .ext in FILNAM changed to .SAV
8          ;
9 002502  026727  0000020 035164  CHKIND:  CMP      FILNAM+2,#^RIND ; IS PROGRAM IND?
10 002510  001021                BNE      9$          ; RETURN IF NOT
11 002512  005767  0000040                TST      FILNAM+4      ; AND ONLY 3 CHARS?
12 002516  001016                BNE      9$          ; RETURN IF NOT
13          ; Program about to be run is IND
14 002520  005767  0000000                TST      INDSAV       ; WAS IND AVAILABLE DURING START-UP?
15 002524  001005                BNE      1$          ; BR IF SO
16 002526  005726                TST      (SP)+        ; THROW AWAY RETURN POINTER IF NOT
17 002530                FABORT  #NOIND    ; AND ABORT COMMAND
18          ; Ensure that IND is always run from "real" SY
19 002540  016767  0000000 0000000 1$:  MOV      SYNAME,FILNAM ; ALWAYS RUN FROM BOOT DEVICE
20 002546  012767  073376  0000060                MOV      #^RSAV,FILNAM+6 ; USE IND. SAV
21 002554  000207                9$:  RETURN
22          ;
23          .END

```

Errors detected: 0

\*\*\* Assembler statistics

Work file reads: 0  
 Work file writes: 0  
 Size of work file: 9848 Words ( 39 Pages)  
 Size of core pool: 17920 Words ( 70 Pages)  
 Operating system: RT-11

Elapsed time: 00:00:16.22  
 DK: TSKM2C, LP: TSKM2C=DK: TSKM2C. MAC/C/N: SYM







