

RMC
VOA

.....

[illegible]

```

LL          IIIIII          SSSSSSSS
LL          IIIIII          SSSSSSSS
LL          II             SS
LL          II             SS
LL          II             SS
LL          II             SS
LL          II             SSSSSS
LL          II             SSSSSS
LL          II             SS
LL          II             SS
LL          II             SS
LL          II             SS
LL          II             SS
LL          II             SS
LLLLLLLLLLLL IIIIII          SSSSSSSS
LLLLLLLLLLLL IIIIII          SSSSSSSS

```


(2) 70
(3) 94

DECLARATIONS
RMSRABCHK - COMMON ARGUMENT AND RAB VALIDATION ROUTINE

```
0000 1          $BEGIN RMORABCHK,000,RMSRMS0,<COMMON RAB CHECKING>
0000 2
0000 3
0000 4 *****
0000 5 *
0000 6 *  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0000 7 *  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0000 8 *  ALL RIGHTS RESERVED.
0000 9 *
0000 10 *  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0000 11 *  ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0000 12 *  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0000 13 *  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0000 14 *  OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0000 15 *  TRANSFERRED.
0000 16 *
0000 17 *  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000 18 *  AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0000 19 *  CORPORATION.
0000 20 *
0000 21 *  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0000 22 *  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0000 23 *
0000 24 *
0000 25 *****
0000 26
0000 27
0000 28 ++
0000 29 Facility: rms32
0000 30
0000 31 Abstract:
0000 32         this routine performs common rab call argument
0000 33         list and rab validation.
0000 34
0000 35 Environment:
0000 36         star processor running starlet exec.
0000 37
0000 38 Author: l f laverdure,      creation date: 4-JAN-1977
0000 39
0000 40 Modified By:
0000 41
0000 42
0000 43 V03-004 DGB0042      Donald G. Blair      02-May-1984
0000 44         If the PIO$V_INHAST bit is set when we start an
0000 45         RMS operation, we conclude that the caller must be
0000 46         at exec AST level or higher and that he would break
0000 47         RMS synchronization rules if allowed to continue.
0000 48         Return Error.
0000 49
0000 50 V03-003 RAS0171      Ron Schaefer      19-Jul-1983
0000 51         Change RAS0162 to a new specific structure-less
0000 52         error.
0000 53
0000 54 V03-002 RAS0162      Ron Schaefer      17-Jun-1983
0000 55         Detect and report the AST/non-AST caller's mode
0000 56         wait hang condition, by checking the low bit of the
0000 57         RAB's BLN field.
```


0000 58 :
0000 59 :
0000 60 :
0000 61 :
0000 62 :
0000 63 :
0000 64 :
0000 65 :
0000 66 :
0000 67 :--
0000 68 :

