

```

RRRRRRRRRRRRRRR      MMM      MMM      SSSSSSSSSSSSSS
RRRRRRRRRRRRRRR      MMM      MMM      SSSSSSSSSSSSSS
RRRRRRRRRRRRRRR      MMM      MMM      SSSSSSSSSSSSSS
RRR      RRR      MMMMMM      MMMMMM      SSS
RRR      RRR      MMMMMM      MMMMMM      SSS
RRR      RRR      MMMMMM      MMMMMM      SSS
RRR      RRR      MMM      MMM      MMM      SSS
RRR      RRR      MMM      MMM      MMM      SSS
RRR      RRR      MMM      MMM      MMM      SSS
RRRRRRRRRRRRRRR      MMM      MMM      SSSSSSSSSS
RRRRRRRRRRRRRRR      MMM      MMM      SSSSSSSSSS
RRRRRRRRRRRRRRR      MMM      MMM      SSSSSSSSSS
RRR      RRR      MMM      MMM      MMM      SSS
RRR      RRR      MMM      MMM      MMM      SSS
RRR      RRR      MMM      MMM      MMM      SSS
RRR      RRR      MMM      MMM      MMM      SSS
RRR      RRR      MMM      MMM      MMM      SSS
RRR      RRR      MMM      MMM      SSSSSSSSSSSSSS
RRR      RRR      MMM      MMM      SSSSSSSSSSSSSS
RRR      RRR      MMM      MMM      SSSSSSSSSSSSSS

```

— 52

Syn

NTS

NTS
NTS

NTS

NTS
NTS

NTS

NTS
NTS

NTS

NTS
NTS

NTS

NTS
NTS

NTS

NTS
NTS

NTS

NTS
NTS

NTS

NTS
NTS

NTS

NTS
NTS

NTS

NTS

NTS
NTS

NTS

NTS
NTS

NTS

NTS

NT
NT

NT

NT
NT

PI

100

10

10

10

10

11. *Journal of the American Medical Association*, 1997; 277: 1001-1005.

1

RRRRRRRR	MM	MM	000000	FFFFFFFF	SSSSSSSS	EEEEEEEE	TTTTTTTT
RRRRRRRR	MM	MM	000000	FFFFFFFF	SSSSSSSS	EEEEEEEE	TTTTTTTT
RR	RR	MMMM	00	FF	SS	EE	TT
RR	RR	MMMM	00	FF	SS	EE	TT
RR	RR	MM	00	FF	SS	EE	TT
RR	RR	MM	00	FF	SS	EE	TT
RRRRRRRR	MM	MM	00	FFFFFFFF	SSSSSS	EEEEEEEE	TTTTTTTT
RRRRRRRR	MM	MM	00	FFFFFFFF	SSSSSS	EEEEEEEE	TTTTTTTT
RR	RR	MM	0000	FF	SS	EE	TT
RR	RR	MM	0000	FF	SS	EE	TT
RR	RR	MM	00	FF	SS	EE	TT
RR	RR	MM	00	FF	SS	EE	TT
RR	RR	MM	000000	FF	SSSSSSSS	EEEEEEEE	TTTTTTTT
RR	RR	MM	000000	FF	SSSSSSSS	EEEEEEEE	TTTTTTTT

LL	IIIIII	SSSSSSSS
LL	IIIIII	SSSSSSSS
LL	II	SS
LL	II	SS
LL	II	SS
LL	II	SS
LL	II	SS
LL	II	SS
LL	II	SS
LL	II	SS
LL	II	SS
LL	II	SS
LLLLLLLLLL	IIIIII	SSSSSSSS
LLLLLLLLLL	IIIIII	SSSSSSSS

