

[illegible]

— 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*, 273, 1994, 1033-1034.

1

```

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
?LLLLLLLLL IIIIIII   SSSSSSSS
LLLLLLLLLL IIIIIII   SSSSSSSS

```


(3)	188
(4)	225
(5)	274
(8)	514
(13)	939
(18)	1207

DECLARATIONS
RM1GETINT - INTERNAL GET/FIND SEQUENTIAL ALL DEVICES
RMSGET_UNIT_REC - UNIT RECORD GET ROUTINE
RMSGET_BLK_DEV - BLOCK DEVICE GET ROUTINE
INTERNAL UTILITY ROUTINES
GTMVRC - SUBROUTINE TO MOVE A RECORD

```

0000 1          $BEGIN RM1GETINT,000,RMSRMS1,<INTERNAL GET SEQUENTIAL>,<QUAD,NOWRT>
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 28 :++
0000 29 : Facility: rms32
0000 30 :
0000 31 : Abstract:
0000 32 : this module performs $get /$find record
0000 33 : processing for the sequential file organization.
0000 34 :
0000 35 :
0000 36 : Environment:
0000 37 : star processor running starlet exec.
0000 38 :
0000 39 : Author: L F Laverdure, creation date: 3-FEB-1977
0000 40 :
0000 41 : Modified By:
0000 42 :
0000 43 : V03-015 RAS0308 Ron Schaefer 16-May-1984
0000 44 : Backout zero-length record optimization made in RAS0278
0000 45 : so that a record buffer addr is always set. Too many
0000 46 : callers depend upon a valid address being returned.
0000 47 : Fix record length computation for VFC ANSI-tape records.
0000 48 :
0000 49 : V03-014 RAS0294 Ron Schaefer 17-Apr-1984
0000 50 : Don't check fixed record length unless FAB$V_BLK set.
0000 51 : Fix to RAS0278.
0000 52 :
0000 53 : V03-013 RAS0278 Ron Schaefer 20-Mar-1984
0000 54 : Try to improve the performance of this beast;
0000 55 : particularly for variable-length record disk files.
0000 56 :
0000 57 : V03-012 JWT0167 Jim Teague 15-Mar-1984
0000 58 : Redo buffer offset, which involves removing it entirely
0000 59 : from this module.
0000 60 :
0000 61 : V03-011 DAS0007 David Solomon 17-Feb-1984
0000 62 : Zero length records read from a mailbox that was SYS$INPUT
0000 63 : were not being returned (another read was posted). This has
0000 64 : been fixed by changing the criteria of when to do logical eof
0000 65 : ($EOD) checking from SYS$INPUT and not a terminal, to SYS$INPUT
0000 66 : and not a record oriented device.
0000 67 :
0000 68 : V03-010 JWT0150 Jim Teague 31-Jan-1983
0000 69 : Implement ANSI buffer offset. Oh, and fix 3 broken
0000 70 : branches, too.
0000 71 :
0000 72 : V03-009 DAS0006 David Solomon 12-Dec-1983
0000 73 : For extended terminal reads, read the number of bytes that the
0000 74 : user asked for (don't maximize read size with device buffer
0000 75 : size).
0000 76 :
0000 77 : V03-008 RAS0159 Ron Schaefer 7-Jun-1983
0000 78 : Change PPF input record checking to allow for records
0000 79 : longer than 257 bytes. No limit is now enforced,
0000 80 : since the $EOD checking can at most require 16 bytes.
0000 81 :
0000 82 : V03-007 DAS0005 David Solomon 24-Mar-1983
0000 83 : New IOSB format for extended terminal reads: terminator
0000 84 : length is now a byte, instead of a word.
```

