


```

RRRRRRRR      MM      MM      000000      XX      XX      AAAAAA      BBBB88888
RRRRRRRR      MM      MM      000000      XX      XX      AAAAAA      BBBB88888
RR      RR      MMMM      MMMM      00      00      XX      XX      AA      AA      BB      BB
RR      RR      MMMM      MMMM      00      00      XX      XX      AA      AA      BB      BB
RR      RR      MM      MM      MM      00      0000      XX      XX      AA      AA      BB      BB
RR      RR      MM      MM      MM      00      0000      XX      XX      AA      AA      BB      BB
RRRRRRRR      MM      MM      00      00      00      XX      XX      AA      AA      BBBB88888
RRRRRRRR      MM      MM      00      00      00      XX      XX      AA      AA      BBBB88888
RR      RR      MM      MM      0000      00      XX      XX      AAAAAAAAAA      BB      BB
RR      RR      MM      MM      0000      00      XX      XX      AAAAAAAAAA      BB      BB
RR      RR      MM      MM      00      00      XX      XX      AA      AA      BB      BB
RR      RR      MM      MM      00      00      XX      XX      AA      AA      BB      BB
RR      RR      MM      MM      000000      XX      XX      AA      AA      BBBB88888
RR      RR      MM      MM      000000      XX      XX      AA      AA      BBBB88888

```

```

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
LLLLLLLLLLLL      IIIIII      SSSSSSSS
LLLLLLLLLLLL      IIIIII      SSSSSSSS

```

| | | |
|------|------|---|
| (3) | 193 | DECLARATIONS |
| (4) | 293 | RM\$OPEN XAB - PROCESS XABS FOR \$OPEN |
| (7) | 402 | XAB_SCAN - COMMON XAB DISPATCH ROUTINE |
| (8) | 549 | FHC-XAB PROCESSING ROUTINES |
| (9) | 664 | DAT XAB ROUTINE |
| (10) | 716 | RDT XAB ROUTINES |
| (11) | 769 | PRO XAB ROUTINES |
| (18) | 998 | ALL XAB ROUTINES |
| (21) | 1102 | RM\$SETALLOC - ROUTINE TO HANDLE ALLOC XAB FOR SEQ. & REL. F.O. |
| (22) | 1190 | JNL XAB ROUTINES |
| (23) | 1354 | RM\$SETEXTRMS - ROUTINE TO HANDLE CONTEXT XAB FOR FAB & RAB |
| (24) | 1565 | RM\$OPNCREXAB - ROUTINE TO RESTART OPEN OR CREATE |
| (25) | 1616 | RM\$CONNXAB - ROUTINE TO RESTART CONNECT |
| (27) | 1764 | Terminal XAB Routines |

```
0000 1          $BEGIN RMOXAB,000,RMSRMS0,<XAB PROCESSING ROUTINES>
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 contains the routines to process xabs
0000 33 :         for open, display, create and modify.
0000 34 :
0000 35 : Environment:
0000 36 :         star processor running starlet exec.
0000 37 :
0000 38 : Author: l f laverdure ,creation date: 27-SEP-1977
0000 39 :
0000 40 : Modified By:
0000 41 :
0000 42 :     V03-040 RAS0291      Ron Schaefer      11-Apr-1984
0000 43 :         Clear XAB$ACLSTS before the ACP operation.
0000 44 :         We really ought to return an alternate status but
0000 45 :         it's impossibly awkward to bubble this obscure failure
0000 46 :         up the chain to turn it into an alternate success/warning.
0000 47 :
0000 48 :     V03-039 RAS0284      Ron Schaefer      29-Mar-1984
0000 49 :         Fix error path and STV value for XABALL processing.
0000 50 :
0000 51 :     V03-038 DGB0029      Donald G. Blair  16-Mar-1984
0000 52 :         Implement ACLs and the PROPAGATE bit -- all in the
0000 53 :         protection XAB.
0000 54 :
0000 55 :     V03-037 DBG0004      Donald G. Blair  23-Feb-1984
0000 56 :         Implement access mode protected files.
0000 57 :
0000 58 :     V03-036 DAS0004      David Solomon    03-Feb-1984
0000 59 :         Tie off journaling: disable processing of $XABJNL on $CREATE.
0000 60 :         Fix incorrect register usage in probe of user AT journal name.
0000 61 :         Fix minor bug in $XABCXR restart code.
0000 62 :
0000 63 :     V03-035 JWT0148      Jim Teague      16-Dec-1983
0000 64 :         Make sure user can't $CREATE file with conflicting
0000 65 :         RU attributes in the JNL XAB.
0000 66 :
0000 67 :     V03-034 JWT0141      Jim Teague      11-Nov-1983
0000 68 :         Change IFB$V_RUM to IFB$V_ONLY_RU
0000 69 :
0000 70 :     V03-033 LMP0133      L. Mark Pilant,  3-Aug-1983  14:58
0000 71 :         Get default protection from PCB instead of PI space. Also
0000 72 :         don't supply the protection attribute unless explicitly
0000 73 :         given in a PROtection XAB.
0000 74 :
0000 75 :     V03-032 RAS0177      Ron Schaefer      29-Jul-1983
0000 76 :         Improve RMS exit performance by simplifying the
0000 77 :         non-context extraction path thru RM$SETEXTRMS.
0000 78 :
0000 79 :     V03-031 DAS0003      David Solomon    29-Jul-1983
0000 80 :         XABJNL JOP bit RUA is now called ONLY_RU; add ASSUMEs.
0000 81 :
0000 82 :     V03-030 KBT0546      Keith B. Thompson 22-Jun-1983
0000 83 :         Stuff the XAB$W_VERLIMIT in the FHCXAB
0000 84 :
```

| | | | | | | |
|------|-----|---|---------|---------|--|-------------|
| 0000 | 85 | : | V03-029 | KPL0001 | Peter Lieberwirth | 17-May-1983 |
| 0000 | 86 | : | | | Fix XABCLSPRO so IFB isn't overwritten when a spurious | |
| 0000 | 87 | : | | | attempt to update the FWA was made. The fix is to | |
| 0000 | 88 | : | | | avoid altogether updating the FWA on CLOSE. | |
| 0000 | 89 | : | | | | |
| 0000 | 90 | : | V03-028 | RAS0153 | Ron Schaefer | 2-May-1983 |
| 0000 | 91 | : | | | Delete reference to \$XABACEDEF missed by RAS0148. | |
| 0000 | 92 | : | | | | |
| 0000 | 93 | : | V03-027 | RAS0148 | Ron Schaefer | 26-Apr-1983 |
| 0000 | 94 | : | | | Add initial support for extended XABPRO and eliminate | |
| 0000 | 95 | : | | | support for XABACE. | |
| 0000 | 96 | : | | | | |
| 0000 | 97 | : | V03-026 | RAS0138 | Ron Schaefer | 22-Mar-1983 |
| 0000 | 98 | : | | | Delete global symbol/data XCONN03_ARGS. | |
| 0000 | 99 | : | | | | |
| 0000 | 100 | : | V03-025 | JWH0190 | Jeffrey W. Horn | 14-Mar-1983 |
| 0000 | 101 | : | | | Add XABACE support. | |
| 0000 | 102 | : | | | | |
| 0000 | 103 | : | V03-024 | MCN0010 | Maria del C. Nasr | 08-Mar-1983 |
| 0000 | 104 | : | | | I forgot to include \$BKTDEF for MCN0009. | |
| 0000 | 105 | : | | | | |
| 0000 | 106 | : | V03-023 | MCN0009 | Maria del C. Nasr | 07-Mar-1983 |
| 0000 | 107 | : | | | Use symbolic name for maximum bucket size. | |
| 0000 | 108 | : | | | | |
| 0000 | 109 | : | V03-022 | LJA0068 | Laurie J. Anderson | 01-Mar-1983 |
| 0000 | 110 | : | | | Check new field in XABCXR, CXRBFZ (the number of bytes | |
| 0000 | 111 | : | | | in length of the user provided buffer CXRBUF) to make | |
| 0000 | 112 | : | | | sure that it is at least XAB\$C_CXRBLN (the length which | |
| 0000 | 113 | : | | | is necessary for storing two keys). | |
| 0000 | 114 | : | | | | |
| 0000 | 115 | : | V03-021 | JWH0189 | Jeffrey W. Horn | 15-Feb-1983 |
| 0000 | 116 | : | | | Fill in UIC and Protection FWA fields when processing | |
| 0000 | 117 | : | | | the protection XAB. | |
| 0000 | 118 | : | | | | |
| 0000 | 119 | : | V03-020 | DAS0002 | David Solomon | 09-Feb-1983 |
| 0000 | 120 | : | | | Add XABTRM support. | |
| 0000 | 121 | : | | | | |
| 0000 | 122 | : | V03-019 | LJA0057 | Laurie J. Anderson | 25-Jan-1983 |
| 0000 | 123 | : | | | Fix a couple more things | |
| 0000 | 124 | : | | | | |
| 0000 | 125 | : | V03-018 | LJA0056 | Laurie J. Anderson | 24-Jan-1983 |
| 0000 | 126 | : | | | Fix a couple things with RMS context extraction and in LJA0048 | |
| 0000 | 127 | : | | | | |
| 0000 | 128 | : | V03-017 | LJA0048 | Laurie J. Anderson | 4-Jan-1983 |
| 0000 | 129 | : | | | - Add NRP for SEQ/REL | |
| 0000 | 130 | : | | | - Fix setting of LRL in IFAB. If this is NOT a create, then | |
| 0000 | 131 | : | | | do NOT copy the LRL value from the FHC XAB into the IFAB. | |
| 0000 | 132 | : | | | | |
| 0000 | 133 | : | V03-016 | JWH0162 | Jeffrey W. Horn | 21-Dec-1982 |
| 0000 | 134 | : | | | Place journal names in separate ACEs. | |
| 0000 | 135 | : | | | | |
| 0000 | 136 | : | V03-015 | MCN0008 | Maria del C. Nasr | 08-Dec-1982 |
| 0000 | 137 | : | | | - Return area id number (AID) in STV instead of XAB | |
| 0000 | 138 | : | | | address for AID and BKZ errors. | |
| 0000 | 139 | : | | | - Maximum bucket size was increased to 127. | |
| 0000 | 140 | : | | | - In XABCREALL1, make sure we only process area 0, and | |
| 0000 | 141 | : | | | not the first one since the XABs can come in any order. | |