(2) 75
(3) 100

DECLARATIONS
RMSFSET - COMMON SETUP FOR FAB FUNCTION ROUTINE

```
0000 1          $BEGIN RMOFSET,000,RM$RMS0,<SETUP FOR A FAB FUNCTION>
0000 2
0000 3
0000 4 :*****
0000 5 :*
0000 6 :*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0000 7 :*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0000 8 :*  ALL RIGHTS RESERVED.
0000 9 :*
0000 10 :*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0000 11 :*  ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0000 12 :*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0000 13 :*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0000 14 :*  OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0000 15 :*  TRANSFERRED.
0000 16 :*
0000 17 :*  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000 18 :*  AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0000 19 :*  CORPORATION.
0000 20 :*
0000 21 :*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0000 22 :*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0000 23 :*
0000 24 :*
0000 25 :*****
0000 26 :
0000 27 :
0000 28 :++
0000 29 : Facility: RMS32
0000 30 :
0000 31 : Abstract:
0000 32 :         routine to perform common setup for a fab function
0000 33 :
0000 34 : Environment:
0000 35 :         star processor running starlet exec.
0000 36 :
0000 37 : Author: L F Laverdure,          creation date: 4-JAN-1977
0000 38 :
0000 39 : Modified By:
0000 40 :
0000 41 :         V03-005 DAS0001          David Solomon          2-Feb-1984
0000 42 :         Don't call RM$RAISE_LOCK unless sharing.
0000 43 :
0000 44 :         V03-004 KBT0319          Keith B. Thompson      8-Sep-1982
0000 45 :         Remove all SO sharing code
0000 46 :
0000 47 :         V03-003 KBT0208          Keith B. Thompson      23-Aug-1982
0000 48 :         Reorganize psects
0000 49 :
0000 50 :         V03-002 TMK0002          Todd M. Katz            02-Aug-1982
0000 51 :         Add a comment emphasizing (spelling ?) that the field FAB$W_IFI
0000 52 :         must have a value if performing an indirect ppf operation.
0000 53 :
0000 54 :         V03-001 TMK0001          Todd M. Katz            27-Jul-1982
0000 55 :         Add the alternate entry point RM$FSET_ALI1. By transferring
0000 56 :         control to this entry point, the call to RM$FABCHK is bypassed
0000 57 :         but the retrieval of the IFAB's address is not.
```



```

0000 58 :
0000 59 : V02-010 REFORMAT Keith B. Thompson 29-Jul-1980
0000 60 :
0000 61 : V009 CDS0077 C D Saether 23-JAN-1980 14:00
0000 62 : clear busy bit in ifab if irab is busy (act error)
0000 63 :
0000 64 : V008 RAN0003 R A Newell 9-NOV-1978 15:22
0000 65 : file sharing code enhancements
0000 66 :
0000 67 : Revision History:
0000 68 :
0000 69 : L F Laverdure 10-OCT-1978 13:30
0000 70 : copy of CHK_IDLE from RMOFILFNC
0000 71 :
0000 72 :--
0000 73 :

```

```

0000 75      .SBTTL  DECLARATIONS
0000 76
0000 77 :
0000 78 : Include Files:
0000 79 :
0000 80
0000 81 :
0000 82 : Macros:
0000 83 :
0000 84
0000 85      $IRBDEF      ; irab data defintions
0000 86      $IMPDEF
0000 87      $FABDEF
0000 88      $IFBDEF
0000 89      $RMSDEF
0000 90
0000 91 :
0000 92 : Equated Symbols:
0000 93 :
0000 94
0000 95 :
0000 96 : Own Storage:
0000 97 :
0000 98