0000	85	:	
0000	86	:	
0000	87	:	
0000	88	:	
0000	89	:	V03-006 KBT0424 Keith B. Thompson 30-Nov-1982
0000	90	:	Change ifb\$w_devbufsiz to ifb\$sl_devbufsiz
0000	91	:	
0000	92	:	
0000	93	:	V03-005 KBT0170 Keith B. Thompson 23-Aug-1982
0000	94	:	Reorganize psects
0000	95	:	
0000	96	:	
0000	97	:	
0000	98	:	
0000	99	:	V03-004 JWH0002 Jeffrey W. Horn 30-Jul-1982
0000	100	:	Fix three broken branches.
0000	101	:	
0000	102	:	
0000	103	:	V03-003 KBT0082 Keith B. Thompson 13-Jul-1982
0000	104	:	Clean up psects
0000	105	:	
0000	106	:	
0000	107	:	
0000	108	:	V03-002 RAS0090 Ron Schaefer 8-Jun-1982
0000	109	:	Correct for odd-aligned NRP_OFF for VAR and VFC formats;
0000	110	:	also clean-up some psect confusion.
0000	111	:	
0000	112	:	
0000	113	:	V03-001 RAS0089 Ron Schaefer 7-Jun-1981
0000	114	:	Delete the reference to the incorrect \$SQUAD_ALIGN macro.
0000	115	:	
0000	116	:	
0000	117	:	V02-050 JWH0001 Jeffrey W. Horn 31-Dec-1981
0000	118	:	Fix broken CASE and yet another branch.
0000	119	:	
0000	120	:	
0000	121	:	
0000	122	:	V02-049 KPL0001 Peter Lieberwirth 31-Dec-1981
0000	123	:	Fix yet another broken branch.
0000	124	:	
0000	125	:	
0000	126	:	
0000	127	:	V02-048 TMK0042 Todd M. Katz 26-Dec-1981
0000	128	:	Fix yet another broken branch by changing a BRW GET_RETREC
0000	129	:	TO A JMP.
0000	130	:	
0000	131	:	
0000	132	:	V02-047 TMK0037 Todd M. Katz 24-Dec-1981
0000	133	:	Fix another broken branch by moving UDFSIZ into the 'middle'
0000	134	:	of GET_ANSI_D.
0000	135	:	
0000	136	:	
0000	137	:	V02-046 TMK0033 Todd M. Katz 22-Dec-1981
0000	138	:	Fix a broken branch by changing a BRW GET00 to a JMP. Also,
0000	139	:	fix another broken branch by moving the STMSIZ labeled code.
0000	140	:	
0000	141	:	
0000	142	:	
0000	143	:	
0000	144	:	
0000	145	:	
0000	146	:	
0000	147	:	
0000	148	:	
0000	149	:	
0000	150	:	
0000	151	:	
0000	152	:	
0000	153	:	
0000	154	:	
0000	155	:	
0000	156	:	
0000	157	:	
0000	158	:	
0000	159	:	
0000	160	:	
0000	161	:	
0000	162	:	
0000	163	:	
0000	164	:	
0000	165	:	
0000	166	:	
0000	167	:	
0000	168	:	
0000	169	:	
0000	170	:	
0000	171	:	
0000	172	:	
0000	173	:	
0000	174	:	
0000	175	:	
0000	176	:	
0000	177	:	
0000	178	:	
0000	179	:	
0000	180	:	
0000	181	:	
0000	182	:	
0000	183	:	
0000	184	:	
0000	185	:	
0000	186	:	
0000	187	:	
0000	188	:	
0000	189	:	
0000	190	:	
0000	191	:	
0000	192	:	
0000	193	:	
0000	194	:	
0000	195	:	
0000	196	:	
0000	197	:	
0000	198	:	
0000	199	:	
0000	200	:	
0000	201	:	
0000	202	:	
0000	203	:	
0000	204	:	
0000	205	:	
0000	206	:	
0000	207	:	
0000	208	:	
0000	209	:	
0000	210	:	
0000	211	:	
0000	212	:	
0000	213	:	
0000	214	:	
0000	215	:	
0000	216	:	
0000	217	:	
0000	218	:	
0000	219	:	
0000	220	:	
0000	221	:	
0000	222	:	
0000	223	:	
0000	224	:	
0000	225	:	
0000	226	:	
0000	227	:	
0000	228	:	
0000	229	:	
0000	230	:	
0000	231	:	
0000	232	:	
0000	233	:	
0000	234	:	
0000	235	:	
0000	236	:	
0000	237	:	
0000	238	:	
0000	239	:	
0000	240	:	
0000	241	:	
0000	242	:	
0000	243	:	
0000	244	:	
0000	245	:	
0000	246	:	
0000	247	:	
0000	248	:	
0000	249	:	
0000	250	:	
0000	251	:	
0000	252	:	
0000	253	:	
0000	254	:	
0000	255	:	
0000	256	:	
0000	257	:	
0000	258	:	
0000	259	:	
0000	260	:	
0000	261	:	
0000	262	:	
0000	263	:	
0000	264	:	
0000	265	:	
0000	266	:	
0000	267	:	
0000	268	:	
0000	269	:	
0000	270	:	
0000	271	:	
0000	272	:	
0000	273	:	
0000	274	:	
0000	275	:	
0000	276	:	
0000	277	:	
0000	278	:	
0000	279	:	
0000	280	:	
0000	281	:	
0000	282	:	
0000	283	:	
0000	284	:	
0000	285	:	
0000	286	:	
0000	287	:	
0000	288	:	
0000	289	:	
0000	290	:	
0000	291	:	
0000	292	:	
0000	293	:	
0000	294	:	
0000	295	:	
0000	296	:	
0000	297	:	
0000	298	:	
0000	299	:	
0000	300	:	
0000	301	:	
0000	302	:	
0000	303	:	
0000	304	:	
0000	305	:	
0000	306	:	
0000	307	:	
0000	308	:	
0000	309	:	
0000	310	:	
0000	311	:	
0000	312	:	
0000	313	:	
0000	314	:	
0000	315	:	
0000	316	:	
0000	317	:	
0000	318	:	
0000	319	:	
0000	320	:	
0000	321	:	
0000	322	:	
0000	323	:	
0000	324	:	
0000	325	:	
0000	326	:	
0000	327	:	
0000	328	:	
0000	329	:	
0000	330	:	
0000	331	:	
0000	332	:	
0000	333	:	
0000	334	:	
0000	335	:	
0000	336	:	
0000	337	:	
0000	338	:	
0000	339	:	
0000	340	:	
0000	341	:	
0000	342	:	
0000	343	:	
0000	344	:	
0000	345	:	
0000	346	:	
0000	347	:	
0000	348	:	
0000	349	:	
0000	350	:	
0000	351	:	
0000	352	:	
0000	353	:	
0000	354	:	
0000	355	:	
0000	356	:	
0000	357	:	
0000	358	:	
0000	359	:	
0000	360	:	
0000	361	:	
0000	362	:	
0000	363	:	
0000	364	:	
0000	365	:	
0000	366	:	
0000	367	:	
0000	368	:	
0000	369	:	
0000	370	:	
0000	371	:	
0000	372	:	
0000	373	:	
0000	374	:	
0000	375	:	
0000	376	:	
0000	377	:	
0000	378	:	
0000	379	:	
0000	380	:	
0000	381	:	
0000	382	:	
0000	383	:	
0000	384	:	
0000	385	:	
0000	386	:	
0000	387	:	
0000	388	:	
0000	389	:	
0000	390	:	
0000	391	:	
0000	392	:	
0000	393	:	
0000	394	:	
0000	395	:	
0000	396	:	
0000	397	:	
0000	398	:	
0000	399	:	
0000	400	:	
0000	401	:	
0000	402	:	
0000	403	:	
0000	404	:	
0000	405	:	
0000	406	:	
0000	407	:	
0000	408	:	
0000	409	:	
0000	410	:	
0000	411	:	
0000	412	:	
0000	413	:	
0000	414	:	
0000	415	:	
0000	416	:	
0000	417	:	
0000	418	:	
0000	419	:	
0000	420	:	
0000	421	:	
0000	422	:	
0000	423	:	
0000	424	:	
0000	425	:	
0000	426	:	
0000	427	:	
0000	428	:	
0000	429	:	
0000	430	:	
0000	431	:	
0000	432	:	
0000	433	:	
0000	434	:	
0000	435	:	
0000	436	:	
0000	437	:	
0000	438	:	
0000	439	:	
0000	440	:	
0000	441	:	
0000	442	:	
0000	443	:	
0000	444	:	
0000	445	:	
0000	446	:	
0000	447	:	
0000	448	:	
0000	449	:	
0000	450	:	
0000	451	:	
0000	452	:	
0000	453	:	
0000	454	:	
0000	455	:	
0000	456	:	
0000	457	:	
0000	458	:	
0000	459	:	
0000	460	:	
0000	461	:	
0000	462	:	
0000	463	:	
0000	464	:	
0000	465	:	
0000	466	:	
0000	467	:	
0000	468	:	
0000	469	:	
0000	470	:	
0000	471	:	
0000	472	:	
0000	473	:	
0000	474	:	
0000	475	:	
0000	476	:	
0000	477	:	
0000	478	:	
0000	479	:	
0000	480	:	
0000	481	:	
0000	482	:	
0000	483	:	
0000	484	:	
0000	485	:	
0000	486	:	
0000	487	:	
0000			


```
0000 142 : Change $EOD processing for SYS$INPUT so that RM1INPUT_SCAN
0000 143 : no longer worry about fitting into the buffer.
0000 144 : This change is necessary to make stream files work later.
0000 145 :
0000 146 : V02-038 KRM0020 K R Malik 1-JUL-1981
0000 147 : Do not clear RFA on network operation.
0000 148 :
0000 149 : V02-037 JAK0051 J A Krycka 23-DEC-1980
0000 150 : For task-to-task get operation, use the smaller of RAB$W_USZ
0000 151 : and IFB$W_DEVBUSIZ to determine the number of bytes to request
0000 152 : in a receive QIO system service call.
0000 153 :
0000 154 : V036 REFORMAT Ken Henderson 30-JUL-1980 5:49
0000 155 : the code was reformatted
0000 156 :
0000 157 : V035 JAK0045 J A Krycka 19-MAR-1980 15:00
0000 158 : fix bug in return of stv for network operation.
0000 159 :
0000 160 : V034 PSK0017 P S Knibbe 14-MAR-1980 11:00
0000 161 : sequential get should only round up record size if the
0000 162 : device is not magtape (checking for ansi is wrong).
0000 163 :
0000 164 : V033 JAK0041 J A Krycka 02-MAR-1980 14:00
0000 165 : issue new get after find if network operation, because
0000 166 : network find does not return a record.
0000 167 :
0000 168 : V032 CDS0012 C D Saether 23-FEB-1980 21:53
0000 169 : use larger of terminal width and user buffer size for read
0000 170 : request to terminals. move data if byte count non-zero.
0000 171 :
0000 172 : V031 PSK0004 P S Knibbe 18-DEC-1979 10:00
0000 173 : if get from terminal has control-y or control-c alternate
0000 174 : success status, don't return the tns error
0000 175 :
0000 176 : V030 PSK0002 P S Knibbe 26-NOV-1979 6:00
0000 177 : terminal escape sequences are placed in the user buffer if
0000 178 : there is room for them.
0000 179 :
0000 180 : V029 PSK0001 P S Knibbe 25-NOV-1979 12:00
0000 181 : a get on foreign magtape clears the irb$eof bit
0000 182 :
0000 183 : V028 CDS0011 C D Saether 26-JUN-1979 17:20
0000 184 : fixes so mailboxes work correctly as sys$input
0000 185 :
0000 186 :--
```

```
0000 188      .SBTTL DECLARATIONS
0000 189
0000 190 :
0000 191 : Include Files:
0000 192 :
0000 193 :
0000 194 :
0000 195 : Macros:
0000 196 :
0000 197 :
0000 198      $RABDEF
0000 199      $FABDEF
0000 200      $IRBDEF
0000 201      $IFBDEF
0000 202      $DEVDEF
0000 203      $BDBDEF
0000 204      $PIODEF
0000 205      $RMSDEF
0000 206      $SSDEF
0000 207
0000 208 :
0000 209 : Equated Symbols:
0000 210 :
0000 211 :
00000020 0000 212      ROP=RAB$L_ROP*8      : bit offset to rop
0000000A 0000 213      LF=10                : line feed
0000000B 0000 214      VT=11                : vertical tab
0000000C 0000 215      FF=12                : form feed
0000000D 0000 216      CR=13                : carriage return
0000001A 0000 217      CTRLZ=26             : control z
0000 218
0000 219 :
0000 220 :
0000 221 : Own Storage:
0000 222 :
0000 223 :
```



```

0000 225      .SBTTL RM1GETINT - INTERNAL GET/FIND SEQUENTIAL ALL DEVICES
0000 226
0000 227      :++
0000 228      : RM$GET_UNIT_REC
0000 229      : RM$GET_BLK_DEV
0000 230      : RM$GETRANDOM
0000 231
0000 232      : this module includes two routines to perform get/find
0000 233      : record processing for the sequential file organization.
0000 234
0000 235      : Calling sequence:
0000 236
0000 237      : bsbw  rm$get_unit_rec ; for unit record device
0000 238      : bsbw  rm$get_blk_dev ; for block-oriented device
0000 239
0000 240      : Input Parameters:
0000 241
0000 242      : r11    impure area addr
0000 243      : r10    ifab addr
0000 244      : r9     irab addr
0000 245      : r8     rab addr
0000 246
0000 247      : Implicit Inputs:
0000 248
0000 249      : the contents of the various structures.
0000 250
0000 251      : Output Parameters:
0000 252
0000 253      : r0      status code
0000 254      : r1-r7  destroyed
0000 255
0000 256      : Implicit Outputs:
0000 257
0000 258      : the internal structures are updated to reflect the
0000 259      : results of the get or find. the rab fields
0000 260      : are updated to correspond to the found or gotten
0000 261      : record (see functional spec for list).
0000 262
0000 263      : Completion Codes:
0000 264
0000 265      : standard rms.
0000 266
0000 267      : Side Effects:
0000 268
0000 269      : none
0000 270
0000 271      :--
0000 272