| | | | | | |
|------|-----|-----|---------|--|--|
| 0000 | 142 | : | | | |
| 0000 | 143 | : | V03-014 | RAS0108 | Ron Schaefer 15-Dec-1982 |
| 0000 | 144 | : | | Change XAB\$B_JOP to XAB\$W_JOP and | |
| 0000 | 145 | : | | fix bad branch for ISAM context restart. | |
| 0000 | 146 | : | | | |
| 0000 | 147 | : | V03-013 | ACG0306 | Andrew C. Goldstein, 13-Dec-1982 14:58 |
| 0000 | 148 | : | | Remove obsolete file header symbols | |
| 0000 | 149 | : | | | |
| 0000 | 150 | : | V03-012 | JWH0140 | Jeffrey W. Horn 30-Nov-1982 |
| 0000 | 151 | : | | Add return length fields for journal XAB. | |
| 0000 | 152 | : | | | |
| 0000 | 153 | : | V03-011 | LJA0037 | Laurie J. Anderson 4-Nov-1982 |
| 0000 | 154 | : | | Add NRP processing for ISAM files. | |
| 0000 | 155 | : | | | |
| 0000 | 156 | : | V03-010 | LJA0027 | Laurie J. Anderson 26-Oct-1982 |
| 0000 | 157 | : | | Delete fields no longer in use. | |
| 0000 | 158 | : | | | |
| 0000 | 159 | : | V03-009 | JWH0112 | Jeffrey W. Horn 12-Oct-1982 |
| 0000 | 160 | : | | Change journal names to be .ASCII strings. | |
| 0000 | 161 | : | | | |
| 0000 | 162 | : | V03-008 | LJA0023 | Laurie J. Anderson 5-Oct-1982 |
| 0000 | 163 | : | | Check for Uniqueness of Context XAB's on Restart. | |
| 0000 | 164 | : | | Error if duplicates. | |
| 0000 | 165 | : | | | |
| 0000 | 166 | : | V03-007 | LJA0022 | Laurie J. Anderson 28-Sep-1982 |
| 0000 | 167 | : | | Use yet another register, R1, instead of R2 in xab_scan. | |
| 0000 | 168 | : | | | |
| 0000 | 169 | : | V03-006 | KBT0343 | Keith B. Thompson 23-Sep-1982 |
| 0000 | 170 | : | | Use R2 instead of R6 in xab_scan (fix krm0061) | |
| 0000 | 171 | : | | | |
| 0000 | 172 | : | V03-005 | KRM0061 | Karl Malik 22-Sep-1982 |
| 0000 | 173 | : | | Modify RMSXAB_SCAN to save R7. | |
| 0000 | 174 | : | | | |
| 0000 | 175 | : | V03-004 | LJA0012 | Laurie Anderson 03-Sep-1982 |
| 0000 | 176 | : | | Add Context XAB Processing for Exiting RMS Services. | |
| 0000 | 177 | : | | Add Create and Open Restart processing of XABCXF. | |
| 0000 | 178 | : | | Add Connect Restart processing of XABCXR. | |
| 0000 | 179 | : | | | |
| 0000 | 180 | : | V03-003 | KBT0302 | Keith B. Thompson 28-Aug-1982 |
| 0000 | 181 | : | | Reorganize psects | |
| 0000 | 182 | : | | | |
| 0000 | 183 | : | V03-002 | JWH0002 | Jeffrey W. Horn 02-Jun-1982 |
| 0000 | 184 | : | | Add Journal XAB processing. | |
| 0000 | 185 | : | | | |
| 0000 | 186 | : | V03-001 | RAS0084 | Ron Schaefer 2-Apr-1982 |
| 0000 | 187 | : | | Return RAT=CR for stream format files even if | |
| 0000 | 188 | : | | the file attribute is none. | |
| 0000 | 189 | : | | | |
| 0000 | 190 | :-- | | | |
| 0000 | 191 | : | | | |

```

0000 193      .SBTTL  DECLARATIONS
0000 194
0000 195      :
0000 196      : Include Files:
0000 197      :
0000 198      :
0000 199      :
0000 200      : Macros:
0000 201      :
0000 202      :
0000 203      $ACEDEF
0000 204      $ATRDEF
0000 205      $BKDEF
0000 206      $FABDEF
0000 207      $RABDEF
0000 208      $FCHDEF
0000 209      $FIBDEF
0000 210      $FWADEF
0000 211      $IFBDEF
0000 212      $IRBDEF
0000 213      $RMSDEF
0000 214      $XABDEF
0000 215      $XABALLDEF
0000 216      $XABCXFDEF
0000 217      $XABCXRDEF
0000 218      $XABDATDEF
0000 219      $XABFHCDEF
0000 220      $XABJNLDEF
0000 221      $XABKEYDEF
0000 222      $XABPRODEF
0000 223      $XABRDTDEF
0000 224      $XABSUMDEF
0000 225      $XABTRMDEF
0000 226      $PCBDEF
0000 227
0000 228      :
0000 229      : Equated Symbols:
0000 230      :
0000 231      :
00000020 0000 232      FOP=FAB$L_FOP*8          ; bit offset to fop field
0000 233
0000 234      :
0000 235      : define case index values for xab dispatching
0000 236      : (note: these must correspond to the offsets in the xab case dispatch)
0000 237      :
0000 238      :
00000000 0000 239      XBC$C_OPNFHC      == 0          ; fhc xab handler for open
00000001 0000 240      XBC$C_OPNFHC1    == 1          ; fhc xab handler for open (part 2)
00000002 0000 241      XBC$C_OPNPRO    == 2          ; pro xab handler for open/display
00000003 0000 242      XBC$C_OPNDAT    == 3          ; dat xab handler for open
00000004 0000 243      XBC$C_CREALLO   == 4          ; all xab handler for create
0000 244      : (for seq. and rel. f.o.)
00000005 0000 245      XBC$C_EXTALL    == 5          ; all xab handler for extend
00000006 0000 246      XBC$C_CREALL1    == 6          ; all xab handler for create (part 2)
00000007 0000 247      XBC$C_CREPRO    == 7          ; pro xab handler for create
00000008 0000 248      XBC$C_OPNRDT    == 8          ; rdt xab handler for open
00000009 0000 249      XBC$C_OPNALL    == 9          ; all xab handler for open

```



```

0000000A 0000 250 XBC$C_CLSPRO == 10 ; pro xab handler for close
0000000B 0000 251 XBC$C_CLSRDT == 11 ; rdt xab handler for close
0000000C 0000 252 XBC$C_DSPFHC == 12 ; fhc xab handler for display
0000000D 0000 253 XBC$C_DSPFHC1 == 13 ; fhc xab handler for display (part 2)
0000000E 0000 254 XBC$C_DSPALL == 14 ; all xab handier for display
0000000F 0000 255 XBC$C_DSPALL1 == 15 ; all xab handler for display (part 2)
00000010 0000 256 XBC$C_CREALL2 == 16 ; all xab handler for create (part 3)
00000011 0000 257 XBC$C_OPNSUM3 == 17 ; summary xab handler / open and display
00000012 0000 258 XBC$C_OPNALL3 == 18 ; alloc xab handler / open & display
00000013 0000 259 XBC$C_OPNKEY3 == 19 ; key xab handler for open and display
00000014 0000 260 XBC$C_OPNJNL == 20 ; journal xab handler for open
00000015 0000 261 XBC$C_CREJNL == 21 ; journal xab handler for create
00000016 0000 262 XBC$C_EXTCXF == 22 ; CXR Context XAB Handler for RMS Exit
00000017 0000 263 XBC$C_EXTCXR == 23 ; CXR Context XAB Handler for RMS Exit
00000018 0000 264 XBC$C_OPNCXF == 24 ; Open and create Restart Handler
00000019 0000 265 XBC$C_CONNCXR == 25 ; Connect Restart Handler
C000001A 0000 266 XBC$C_XCONN03 == 26 ; ISAM NRP processing routine for XABCXR
0000001B 0000 267 XBC$C_GETPUTTRM == 27 ; TRM XAB handler for get/put
0000001C 0000 268 XBC$C_OPNPRO1 == 28 ; pro xab handler for open/disp (part 2)
0000001D 0000 269 XBC$C_ENTPRO == 29 ; protecton xab handler for enter/rename
0000001E 0000 270 XBC$C_CREPRO1 == 30 ; pr xab handler for create (part 2)
0000 271 ; Watch out -- these xbc$ constants cannot be larger than 31!!
0000 272 ;
0000 273 ;
0000 274 ; Own Storage:
0000 275 ;
0000 276 ;
0000 277 ; table with all valid FAB xab codes for validation
C000 278 ;
0000 279 ;
20 22 1F 16 1E 13 15 1D 12 14 0000 280 XABTBL: .BYTE XAB$C_ALL,XAB$C_DAT,XAB$C_FHC,XAB$C_KEY,XAB$C_PRO,-
000A 281 XAB$C_RDT,XAB$C_SUM,XAB$C_TRM,XAB$C_JNL,XAB$C_CXF
0000000A 000A 282 XABTBLEN=-XABTBL
000A 283 ;
000A 284 ;
000A 285 ; Table with all valid RAB xab codes for validation
000A 286 ;
000A 287 ;
000A 288 RXABTBL:
1F 21 000A 289 .BYTE XAB$C_CXR, XAB$C_TRM
00000002 000C 290 RXABTBLEN=-RXABTBL
000C 291