```



```
0000 100      .SBTTL RMSFSET - COMMON SETUP FOR FAB FUNCTION ROUTINE
0000 101
0000 102 :++
0000 103 :
0000 104 : RMSFSET      - Set up for a FAB function call
0000 105 : RMSFSET_ALT1 - Bypass RMSFABCHK call but obtain IFAB's address
0000 106 : RMSFSET_ALT  - Bypass RMSFABCHK call
0000 107 :
0000 108 : this routine performs common setup for a fab function call
0000 109 : including the following:
0000 110 :
0000 111 :     1. call RMSFABCHK to check arglist, set base regs,
0000 112 :        and zero sts and stv fields in fab
0000 113 :     2. check for valid ifi and set ifab addr
0000 114 :     3. check for stream idle and set to busy
0000 115 :     4. store the arglist addr and caller's mode in the ifab
0000 116 :     5. save sp entry value in IMP$L_SAVED_SP
0000 117 :
0000 118 :
0000 119 : Calling sequence:
0000 120 :
0000 121 :     BSBW      RMSFSET
0000 122 :
0000 123 :     alternate entry at RMSFSET_ALT to perform functions 3, 4, & 5 only
0000 124 :     R7, R8, R9, and R11 must be set as per output prior to call.
0000 125 :
0000 126 :     alternate entry at RMSFSET_ALT1 to perform functions 2, 3, 4, & 5 only
0000 127 :     R7, R8, R9, and R11 must be set as per output prior to call.
0000 128 :
0000 129 :
0000 130 : Input Parameters:
0000 131 :
0000 132 :     R9      IFI of IFAB if enter at RMSFSET_ALT1
0000 133 :     SP      stack pointer
0000 134 :     AP      argument list addr
0000 135 :
0000 136 : Implicit Inputs:
0000 137 :
0000 138 :     The contents of the FAB.
0000 139 :
0000 140 :     NOTE: If performing an indirect PPF operation, the field FAB$W_IFI
0000 141 :           must have a value.
0000 142 :
0000 143 : Output Parameters:
0000 144 :
0000 145 :     R11      impure area address
0000 146 :     R10      ifab address
0000 147 :     R9       ifab address
0000 148 :     R8       fab address
0000 149 :     R7       caller's mode
0000 150 :     R0 thru R5 destroyed
0000 151 :
0000 152 : Implicit Outputs:
0000 153 :
0000 154 :     IMP$L_SAVED_SP is set to value of SP+4
0000 155 :
0000 156 : Completion Codes:
```

```

0000 157 :
0000 158 :      none.  if an error is detected returns to user (not caller)
0000 159 :      after appropriate cleanup.  the user receives a standard
0000 160 :      rms error code, in particular, ifi and those returned by
0000 161 :      RMSFABCHK.
0000 162 :
0000 163 :      Side Effects:
0000 164 :
0000 165 :      none
0000 166 :
0000 167 :--
0000 168 :