```

```
0000 274 .SBTTL RMSGET_UNIT_REC - UNIT RECORD GET ROUTINE
0000 275
0000 276 RMSGET_UNIT_REC::
54 50 01 D0 0000 277      MOVCL #1,R0          ; get a correct status in r0
20 A9 D0 0003 278      MOVL  IRB$$_CURBDB(R9),R4      ; get current bdb addr
15 13 0007 279      BEQL  NEXT          ; branch if none exists
05 6A 2E E1 0009 280      BBC   #IFB$$_PPF_INPUT,(R10),10$; br if not 'sys$input'
000D 281
000D 282 ;
000D 283 ; if this is the 'input' stream for this process, then retrec will perform
000D 284 ; end of deck checking. it indicates that the current record is to be
000D 285 ; re-retrieved on the next get by resetting the nrp context from the rp
000D 286 ; context (makes much sense for disk files). to detect that same condition
000D 287 ; here (mailboxes are reasonably used as sys$input), the nrp field is stuffed
000D 288 ; with a 1 before calling retrec. the rp field is left clear. therefore, if
000D 289 ; the nrp field is clear at this point, we are supposed to re-retrieve the
000D 290 ; same record as last time (i.e., get it out of the buffer). note that the
000D 291 ; nrp field can be anything following connect if curbdb is zero so that this
000D 292 ; check is skipped the first time through after connect.
000D 293 ;
000D 294
40 A9 D5 000D 295      TSTL  IRB$$_NRP(R9)          ; re-retrieve this record?
57 13 0010 296      BEQL  MOVEDATA          ; yep
0012 297
0012 298 ;
0012 299 ; if last operation was a find and this is a get, then re-use the record
0012 300 ; in curbdb, otherwise skip it and get a new one
0012 301 ;
0012 302
08 6A 3E E0 0012 303 10$: BBS   #IFB$$_DAP,(R10),NEXT      ; network find does not return
0016 304 ; any data, so get new record
0016 305
04 69 25 E1 0016 306      BBC   #IRB$$_FIND_LAST,(R9),NEXT; get another
4B 69 29 E1 001A 307      BBC   #IRB$$_FIND,(R9),MOVEDATA; this is a get, use rec found
001E 308
001E 309 ;
001E 310 ; irb$$_eof is used to mark whether or not the device is a end of file.
001E 311 ; because foreign magtapes permit a user to read past the end of the file,
001E 312 ; they must clear this bit on a read. foreign magtapes which were mounted
001E 313 ; with a record size specified do not go through this path.
001E 314 ;
001E 315
08 6A 05 E1 001E 316 NEXT: BBC   #DEV$$_SQD,IFB$$_PRIM_DEV(R10),10$; branch if not magtape
04 6A 18 E1 0022 317      BBC   #DEV$$_FOR,IFB$$_PRIM_DEV(R10),10$; branch if not foreign
0026 318      CSB   #IRB$$_EOF,(R9)          ; always clear eof on get
002A 319
6D 69 21 E0 002A 320 10$: BBS   #IRB$$_EOF,(R9),RMSGET_EOF      ; exit if at eof
002E 321
002E 322 ;
002E 323 ; get next record
002E 324 ;
002E 325
52 48 AA D0 002E 326      MOVL  IFB$$_DEVBUFSIZ(R10),R2      ; size of buffer
0C 6A 3F E1 0032 327      BBC   #IFB$$_NSP,(R10),15$          ; branch if not task-to-task operation
52 20 A8 B1 0036 328      CMPW  RAB$$_JSZ(R8),R2          ; use the smaller of user buffer size
18 1E 003A 329      BGEQU  20$          ; and device width (which for
52 20 A8 3C 003C 330      MOVZWL RAB$$_USZ(R8),R2          ; task-to-task communication is a
```



```
INTERNAL GET SEQUENTIAL
RM$GET_UNIT_REC - UNIT RECORD GET ROUTIN

12 11 0040 331 BRB 20$ ; predefined data transfer size limit)
02 E1 0042 332 15$: BBC ; branch if not terminal
06 68 OE 6A 0044 333 IFB$L PRIM_DEV(R10),20$ ;
3C E0 0046 334 BBS #RAB$V_ETOTROP,(R8),17$ ; use user buffer size for extended
004A 335 ; terminal reads
20 A8 52 B1 004A 336 CMPW R2,RAB$W_USZ(R8) ; use the larger of user buffer size
04 1E 004E 337 BGEQU 20$ ; and terminal width (as user request
52 20 A8 3C 0050 338 17$: MOVZWL RAB$W_USZ(R8),R2 ; may exceed one line of data)
OC A9 7C 0054 339 20$: CLRQ IRB$L_IOS(R9) ; init io status block
53 D4 0057 340 CLRL R3 ; read required
FFA4' 30 0059 341 BSBW RM$SEQRD ; get the record
005C 342
005C 343 ASSUME <BDB$B_REL_VBN+1> EQ BDB$B_VAL_VBNS
005C 344
48 A4 B4 005C 345 CLRW BDB$B_REL_VBN(R4) ; clear both fields
49 A4 96 005F 346 INCB BDB$B_VAL_VBNS(R4) ; make this a one (was zero)
0062 347 ; so rm$input_scan works
0062 348
0062 349
0062 350 ; if terminal device, make further checks.
0062 351
0062 352
5B 6A 02 E0 0062 353 BBS #DEV$V_TRM, IFB$L_PRIM_DEV(R10), TRM ; branch if terminal
0066 354
05 50 E9 0066 355 BLBC R0, ERR_CONT ; don't modify STV on error
0069 356
0069 357 ;
0069 358 ; return the record
0069 359
0069 360
OC A8 10 A9 D0 0069 361 MOVEDATA:
006E 362 MOVL IRB$L_IOS4(R9), RAB$L_STV(R8); return 2nd lngwrđ iosb to user
40 A9 01 D0 006E 363 ERR_CONT:
51 18 A4 D0 0072 364 MOVL #1,IRB$L_NRP(R9) ; stuff with 1 (see note above)
56 14 A4 3C 0076 365 MOVL BDB$L_ADDR(R4),R1 ; set up r1, r6, r7 for retrec
57 56 51 C1 007A 366 MOVZWL BDB$W_NUMB(R4),R6
03 6A 3E E1 007E 367 ADDL3 R1,R6,R7
00A8 30 0082 368 BBC #IFB$V_DAP,(R10),10$
50 DD 0085 369 BSBW NTVFC ; branch if network request
034C 30 0087 370 10$: PUSHL R0 ; save status
03 50 E8 008A 371 BSBW RETREC ; return the record
6E 50 D0 008D 372 BLBS R0, 20$ ; branch on ok
50 8ED0 0090 373 MOVL R0,(SP) ; set status
0093 374 20$: POPL R0 ; restore status
03 6A 3E E0 0093 375 ZERO_RFA1:
10 A8 7C 0097 376 BBS #IFB$V_DAP,(R10),10$ ; Branch if network request
05 009A 377 CLRQ RAB$W_RFA(R8) ; zero rfa
378 10$: RSB ; return to caller
```



```
009B 380
009B 381 :++
009B 382 :
009B 383 : eof error
009B 384 :
009B 385 :--
009B 386 :
009B 387 RMSGET_EOF::
009B 388 RMSERR EOF
00A0 389
00A0 390 :
00A0 391 : if device is a terminal reset eof flag so that eof error is given only once
00A0 392 :
00A0 393 :
06 6A 02 E1 00A0 394 BBC #DEV$V_TRM,IFB$$_PRIM_DEV(R10),10$; branch unless terminal
00A4 395 CSB #IRB$V_ECF,(R9) ; reset eof flag
14 11 00A8 396 BRB 20$ ; and branch to common exit
00AA 397
00AA 398 :
00AA 399 : if this is sys$input (on other than a terminal) reset end of data scan string
00AA 400 :
00AA 401 :
10 6A 2E E1 00AA 402 10$: BBC #IFB$V_PPF_INPUT,(R10),20$ ; branch if not sys$input
00000000'9F 02 8A 00AE 403 BICB2 #1@PIOS$V_EOD,@#PIOS$GW_STATUS ; clear '$eod' flag
00000000'9F 2401 8F B0 00B5 404 MOVW #1+<^A/$7 @ 8>,@#PIOS$GT_ENDSTR ; reset eod string to single '$'
015F 31 00BE 405 20$: BRW ZERO_RFA
00C1 406
00C1 407 :++
00C1 408 :
00C1 409 : if a sequential read returned a partial escape or a bad escape error
00C1 410 : map to rms error. this will be overridden by retrec errors (e.g., rtb).
00C1 411 : in any case, continue to move any data returned (byte count in iosb).
00C1 412 :
00C1 413 :--
00C1 414 :
00C1 415 TRM:
COF4 8F 21 50 E8 00C1 416 BLBS R0,20$ ; successful read?
00C4 417 CMPW R0,#<RMS$_RER&^XFFFF> ; read error ?
3C 12 00C9 418 BNEQ 35$ ; read error - check for specl
00CB 419
00CB 420 :
00CB 421 : continue to return any data present. this error will be preserved unless
00CB 422 : retrec gets an error.
00CB 423 :
00CB 424 :
3C 0C A9 B1 00CB 425 CMPW IRB$$_IOS(R9), #SS$_BADESCAPE; yes - bad escape sequence ?
07 12 00CF 426 BNEQU 10$ ; no - check next
00D1 427 RMSERR BES,R0 ; yes - map error
0D 11 00D6 428 BRB 20$ ; continue
00D8 429
00D8 430 :
00D8 431 : continue to return any data present. this error will be preserved unless
00D8 432 : retrec gets an error.
00D8 433 :
00D8 434 :
01FC 8F 0C A9 B1 00D8 435
00D8 436 10$: CMPW IRB$$_IOS(R9), #SS$_PARTESCAPE; is it partial escape sequence?
```



```
27 12 00DE 437      BNEQU 35$      ; no - continue
      00E0 438      RMSERR PES,R0    ; use rms error
      00E5 439
      00E5 440
      00E5 441      ; special checks for terminal devices after qio is done. adjust
      00E5 442      ; bdb$w_numb to include escape sequences and cause them to be returned.
      00E5 443
      00E5 444
01 12 A9 91 00E5 445 20$: CMPB IRB$ _IOS4+2(R9), #1 ; 1 byte terminator?
      08 13 00E9 446      BEQL 30$      ; if so don't copy it
52 12 A9 9A 00EB 447      MOVZBL IRB$ _IOS4+2(R9), R2 ; extract terminator length
14 A4 52 A0 00EF 448      ADDW2 R2, BDB$W_NUMB(R4) ; add in escape sequence length
1A 10 A9 91 00F3 449 30$: CMPB IRB$ _IOS4(R9), #CTRLZ ; record terminate with ^z?
      09 12 00F7 450      BNEQ 34$      ; branch if not
      0E A9 B5 00F9 451      TSTW IRB$W _IOS2(R9) ; test size returned
      9D 13 00FC 452      BEQL RMSGET_EOF ; make this eof if zero
      00FE 453      SSB #IRB$V_EOF, (R9) ; set eof flag
OC A8 10 A9 D0 0102 454 34$: MOVL IRB$ _IOS4(R9), RAB$ _STV(R8); return 2nd lngwrds iosb to user
      FF64 30 0107 455 35$: BSBW ERR CONT ; move the data
1B 69 29 E0 010A 456      BBS #IRB$V_FIND, (R9), 50$ ; is this is find, leave now
01 12 A9 91 010E 457      CMPB IRB$ _IOS4+2(R9), #1 ; 1 byte terminator?
      15 13 0112 458      BEQL 50$      ; no sweat, all done
      0B 1A 0114 459      BGTRU 40$      ; branch if escape sequence
OC A9 01 B1 0116 460      CMPW #1, IRB$ _IOS(R9) ; straight success so far?
      011A 461
      011A 462
      011A 463      ; check in iosb so that a get following find that got
      011A 464      ; cntrl^c or y alt succ does not get tns error.
      011A 465
      011A 466
      0D 12 011A 467      BNEQU 50$      ; no, then leave error as is
      011C 468      RMSERR TNS ; else say no terminator seen
52 12 A9 9A 0121 469 40$: MOVZBL IRB$ _IOS4+2(R9), R2 ; extract terminator length
22 A8 52 A2 0125 470      SUBW2 R2, RAB$W_RSZ(R8) ; adjust record size reported
      05 0129 471 50$: RSB ; so that escape sequence is
      012A 472      ; not included.
      012A 473
      012A 474 ZERO_RFA1 BR:
      FF66 31 012A 475 BRW ZERO_RFA1 ; and out
      012D 476
      012D 477 :++
      012D 478
      012D 479 ; network specific code to update rhb if record in vfc format
      012D 480
      012D 481 :--
      012D 482
03 50 AA 91 012D 483 NTVFC: CMPB IFB$B _RFMORG(R10), #FAB$C_VFC; vfc format?
      2B 12 0131 484      BNEQ 30$      ; branch if not
      50 DD 0133 485      PUSHL R0 ; save old status
52 5F AA 9A 0135 486      MOVZBL IFB$B _FSZ(R10), R2 ; pick up fixed header size
62 A9 52 A0 0139 487      ADDW2 R2, IRB$W_CSIZ(R9) ; adjust total record size
      56 52 C2 013D 488      SUBL2 R2, R6 ; adjust remaining rec size
      1D 1F 0140 489      BLSSU 40$ ; branch if fsz > available data
53 2C A8 D0 0142 490      MOVL RAB$ _RHB(R8), R3 ; get user's header buffer
      10 13 0146 491      BEQL 10$ ; and branch if none
      0148 492
      0148 493 ;
```

```

0148 494 : move vfc header to user
0148 495 :
0148 496 :
OC 69 29 E0 0148 497 BBS #IRB$V_FIND,(R9),10$ ; branch if this is a find
      0319 30 014C 498 BSBW GTMVRTC ; move just like record
      06 50 E8 014F 499 BLBS R0,10$ ; branch if ok
      0212 30 0152 500 BSBW ERRRHB ; map the error maybe
      6E 50 D0 0155 501 MOVL R0,(SP) ; set status
      0158 502 :
      0158 503 :
      0158 504 : skip over unwanted rhb
      0158 505 :
      0158 506 :
      51 52 C0 0158 507 10$: ADDL2 R2,R1 ; bump source pointer
      50 8ED0 015B 508 20$: POPL R0 ; restore status
      05 015E 509 30$: RSB ; rejoin mainline
      015F 510 :
      56 D4 015F 511 40$: CLRL R6 ; show no data
      F8 11 0161 512 BRB 20$