V03-001	KBT0213	Keith B. Thompson	23-Aug-1982
	Reorganize psects		
V02-010	CDS0001	C Saether	10-Dec-1981
	Rename psect.		
V02-009	REFORMAT	Maria del C. Nasr	24-Jul-1980

```
0000 70          .SBTTL DECLARATIONS
0000 71
0000 72 :
0000 73 : Include Files:
0000 74 :
0000 75 :
0000 76 :
0000 77 : Macros:
0000 78 :
0000 79 :
0000 80          $PIODEF
0000 81          $PSLDEF
0000 82          $RABDEF
0000 83          $RMSDEF
0000 84
0000 85 :
0000 86 : Equated Symbols:
0000 87 :
0000 88 :
0000 89 :
0000 90 : Own Storage:
0000 91 :
0000 92
```

```
0000 94      .SBTTL  RMSRABCHK - COMMON ARGLIST AND RAB VALIDATION ROUTINE
0000 95
0000 96      :++
0000 97      :
0000 98      : RMSRABCHK
0000 99      : RMSNULL
0000 100     : RMSBASIC_ERR
0000 101     :
0000 102     : This routine performs the following functions:
0000 103     :
0000 104     : 1. setup r11 to point to the image i/o impure area
0000 105     :    (may be changed by rms0conn or rm0ifisi)
0000 106     : 2. check argument list for accessibility and validity
0000 107     : 3. check rab for accessibility and validity
0000 108     : 4. set r8 to address of rab
0000 109     : 5. inhibit rms internal asts
0000 110     : 6. set r9 to the value of isi
0000 111     :
0000 112     :
0000 113     :
0000 114     : Calling sequence:
0000 115     :
0000 116     :     bsbw    rm$rabchk
0000 117     :
0000 118     : Input Parameters:
0000 119     :
0000 120     :     ap      arglist addr
0000 121     :
0000 122     : Implicit Inputs:
0000 123     :
0000 124     :     the contents of the arglist and the bid, bln, and isi fields
0000 125     :     of the rab.
0000 126     :
0000 127     : Output Parameters:
0000 128     :
0000 129     :     r11     impure area address
0000 130     :     r9      isi value
0000 131     :     r8      rab address
0000 132     :     r7      mode of caller
0000 133     :
0000 134     : Implicit Outputs:
0000 135     :
0000 136     :     none
0000 137     :
0000 138     : Completion Codes:
0000 139     :
0000 140     :     z-bit set if isi = zero, else clear.
0000 141     :     if any errors, the rms error code is set into r0
0000 142     :     and return is made to the user (not caller).
0000 143     :
0000 144     : Side Effects:
0000 145     :
0000 146     :     rms internal asts are inhibited.
0000 147     :
0000 148     : --
0000 149
```



```
0000 151
0000 152 :++
0000 153 :
0000 154 : set up pointer to image i/o impure area and set r7 to the mode of the caller
0000 155 :
0000 156 :--
0000 157
0000 158 RMSRABCHK::
5B DC 0000 159 MOVPSL R11 ; get psl
16 EF 0002 160 EXTZV #PSL$V_PRVMOD,-
57 5B 02 0004 161 #PSL$$_PRVMOD,R11,R7 ; extract the previous mode
0007 162
5B 00000000'9F DE 0007 163 MOVAL @#PIO$GW_IIOIMPA,R11 ; image io impure area addr
000E 164
000E 165 :++
000E 166 :
000E 167 : perform accessibility checks
000E 168 :
000E 169 :--
58 04 AC D0 000E 170
0012 171 MOVL 4(AP),R8 ; get rab address
001A 172 IFNOWRT #RAB$C_BLN,(R8),ERRRAB ; rab writeable?
001A 173 ASSUME RAB$B_BID EQ 0
001A 174 ASSUME RAB$B_BLN EQ 1
4401 8F 68 B1 001A 175 CMPW (R8),#RAB$C_BID+<RAB$C_BLN@8>; rab block id & length?
OB 12 001F 176 BNEQ CHKRAB ; branch if perhaps not
0021 177
0021 178 :++
0021 179 :
0021 180 : Disable AST's. If the PIO$V_INHA$T bit is already set, we
0021 181 : conclude that the caller must be at exec ast level or higher
0021 182 : (otherwise, he could not have kicked off an RMS operation
0021 183 : while RMS was already in progress) and would break RMS
0021 184 : synchronization rules if allowed to continue. Return RMS$_BUSY
0021 185 : status when this happens.
0021 186 :
0021 187 :--
0021 188
15 0000'CB 00 E2 0021 189 INHA$T: BBSS #PIO$V_INHA$T,- ; disable asts
0023 190 W^<PIO$GW_STATUS-PIO$GW_IIOIMPA>(R11),ERRBUSY
0027 191
0027 192
59 02 A8 3C 0027 193 MOVZWL RAB$W_ISI(R8),R9 ; set r9 = isi value
002B 194
002B 195 :++
002B 196 :
002B 197 : dummy routine to simply return via an rsb
002B 198 :
002B 199 :--
002B 200
05 002B 201 RM$NULL::
002B 202 RSB
002C 203
002C 204 :++
002C 205 :
002C 206 : make longer version of rab validity check
002C 207 :
```



```
002C 208 ;--
002C 209
01 68 91 002C 210 CHKRAB: CMPB (R8),#RAB$C_BID ; is it a rab?
19 12 002F 211 BNEQ ERRRAB
44 8F 01 A8 91 0031 212 CMPB RAB$B_BLN(R8),#RAB$C_BLN; is it long enough?
0B 1F 0036 213 BLSSU ERRBLN ; nope, report error
E5 01 A8 E9 0038 214 BLBC RAB$B_BLN(R8),INHAST ; is this RAB busy?
003C 215 ; continue if not
003C 216
003C 217 ;++
003C 218
003C 219 : an error has occurred in validating the argument list or rab
003C 220
003C 221 : Since an error code cannot be safely stored in the rab,
003C 222 : no attempt to generate an err= ast will be made.
003C 223 : R0 will be set to the appropriate error code and an
003C 224 : exception, if enabled, will be generated upon ret.
003C 225
003C 226 ;--
003C 227
003C 228 ERRBUSY:
003C 229 RMSERR BUSY ; synchronization problem
0C 11 0041 230 BRB BASIC_ERR
0043 231
0043 232 ERRBLN:
0043 233 RMSERR BLN ; invalid block length
05 11 0048 234 BRB BASIC_ERR
004A 235
004A 236 ERRRAB:
004A 237 RMSERR RAB ; invalid rab
004F 238
004F 239 BASIC_ERR:
004F 240 SSB #16,R0 ; prefix the facility code
0053 241 ; ... to the error code
0053 242
04 0053 243 RET ; and return to caller
0054 244
0054 245 .END
```