```

```
000C 293 .SBTTL RMSOPEN_XAB - PROCESS XABS FOR $OPEN
000C 294
000C 295 :++
000C 296 :
000C 297 : RMSOPEN_XAB: Process XABs for $OPEN, first part
000C 298 : RMSOPEN_XAB1: Process XABs for $OPEN, second part
000C 299 :
000C 300 : these subroutines process the xabs for open. they are handled in
000C 301 : two parts, the first executing before doing the access qio,
000C 302 : the other after.
000C 303 :
000C 304 : this section of the module also includes the common xab chain
000C 305 : following and dispatch code.
000C 306 :
000C 307 : Calling sequence:
000C 308 :
000C 309 :     bsbw  rmsopen_xab      ; first part
000C 310 :     bsbw  rmsopen_xab1    ; second part
000C 311 :
000C 312 : Input Parameters: (For RMSXAB_SCAN, also)
000C 313 :
000C 314 :     r11   impure area address
000C 315 :     r10   file work page address or ifab (if r9 is irab)
000C 316 :     r9    ifab address or irab address (if rab xab)
000C 317 :     r8    fab address or rab address (if rab xab)
000C 318 :     r7    nwa address (if any)
000C 319 :     Or
000C 320 :     If called from EXTRMS, and there is not an IFB/IRB structure
000C 321 :     available, then (r7) has the user calling mode.
000C 322 :     r6    fib address (if any)
000C 323 :     r5    next attribute address (if applicable)
000C 324 :
000C 325 : Implicit Inputs:
000C 326 :
000C 327 :     fab$l_xab and the xabs within the chain thus defined
000C 328 :
000C 329 : Output Parameters:
000C 330 :
000C 331 :     attribute list entries are added such that the requested file
000C 332 :     attributes are filled in.
000C 333 :     r5    updated to point to the next available attribute address.
000C 334 :     For call to scan for XABTRM, R5 points to XABTRM.
000C 335 :     r4    zero if no xabs processed, otherwise non-zero
000C 336 :     (in general, bit corresponding to case handler index
000C 337 :     is set for each handler called)
000C 338 :     r1-r3,ap destroyed
000C 339 :     r0    status code
000C 340 :
000C 341 : Implicit Outputs:
000C 342 :
000C 343 :     the various fields of the xabs are filled in.
000C 344 :
000C 345 : Completion Codes:
000C 346 :
000C 347 :     standard rms, in particular suc, xab, cod, imx and various others.
000C 348 :
000C 349 : Side Effects:
```

RMOXAB
V04-000

D 13
XAB PROCESSING ROUTINES
RMSOPEN_XAB - PROCESS XABS FOR \$OPEN

16-SEP-1984 00:41:40 VAX/VMS Macro V04-00
5-SEP-1984 16:22:47 [RMS.SRC]RMOXAB.MAR;1

Page 8
(4)

RM
VO

```
000C 350 :  
000C 351 : none  
000C 352 :  
000C 353 :--  
000C 354
```

```

000C 356
000C 357
000C 358 : the following table provides the dispatching information for those xabs
000C 359 : that are input to $open and require an entry in the files attribute list.
000C 360 : the entries consist of the xab code, its length, and its case index.
000C 361 :
000C 362 :
000C 363 OPNXAB_ARGS:
00 2C 1D 000C 364 .BYTE XAB$C_FHC,XAB$C_FHCLEN,XBC$C_OPNFHC
02 10 13 000F 365 .BYTE XAB$C_PRO,XAB$C_PROLEN_V3,XBC$C_OPNPRO
03 24 12 0012 366 .BYTE XAB$C_DAT,XAB$C_DATLEN_V2,XBC$C_OPNDAT
08 14 1E 0015 367 .BYTE XAB$C_RDT,XAB$C_RDTLEN,XBC$C_OPNRDT
      00 0018 368 .BYTE 0 ; end of table flag
      0019 369
      0019 370 :
      0019 371 : the following table specifies those xabs to open that require processing
      0019 372 : after the access qio has been performed.
      0019 373 :
      0019 374 :
      0019 375 OPNXAB1_ARGS:
01 2C 1D 0019 376 .BYTE XAB$C_FHC,XAB$C_FHCLEN,XBC$C_OPNFHC1
09 20 14 001C 377 .BYTE XAB$C_ALL,XAB$C_ALLLEN,XBC$C_OPNALL
14 3C 22 001F 378 .BYTE XAB$C_JNL,XAB$C_JNLLEN,XBC$C_OPNJNL
1C 10 13 0022 379 .BYTE XAB$C_PRO,XAB$C_PROLEN_V3,XBC$C_OPNPRO1
      00 0025 380 .BYTE 0
      0026 381
      0026 382 :
      0026 383 : entry point for processing open xabs (part 2)
      0026 384 :
      0026 385 :
      5C FO AF 9E 0026 386 RMSOPEN_XAB1::
      04 11 0026 387 .MOVAB OPNXAB1_ARGS,AP ; set arg list addr
      002A 388 .BRB XAB_SCAN ; and go process
      002C 389
      002C 390 :
      002C 391 : entry point for processing open xabs (part 1)
      002C 392 :
      002C 393 :
      5C DD AF 9E 002C 394 RMSOPEN_XAB::
      0030 395 .MOVAB OPNXAB_ARGS,AP ; set arg list addr
      0030 396
      0030 397 :
      0030 398 : and fall thru to xab_scan
      0030 399 :
      0030 400

```

```

0030 402          .SBTTL XAB_SCAN - COMMON XAB DISPATCH ROUTINE
0030 403
0030 404      :++
0030 405      : RMSXAB_SCAN: Scan XAB chain for processing
0030 406      :
0030 407      : this routine scans the xab chain for a match on any of those xabs
0030 408      : specified in the argument list.  if an xab is found it is probed
0030 409      : and dispatch is made to the routine indicated by the argument list
0030 410      : case index.
0030 411      :--
0030 412
0030 413  RMSXAB_SCAN::
0030 414  XAB_SCAN:
      54  D4 0030 415          CLRL    R4          ; clear uniqueness vector
0032 416
0032 417          ASSUME  FAB$B_BID    EQ    RAB$B_BID
0032 418
      03  68  91 0032 419          CMPB   FAB$B_BID(R8),#FAB$C_BID ; Is the user structure a FAB or RAB?
0035 420          BNEQ   5$          ; Not a FAB, Must be a RAB
      53  24  A8  D0 0037 421          MOVL   FAB$L_XAB(R8),R3      ; get 1st xab address from FAB
003B 422          BRB    CHKXAB      ; Continue - check if XAB present
      53  40  A8  D0 003D 423 5$: MOVL   RAB$L_XAB(R8),R3      ; get 1st XAB address from RAB
0041 424  CHKXAB: BNEQ   6$          ; (another) XAB present
009F 31  0043 425          BRW    SUCXIT ; All done if none.
      59  D5  0046 426 6$: TSTL   R9          ; If called from EXTRMS, then could..
      06  13  0048 427          BEQL   7$          ; ..not have an IFB/IRB structure.
004A 428
004A 429          ASSUME  IFB$B_MODE    EQ    IRB$B_MODE
004A 430
      51  0A  A9  9A 004A 431          MOVZBL IFB$B_MODE(R9),R1      ; get mode in R1 for probe
004E 432          BRB    8$          ; Continue - probe structure
0050 433
      63  51  57  D0 0050 434 7$: MOVL   R7,R1          ; Move the user calling mode to R1
0053 435 8$: PROBER  R1,#XAB$L_NXT+4,(R3) ; probe readability
0057 436          BNEQ   9$          ;
0094 31  0059 437          BRW    ERRXAB      ; error if no access.
      52  5C  D0 005C 438 9$: MOVL   AP,R2          ; get address of scan table
      63  82  91 005F 439 10$: CMPB   (R2)+,XAB$B_COD(R3) ; xab code match this entry?
0062 440          BEQL   GOTXAB      ; branch if yes
      52  02  C0 0064 441          ADDL2  #2,R2          ; next table entry
      62  95  0067 442          TSTB   (R2)          ; end of table?
      F4  12  0069 443          BNEQ   10$         ; branch if not
006B 444
006B 445      :
006B 446      : found no match in the table.
006B 447      : check against all xab codes for at least valid and if so ignore it.
006B 448      :
006B 449
006B 450          ASSUME  FAB$B_BID    EQ    RAB$B_BID
006B 451
      03  68  91 006B 452          CMPB   FAB$B_BID(R8),#FAB$C_BID ; Is the user structure a FAB or RAB?
006E 453          BNEQ   15$         ; Not a FAB, Must be a RAB
      8B  AF  0A  63  3A 0070 454          LOCC   XAB$B_COD(R3),#XABTBLEN,XABTBL ; known FAB XAB code?
0075 455          BRB    20$         ; Continue
      8E  AF  02  63  3A 0077 456 15$: LOCC   XAB$B_COD(R3),#RXABTBLEN,RXABTBL ; known RAB XAB code?
007C 457 20$: BEQL   ERPCOD      ; branch if not
      53  04  A3  D0 007E 458  NXTXAB: MOVL   XAB$L_NX1(R3),R3      ; get next xab in chain

```

```

BD 11 0082 459 BRB CHKXAB ; go process
      0084 460
      0084 461 ;
      0084 462 ; found a match in the xab table.
      0084 463 ; check the length, probe it, and dispatch to the handler.
      0084 464 ;
      0084 465
50 01 A3 9A 0084 466 GOTXAB: MOVZBL XAB$B_BLN(R3),R0 ; get xab length field
      82 50 91 0088 467 CMPB R0,(R2)+ ; is length < min.?
      63 1F 008B 468 BLSSU ERRXAB ; branch if yes
      59 D5 008D 469 TSTL R9 ; If called from EXTRMS, then could..
      06 13 008F 470 BEQL 30$ ; ..not have an IFB/IRB structure.
      0091 471
      0091 472 ASSUME IFB$B_MODE EQ IRB$B_MODE
      0091 473
51 0A A9 9A 0091 474 MOVZBL IFB$B_MODE(R9),R1 ; get mode in R1 for probe
      03 11 0093 475 BRB 35$ ; continue - probe structure
      0097 476
63 51 57 D0 0097 477 30$: MOVL R7,R1 ; Move the user calling mode to R1
      50 51 OD 009A 478 35$: PROBEW R1,R0,(R3) ; probe writeability
      50 13 009E 479 BEQL ERRXAB ; branch if no access
      DB AF 9F 00A0 480 PUSHAB NXTXAB ; push return pc
      00A3 481 CASE TYPE=B, SRC=(R2),- ; and go process
      00A3 482 DISPLIST=<XABOPNFHC,XABOPNFHC1,XABOPNPRO,XABOPNDAT,-
      00A3 483 XABCREALLO,RM$EXTEND,XAB,XABCREALL1,XABCREPRO,-
      00A3 484 XABOPNRDT,XABOPNALL,XABCLSPRO,XABCLSRDT,-
      00A3 485 XABDSPFHC,XABDSPFHC1,XABDSPALL,XABDSPALL1,-
      00A3 486 XABCREALL2,RM$ISUM03,RM$IALLO3,RM$IKEY03,-
      00A3 487 XABOPNJNL,XABCREJNL,XABEXTCXF,XABEXTCXR,-
      00A3 488 XABOPNCXF,XABCONNCLR,XCONN03,XABTRM,XABOPNPRO1,-
      00A3 489 XABENTPRO,XABCREPRO1>
      00E5 490
      00E5 491 ;
      00E5 492 ; (note: should never fall thru)
      00E5 493 ;
      00E5 494 ; dispatch with standard r8 thru r11 and
      00E5 495 ; r6 = fib address
      00E5 496 ; r5 = addr to build next attribute list entry
      00E5 497 ; r4 = xab uniqueness bit vector
      00E5 498 ; r3 = xab address
      00E5 499 ;
      00E5 500 ; must return with r4 and r5 updated as necessary and only r0 thru r2
      00E5 501 ; modified. note: ap must not be modified.
      00E5 502 ;
      00E5 503 ; error routine should clean return pc from stack and rsb with error code in r0.
      00E5 504 ;
      00E5 505 ;
      05 00E5 506 SUCXIT: RMSSUC
      00E8 507 RSB
      00E9 508
      00E9 509 ;
      00E9 510 ; Conflicting XABJNL RU attributes
      00E9 511 ;
      00E9 512 ;
      00E9 513 ERR_CONFRU:
      00E9 514 RMSERR XCR ; no arguments
      16 11 00EE 515 BRB CLNSTK