```



```
0163 514 .SBTTL RMSGET_BLK_DEV - BLOCK DEVICE GET ROUTINE
0163 515
0163 516 :++
0163 517 :
0163 518 : perform get from a block device.
0163 519 :
0163 520 :--
0163 521 :
0163 522 :++
0163 523 :
0163 524 : make longer version of eof check.
0163 525 : (note: this label is not a global entry point)
0163 526 :
0163 527 : this module checks for the current record position at or
0163 528 : beyond the logical end of file, setting
0163 529 : the irb$eof bit if so.
0163 530 :
0163 531 :--
0163 532 :
0163 533 CHKEOF1:
0163 534 BBS #IRB$V_EOF,(R9),10$ ; branch if already eof
74 AA 69 21 E0 0167 535 CMPL IRB$L_NRP_VBN(R9),IFB$L_EBK(R10) ; at eof?
40 A9 D1 016C 536 BLSSU GETNRP ; branch if not yet
25 1F 016E 537 BGTRU 10$ ; branch if definitely
07 1A 0170 538 ; in current eof block
5C AA 44 A9 B1 0170 539 CMPW IRB$W_NRP_OFF(R9),IFB$W_FFB(R10) ; how about byte position?
1C 1F 0175 540 BLSSU GETNRP ; nope - not there yet
FF1D 31 0177 541 10$: SSB #IRB$V_EOF,(R9) ; eof - set the flag
017B 542 BRW RMSGET_EOF ; report error
017E 543
017E 544 :++
017E 545 :
017E 546 : get next block (note: this label is not a global entry point)
017E 547 :
017E 548 :--
017E 549 :
0344 30 017E 550 GET00: BSBW NXTBLK
A6 50 E9 0181 551 BLBC R0,ZERO_RFA1_BR ; get out on error
0184 552
0184 553 :++
0184 554 :
0184 555 : entry point to read a full buffer (i.e., mbc+1 blocks)
0184 556 :
0184 557 :--
0184 558 :
53 D4 0184 559 RMSGET_BLK_DEV::
0184 560 CLRL R3 ; flag full read required
0186 561
0186 562 :++
0186 563 :
0186 564 : entry point for random reads to transfer less than mbc+1 blocks
0186 565 : r3=short read flag (#2) and r2=# blocks to read minus 1
0186 566 :
0186 567 :--
0186 568 :
0186 569 RMSGETRANDOM::
0186 570 $STPT GETBLKD
```

```
74 AA 40 A9 D1 018C 571      CMPL  IRB$N_NRP_VBN(R9),IFB$N_NRP_VBN(R9); at or past eof block?
      D0 1E 0191 572      BGEQU  CHKEOF1      ; yes - make long check
      0193 573
      0193 574      :
      0193 575      : must get the block specified by nrp
      0193 576      :
      0193 577
      FE6A' 30 0193 578 GETNRP: BSBW  RMSGETBLKNRP      ; get the block
      91 50 E9 0196 579      BLBC  R0,ZERO RFA1 BR      ; get out on error
      51 44 A9 C0 0199 580      ADDL2 IRB$N_NRP_OFF(R9),R1 ; make offset into address
48 A9 40 A9 7D 019D 581      MOVQ  IRB$N_NRP_VBN(R9),IRB$N_NRP_VBN(R9); set rp from nrp
      01A2 582
      01A2 583      :++
      01A2 584      :
      01A2 585      : dispatch based on record format
      01A2 586      :
      01A2 587      :--
      01A2 588
      01A2 589      ASSUME  FAB$C_UDF      EQ  0
      01A2 590      ASSUME  FAB$C_FIX      EQ  1
      01A2 591      ASSUME  FAB$C_VAR      EQ  2
      01A2 592      ASSUME  FAB$C_VFC      EQ  3
      01A2 593      ASSUME  FAB$C_STM      EQ  4
      01A2 594      ASSUME  FAB$C_STMCR    GT  FAB$C_STM
      01A2 595      ASSUME  FAB$C_STMLF    GT  FAB$C_STM
      01A2 596
      55 50 AA 9A 01A2 597      MOVZBL IFB$B_RFMORG(R10),R5 ; case on record format
      OF 13 01A6 598      BEQL  10$      ; UDF
      02 55 91 01A8 599      CMPB  R5,#FAB$C_VAR
      18 13 01AB 600      BEQL  VARSIZ      ; VAR
      7C 1F 01AD 601      BLSSU  FIX$IZ      ; FIX
      04 55 91 01AF 602      CMPB  R5,#FAB$C_STM
      06 1F 01B2 603      BLSSU  20$      ; VFC
      00F0 31 01B4 604      BRW  STMSIZ      ; STM11, STMLF, STMCR
      01B7 605
      00B2 31 01B7 606 10$: BRW  UDFSIZ
      01BA 607
      0140 31 01BA 608 20$: BRW  VFCSIZ
```



```
01BD 610 :++
01BD 611 :
01BD 612 : Variable or vfc size
01BD 613 :
01BD 614 : the total size field is, by definition (i.e., $put), contained in a single
01BD 615 : block, thus it can be accessed without regard for crossing block boundaries.
01BD 616 :
01BD 617 : the size field is in one of the following two formats:
01BD 618 :
01BD 619 :     binary - word-aligned signed 15-bit integer (size of data)
01BD 620 :             - -1 = end of block
01BD 621 :             - -n illegal
01BD 622 :
01BD 623 :     ansi d - byte-aligned 4 decimal ascii char count (size of data + 4)
01BD 624 :             - "x" = end of block
01BD 625 :             - all other chars illegal (rms$_ani)
01BD 626 :
01BD 627 :
01BD 628 : .ENABLE LSB
01BD 629 :
0102 30 01BD 630 10$: BSBW GET_ANSI_D ; get decimal byte count
BB 50 E9 01C0 631 BLBC R0,GET00 ; error implies reached end-of-block
16 11 01C3 632 BRB 20$ ; continue with checking
F4 6A 26 E0 01C5 633 VARSIZ: BBS #IFB$V_ANSI_D,(R10),10$ ; binary size field?
01C9 634
01C9 635 :
01C9 636 : binary size field
01C9 637 :
01C9 638 :
01C9 639 :
52 57 51 C3 01C9 640 SUBL3 R1,R7,R2 ; get size of buffer remaining
02 52 D1 01CD 641 CMPL R2,#2 ; room for count?
AC 19 01D0 642 BLSS GET00 ; no, onto next block
62 A9 02 B0 01D2 643 MOVW #2,IRB$W_CSIZ(R9) ; note overhead for size field
56 81 32 01D6 644 CVTWL (R1)+,R6 ; get rec length
49 19 01D9 645 BLSS GET_CHKMINUS ; branch if negative
01DB 646
01DB 647 :++
01DB 648 :
01DB 649 : if records are not allowed to cross block boundaries, see if it
01DB 650 : does and is therefore illegal.
01DB 651 :
01DB 652 :--
01DB 653 :
09 51 AA 03 E1 01DB 654 20$: BBC #FAB$V_BLK,IFB$B_RAT(R10),GET_RETREC
50 57 51 C3 01E0 655 SUBL3 R1,R7,R0 ; get size of buffer remaining
50 56 D1 01E4 656 CMPL R6,R0 ; rec len > # left?
76 1A 01E7 657 BGTRU ERRIRC ; branch if so
01E9 658
01E9 659 :++
01E9 660 :
01E9 661 : now return record to user
01E9 662 :
01E9 663 :--
01E9 664 :
01EA 30 01E9 665 GET_RETREC: BSBW RETREC ; return record to user
01E9 666
```

```
2A 50 E9 01EC 667 BLBC R0,60$ ; get out on error
01EF 668
01EF 669 :
01EF 670 : at this point get has been done for user.
01EF 671 : now just round up nrp_off to a word boundary unless magtape
01EF 672 :
01EF 673
0A 6A 05 E0 01EF 674 30$: BBS #DEV$V_SQD,IFB$$_PRIM_DEV(R10),40$; branch if magtape
01F3 675
01F3 676 ASSUME FAB$$_UDF EQ 0
01F3 677
50 AA 95 01F3 678 TSTB IFB$$_RFMORG(R10) ; UDF format?
05 13 01F6 679 BEQL 40$ ; no rounding needed
51 D6 01F8 680 INCL R1 ; word align next byte pointer
51 01 8A 01FA 681 BICB2 #1,R1
54 20 A9 D0 01FD 682 40$: MOVL IRB$$_CURBDB(R9),R4 ; get current bdb address
44 A9 51 4C A4 C3 0201 683 SUBL3 BDB$$_CURBUFADR(R4),R1,IRB$$_NRP_OFF(R9); make byte pointer relative
10 A8 48 A9 7D 0207 684 MOVQ IRB$$_RP_VBN(R9),RAB$$_RFA(R8); set rfa
57 51 B1 020C 685 CMPW R1,R7 ; check for at end of block
01 13 020F 686 BEQL 50$ ; branch if not
05 0211 687 RSB ; and return
0212 688
40 A9 D6 0212 689 50$: INCL IRB$$_NRP_VBN(R9) ; at end of block
44 A9 B4 0215 690 CLRW IRB$$_NRP_OFF(R9) ; bump vbn and clear offset in block
05 0218 691 RSB ; and return
0219 692
0219 693 :
0219 694 : handle errors
0219 695 : received error from retrec.
0219 696 : if error is rtb, continue anyway, else get out.
0219 697 :
0219 698
81A8 8F 50 B1 0219 699 60$: CMPW R0,#RMS$$_RTB&^XFFFF ; is this the rtb error?
CF 13 021E 700 BEQL 30$ ; branch if yes
0220 701
0220 702 :++
0220 703 :
0220 704 : error exit for block device
0220 705 :
0220 706 :--
0220 707
0220 708 ZERO_RFA:
10 A8 7C 0220 709 CLRQ RAB$$_RFA(R8) ; zero rfa
05 05 0223 710 RSB
0224 711
0224 712 :++
0224 713 :
0224 714 : check for size = -1 (end of block flag)
0224 715 :
0224 716 :--
0224 717
0224 718 GET_CHKMINUS:
56 D6 0224 719 INCL R6 ; size = -1?
37 12 0226 720 BNEQ ERRIRC ; handle error
FF53 31 0228 721 BRW GET00 ; branch if yes -
022B 722 ; says get next block
022B 723 .DISABLE LSB
```



```
022B 725 :++
022B 726 :
022B 727 : fixed size record - pick up the record length from ifb
022B 728 : and check for a record of all "" chars on magtape, which is ignored.
022B 729 : if record format is fixed, there is no size field
022B 730 :
022B 731 :--
022B 732 :
56 52 AA 32 022B 733 FIXSIZ: CVTWL IFB$W_LRL(R10),R6 ; get fixed rec length
16 6A 2E 19 022F 734 BLSS ERRIRC ; handle unreasonable size error
50 53 51 D0 0231 735 BBC #IFB$V_ANSI_D,(R10),20$ ; proceed if not tape
61 56 5E 8F 3B 0238 736 10$: MOVL R1,R3 ; save buffer ptr
50 57 51 C3 0238 737 SUBL3 R1,R7,R0 ; #bytes left=end-start
50 50 56 D1 023C 738 CMPL R6,R0 ; rec len > # left?