RMORABCHK
Symbol table

COMMON RAB CHECKING

C 9

16-SEP-1984 00:31:34 VAX/VMS Macro V04-00
5-SEP-1984 16:22:13 [RMS.SRC]RMORABCHK.MAR;1

Page 7
(5)

```
$$PSECT EP      = 00000000
$$RMSTEST       = 0000001A
$$RMS_PBUGCHK   = 00000010
$$RMS_TBUGCHK   = 00000008
$$RMS_UMODE     = 00000004
BASIC_ERR       0000004F R      01
CHKRAB          0000002C R      01
ERRBLN          00000043 R      01
ERRBUSY         0000003C R      01
ERRRAB          0000004A R      01
INHAST          00000021 R      01
PIO$GW_IIOIMPA ***** X      01
PIO$GW_STATUS ***** X      01
PIO$V_INHAST    = 00000000
PSL$S_PRVMOD    = 00000002
PSL$V_PRVMOD    = 00000016
RAB$B_BID       = 00000000
RAB$B_BLN       = 00000001
RAB$C_BID       = 00000001
RAB$C_BLN       = 00000044
RAB$W_ISI       = 00000002
RMS$NULL        0000002B RG     01
RMSRABCHK       00000000 RG     01
RMS$BLN         = 0001842C
RMS$BUSY         = 0001848C
RMS$RAB         = 0001863C
```

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

PSECT name	Allocation	PSECT No.	Attributes
ABS	00000000 (0.)	00 (0.)	NOPIC USR
RMSRMSO	00000054 (84.)	01 (1.)	PIC USR
\$ABSS	00000000 (0.)	02 (2.)	NOPIC USR

CON	ABS	LCL	NOSHR	NOEXE	NORD	NOWRT	NOVEC	BYTE
CON	REL	GBL	NOSHR	EXE	RD	NOWRT	NOVEC	BYTE
CON	ABS	LCL	NOSHR	EXE	RD	WRT	NOVEC	BYTE

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

Phase	Page faults	CPU Time	Elapsed Time
Initialization	29	00:00:00.08	00:00:01.52
Command processing	115	00:00:00.65	00:00:04.03
Pass 1	200	00:00:04.35	00:00:15.50
Symbol table sort	0	00:00:00.38	00:00:00.79
Pass 2	55	00:00:00.97	00:00:02.95
Symbol table output	4	00:00:00.05	00:00:00.33
Psect synopsis output	1	00:00:00.02	00:00:00.02
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	406	00:00:06.50	00:00:25.14

The working set limit was 1350 pages.
21919 bytes (43 pages) of virtual memory were used to buffer the intermediate code.
There were 20 pages of symbol table space allocated to hold 399 non-local and 1 local symbols.
245 source lines were read in Pass 1, producing 13 object records in Pass 2.

18 pages of virtual memory were used to define 17 macros.

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

Macro library name	Macros defined
-----	-----
\$255\$DUA28:[RMS.OBJ]RMS.MLB;1	7
\$255\$DUA28:[SYS.OBJ]LIB.MLB;1	1
\$255\$DUA28:[SYSLIB]STARLET.MLB;2	5
TOTALS (all libraries)	13

501 GETS were required to define 13 macros.

There were no errors, warnings or information messages.

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

0319 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