```

```

00F0 516
00F0 517 :
00F0 518 : handle errors. problem could be:
00F0 519 :
00F0 520 : - non-accessible xab
00F0 521 : - invalid xab length
00F0 522 :
00F0 523 :
00F0 524 ERRXAB:
OC A8 53 D0 00F0 525 RMSERR XAB ; declare error
05 00F5 526 SETSTV: M(VL R3,FAB$L_STV(R8) ; give address
00F9 527 RSB
00FA 528
00FA 529 :
00FA 530 : handle invalid xab code error
00FA 531 :
00FA 532 :
00FA 533 ERRCOD:
F4 11 00FA 534 RMSERR COD
00FF 535 BRB SETSTV
0101 536
0101 537 :
0101 538 : handle invalid duplicate xab error
0101 539 :
0101 540 :
0101 541 ERRIMX:
0101 542 RMSERRIMX::
0101 543
8E D5 0106 544 CLNSTK: TSTL IMX ; remove return pc from stack
EB 11 0108 545 BRB SETSTV
010A 546
010A 547

```

```

010A 549      .SBTTL  FHC XAB PROCESSING ROUTINES
010A 550
010A 551      :++
010A 552      :
010A 553      : XABDSPFHC:  Process file header XAB for $DISPLAY, part 1
010A 554      : XABOPNFHC:  Process file header XAB for $OPEN and $CREATE, part 1
010A 555      : XABDSPFHC1: Process file header XAB for $DISPLAY, part 2
010A 556      : XABOPNFHC1: Process file header XAB for $OPEN and $CREATE, part 2
010A 557      :
010A 558      : xab routine to process the file header characteristics xab for $open,
010A 559      : $create, and $display
010A 560      :
010A 561      : called in two parts.  the first part merely builds an attribute list
010A 562      : entry to read the record attributes into the user's xab.
010A 563      :
010A 564      : the second part fills in the sbn and swaps the ebk and hbk fields.
010A 565      :
010A 566      :--
010A 567      :
010A 568      :++
010A 569      :
010A 570      : entry point for $display processing.  (must read in statistics block)
010A 571      :
010A 572      :--
010A 573      :
010A 574      XABDSPFHC:
85 04 80 010A 575      MOVW  #4,(R5)+      ; read only sbn longword
85 09 80 010D 576      MOVW  #ATRSC_STATBLK,(R5)+ ; of statistics block
85 28 A3 DE 0110 577      MOVAL  XAB$L_SBN(R3),(R5)+ ; read directly into xab
0114 578      :
0114 579      :++
0114 580      :
0114 581      : entry point for $open and $create processing.  (read in record attributes)
0114 582      :
0114 583      :--
0114 584      :
0114 585      XABOPNFHC:
23 A9 02 91 0114 586      CMPB  #IFBSC_IDX,IFB$B_ORGCASE(R9) ; if this is not an indexed file
      05 12 0118 587      BNEQ  5$ ; process xab
      17 A3 95 011A 588      TSTB  XAB$B_AID(R3) ; if not 1st area xab
      1D 12 011D 589      BNEQ  10$ ; skip processing
DE 54 00 E2 011F 590 5$:  BBSS  #XBCSC_OPNFHC,R4,ERRIMX ; flag xab seen, error
0123 591      :
0123 592      :
0123 593      : if already one
0123 594      :
0123 595      :
85 20 80 0123 596      MOVW  #32,(R5)+ ; return all 32 bytes
85 04 80 0126 597      MOVW  #ATRSC_RECATTR,(R5)+ ; read record attributes
85 08 A3 DE 0129 598      MOVAL  XAB$B_RFO(R3),(R5)+ ; address to read attr.s
01 1F A8 91 012D 599      CMPB  FAB$B_RFM(R8),#FABSC_FIX ; fixed rec. format?
      09 13 0131 600      BEQL  10$ ; branch if yes
05 69 32 E1 0133 601      BBC   #IFB$V_CREATE,(R9),10$ ; skip copy of LRL if not create
52 A9 0A A3 80 0137 602      MOVW  XAB$W_LRL(R3),IFB$W_LRL(R9); copy longest rec. len.
013C 603      :
013C 604      :
013C 605      : in case this is $create

```



```

013C 606 :
013C 607 :
05 013C 608 10$: RSB
013D 609 :
013D 610 :++
013D 611 :
013D 612 : part 2 (called after the access qio)
013D 613 :
013D 614 :--
013D 615 :
013D 616 :++
013D 617 :
013D 618 : entry point for $open
013D 619 :
013D 620 :--
013D 621 :
013D 622 XABOPNFHC1:
28 A3 01A8 CA D0 013D 623          MOVL   FWASL_SBN(R10),XABSL_SBN(R3) ; set starting lbn
      2C A6 B0 0143 624          MOVW   FIBSW_VERLIMIT(R6),- ; and version limit
      26 A3      0146 625          XABSW_VERLIMIT(R3)
0148 626 :
0148 627 :++
0148 628 :
0148 629 : entry point for $display
0148 630 :
0148 631 :--
0148 632 :
0148 633 XABDSPFHC1:
28 A3 28 A3 10 9C 0148 634          ROTL   #16,XABSL_SBN(R3),XABSL_SBN(R3) ; swap halves of start lbn
10 A3 10 A3 10 9C 014E 635          ROTL   #16,XABSL_EBK(R3),XABSL_EBK(R3) ; swap halves of ebk
      0B 12 0154 636          BNEQ   10$ ; branch if non-zero
0156 637 :
0156 638          ASSUME  FABSC_SEQ      EQ      0
0156 639 :
      23 A9 95 0156 640          TSTB   IFBSB_ORGCASE(R9) ; sequential file org?
      06 13 0159 641          BEQL   10$ ; branch if yes
10 A3 70 A9 01 C1 015B 642          ADDL3  #1,IFBSL_HBK(R9),XABSL_EBK(R3) ; set ebk from hbk + 1
      0C A3 70 A9 D0 0161 643 10$: MOVL   IFBSL_HBR(R9),XABSL_HBR(R3) ; set hbk from ifab
0166 644 :
0166 645 :
0166 646 : force stream format files to appear to have RAT non-null,
0166 647 : even if they don't.
0166 648 :
0166 649 :
0166 650          ASSUME  FABSC_STM      LT      FABSC_STMLF
0166 651          ASSUME  FABSC_STM      LT      FABSC_STMCR
0166 652          ASSUME  FABSV_ORG      EQ      4
0166 653          ASSUME  FABSS_ORG      EQ      4
0166 654 :
50 08 A3 04 00 EF 0166 655          EXTZV  #0,#4,XABSB_RFO(R3),R0 ; extract record format
      04 50 91 016C 656          CMPB   R0,#FABSC_STM ; stream format?
      0A 1F 016F 657          BLSSU  30$ ; nope
      07 93 0171 658          BITB   #<FABSM_CR!FABSM_FTN!FABSM_PRN>,-
      09 A3 0173 659          XABSB_ATR(R3) ; carriage control already set?
      04 12 0175 660          BNEQ   30$ ; ok
      09 A3 02 88 0177 661          BISB2  #FABSM_CR,XABSB_ATR(R3) ; force RAT=CR
05 017B 662 30$: RSB

```

```

017C 664          .SBTTL  DAT XAB ROUTINE
017C 665
017C 666      :++
017C 667      :XABOPNDAT:  Process the date and time xab for $open, $create, & $display.
017C 668      :
017C 669      : in all three cases attribute list entries are made for creation date & time,
017C 670      : expiration date & time, revision date & time, and revision count.
017C 671      :
017C 672      : !!! \note: this routine should be modified when a single attribute can
017C 673      :         handle all four.\ !!!
017C 674      :
017C 675      :--
017C 676
017C 677 XABOPNDAT:
81 54 03 E2 017C 678      BBSS      #XBC$C_OPNDAT,R4,ERRIMX ; flag xab seen, error
0180 679
0180 680      :
0180 681      : if already one
0180 682      :
0180 683      : make attribute list entries for creation & expiration date & time
0180 684      :
0180 685
85 08 B0 0180 686      MOVW      #8,(R5)+          ; date length
85 11 B0 0183 687      MOVW      #ATR$C_CREDATE,(R5)+      ; xfer creation date
0186 688
50 18 A3 14 A3 C9 0186 689      BISL3      XAB$L_CDT0(R3),XAB$L_CDT4(R3),R0 ; is date 0?
018C 690      BNEQU      10$          ; no - use date
018E 691      $GETTIM_S      TIMADR=XAB$Q_CDT(R3) ; yes - use current date
85 14 A3 DE 0198 692 10$:      MOVAL      XAB$Q_CDT(R3),(R5)+      ; xfer address
019C 693
019C 694      :
019C 695      : don't default expiration date
019C 696      :
019C 697
85 08 B0 019C 698      MOVW      #8,(R5)+          ; date length
85 13 B0 019F 699      MOVW      #ATR$C_EXPDATE,(R5)+      ; xfer expiration date
85 1C A3 DE 01A2 700      MOVAL      XAB$Q_EDT(R3),(R5)+      ; xfer address
01A6 701
01A6 702      :
01A6 703      : Only process the backup date and time if the length of the XAB block
01A6 704      : includes it. Do not default it.
01A6 705      :
01A6 706
2C 01 A3 91 01A6 707      CMPB      XAB$B_BLN(R3),#XAB$C_DATLEN ; does it include backup date?
01AA 708      BLSSU      20$          ; branch, if it does not
85 08 B0 01AC 709      MOVW      #8,(R5)+          ; date length
85 14 B0 01AF 710      MOVW      #ATR$C_BAKDATE,(R5)+      ; transfer backup date attribute code
85 24 A3 DE 01B2 711      MOVAL      XAB$Q_EDT(R3),(R5)+      ; transfer address
01B6 712 20$:      BRB      REV_DATE_COUNT      ; make attr entries for revision
01B8 713      ; date & count and return
01B8 714