50 50 56 D1 023F 739 BGTRU 20$ ; proceed if so
99 51 AA 03 E1 0241 740 SKPC #^A\^\\,R6,(R1) ; is record all "" chars?
50 57 51 C3 0246 741 BEQL 10$ ; no, so have valid record
50 50 56 D1 0248 742 MOVL R3,R1 ; restore buffer ptr
99 51 AA 03 E1 024B 743 20$: BBC #FAB$V_BLK,IFB$B_RAT(R10),GET_RETREC
50 57 51 C3 0250 744 SUBL3 R1,R7,R0 ; #bytes left=end-start
50 50 56 D1 0254 745 CMPL R6,R0 ; rec len > # left?
50 50 90 1B 0257 746 BLEQU GET_RETREC ; branch if not
0259 747 :
0259 748 :
0259 749 : fixed length record format, no crossing block boundaries allowed.
0259 750 : if size not larger than a block then go back and get next block.
0259 751 :
0259 752 :
48 AA 56 B1 0259 753 CMPW R6,IFB$L_DEVBUFSIZ(R10) ; record len > 1 block?
50 60 1B 025D 754 BLEQU GET01 ; branch if yes
025F 755 :
025F 756 :++
025F 757 :
025F 758 : illegal record format
025F 759 :
025F 760 :--
025F 761 :
025F 762 ERRIRC:
OC A8 40 A9 D0 025F 763 RMSERR IRC
50 57 51 C3 0264 764 MOVL IRB$L_NRP_VBN(R9),RAB$L_STV(R8); indicate vbn having bad record
50 50 90 1B 0269 765 BRW ZERO_RFA
```

```
026C 767 :  
026C 768 : for UDF format, the users buffer size IS the record size.  
026C 769 : however, we must perform our own logical end-of-file checking  
026C 770 : and reduce the last record length accordingly.  
026C 771 : Eof will be detected on the next $GET.  
026C 772 :  
026C 773 :  
026C 774 UDFSIZ:  
56 20 A8 3C 026C 775 MOVZWL RAB$W_USZ(R8),R6 : get requested size  
50 48 AA D0 0270 776 MOVL IFB$W_DEVBUFSIZ(R10),R0 : get magic buffersize number  
40 A9 C3 0274 777 SUBL3 IRB$W_NRP_VBN(R9),-  
52 74 AA A3 0277 778 IFB$W_EBK(R10),R2 : get # blks till EOF  
44 A9 A3 027A 779 SUBW3 IRB$W_NRP_OFF(R9),-  
53 5C AA 027D 780 IFB$W_FFBR10),R3 : and offset too  
05 18 0280 781 BGEQ 20$ : negative means blk underflow  
52 D7 0282 782 DECL R2 : reduce # blks  
53 50 A0 0284 783 ADDW2 R0,R3 : and add a blk  
54 56 D0 0287 784 20$: MOVL R6,R4 : prepare for divide  
55 54 55 D4 028A 785 CLRL R5 : convert size to blk,byte format  
54 54 50 7B 028C 786 EDIV R0,R4,R4,R5 : in R4/R5  
52 54 D1 0291 787 CMPL R4,R2 : adequate # of blks avail?  
07 1A 0294 788 BGTRU 30$ : no room  
0C 1F 0296 789 BLSSU 40$ : lots of room  
53 55 B1 0298 790 CMPW R5,R3 : enough bytes maybe?  
07 1F 029B 791 BLSSU 40$ : yes  
56 52 50 C5 029D 792 30$: MULL3 R0,R2,R6 : use the remaining space up  
56 53 A0 02A1 793 ADDW2 R3,R6 : for this record  
FF42 31 02A4 794 40$: BRW GET_RETREC  
02A7 795 :  
02A7 796 :  
02A7 797 : stream format -- skip the trailing '^' chars in the block for  
02A7 798 : magtape.  
02A7 799 :  
02A7 800 STMSIZ:  
11 6A 26 E1 02A7 801 BBC #IFB$W_ANSI_D,(R10),20$ : only check for ansi tape  
52 04 D0 02AB 802 MOVL #4,R2 : record count  
57 51 D1 02AE 803 10$: CMPL R1,R7 : end-of-block?  
0C 13 02B1 804 BEQL GET01 : yes, go to next block  
5E 8F 81 91 02B3 805 CMPB (R1)+,^A^ : is char eql '^'?  
06 13 02B7 806 BEQL GET01 : yes, go to next block  
F2 52 F5 02B9 807 SOBGTR R2,10$ : for entire record count  
FD41' 31 02BC 808 20$: BRW RM$GET_STM_FMT : retrieve the record  
FEBC 31 02BF 809 GET01: BRW GET00 : go to next block  
02C2 811 :  
02C2 812 :++  
02C2 813 :  
02C2 814 : this is a get of a variable (or vfc) ansi magtape record.  
02C2 815 :  
02C2 816 : interpret the first four bytes of the record as the length in decimal ascii.  
02C2 817 :  
02C2 818 :--  
02C2 819 :  
02C2 820 GET_ANSI_D:  
62 A9 04 B0 02C2 821 MOVW #4,IRB$W_CSIZ(R9) : note overhead  
52 04 7D 02C6 822 MOVQ #4,R2 : pick up 4 bytes and clear temp  
56 D4 02C9 823 CLRL R6 : build resultant rec. len. here
```



```
INTERNAL GET SEQUENTIAL
RMSGET_BLK_DEV - BLOCK DEVICE GET ROUTIN

53  81  50  D4  02CB  824  10$:  CLRL  R0          ; assume failure
      30  83  02CD  825          SUBB3  #^A/0/,(R1)+,R3 ; pick up digit & remove bias
      1B  19  02D1  826          BLSS  30$          ; branch if bad
09    53  91  02D3  827          CMPB  R3,#9         ; is it a digit?
      11  1A  02D6  828          BGTRU 20$          ; branch if not
56    0A  A4  02D8  829          MULW2  #10,R6       ; shift previous partial result
56    53  A0  02DB  830          ADDW2  R3,R6       ; add in new digit
      EC  52  F5  02DE  831          SOBGTR R2,10$    ; continue if more digits
56    04  C2  02E1  832          SUBL2  #4,R6       ; adjust for size field itself
      12  19  02E4  833          BLSS  15$          ; bad size?
      50  D6  02E6  834          INCL  R0           ; set success
      05  02E8  835 12$:  RSB
      02E9  836
      02E9  837
      02E9  838 ; character was > 9. check for '^' implying end of block.
      02E9  839 ;
      02E9  840
2E    53  91  02E9  841 20$:  CMPB  R3,#<^A/^/ - ^A/0/> ; was invalid character '^'?
      FA  13  02EC  842          BEQL  12$          ; return failure which implies
      02EE  843          ; go read next block
      02EE  844
      02EE  845 ;
      02EE  846 ; invalid ansi d format.
      02EE  847 ;
      02EE  848
      01  BA  02EE  849 30$:  POPR  #^M<R0>         ; discard return pc
      FF28 31  02F0  850          RMSERR ANI
      02F5  851          BRW  ZERO_RFA             ; get out on error
      01  BA  02F8  852
      02F8  853 15$:  POPR  #^M<R0>         ; discard return pc
      FF62 31  02FA  854 ERRIRC1: BRW  ERRIRC       ; report bad record size
      02FD  855
      02FD  856
```

```
02FD 858 :++
02FD 859 :
02FD 860 : if vfc, process fixed header
02FD 861 :
02FD 862 :--
02FD 863 :
08 6A 26 E1 02FD 864 VFCSIZ: BBC #IFBSV ANSI_D,(R10),5$
      FFBE 30 0301 865 BSBW GET ANSI_D ; branch if ansi d
      B8 50 E9 0304 866 BLBC R0,GET01- ; next block on failure
      15 11 0307 867 BRB 20$ ; continue with checking
      0309 868 :
      0309 869 :
      0309 870 : binary size field
      0309 871 :
      0309 872 :
50 57 51 C3 0309 873 5$: SUBL3 R1,R7,R0 ; get size of buffer remaining
      02 50 D1 030D 874 CMPL R0,#2 ; room for count?
      AD 19 0310 875 BLSS GET01 ; no, onto next block
62 A9 02 B0 0312 876 MOVW #2,IRBSW_CSIZ(R9) ; note overhead for size field
      56 81 32 0316 877 CVTWL (R1)+,R6 ; get rec length
      03 18 0319 878 BGEQ 20$ ; branch if okay
      FF06 31 031B 879 BRW GET_CHKMINUS ; chk for end-of-block
      031E 880 :
      031E 881 :++
      031E 882 :
      031E 883 : if records are not allowed to cross block boundaries, see if it
      031E 884 : does and is therefore illegal.
      031E 885 :
      031E 886 :--
      031E 887 :
09 51 AA 03 E1 031E 888 20$: BBC #FABS$V BLK,IFBSB_RAT(R10),30$
50 57 51 C3 0323 889 SUBL3 R1,R7,R0 ; #bytes left=end-start
      50 56 D1 0327 890 CMPL R6,R0 ; rec len > # left?
      CE 1A 032A 891 BGTRU ERRIRC1 ; bad record if so
      52 5F AA 9A 032C 892 30$: MOVZBL IFBSB_FSZ(R10),R2 ; pick up fixed header size
62 A9 52 A0 0330 893 ADDW2 R2,IRBSW_CSIZ(R9) ; adjust total record size
      56 52 C2 0334 894 SUBL2 R2,R6 ; adjust remaining rec size
      06 69 29 E0 0337 895 BBS #IRBSV FIND,(R9),40$ ; branch if this is a find
      53 2C A8 D0 033B 896 MOVL RABS$ _RHB(R8),R3 ; get user's header buffer
      1B 12 033F 897 BNEQ 60$ ; and branch if there is one
      0341 898 :
      0341 899 :
      0341 900 : skip over unwanted rhb
      0341 901 :
      0341 902 :
50 57 51 C3 0341 903 40$: SUBL3 R1,R7,R0 ; get # bytes left in buffer
      52 50 D1 0345 904 CMPL R0,R2 ; less than fsz?
      0C 1E 0348 905 BGEQU 50$ ; branch if not
7E 52 50 C3 034A 906 SUBL3 R0,R2,-(SP) ; get remaining count
      0174 30 034E 907 BSBW NXTBLK ; get new buffer
      04 BA 0351 908 POPR #^M<R2> ; restore remaining count
      0E 50 E9 0353 909 BLBC R0,100$ ; get out on error
      0356 910 :
      0356 911 :
      0356 912 : (FSZ DEFINITELY IN THIS BLOCK)
      0356 913 :
      0356 914 :
```



```
INTERNAL GET SEQUENTIAL
RMSGET_BLK_DEV - BLOCK DEVICE GET ROUTIN

51 52 C0 0356 915 50$: ADDL2 R2,R1 ; bump source pointer
FE8D 31 0359 916 55$: BRW GET_RETREC ; rejoin var line
      035C 917 ;
      035C 918 ;
      035C 919 ; move vfc header to user