```



```
FFFD' 30 0000 170 RMSFSET::
0000 171 BSBW RMSFABCHK ; valid fab?
0003 172 ; returns only if o.k.
0003 173 ; NOTE:
0003 174 ; note: sets R11 to impure addr
0003 175 ; R9 to ifi
0003 176 ; R8 to fab addr
0003 177 ; R7 to caller's mode
0003 178
0003 179 :
0003 180 : Alternate entry point.
0003 181 :
0003 182 : Get the IFAB address and check for a valid IFAB.
0003 183 :
0003 184 :
0003 185 RMSFSET_ALT1::
50 06 D0 0003 186 MOVL #IMP$L_IFABTBL/4,R0 ; ifab table offset divided by 4
FFF7' 30 0006 187 BSBW RMSGTIADR ; get ifab addr
46 13 0009 188 BEQL ERRIFI ; branch if bad
000B 189 ASSUME IFB$B_BID EQ IFB$B_BLN-1
000B 190 .IF NE $$RMS$TEST&$$RMS_TBUGCHK
2E0B 8F 08 A9 B1 000B 191 CMPW IFB$B_BID(R9),#IFB$C_BID+<<IFB$C_BLN/4>>*256>
51 12 0011 192 BNEQ ERRBUG ; branch if not a valid ifab
0013 193 .ENDC
0013 194
0013 195 :
0013 196 : alternate entry from fseti here
0013 197 :
0013 198 : set busy, checking if already active
0013 199 : store caller's mode and arglist addr in ifab
0013 200 :
0013 201 :
0013 202 RMSFSET_ALT::
45 69 20 E2 0013 203 BBSS #IFB$V_BUSY,(R9),ERRACT
30 68 1E E0 0017 204 BBS #FAB$V_PPF_IND+<FAB$W_IFI*8>,(R8),CHKIND; branch if indirect ppf
001B 205 CSB #IFB$V_PPF_IMAGE,(R9) ; make sure indirect bit off
0A A9 57 90 001F 206 SETMOD: MOVB R7,IFB$B_MODE(R9) ; save caller's mode
18 A9 5C D0 0023 207 MOVL AP,IFB$L_ARGLIST(R9) ; save pointer to arglist
5A 59 D0 0027 208 MOVL R9,R10 ; copy ifab addr
14 AB 5E 04 C1 002A 209 ADDL3 #4,SP,IMP$L_SAVED_SP(R11); save stack entry value
002F 210
002F 211 :
002F 212 : (less return pc)
002F 213 :
002F 214 :
002F 215 .IF NE $$RMS$TEST&$$RMS_TBUGCHK
24 A9 58 D0 002F 216 MOVL R8,IFB$L_LAST_FAB(R9) ; save addr this fab
0033 217 .ENDC
0033 218
0033 219 :
0033 220 : check that all irabs connected to an ifab are not busy.
0033 221 :
0033 222 :
50 5A D0 0033 223 MOVL R10,R0 ; ifab addr to temp reg
1C 60 04 11 0036 224 BRB 20$ ; go check if any irabs linked
0038 225 10$: BBS #IRB$V_BUSY,(R0),ERRACT0; error if busy
003C 226 ASSUME IRB$L_IRAB_LNK EQ IFB$L_IRAB_LNK
```

```
50 1C A0 D0 003C 227 20$: MOVL IRB$L_IRAB_LNK(R0),R0 ; get next irab
    F6 12 0040 228 ; BNEQ 10$ ; branch if we got one
    78 AA D5 0042 229 ; TSTL IFB$L_SFSB_PTR(R10) ; are we sharing?
    03 13 0045 230 ; BEQL 30$ ; no, no need to lock file
    FFB6' 30 0047 231 ; BSBW RMSRAISE_LOCK ; take lock on file
    05 004A 232 30$: RSB
    004B 233
    004B 234 ;
    004B 235 ; the ifi value indicates indirect processing of a process permanent file
    004B 236 ;
    004B 237 ; set PPF_IMAGE flag
    004B 238 ;
    004B 239 ;
    CE 11 004B 240 CHKIND: SSB #IFB$V_PPF_IMAGE,(R9) ; set indirect operation flag
    004F 241 BRB SETMOD ; and continue
```



```
0051 243
0051 244 :
0051 245 : error returns
0051 246 :
0051 247 :
0051 248 ERRIFI:
0051 249          RMSERR IFI          ; invalid ifi value
09 11 0056 250          BRB ERROR
0058 251
0058 252 ERRACT0:
0058 253          CSB          #IFB$V_BUSY,(R10)      ; don't leave ifab busy on irab busy
005C 254
005C 255 ERRACT:
005C 256          RMSERR ACT          ; stream already active
FF9C' 31 0061 257 ERROR: BRW RMSEX_NOSTR
0064 258
0064 259 :
0064 260 : internal rms problem - ifab table pointed to an invalid ifab!
0064 261 :
0064 262
0064 263 ERRBUG: RMSTBUG FTL$_BADIFAB
006B 264
006B 265
006B 266          .END
```