```

```

01B8 716          .SBTTL RDT XAB ROUTINES
01B8 717
01B8 718 :++
01B8 719 : XABOPNRDT: process the revision date and time xab for $oper, $create, & $display.
01B8 720 : XABCLSRDT: process the revision date and time xab for $close
01B8 721 :
01B8 722 : in all three cases attribute list entries are made for revision date & time
01B8 723 : and revision count.
01B8 724 :
01B8 725 : !!! \note: this routine should be modified when a single attribute can
01B8 726 : handle both.\ !!!
01B8 727 :
01B8 728 :--
01B8 729
01B8 730 XABOPNRDT:
03 54 08 E3 01B8 731 BBCS #XBCSC_OPNRDT,R4,10$ ; flag xab seen,
0101 31 01BC 732 BRW ERRIMX_BR1 ; error if already set
01BF 733 10$:
01BF 734 REV_DATE_COUNT:
85 08 B0 01BF 735 MOVW #8,(R5)+ ; date length
85 12 B0 01C2 736 MOVW #ATRSC_REVDATE,(R5)+ ; xfer revision date
01C5 737
50 10 A3 0C A3 C9 01C5 738 BISL3 XABSL_RDT0(R3),XABSL_RDT4(R3),R0 ; is date 0 ?
0A 12 01CB 739 BNEQU 30$ ; no - use it
01CD 740 $GETTIM_S TIMADR=XAB$Q_RDT(R3) ; yes - get current date
85 0C A3 DE 01D7 741 30$: MOVAL XAB$Q_RDT(R3),(R5)+ ; xfer address
01DB 742
85 02 B0 01DB 743 MOVW #2,(R5)+ ; revision count length
85 0D B0 01DE 744 MOVW #ATRSC_ASCDATES,(R5)+ ; revision count attribute code
85 08 A3 3E 01E1 745 MOVAW XAB$W_RVN(R3),(R5)+ ; address to xfer rvn
05 01E5 746 RSB
01E6 747
01E6 748 :++
01E6 749 : routine to process rdt xab for $close.
01E6 750 :
01E6 751 : same processing as for $open rdt except that attribute list entries are
01E6 752 : built on the stack.
01E6 753 :
01E6 754 :--
01E6 755
01E6 756 XABCLSRDT:
03 54 08 E3 01E6 757 BBCS #XBCSC_CLSRDT,R4,10$ ; branch xab not seen
FF14 31 01EA 758 BRW ERRIMX ; flag xab seen , error
01ED 759
01ED 760 :
01ED 761 : if already one
01ED 762 :
U:FD 763
7E 5E 08 C2 01E9 764 10$: SUBL2 #8,SP ; create 8 bytes on stack
55 5E D0 01F0 765 MOVL SP,R5 ; build attr. list entries here
08 AE 7D 01F3 766 MOVQ 8(SP),-(SP) ; return addr to bottom of stk
C6 11 01F7 767 BRB REV_DATE_COUNT ; build attr list entries

```

```

                                .SBTTL  PRO XAB ROUTINES
01F9 769
01F9 770
01F9 771 :++
01F9 772 : XABOPNPRO: Process the protection xab for $open and $display. Make
01F9 773 : entries in the attribute list to have the ACP fill in the xab fields
01F9 774 : directly.
01F9 775 :
01F9 776 :--
01F9 777
01F9 778 XABOPNPRO:
03 54 02 E3 01F9 779 BBCS #XBC$C_OPNPRO,R4,5$ ; br if no xab seen (okay)
      FF01 31 01FD 780 BRW ERRIMX ; else flag duplicate xabpro
      0200 781
      4F 10 0200 782 5$: BSBB UIC ; make uic attr. list entry
      59 10 0202 783 BSBB MTACC ; make ansi accessibility entry
      3F 10 0204 784 BSBB PRO ; make file protection entry
58 8F 01 A3 91 0206 785 CMPB XAB$B_BLN(R3),#XAB$K_PROLEN ; Extended length XABPRO? (New V4)
      25 1F 0208 786 BLSSU 10$ ; skip if not
85 00260002 8F D0 020D 787 BSBB PROT MODE ; make access mode protection entry
      85 1E A3 DE 020F 788 MOVL #<ATR$C_ACLLENGTH@16>+2,(R5)+ ; attribute to return acl length
      18 A3 D5 021A 789 MOVAL XAB$W_ACLLEN(R3),(R5)+ ; xfer directly to xab
      13 13 021D 791 TSTL XAB$L_ACLBUF(R3) ; did user specify an acl buffer?
      85 1C A3 B0 021F 792 BEQLU 10$ ; branch if not
      85 85 25 B0 0223 793 MOVW XAB$W_ACLSIZ(R3),(R5)+ ; size of acl buffer
      18 A3 D0 0226 794 MOVL #ATR$C_READACL,(R5)+ ; attribute to place acl in user buffer
      20 A3 D0 022A 795 MOVL XAB$L_ACLBUF(R3),(R5)+ ; addr of acl buffer
      30 A6 022D 796 MOVL XAB$L_ACLCTX(R3),- ; pass acl context to file system
      24 A3 D4 022F 797 CLRL FIB$L_ACLCTX(R6)
      05 0232 798 10$: RSB XAB$L_ACLSTS(R3) ; clear acl status
      0233 799

```

```

0233 801 :++
0233 802 : XABOPNPRO1: This routine contains protection xab processing which must
0233 803 : follow the call to the acp for $open/$display operations.
0233 804 :
0233 805 :--
0233 806 :
01 A3 91 0233 807 XABOPNPRO1:
58 8F 0233 808         CMPB     XAB$B_BLN(R3),-      ; extended-length xabpro?
0A 1F 0236 809         #XAB$R_PROLEN
30 A6 1F 0238 810         BLSSU    10$          ; branch if not.
20 A3 D0 023A 811         MOVL     FIB$L_ACLCTX(R6),-  ; place acl context in xab
34 A6 D0 023D 812         MOVL     XAB$L_ACLCTX(R3)
24 A3 D0 023F 813         MOVL     FIB$L_ACL_STATUS(R6),- ; return status for acl operation
05 0242 814         XAB$L_ACLSTS(R3)
0244 815 10$:      RSB
0245 816
0245 817
0245 818
0245 819
0245 820 :
0245 821 : Miscellaneous subroutines used by protection xab routines.
0245 822 :
0245 823 :
0245 824 :
0245 825 : put protection attribute code into attribute list
0245 826 :
85 00160002 8F D0 0245 827 PRO:  MOVL     #<ATR$C_FPRO@16>+2,(R5)+ ; display SOWG protection - 2-byte field
85 08 A3 DE 024C 828         MOVAL    XAB$W_PRO(R3),(R5)+      ; xfer directly to xab
05 0250 829         RSB
0251 830
0251 831 :
0251 832 : put uic attribute code into attribute list
0251 833 :
85 00150004 8F D0 0251 834 UIC:  MOVL     #<ATR$C_UIC@16>+4,(R5)+ ; display uic in longword field
85 0C A3 DE 0258 835         MOVAL    XAB$L_UIC(R3),(R5)+      ; xfer directly to xab
05 025C 836         RSB
025D 837
025D 838 :
025D 839 : put ansi accessibility attribute code into attribute list
025D 840 :
85 001E0001 8F D0 025D 841 MTACC: MOVL     #<ATR$C_HDR1_ACC@16>+1,(R5)+ ; display ansi accessibility byte
85 0A A3 DE 0264 842         MOVAL    XAB$B_MTACC(R3),(R5)+      ; xfer directly to xab
05 0268 843         RSB
0269 844
0269 845 :
0269 846 : put access mode protection attribute code into attribute list
0269 847 :
0269 848 PF T_MODE:
85 00180001 8F D0 0269 849         MOVL     #<ATR$C_ACLEVEL@16>+1,(R5)+ ; display access mode prot. byte
85 10 A3 DE 0270 850         MOVAL    XAB$B_PROT_MODE(R3),(R5)+      ; xfer directly to xab
05 0274 851         RSB

```

```

0275 853 :++
0275 854 : XABCREPRO: Process the protection xab for $create
0275 855 :
0275 856 : If nonzero uic, then enter attribute to write the uic. If uic=0
0275 857 : and the fab$u_cif bit is set,
0275 858 : make the attribute atr$u_cif_ro which means although this is a
0275 859 : WRITE ATTRIBUTES subfunction, but won't you please READ ATTRIBUTES
0275 860 : the uic if this turns out to be an open of an existing file so that the
0275 861 : uic isn't altered.
0275 862 :
0275 863 : Enter attribute to write the protection field unless the default -1
0275 864 : is set. Note that since -1 (^XFFFF) indicates 'No access', you cannot
0275 865 : $CREATE a file for no access using RMS.
0275 866 :
0275 867 :--
0275 868 :
0275 869 XABCREPRO:
47 54 07 E2 0275 870 BBSS #XBC$C_CREPRO,R4,ERRIMX_BR1 ; flag xab seen, error
0279 871 :
0279 872 :
0279 873 : handle uic
0279 874 :
0279 875 :
28 AA 0C A3 D0 0279 876 MOVL XAB$UIC(R3),FWA$UIC(R10) ; copy uic to FWA
027E 877 BNEQ 20$ ; branch if not zero
0280 878 :
0280 879 :
0280 880 : zero uic
0280 881 :
0280 882 :
0A 68 39 E1 0280 883 BBC #FAB$V_CIF+FOP,(R8),40$ ; branch unless cif set
0284 884 BSBB UIC ; specify uic read
FA A5 1A B0 0286 885 MOVW #ATR$C_UIC_RO,-6(R5) ; make sure uic not written
028A 886 BRB 40$
028C 887 :
028C 888 20$: BSBB UIC ; use non-zero uic
028E 889 :
028E 890 :
028E 891 : handle ansi accessibility
028E 892 :
028E 893 :
0A A3 95 028E 894 40$: TSTB XAB$B_MTACC(R3) ; any specified?
02 13 0291 895 BEQL 50$ ; branch if yes
02 10 0293 896 BSBB MTACC
0295 897 :
0295 898 :
0295 899 : handle protection
0295 900 :
0295 901 :
2C AA 08 A3 B0 0295 902 50$: MOVW XAB$W_PRO(R3),FWA$W_PRO(R10) ; put prot into FWA
50 2C AA 01 A1 029A 903 ADDW3 #1,FWA$W_PRO(R10),R0 ; is protection = -1? (i.e. default)
02 13 029F 904 BEQL 60$ ; branch if so
02 A2 10 02A1 905 BSBB PRO ; include attribute to write protection
58 8F 01 A3 91 02A3 906 60$: CMPB XAB$B_BLN(R3),#XAB$K_PROLEN ; extended length XABPRO? (New V4)
02 A8 1F 02A8 907 BLSSU 70$ ; skip if not
02 AA 10 02AA 908 BSBB PROT_MODE ; attribute for access mode protection
18 A3 D5 02AC 909 TSTL XAB$C_ACLBUF(R3) ; did user specify an acl buffer?