      035C 920 ;
      035C 921 ;
0109 30 035C 922 60$: BSBW GTMVRC ; move just like record
F7 50 E8 035F 923 BLBS RO,55$ ; get out on error
03 10 0362 924 BSBB ERRRHB ; map the error maybe
FEB9 31 0364 925 100$: BRW ZERO_RFA
      0367 926
      0367 927 ;++
      0367 928 ;
      0367 929 ; handle invalid record header buffer
      0367 930 ;
      0367 931 ;--
      0367 932
86EC 8F 50 R1 0367 933 ERRRHB:
05 12 036C 934 CMPW RO,#RMS$_UBF&^XFFFF ; was error bad ubf?
      036E 935 BNEQ 100$ ; branch if not
05 0373 936 RMSERR RHB ; switch error code to bad rhb
      937 100$: RSB
```



```
0374 939 .SBTTL INTERNAL UTILITY ROUTINES
0374 940 :++
0374 941 : CHKEOD
0374 942 : RETREC
0374 943 : RETREC1
0374 944 : SUCXIT
0374 945 : UPDSRC
0374 946 : MOVEMODE
0374 947 : SETRSZ
0374 948 : ERRUSZ
0374 949 : ERRUBF
0374 950 : ERRUBF1
0374 951 : SKPREC
0374 952 :
0374 953 : retrec routine to return record data to user.
0374 954 : handles crossing block boundaries and locate mode.
0374 955 :
0374 956 : Input Parameters:
0374 957 :
0374 958 :     r11      impure area addr
0374 959 :     r10      ifab addr
0374 960 :     r9       irab addr
0374 961 :     r8       rab addr
0374 962 :     r7       addr of end of buffer+1
0374 963 :     r6       size of record in bytes
0374 964 :     r1       addr of record in the buffer
0374 965 :
0374 966 : Implicit Inputs:
0374 967 :
0374 968 : the contents of the various structures.
0374 969 :
0374 970 : outputs:
0374 971 :
0374 972 :     r7       end addr of current i/o buffer+1 (updated)
0374 973 :     r2-r6    destroyed
0374 974 :     r1       pointer to next byte in block
0374 975 :     r0       status code
0374 976 :
0374 977 : Implicit Outputs:
0374 978 :
0374 979 :     irb$l_curbdb      may be updated
0374 980 :     irb$w_cursiz=irb$w_cursiz+record size
0374 981 :     rab$l_rbf
0374 982 :     rab$w_rsz
0374 983 :     rab$l_stv      =total record size if rtb error
0374 984 :     rab$l_ubf      record moved here if move mode
0374 985 :
0374 986 : completion code:
0374 987 :
0374 988 :     standard rms.    upon an error irb$l_ios and irb$l_ios4
0374 989 :     may have additional information.
0374 990 :
0374 991 : Side Effects:
0374 992 :
0374 993 :     may cause i/o if record crosses block boundary.
0374 994 :
0374 995 :--
```



```
0374 996
0374 997 :++
0374 998
0374 999 : this is get from sys$input - make additional logical eof checks
0374 1000 : (note: this is not the entry point to the retrec subroutine)
0374 1001 :
0374 1002 :--
0374 1003
69 22 E1 0374 1004 CHKEOD: BBC #IRB$V_PPF_IMAGE,(R9),- ; branch if not indirect
62 00 E0 0377 1005 RETRECT
00 6A 0378 1006 BBS #DEV$V_REC,- ; branch if record oriented device
5E 037A 1007 IFB$L_PRIM_DEV(R10),- ; (terminal, mailbox, network)
037B 1008 RETRECT1
037C 1009
037C 1010 :++
037C 1011
037C 1012 : Fix up the buffer pointer so that the $EOD scan will complete
037C 1013 : successfully.
037C 1014 : This may involve re-positioning the buffer so that the current record is
037C 1015 : in the first block. We assume the RM$CONNECT1 has forced MBC>0 so that
037C 1016 : at least 2 blocks are present in each buffer.
037C 1017
037C 1018 : Check if entire record is contained in the buffer. If not, check that
037C 1019 : at least 16 characters of the current record are contained in the block
037C 1020 : as that is sufficient for the $EOD string checking.
037C 1021 : Note that the current record is not entirely contained within the buffer
037C 1022 : can only happen with disk files.
037C 1023
037C 1024 : This check assumes that mbc is non-zero for disk files where records cross
037C 1025 : block boundaries. in this case, bdb$b_val_vbns gives the number of valid
037C 1026 : blocks in the buffer, and bdb$b_rel_vbn specifies which block within the
037C 1027 : buffer is currently being processed, in the range of 0 to irb$b_mbc.
037C 1028
037C 1029 : Calculate in r5 the total # of bytes remaining in this and any subsequent
037C 1030 : blocks in the buffer.
037C 1031
037C 1032 : note: if sys$input is from a disk file and records are allowed to
037C 1033 : cross block boundaries, irb$b_mbc must be greater than 0,
037C 1034 : otherwise the processing of a record crossing a block boundary
037C 1035 : will cause rms to loop. it is assumed that rm$connect1 has
037C 1036 : forced mbc > 0.
037C 1037
037C 1038 : If the record is not contained in the current buffer,
037C 1039 : cause the buffer to be read again, but with the current block as the
037C 1040 : first block in the buffer. Since the buffer is a multiple block buffer,
037C 1041 : the new read will cause the next part of the current record to be
037C 1042 : resident also.
037C 1043
037C 1044 : note: worst case is a vfc record with a 255 byte fixed part, with the
037C 1045 : byte count coming as the last word of a block. in this case
037C 1046 : and with a 2-block buffer, the actual record portion to be
037C 1047 : scanned must occur in the second block
037C 1048 :--
037C 1049
57 51 D1 037C 1050 CMPL R1,R7 ; is this the end of the buffer?
08 12 037F 1051 BNEQ 10$ ; branch if not
53 D4 0381 1052 CLRL R3 ; flag read required
```



```
FC7A' 30 0383 1053 BSBW RM$NXTBLK1 ; yes - go read another block
4C 50 E9 0386 1054 BLBC R0,50$ ; get out on error
55 57 51 C3 0389 1055 10$: SUBL3 R1,R7,R5 ; compute # bytes left this block
54 20 A9 D0 038D 1056 MOVL IRB$L_CURBDB(R9),R4 ; get current bdb address
50 49 A4 9A 0391 1057 MOVZBL BDB$B_VAL_VBNS(R4),R0 ; total # valid vbns in buffer
50 48 A4 82 0395 1058 SUBB2 BDB$B_REL_VBN(R4),R0 ; # blocks to end of buffer
50 50 D7 0399 1059 DECL R0 ; adjust for current block
50 48 AA C4 039B 1060 MULL2 IFB$L_DEVBUSIZ(R10),R0 ; # bytes after current block
55 50 C0 039F 1061 ADDL2 R0,R5 ; total # valid bytes left
55 56 DD 03A2 1062 PUSHL R6 ; save actual record length
55 56 B1 03A4 1063 CMPW R6,R5 ; is entire record within buff?
56 23 1B 03A7 1064 BLEQU 40$ ; no need to re-position if so
56 55 D0 03A9 1065 MOVL R5,R6 ; try to use amount present
55 00' B1 03AC 1066 CMPW S^#PIOSS_EODSTR,R5 ; enough chars to check?
55 1B 1B 03AF 1067 BLEQU 40$ ; yes, go for it
40 A9 D1 03B1 1068 CMPL IRB$L_NRP_VBN(R9),- ; at or past eof block?
74 AA 03B4 1069 IFB$L_EBK(R10) ; use avail space if so
5E 14 1E 03B6 1070 BGEQU 40$ ; since there isn't anything to remap.
5E 04 C0 03B8 1071 ADDL2 #4,SP ; discard saved reclen
FC42' 30 03BB 1072 BSBW RM$RELBLK1 ; release buffer
14 50 E9 03BE 1073 BLBC R0,50$ ; get out on error
40 A9 48 A9 7D 03C1 1074 MOVQ IRB$L_RP_VBN(R9),IRB$L_NRP_VBN(R9); re-read this record
20 A9 D4 03C6 1075 CLRL IRB$L_CURBDB(R9) ; make sure this bdb not reused
FDB8 31 03C9 1076 BRW RM$GET_BLK_DEV ; start over, re-mapping the buffer
FC31' 30 03CC 1077 BSBW RM$INPUT_SCAN ; compare string for
03CF 1078 40$: ; match with eod string
03CF 1079 ; or '$deck' or '$eod'
56 8E D0 03CF 1081 MOVL (SP)+,R6 ; restore reclen
05 50 E8 03D2 1082 BLBS R0,RETREC1 ; continue on success
05 03D5 1083 50$: RSB
03D6 1084
03D6 1085 ;++
03D6 1086 ;
03D6 1087 ; entry point for retrec subroutine
03D6 1088 ;
03D6 1089 ;--
03D6 1090
03D6 1091 RETREC:
9A 6A 2E E0 03D6 1092 BBS #IFB$V_PPF_INPUT,(R10),CHKEOD; branch if 'sys$input'
03DA 1093 RETREC1:
62 A9 56 A0 03DA 1094 ADDW2 R6,IRB$W_CSIZ(R9) ; add into total record size
21 68 30 E1 03DE 1095 BBC #RAB$V_LOC+ROP,(R8),MOVEMODE; branch if move mode
03E2 1096
03E2 1097 ;
03E2 1098 ; user wants locate mode
03E2 1099 ; -allow only if record does not cross block boundary
03E2 1100 ; and file is not update accessed
03E2 1101 ; and not indirect access to ppf
03E2 1102 ;
03E2 1103
1D 69 22 E0 03E2 1104 BBS #IRB$V_PPF_IMAGE,(R9),MOVEMODE; force move on ind
03E6 1105 ; access of ppf
03E6 1106
18 22 AA 03 E0 03E6 1107 BBS #IFB$V_UPD,IFB$B_FAC(R10),MOVEMODE; if update access, move mode
50 57 51 C3 03EB 1108 SUBL3 R1,R7,R0 ; get end address of buffer
50 56 D1 03EF 1109 CMPL R6,R0 ; past end of block?
```



```
OF 1A 03F2 1110          BGTRU  MOVEMODE          ; if so, must move record
      03F4 1111
      03F4 1112 :