RMOFSET
Symbol table

SETUP FOR A FAB FUNCTION

J 12

16-SEP-1984 00:22:36 VAX/VMS Macro V04-00
5-SEP-1984 16:21:48 [RMS.SRC]RMOFSET.MAR;1

Page 9
(6)

```

$$PSECT_EP      = 00000000
$$RMSTEST       = 0000001A
$$RMS_PBUGCHK   = 00000010
$$RMS_TBUGCHK   = 00000008
$$RMS_UMODE     = 00000004
CHKIND          = 0000004B R      01
ERRACT          = 0000005C R R    01
ERRACTO        = 00000058 R R    01
ERRBUG          = 00000064 R R    01
ERRIFI          = 00000051 R R    01
ERROR           = 00000061 R      01
FABSV_PPF_IND   = 0000000E
FABSW_IFI       = 00000002
FTLS_BADIFAB    = FFFFFFFD
IFBSB_BID       = 00000008
IFBSB_BLN       = 00000009
IFBSB_MODE      = 0000000A
IFBSC_BID       = 0000000B
IFBSC_BLN       = 000000B8
IFBSL_ARGLST    = 00000018
IFBSL_IRAB_LNK  = 0000001C
IFBSL_LAST_FAB  = 00000024
IFBSL_SF SB_PTR = 00000078
IFBSV_BUSY      = 00000020
IFBSV_PPF_IMAGE = 00000022
IMPSL_IFABTBL   = 00000018
IMPSL_SAVED_SP  = 00000014
IRBSL_IRAB_LNK  = 0000001C
IRBSV_BUSY      = 00000020
RMSBUG          ***** X      01
RMSEX_NOSTR     ***** X      01
RMSFABCHK       ***** X      01
RMSFSET         00000000 RG      01
RMSFSET_ALT     00000013 RG      01
RMSFSET_ALT1    00000003 RG      01
RMSGTIADR       ***** X      01
RMSRAISE_LOCK   ***** X      01
RMSS_ACT        = 0001825A
RMSS_IFI        = 00018564
SETMOD          0000001F R      01

```

+-----+
! Psect synopsis !
+-----+

PSECT name	Allocation	PSECT No.	Attributes														
ABS	00000000 (0.)	00 (0.)	NOPIC	USR	CON	ABS	LCL	NOSHR	NOEXE	NORD	NOWRT	NOVEC	BYTE				
RM\$RMSO	0000006B (107.)	01 (1.)	PIC	USR	CON	REL	GBL	NOSHR	EXE	RD	NOWRT	NOVEC	BYTE				
\$ABSS	00000000 (0.)	02 (2.)	NOPIC	USR	CON	ABS	LCL	NOSHR	EXE	RD	WRT	NOVEC	BYTE				

+-----+
! Performance indicators !
+-----+

Phase	Page faults	CPU Time	Elapsed Time
-----	-----	-----	-----
Initialization	35	00:00:00.06	00:00:01.41
Command processing	151	00:00:00.71	00:00:04.26
Pass 1	263	00:00:07.75	00:00:19.89
Symbol table sort	0	00:00:01.05	00:00:01.47
Pass 2	59	00:00:01.44	00:00:03.70
Symbol table output	6	00:00:00.06	00:00:00.06
Psect synopsis output	2	00:00:00.03	00:00:00.02
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	518	00:00:11.10	00:00:30.82

The working set limit was 1350 pages.

42527 bytes (84 pages) of virtual memory were used to buffer the intermediate code.

There were 50 pages of symbol table space allocated to hold 849 non-local and 6 local symbols.

266 source lines were read in Pass 1, producing 13 object records in Pass 2.

21 pages of virtual memory were used to define 20 macros.

+-----+
! Macro library statistics !
+-----+

Macro library name	Macros defined
-----	-----
_\$255\$DUA28:[RMS.OBJ]RMS.MLB;1	11
_\$255\$DUA28:[SYS.OBJ]LIB.MLB;1	1
_\$255\$DUA28:[SYSLIB]STARLET.MLB;2	4
TOTALS (all libraries)	16

961 GETS were required to define 16 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LISS\$:RMOFSET/OBJ=OBJ\$:RMOFSET MSRC\$:RMOFSET/UPDATE=(ENH\$:RMOFSET)+EXECML\$/LIB+LIB\$:RMS/LIB

0318 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