```

```
85      0E      13 02AF  910      BEQLU  70$      ; branch if not
      1C A3     B0 02B1  911      MOVW   XAB$W_ACLSIZ(R3),(R5)+ ; siz of acl buffer
85      85 1F     B0 02B5  912      MOVW   #ATR$C_ADDACLENT,(R5)+ ; attribute to create acl
      18 A3     D0 02B8  913      MOVL   XAB$L_ACLBUF(R3),(R5)+ ; address of user's acl buffer
      24 A3     D4 02BC  914      CLRL   XAB$L_ACLSTS(R3)      ; clear acl status
      05      05 02BF  915 70$:   RSB
      02C0  916
      02C0  917 ;
      02C0  918 ; extended branch for error
      02C0  919 ;
      02C0  920
      FE3E 31 02C0  921 ERRIMX_BR1
      02C0  922 BRW      ERRIMX
```

```
02C3 924 :++
02C3 925 : XABCREPRO1: Process the protection xab for $create, part 2.
02C3 926 :
02C3 927 : If extended xabpro then return acl status.
02C3 928 :--
02C3 929 :
02C3 930 XABCREPRO1:
01 A3 91 02C3 931      CMPB  XAB$B_BLN(R3),-      ; extended-length xabpro?
58 8F      02C6 932      #XAB$R_PROLEN
      05 1F 02C8 933      BLSSU  10$      ; branch if not
34 A6 00 02CA 934      MOVL   FIB$L_ACL_STATUS(R6),- ; return status for acl operation
24 A3      02CD 935
      05 02CF 936 10$:  RSB   XAB$L_ACLSTS(R3)
      02D0 937
```



```

02D0 939
02D0 940 :++
02D0 941 : XABCLSPRO: Process the pro xab for $close
02D0 942 :
02D0 943 : checks for default uic (i.e., 0) and if found then don't process uic
02D0 944 :
02D0 945 : also check for default pro (i.e., -1 - no access to anyone), and if found
02D0 946 : don't touch the protection
02D0 947 :
02D0 948 :--
02D0 949 :
02D0 950 XABCLSPRO:
EC 54 0A E2 02D0 951 BBSS #XBC$C_CLSPRO,R4,ERRIMX_BR1 ; flag xab seen, error
02D4 952 :
02D4 953 :
02D4 954 : if already one
02D4 955 :
02D4 956 : handle uic
02D4 957 :
02D4 958 :
OC A3 D5 02D4 959 TSTL XAB$L_UIC(R3) ; uic = 0?
09 13 02D7 960 BEQL 10$ ; branch if yes
55 5E D0 02D9 961 MOVL SP,R5 ; point to attr. list entry on stack
7E 6E 7D 02DC 962 MOVQ (SP),-(SP) ; create 8 bytes on stack
02DF 963 :
02DF 964 :
02DF 965 : for uic attribute entry
02DF 966 :
02DF 967 :
FF6F 30 02DF 968 BSBW UIC ; specify uic write
02E2 969 10$:
02E2 970 :
02E2 971 :
02E2 972 : process pro
02E2 973 :
02E2 974 :
50 08 A3 01 A1 02E2 975 ADDW3 #1,XAB$W_PRO(R3),R0 ; is pro spec = -1?
09 13 02E7 976 BEQL 20$ ; branch if so
55 5E D0 02E9 977 MOVL SP,R5 ; point to attr. list entry on stack
7E 6E 7D 02EC 978 MOVQ (SP),-(SP) ; create 8 free bytes on stack
FF53 31 02EF 979 BRW PRO ; go process protection
05 02F2 980 20$: RSB
02F3 981

```

```
02F3 983 :++
02F3 984 : XABENTPRO: process the protection xab for the directory 'enter' which
02F3 985 : occurs during an $enter or a $rename operation. Currently, the only
02F3 986 : thing to do is to pass the xab$v_propagate bit on to the file system.
02F3 987 :
02F3 988 :--
02F3 989 :
02F3 990 XABENTPRO:
C9 54 1D E2 02F3 991 BBSS #XBCSC_ENTPRO,R4,ERRIMX_BR1 ; only allow 1 xabpro
06 0B A3 E9 02F7 992 ASSUME XAB$v_PROPAGATE EQ 0
02FB 993 BLBC XAB$b_PROT_OPT(R3),20$ ; skip if propagate not set
02FB 994 SSB #FIB$v_PROPAGATE - ; set propagate bit in fib
05 0301 995 FIB$L_STATUS+FWA$t_FIBBUF(R10)
02FB 996 20$: RSB
```

```

0302 998      .SBTTL  ALL XAB ROUTINES
0302 999
0302 1000    :++
0302 1001    : XABDSPALL:  Process allocation xab for $display, part 1
0302 1002    : XABOPNALL:  Process allocation xab for $open
0302 1003    : XABDSPALL:  Process allocation xab for $display, part 2
0302 1004    :
0302 1005    : sets the alq, deq, bkz, and aop (ctg and cbt bits only) fields of the xab
0302 1006    :
0302 1007    :--
0302 1008    :
0302 1009    :++
0302 1010    :
0302 1011    : entry point for $display to cause the user characteristics bits to
0302 1012    : be read into the xab.
0302 1013    :
0302 1014    :--
0302 1015    :
0302 1016    XABDSPALL:
167 0302 1017    CMPB      #IFB$C_IDX,IFB$B_ORGCASE(R9) ; indexed file?
168 0306 1018    BNEQ      10$ ; branch if not
169 17 0308 1019    TSTB      XAB$B_AID(R3) ; is this the first area id?
170 030B 1020    BNEQ      20$ ; branch if not
171 AF 54 030D 1021 10$:  BBSS      #XBC$C_DSPALL,R4,ERRIMX_BR1 ; error is xab already seen
172 85 0311 1022    MOVW      #1,(R5)+ ; just read 1st byte
173 85 0314 1023    MOVW      #ATR$C_UCHAR,(R5)+ ; user char. attr. code
174 85 0317 1024    MOVAB     XAB$B_AOP(R3),(R5)+ ; addr for read
175 031B 1025 20$:  RSB
031C 1026
031C 1027    :++
031C 1028    :
031C 1029    : entry point for $open processing
031C 1030    :
031C 1031    :--
031C 1032    :
031C 1033    XABOPNALL:
176 08 A3 44 AA 90 031C 1034    MOVB      FWAS$UCHAR(R10),XAB$B_AOP(R3) ; set aop byte from user char.
0321 1035
0321 1036    :++
0321 1037    :
0321 1038    : entry point for 2nd part of $display processing
0321 1039    :
0321 1040    :--
0321 1041    :
0321 1042    XABDSPALL1:
0321 1043
0321 1044    ASSUME     FCH$V_CONTIG EQ XAB$V_CTG
0321 1045    ASSUME     FCH$V_CONTIGB EQ XAB$V_CBT
0321 1046
177 08 A3 5F 8F 8A 0321 1047    BICB2     #255 \ <XAB$M_CTG!XAB$M_CBT>,XAB$B_AOP(R3) ; clear other bits
178 23 A9 02 91 0326 1048    CMPB      #IFB$C_IDX,IFB$B_ORGCASE(R9) ; indexed file?
179 032A 1049    BEQL      IDXRET ; branch if indexed
180 14 A3 62 A9 80 032C 1050    MOVW      IFB$W_DEQ(R9),XAB$W_DEQ(R3) ; set deq
181 16 A3 5E A9 90 0331 1051    MOVB      IFB$B_BKS(R9),XAB$B_BKZ(R3) ; set bkz
182 10 A3 70 A9 80 0336 1052    SETALQ:  MOVL     IFB$L_HBK(R9),XAB$L_ALQ(R3) ; set alq
183 05 033B 1053    IDXRET:  RSB
033C 1054

```

```

033C 1056
033C 1057 :++
033C 1058 : XABCREALL1: Process allocation xab for $create, part 1
033C 1059 : XABCREALL2: Process allocation xab for $create, part 2
033C 1060 :
033C 1061 : Process allocation XAB for area 0 only, since called by RM$CREATECOM
033C 1062 : when file created. Uses the allocation xab parameters to set up the FIB,
033C 1063 : whose address must be in r6.
033C 1064 :
033C 1065 :--
033C 1066
033C 1067 XABCREALL1:
00 54 06 E2 033C 1068 BBSS #XBC$C_CREALL1,R4,10$ ; set alloc XAB found
      17 A3 95 0340 1069 10$: TSTB XAB$B_AID(R3) ; area 0?
      25 12 0343 1070 BNEQ 50$ ; branch if not (do not process)
      56 DD 0345 1071 PUSHL R6 ; save fib address
      51 56 D0 0347 1072 MOVL R6,R1 ; copy it to r1
      56 53 D0 034A 1073 MOVL R3,R6 ; xab address to r6
      FCBO' 30 034D 1074 BSBW RM$SET_XABALL ; set up fib from xab
      56 8E D0 0350 1075 MOVL (SP)+,R6 ; restore fib addr
      15 50 E9 0353 1076 BLBC R0,60$ ; branch if bad values
05 16 A6 01 E1 0356 1077 BBC #FIB$V_ALCONB,FIB$W_EXCTL(R6),20$ ; branch if not cbt
0A 44 AA 05 E3 035B 1078 BBCS #FCH$V_CONTIGB,FWA$W_UCHAR(R10),50$ ; set cbt file attr. & branch
05 16 A6 00 E1 0360 1079 20$: BBC #FIB$V_ALCON,FIB$W_EXCTL(R6),50$ ; branch if not ctg
44 AA 80 8F 88 0365 1080 50$: BISB2 #1aFCH$V_CONTIG,FWA$W_UCHAR(R10) ; set ctg file attribute
      05 036A 1081 RSB
      036B 1082
      036B 1083 ;
      036B 1084
      0067 31 036B 1085 60$: BRW STVRSB ; load stv with AID number and return
      036E 1086
      036E 1087 :++
      036E 1088 :
      036E 1089 : xabcreall2 routine to process filling in of alq field with actual
      036E 1090 : first extent size on $create.
      036E 1091 :
      036E 1092 :--
      036E 1093
      036E 1094 XABCREALL2:
      23 A9 02 91 036E 1095 CMPB #IFB$C_IDX,IFB$B_ORGCASE(R9) ; indexed file?
      06 12 0372 1096 BNEQ 5$ ; branch if not
      17 A3 95 0374 1097 TSTB XAB$B_AID(R3) ; first one?
      01 13 0377 1098 BEQL 5$ ; branch if so
      05 0379 1099 RSB ; return
      BA 11 037A 1100 5$: BRB SETALQ