      03F4 1113 : o.k. for locate mode
      03F4 1114 :
      03F4 1115
22 A8 56 B0 03F4 1116          MOVW  R6,RAB$W_RSZ(R8)      ; set record size
28 A8 51 D0 03F8 1117          MOVL  R1,RAB$L_RBF(R8)      ; and record address
      03FC 1118 SUCXIT: RMSSUC
51 56 C0 03FF 1119 UPDSRC: ADDL2  R6,R1          ; adjust r1 to point past
      0402 1120          ; record in buffer
      05 0402 1121          RSB
```

```
0403 1123
0403 1124 :++
0403 1125 :
0403 1126 : move mode - move record to user's buffer
0403 1127 :
0403 1128 :--
0403 1129 :
0403 1130 MOVEMODE:
3C 69 29 E0 0403 1131 BBS #IRB$V FIND,(R9),SKPREC ; branch if find, skipping record
53 24 A8 D0 0407 1132 MOVL RAB$L_UBF(R8),R3 ; get user buffer addr
54 20 A8 3C 040B 1133 MOVZWL RAB$W_USZ(R8),R4 ; and length
52 24 13 040F 1134 BEQL ERRUSZ ; branch if zero
52 56 D0 0411 1135 MOVL R6,R2 ; copy record size
52 54 D1 0414 1136 CMPL R4,R2 ; user buffer long enough?
07 1E 0417 1137 BGEQU 10$ ; branch if so
0419 1138
0419 1139 :
0419 1140 : user buffer not long enough
0419 1141 : readjust count to fill it and advise of difference
0419 1142 : (error status code generated later)
0419 1143 :
0419 1144 :
0C A8 52 3C 0419 1145 MOVZWL R2,RAB$L_STV(R8) ; indicate total record size
52 54 D0 041D 1146 MOVL R4,R2 ; just use user buffer size
0420 1147
0420 1148 :
0420 1149 : store the record's address and size and move the data
0420 1150 :
0420 1151 :
22 A8 52 B0 0420 1152 10$: MOVW R2,RAB$W_RSZ(R8) ; set rsz
28 A8 53 D0 0424 1153 MOVL R3,RAB$L_RBF(R8) ; set rbf from ubf addr
56 52 C2 0428 1154 SUBL2 R2,R6 ; compute remaining byte count
38 10 042B 1155 BSBB GTMVR ; move the record
04 50 E9 042D 1156 BLBC R0,100$ ; get out on error
56 D5 0430 1157 TSTL R6 ; part of record yet to be moved?
0F 12 0432 1158 BNEQ SKPREC ; branch if yes
05 0434 1159 100$: RSB ; return to retrec caller
0435 1160
```



```
0435 1162
0435 1163 :++
0435 1164 :
0435 1165 : user buffer errors
0435 1166 :
0435 1167 :--
0435 1168
0435 1169 ERRUSZ:
05 043A 1170 RMSERR USZ ; user buffer size = 0
043B 1171 RSB
043B 1172
043B 1173 :
043B 1174 : user buffer not writeable from caller's mode.
043B 1175 :
043B 1176
18 BA 043B 1177 ERRUBF: POPR #^M<R3,R4> ; clean up stack
043D 1178 ERRUBF1:
043D 1179 RMSERR UBF ; no access to user buffer
05 0442 1180 RSB
0443 1181
0443 1182 :++
0443 1183 :
0443 1184 : must skip over that part of record not moved
0443 1185 : due to short user record buffer or doing find
0443 1186 :
0443 1187 :--
0443 1188
50 57 51 C3 0443 1189 SKPREC: SUBL3 R1,R7,R0 ; #bytes left=end-start
50 50 56 D1 0447 1190 CMPL R6,R0 ; rec len > # left?
56 0A 1B 044A 1191 BLEQU 60$ ; branch if not
56 50 C2 044C 1192 SUBL2 R0,R6 ; compute remaining recordsize
044F 1193 ; =old size - part in buffer
0073 30 044F 1194 BSBW NXTBLK ; get another buffer full
EE 50 E8 0452 1195 BLBS R0,SKPREC ; and loop if no error
05 0455 1196 RSB ; get out on error
0456 1197
0456 1198 :
0456 1199 : return record too big error
0456 1200 :
0456 1201
A2 69 29 E0 0456 1202 60$: BBS #IRB$V_FIND,(R9),SUCXIT ; branch if find
045A 1203 RMSERR RTB
9E 11 045F 1204 BRB UPDSRC ; rejoin main line
0461 1205
```

```
0461 1207 .SBTTL GTMVRC - SUBROUTINE TO MOVE A RECORD
0461 1208
0461 1209 :++
0461 1210 : GTMVRC
0461 1211 :
0461 1212 : this subroutine moves a record from rms i/o buffer
0461 1213 : to user's record buffer, crossing block boundaries
0461 1214 : as needed.
0461 1215 :
0461 1216 : Calling sequence:
0461 1217 :
0461 1218 : bsbw gtmvrc
0461 1219 :
0461 1220 : Input Parameters:
0461 1221 :
0461 1222 : r7 end addr of block buffer
0461 1223 : r3 destination addr
0461 1224 : r2 size of record in bytes
0461 1225 : r1 source addr (in block buffer)
0461 1226 :
0461 1227 : Implicit Inputs:
0461 1228 :
0461 1229 : none
0461 1230 :
0461 1231 : outputs:
0461 1232 :
0461 1233 : r1 address of next record in block buffer
0461 1234 : r0 status code
0461 1235 : r2-r5,ap destroyed
0461 1236 :
0461 1237 : Implicit Outputs:
0461 1238 :
0461 1239 : the next block will be in the i/o buffer if
0461 1240 : the record crossed the block boundary
0461 1241 :
0461 1242 : Side Effects:
0461 1243 :
0461 1244 : may have switched to running at ast level.
0461 1245 :--
0461 1246 :
0461 1247 : .ALIGN QUAD ; align for performance
0468 1248 GTMVRC:
0468 1249 TSTL R2 ; get record size
046A 1250 BEQL 25$ ; branch on zero size
046C 1251 SUBL3 R1,R7,R0 ; get # bytes left in block
0470 1252 BEQL 40$ ; read another block if none left
0472 1253 CMPL R0,R2 ; less than move count?
0475 1254 BLSSU 10$ ; branch if yes (use # left)
0477 1255 MOVL R2,R0 ; no - use the move count
047A 1256 BEQL 25$ ; branch if zero length record
0200 8F 50 B1 047C 1257 10$: CMPW R0,#512 ; buffer require long probe?
18 1A 0481 1258 BGTRU 30$ ; branch if yes
0483 1259 20$: IFNOWRT R0,(R3),ERRUBF1,- ; probe this part of buffer,
0483 1260 IRB$B_MODE(R9) ; branching if bad
5C 52 50 C3 048A 1261 SUBL3 R0,R2-AP ; adjust remaining count
63 61 50 28 048E 1262 MOVC3 R0,(R1),(R3) ; move the record
52 5C D0 0492 1263 MOVL AP,R2 ; any remaining bytes?
```



```
24 12 0495 1264 BNEQ 40$ ; branch if yes
      0497 1265 25$: RMSSUC
      05 049A 1266 RSB
      049B 1267
      049B 1268 ;
      049B 1269 ; long probe for devices with a default buffer size greater than 512
      049B 1270 ;
      049B 1271
54 09 BB 049B 1272 30$: PUSH R0,R3 ; save byte count and address
FE00 8F 32 049D 1273 CVTWL #-512,R4 ; address computation constant
53 54 C2 04A2 1274 35$: IFNOWRT R0,(R3),ERRUBF,IRB$B_MODE(R9); branch if not writeable
50 6044 3E 04A9 1275 SUBL2 R4,R3 ; get address next page
      F0 14 04AC 1276 MOVAW (R0)[R4],R0 ; adjust count (- 2 pages)
50 54 C2 04B0 1277 BGTR 35$ ; branch if more to probe
      EB 14 04B2 1278 SUBL2 R4,R0 ; 'add' back in 1 page
      09 BA 04B5 1279 BGTR 35$ ; branch if more to probe
      C8 11 04B7 1280 POPR #^M<R0,R3> ; restore byte count and address
      04B9 1281 BRB 20$ ; rejoin main line
      04BB 1282
      04BB 1283 ;++
      04BB 1284 ;
      04BB 1285 ; move next part of record
      04BB 1286 ;
      04BB 1287 ;--
      04BB 1288
      0C BB 04BB 1289 40$: PUSH R2,R3 ; save remaining byte count
      04BD 1290 ; and destination address
      06 10 04BD 1291 BSBB NXTBLK ; read next block
      0C BA 04BF 1292 POPR #^M<R2,R3> ; restore remaining count and addr
A4 50 E8 04C1 1293 BLBS R0,GTMVRC ; and go again
      05 04C4 1295 RSB
```

```
04C5 1297
04C5 1298 :++
04C5 1299 :nxtblk subroutine - change buffer/blocks
04C5 1300 :
04C5 1301 :
04C5 1302 :inputs:
04C5 1303 :    r11    impure area addr
04C5 1304 :    r10    ifab addr
04C5 1305 :    r9     irab addr
04C5 1306 :    r8     rab addr
04C5 1307 :
04C5 1308 :Implicit Inputs:
04C5 1309 :
04C5 1310 :    irb$l_curbdb
04C5 1311 :    irb$l_nrp_vbn
04C5 1312 :    irb$v_find
04C5 1313 :
04C5 1314 :outputs:
04C5 1315 :
04C5 1316 :if last operation was a find,then:
04C5 1317 :    r0      success code
04C5 1318 :    irb$l_nrp_vbn = irb$l_nrp_vbn+1
04C5 1319 :    irb$w_nrp_off = 0
04C5 1320 :
04C5 1321 :if last operation was not a find, then:
04C5 1322 :
04C5 1323 :    r7      end of buffer addr+1
04C5 1324 :    r4      bdb addr of new block
04C5 1325 :    r1      block buffer addr of new block
04C5 1326 :    r0      status code
04C5 1327 :    r2-r3   destroyed
04C5 1328 :
04C5 1329 :--
04C5 1330 :
04 6A 2E E0 04C5 1331 NXTBLK: BBS    #IFB$V_PPF_INPUT,(R10),10$ ; branch if this is sys$input
05 69 29 E0 04C9 1332 BBS    #IRB$V_FIND,(R9),20$ ; branch if find
      53 D4 04CD 1333 10$: CLRL   R3 ; flag read required
      FB2E' 31 04CF 1334 BRW    RM$NXTBLK1 ; go to next block routine
04D2 1335 :
04D2 1336 :
04D2 1337 :this is a find call for disk so just bump nrp data and reset buffer pointer
04D2 1338 :
04D2 1339 :
04D2 1340 20$: INCL   IRB$L_NRP_VBN (R9)
      40 A9 D6 04D5 1341 CLRW   IRB$W_NRP_OFF(R9)
      44 A9 B4 04D8 1342 MOVAL  -512(R7),R1 ; reset start of buffer addr
51 FE00 C7 DE 04DD 1343 RMSSUC
      05 04E0 1344 RSB
04E1 1345
04E1 1346
04E1 1347 .END
```