```

```

037C 1102      .SBTTL  RMSSETALLOC - ROUTINE TO HANDLE ALLOC XAB FOR SEQ. & REL. F.O.
037C 1103
037C 1104      :++
037C 1105      : RMSSETALLOC: Process the allocation xab for the sequential and relative
037C 1106      : file organizations.
037C 1107      :
037C 1108      : XABCREALLO: routine called by xab_scan if an allocation xab is found
037C 1109      :
037C 1110      : if an allocation xab is found, it is checked for defining area 0 only and
037C 1111      : it is used to provide the alq, deq, and bks fields, these being copied to
037C 1112      : the fab for later processing. placement information is processed when the
037C 1113      : file creation is actually done.
037C 1114      :
037C 1115      : the entire xab chain is checked for validity.
037C 1116      :
037C 1117      :--
037C 1118
037C 1119  CREXABO_ARGS:
00 04 20 14 037C 1120      .BYTE  XABSC_ALL,XABSC_ALLEN,XBSC_CREALLO,0
0380 1121
0380 1122  RMSSETALLOC::
5C  F9 AF 9E 0380 1123      MOVAB  CREXABO_ARGS,AP      ; set arg list addr
   FCA9 30 0384 1124      BSBW   XAB_SCAN      ; go scan xab chain
0387 1125
0387 1126      :
0387 1127      : use fab deq field to set both deq and run-time deq fields of ifab
0387 1128      :
0387 1129
62 A9 14 A8 B0 0387 1130      MOVW   FAB$W_DEQ(R8),IFB$W_DEQ(R9)
4C A9 14 A8 B0 038C 1131      MOVW   FAB$W_DEQ(R8),IFB$W_RTDEQ(R9)
   05 0391 1132      RSB
0392 1133
0392 1134      :++
0392 1135      : xabcreallo routine called by xab_scan if an allocation xab is found
0392 1136      :
0392 1137      : verify this is the only allocation xab and that its aid = 0.
0392 1138      : copy the alq, deq, and bkz fields from the xab to the fab.
0392 1139      :
0392 1140      :--
0392 1141
0392 1142  XABCREALLO:
23 A9 02 91 0392 1143      CMPB   #IFBSC_IDX,IFB$B_ORGCASE(R9); indexed file?
   05 12 0396 1144      BNEQ   $$      ; branch if not
   17 A3 95 0398 1145      TSTB   XAB$B_AID(R3)      ; is this the first area id?
   28 12 0398 1146      BNEQ   10$      ; branch if not
25 54 04 E2 039D 1147  5$:  BBSS   #XBSC_CREALLO,R4,ERRIMX_BR ; error is all. xab already seen
   17 A3 95 03A1 1148      TSTB   XAB$W_AID(R3)      ; aid = 0?
   23 12 03A4 1149      BNEQ   ERRAID      ; branch if not (error)
10 A8 10 A3 D0 03A6 1150      MOVL   XAB$L_ALQ(R3),FAB$L_ALQ(R8) ; copy alq to fab
14 A8 14 A3 B0 03AB 1151      MOVW   XAB$W_DEQ(R3),FAB$W_DEQ(R8) ; and deq
   16 A3 91 03B0 1152      CMPB   XAB$B_BKZ(R3),-
   3F 03B3 1153      #BKTSC_MAXBKFSIZ      ; bkz in range?
3E A8 16 A3 90 03B4 1154      BGTRU  ERRBKZ      ; branch if not
   03B6 1155      MOVB   XAB$B_BKZ(R3),FAB$B_BKS(R8) ; and bks
   03BB 1156
   03BB 1157      ASSUME FAB$C_EQ      EQ      0
   03BB 1158

```


| | | | | | | | | | | |
|----|----|------|------|------|------|------|------|--------|--|---------------------------------|
| | | | 03 | 12 | 04AF | 1304 | BNEQ | 5\$ | | |
| | | | 0095 | 31 | 04B1 | 1305 | BRW | 40\$ | | ; branch if none |
| | | | | | 04B4 | 1306 | | | | |
| | | | | | 04B4 | 1307 | | | | |
| | | | | | 04B4 | 1308 | | | | |
| | | | | | 04B4 | 1309 | | | | |
| | | | | | 04B4 | 1310 | | | | |
| | | | | | 04B4 | 1311 | 5\$: | PUSHR | #^M<R2,R3,R4,R5,R6> | ; save registers |
| | 28 | 00A0 | C9 | 02 | E1 | 04B8 | | BBC | #IFBSV BI,IFBSB_JNLFLG(R9),10\$ | ; branch if no BI |
| | | 52 | 0C | A3 | 9A | 04BE | | MOVZBL | XAB\$B_BIS(R3),R2 | ; get buffr size |
| | | | | 22 | 13 | 04C2 | | BEQL | 10\$ | ; branch if no buff |
| | 10 | B3 | 52 | 0A | 0D | 04C4 | | PROBEW | IFBSB_MODE(R9),R2,@XAB\$L_BIA(R3) | ; probe buff |
| | | | | C8 | 13 | 04CA | | BEQL | ERRXAB_BRJ | ; branch on error |
| | | | | | | 04CC | | | | |
| | | | | | | 04CC | | ASSUME | ACESB_SIZE EQ 0 | |
| | | | | | | 04CC | | | | |
| | | 56 | 08E0 | CA | 9A | 04CC | | MOVZBL | FWAST_BIACE(R10),R6 | ; get ACE size |
| | | | 56 | 04 | 82 | 04D1 | | SUBB | #ACES ACCESS,R6 | ; calculate BI name size |
| 10 | B3 | 52 | 20 | 08E4 | 2C | 04D4 | | MOVCS | R6,FWAST_BIJNLN(R10),#^A/ /,R2,@XAB\$L_BIA(R3) | |
| | | | | 53 | 04 | 04DD | | MOVL | 4(SP),R3 | ; Restore XAB addr |
| | | | | 0D | A3 | 04E1 | | SUBB3 | R0,R6,XAB\$B_BIL(R3) | ; calculate moved size |
| | | | | | | 04E6 | | | | |
| | | 28 | 00A0 | C9 | 03 | E1 | 04E6 | 10\$: | BBC | #IFBSV AI,IFBSB_JNLFLG(R9),20\$ |
| | | | 52 | 14 | A3 | 9A | 04EC | | MOVZBL | XAB\$B_AIS(R3),R2 |
| | | | | 22 | 13 | 04F0 | | BEQL | 20\$ | ; branch if no buff |
| | 18 | B3 | 52 | 0A | 0D | 04F2 | | PROBEW | IFBSB_MODE(R9),R2,@XAB\$L_AIA(R3) | ; probe buff |
| | | | | 9A | 13 | 04F8 | | BEQL | ERRXAB_BRJ | ; branch on error |
| | | | 56 | 08F4 | 9A | 04FA | | MOVZBL | FWAST_AIACE(R10),R6 | ; get ACE size |
| | | | | 56 | 04 | 04FF | | SUBB | #ACES ACCESS,R6 | ; calculate AI name size |
| 18 | B3 | 52 | 20 | 08F8 | 2C | 0502 | | MOVCS | R6,FWAST_AIJNLN(R10),#^A/ /,R2,@XAB\$L_AIA(R3) | |
| | | | | 53 | 04 | 050B | | MOVL | 4(SP),R3 | ; Restore XAB addr |
| | | | 15 | A3 | 56 | 050F | | SUBB3 | R0,R6,XAB\$B_AIL(R3) | ; calculate moved size |
| | | | | | | 0514 | | | | |
| | | 28 | 00A0 | C9 | 04 | E1 | 0514 | 20\$: | BBC | #IFBSV AT,IFBSB_JNLFLG(R9),30\$ |
| | | | 52 | 1C | A3 | 9A | 051A | | MOVZBL | XAB\$B_ATS(R3),R2 |
| | | | | 25 | 13 | 051E | | BEQL | 30\$ | ; branch if no buff |
| | 20 | B3 | 52 | 0A | 0D | 0520 | | PROBEW | IFBSB_MODE(R9),R2,@XAB\$L_ATA(R3) | ; probe buff |
| | | | | 03 | 12 | 0526 | | BNEQ | 25\$ | |
| | | | | FF | 69 | 0528 | | BRW | ERRXAB_BRJ | ; branch on error |
| | | | 56 | 0908 | 9A | 052B | | MOVZBL | FWAST_ATAACE(R10),R6 | ; get ACE size |
| | | | | 56 | 04 | 0530 | | SUBB | #ACES ACCESS,R6 | ; calculate AT name size |
| 20 | B3 | 52 | 20 | 090C | 2C | 0533 | | MOVCS | R6,FWAST_ATJNLN(R10),#^A/ /,R2,@XAB\$L_ATA(R3) | |
| | | | | 53 | 04 | 053C | | MOVL | 4(SP),R3 | ; Restore XAB addr |
| | | | 1D | A3 | 56 | 0540 | | SUBB3 | R0,R6,XAB\$B_ATL(R3) | ; calculate moved size |
| | | | | | | 0545 | | | | |
| | | | | 007C | 8F | BA | 0545 | 30\$: | POPR | #^M<R2,R3,R4,R5,R6> |
| | | | | | | 0549 | | | | ; restore registers |
| | | | | | 05 | 0549 | | 40\$: | RSB | |
| | | | | | | 054A | | | | |

```

054A 1354          .SBTTL RMS$SETEXTRMS - ROUTINE TO HANDLE CONTEXT XAB FOR FAB & RAB
054A 1355
054A 1356 :++
054A 1357 : RMS$SETEXTRMS: Process the Context XABs for the FAB and RAB on RMS exit
054A 1358 :
054A 1359 :
054A 1360 : if a context xab is found, it is filled in with information from the
054A 1361 : IFB or IRB, and FAB or RAB, as needed.
054A 1362 :
054A 1363 : the entire xab chain is checked for validity by XAB_SCAN
054A 1364 :
054A 1365 :--
054A 1366
054A 1367 EXT_XAB_ARGS:
16 3C 20 054A 1368          .BYTE  XAB$C_CXF,XAB$C_CXLEN,XBC$C_EXTCXF
054D 1369          .BYTE  0
054E 1370
054E 1371 EXT_XABR_ARGS:
17 54 21 054E 1372          .BYTE  XAB$C_CXR,XAB$C_CXRLLEN,XBC$C_EXTCXR
0551 1373          .BYTE  0
0552 1374
0552 1375 RMS$SETEXTRMS::
0552 1376
0552 1377          ASSUME  FAB$B_BID      EQ      RAB$B_BID
0552 1378          ASSUME  FAB$L_STS      EQ      RAB$L_STS
0552 1379
03   68   91 0552 1380          CMPB   FAB$B_BID(R8),#FAB$C_BID ; Is the user structure a FAB or RAB?
0555 1381          BNEQ   10$ ; Not a FAB, Must be a RAB
24  A8   D5 0557 1382          TSTL   FAB$L_XAB(R8) ; any XABs?
055A 1383          BEQL   40$ ; don't bother checking
5C   E9  AF 9E 055C 1384          PUSHL  AP ; save register
055E 1385          MOVAB  EXT_XAB_ARGS,AP ; set arg list addr for a FAB
0567 1386          BRB   20$ ; Continue - Call XAB_SCAN
40  A8   D5 0564 1387 10$:  TSTL   RAB$L_XAB(R8) ; any XABs?
0567 1388          BEQL   40$ ; don't bother checking
5C   E0  AF 9E 0569 1389          PUSHL  AP ; save register
5C   E0  AF 9E 056B 1390          MOVAB  EXT_XABR_ARGS,AP ; set arg list addr for a RAB
5C   F0  AF 9E 056F 1391 20$:  BSBW   XAB_SCAN ; go scan xab chain
0572 1392          POPL  AP ; restore register
08  50   E8 0575 1393          BLBS   R0,30$ ; success
0578 1394          SSB   #16,R0 ; add facility code on failure
08  A8   50  D0 057C 1395          MOVL  R0,FAB$L_STS(R8) ; save failure status
50  08  A8  D0 0580 1396 30$:  MOVL  FAB$L_STS(R8),R0 ; restore actual status
0584 1397 40$:  RSB
0585 1398
0585 1399
0585 1400 :
0585 1401 : Fill in the contents of the FAB associated Context XAB,
0585 1402 : on exit from the RMS service.
0585 1403 :
0585 1404 : (R3) = address of the context XAB, already probed.
0585 1405 :
0585 1406 : If this is a restart operation, clear the restart bit and copy the
0585 1407 : bookkeeping bits to the IFAB (and anything else that has to be copied
0585 1408 : to the IFAB last). For optimization's sake, do not copy context into
0585 1409 : XAB for a restart.
0585 1410 :