RM1GETINT
Symbol table

INTERNAL GET SEQUENTIAL

L 14

16-SEP-1984 00:48:17 VAX/VMS Macro V04-00
5-SEP-1984 16:23:21 [RMS.SRC]RM1GETINT.MAR;1

Page 30
(20)

\$\$PSECT EP	= 00000000			IRBSL_CURBDB	= 00000020		
\$\$RMSTEST	= 0000001A			IRBSL_IOS	= 0000000C		
\$\$RMS_PBUGCHK	= 00000010			IRBSL_IOS4	= 00000010		
\$\$RMS_TBUGCHK	= 00000008			IRBSL_NRP	= 00000040		
\$\$RMS_UMODE	= 00000004			IRBSL_NRP_OFF	= 00000044		
BDB\$B_REL_VBN	= 00000048			IRBSL_NRP_VBN	= 00000040		
BDB\$B_VAL_VBNS	= 00000049			IRBSL_RP_VBN	= 00000048		
BDB\$B_ADDR	= 00000018			IRBSV_EOF	= 00000021		
BDB\$B_CURBUFADR	= 0000004C			IRBSV_FIND	= 00000029		
BDB\$W_NUMB	= 00000014			IRBSV_FIND_LAST	= 00000025		
CHKEOD	00000374	R	01	IRBSV_PPF_IMAGE	= 00000022		
CHKEOF1	00000163	R	01	IRBSW_CSIZ	= 00000062		
CR	= 0000000D			IRBSW_IOS2	= 0000000E		
CTRLZ	= 0000001A			IRBSW_NRP_OFF	= 00000044		
DEV\$V_FOR	= 00000018			LF	= 0000000A		
DEV\$V_REC	= 00000000			MOVEDATA	00000069	R	01
DEV\$V_SQD	= 00000005			MOVEMODE	00000403	R	01
DEV\$V_TRM	= 00000002			NEXT	0000001E	R	01
ERRIRC	0000025F	R	01	NTVFC	0000012D	R	01
ERRIRC1	000002FA	R	01	NXTBLK	000004C5	R	01
ERRRHB	00000367	R	01	PIOSA_TRACE	*****	X	01
ERRUBF	0000043B	R	01	PIOSGT_ENDSTR	*****	X	01
ERRUBF1	0000043D	R	01	PIOSGW_STATUS	*****	X	01
ERRUSZ	00000435	R	01	PIOSS_EODSTR	*****	X	01
ERR CONT	0000006E	R	01	PIOSV_EOD	= 00000001		
FAB\$C_FIX	= 00000001			RAB\$B_RBF	= 00000028		
FAB\$C_STM	= 00000004			RAB\$B_RHB	= 0000002C		
FAB\$C_STMCR	= 00000006			RAB\$B_RBP	= 00000004		
FAB\$C_STMLF	= 00000005			RAB\$B_STV	= 0000000C		
FAB\$C_UDF	= 00000000			RAB\$B_UBF	= 00000024		
FAB\$C_VAR	= 00000002			RAB\$V_ETO	= 0000001C		
FAB\$C_VFC	= 00000003			RAB\$V_LOC	= 00000010		
FAB\$V_BLK	= 00000003			RAB\$W_RFA	= 00000010		
FF	= 0000000C			RAB\$W_RSZ	= 00000022		
FIXSIZ	0000022B	R	01	RAB\$W_USZ	= 00000020		
GET00	0000017E	R	01	RETREC	000003D6	R	01
GET01	000002BF	R	01	RETREC1	000003DA	R	01
GETNRP	00000193	R	01	RMSGETBLKNRP	*****	X	01
GET_ANSI_D	000002C2	R	01	RMSGETRANDOM	00000186	RG	01
GET_CHKMINUS	00000224	R	01	RMSGET_BLK_DEV	00000184	RG	01
GET_RETREC	000001E9	R	01	RMSGET_EOF	0000009B	RG	01
GTMVRC	00000468	R	01	RMSGET_STM_FMT	*****	X	01
IFB\$B_FAC	= 00000022			RMSGET_UNIT_REC	00000000	RG	01
IFB\$B_FSZ	= 0000005F			RMSINPUT_SCAN	*****	X	01
IFB\$B_RAT	= 00000051			RMSNXTBLK1	*****	X	01
IFB\$B_RFMORG	= 00000050			RMSRELBLK1	*****	X	01
IFB\$B_DEVBUFSIZ	= 00000048			RMSSEQRD	*****	X	01
IFB\$B_EBK	= 00000074			RMS\$ANI	= 0001840C		
IFB\$B_PRIM_DEV	= 00000000			RMS\$BES	= 000181C0		
IFB\$V_ANSI_D	= 00000026			RMS\$EOF	= 0001827A		
IFB\$V_DAP	= 0000003E			RMS\$IRC	= 0001857C		
IFB\$V_NSP	= 0000003F			RMS\$PES	= 000181C8		
IFB\$V_PPF_INPUT	= 0000002E			RMS\$RER	= 0001C0F4		
IFB\$V_UPD	= 00000003			RMS\$RHB	= 0001866C		
IFB\$W_FFB	= 0000005C			RMS\$RTB	= 000181A8		
IFB\$W_LRL	= 00000052			RMS\$TNS	= 000181B8		
IRB\$B_MODE	= 0000000A			RMS\$UBF	= 000186EC		

RM1GETINT
Symbol table

INTERNAL GET SEQUENTIAL

M 14

16-SEP-1984 00:48:17 VAX/VMS Macro V04-00
5-SEP-1984 16:23:21 [RMS.SRC]RM1GETINT.MAR;1

Page 31
(20)

RMS\$_USZ	= 000186F4		
ROP	= 00000020		
SKPREC	00000443	R	01
SS\$_BADESCAPE	= 0000003C		
SS\$_PARTESCAPE	= 000001FC		
STMSIZ	000002A7	R	01
SUCXIT	000003FC	R	01
TPT\$\$_GETBLKD	*****	X	01
TRM	000000C1	R	01
UDFSIZ	0000026C	R	01
UPDSRC	000003FF	R	01
VARSIZ	000001C5	R	01
VFC\$IZ	000002FD	R	01
VT	= 0000000B		
ZERO_RFA	00000220	R	01
ZERO_RFA1	00000093	R	01
ZERO_RFA1_BR	0000012A	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					
RMSRMS1	000004E1 (1249.)	01 (1.)	PIC	USR	CON	REL	GBL	NOSHR	EXE	RD	NOWRT	NOVEC	QUAD					
\$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	29	00:00:00.07	00:00:01.24
Command processing	117	00:00:00.72	00:00:04.19
Pass 1	413	00:00:15.19	00:00:41.81
Symbol table sort	0	00:00:02.12	00:00:03.36
Pass 2	224	00:00:04.15	00:00:11.78
Symbol table output	16	00:00:00.16	00:00:00.42
Psect synopsis output	1	00:00:00.02	00:00:00.14
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	802	00:00:22.44	00:01:03.17

The working set limit was 1950 pages.
88550 bytes (173 pages) of virtual memory were used to buffer the intermediate code.
There were 80 pages of symbol table space allocated to hold 1477 non-local and 69 local symbols.
1347 source lines were read in Pass 1, producing 16 object records in Pass 2.
26 pages of virtual memory were used to define 25 macros.

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

Macro library name	Macros defined
-----	-----
\$255\$DUA28:[RMS.OBJ]RMS.MLB;1	14
\$255\$DUA28:[SYS.OBJ]LIB.MLB;1	1
\$255\$DUA28:[SYSLIB]STARLET.MLB;2	6
TOTALS (all libraries)	21

1575 GETS were required to define 21 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS\$:RM1GETINT/OBJ=OBJ\$:RM1GETINT MSRC\$:RM1GETINT/UPDATE=(ENH\$:RM1GETINT)+EXECML\$/LIB+LIB\$:RMS/LIB

0321 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