```

```

0585 1411
0585 1412 XABEXTCXF:
59 D5 0585 1413 TSTL R9 ; Is this a structureless exit?
04 13 0587 1414 BEQL 2$ ; Yes - Skip bit clear
0589 1415
0589 1416 ASSUME IRBSV_RESTART EQ IFBSV_RESTART
43 69 3B E4 0589 1417
0589 1418 BBSC #IRBSV_RESTART,(R9),50$ ; Clear restart bit and return
058D 1419
058D 1420 ASSUME XAB$L_CXFSTS+4 EQ XAB$L_CXFS1V
058D 1421
08 A3 08 A8 7D 058D 1422 2$: MOVQ FAB$L_STS(R8),XAB$L_CXFSTS(R3) ; copy STS/STV fields.
18 A3 02 A8 80 0592 1423 MOVW FAB$W_IFI(R8),XAB$W_CXFIFI(R3) ; copy over IFI.
1A A3 01 90 0597 1424 MOVB #XAB$C_CXT_VER1,XAB$B_CXFVER(R3) ; Record the version.
059B 1425
059B 1426 : If there is not an internal RMS structure available, this exit is
059B 1427 : probably an error, or a close. Does not make sense to copy anything
0596 1428 : else into the context XAB. (cannot copy most things due to no structure)
059B 1429
059B 1430 TSTL R9 ; Is this a structureless ex
059D 1431 BEQL EX_NOIFB ; Yes, No IFAB available.
14 A3 04 A9 D0 059F 1432 MOVL IFB$L_BKPBITS(R9),XAB$L_CXFBKP(R3) ; Copy bookkeeping bits
20 A3 4C A9 80 05A4 1433 MOVW IFB$W_RTDEQ(R9),XAB$W_CXFDEQ(R3) ; Run time Default Extend Qu
22 A3 22 A9 90 05A9 1434 MOVW IFB$B_FAC(R9),XAB$B_CXFFAC(R3) ; File Access
23 A3 4E A9 90 05AE 1435 MOVW IFB$B_SHR(R9),XAB$B_CXFVSHR(R3) ; Share
50 23 A9 9A 05B3 1436 MOVZBL IFB$B_ORGCASE(R9),R0 ; File Organization
27 A3 50 04 78 05B7 1437 ASHL #4,R0,XAB$B_CXFORG(R3) ; Shift over for XAB
28 A3 64 A9 80 05BC 1438 MOVW IFB$W_GBC(R9),XAB$W_CXFGBC(R3) ; Global Buffer count
50 38 A9 D0 05C1 1439 MOVL IFB$L_FWA_PTR(R9),R0 ; Get FWA address
50 14 B0 9E 05C5 1440 MOVAB @FWA$Q_FIB+4(R0),R0 ; Get ptr to FIB in FWA
2A A3 03 A0 90 05C9 1441 MOVW FIB$B_QSIZE(R0),XAB$B_CXFRTV(R3) ; Copy Retrieval Window from
05 11 05CE 1442 BRB EX_NOIFB ; return
04 A9 14 A3 D0 05D0 1444 50$: MOVL XAB$L_CXFBKP(R3),IFB$L_BKPBITS(R9) ; transfer bookkeeping bits
05D5 1445
05D5 1446 EX_NOIFB:
05 D5 05D5 1447 RSB ; Return
05D6 1448
05D6 1449 :
05D6 1450 : Fill in the contents of the RAB associated Context XAB.
05D6 1451 : on exit from the RMS service.
05D6 1452 : (R3) = address of the context XAB, already probed.
05D6 1453 :
05D6 1454 : For optimization's sake, do not copy context into the XAB for a restart
05D6 1455 : operation.
05D6 1456 :
05D6 1457
05D6 1458 XABEXTCXR:
59 D5 05D6 1459 TSTL R9 ; Is this a structureless exit?
04 13 05D8 1460 BEQL 2$ ; Yes - Skip bit clear.
05DA 1461
05DA 1462 ASSUME IRBSV_RESTART EQ IFBSV_RESTART
37 69 3B E4 05DA 1463
05DA 1464 BBSC #IRBSV_RESTART,(R9),30$ ; Clear restart bit while noting res
05DE 1465
05DE 1466 ASSUME XAB$L_CXRSTS+4 EQ XAB$L_CXRSTV
05DE 1467

```

```

08 A3 08 A8 7D 05DE 1468 2$: MOVQ RAB$S_STS(R8),XAB$S_CXRSTS(R3) ; copy STS/STV fields.
18 A3 02 A8 80 05E3 1469 MOVW RAB$W_ISI(R8),XAB$W_CXRISI(R3) ; copy over IFI.
1A A3 01 90 05E8 1470 MOVB #XAB$C_CXT_VER1,XAB$B_CXRVER(R3) ; Record version
05EC 1471 :
05EC 1472 : If there is not an internal RMS structure available, this exit is
05EC 1473 : probably an error, or a disconnect. Does not make sense to copy anything
05EC 1474 : else into the context XAB (cannot copy some things due to no struture).
05EC 1475 :
59 D5 05EC 1476 TSTL R9 ; Is this a structureless ex
03 12 05EE 1477 BNEQ 20$ ; No, continue
00C3 31 05F0 1478 BRW EX_NOIRB ; Yes, No IRAB available.
14 A3 04 A9 D0 05F3 1479 20$: MOVL IRB$S_BKPBITS(R9),XAB$S_CXRBKP(R3) ; Copy bookkeeping bits
21 A3 55 A9 90 05F6 1480 MOVB IRB$B_MBC(R9),XAB$B_CXRMBC(R3) ; Copy Multi-block cnt
20 A3 5C A9 90 05FB 1481 MOVB IRB$B_MBF(R9),XAB$B_CXRMBF(R3) ; Copy Multi-buffer cnt
0602 1482 CASE TYPE=B,SRC=IFB$B_ORGCASE(R10),- ; Case on organization
0602 1483 DISPLIST=<EX_SEQ,EX_REL,EX_ISAM> ; copy different stuff
060D 1484 RMSERR ORG ; Organization error
FAF1 31 0612 1485 BRW CLNSTK ; Remove return PC from
0615 1486 ; stack and return.
0615 1487 :
0615 1488 : finish coping the last of the items for SEQ and REL restart connert.
0615 1489 :
0615 1490 30$:
40 A9 24 A3 D0 0615 1491 MOVL XAB$S_CXRVBN(R3),IRB$S_NRP_VBN(R9) ; copy NRP information -
44 A9 28 A3 80 061A 1492 MOVW XAB$W_CXROFF(R3),IRB$W_NRP_OFF(R9) ; VBN and offset
04 A9 14 A3 D0 061F 1493 MOVL XAB$S_CXRBKP(R3),IRB$S_BKPBITS(R9) ; copy bookkeeping bits
008F 31 0624 1494 BRW EX_NOIRB ; branch to return
0627 1495 :
0627 1496 : For each file organization, need to save different NRP context
0627 1497 :
0627 1498 : For Sequential and Relative files, copy the same stuff
0627 1499 :
0627 1500 EX_SEQ: ; Sequential files
0627 1501 :
0627 1502 EX_REL: ; Relative files
24 A3 40 A9 D0 0627 1503 MOVL IRB$S_NRP_VBN(R9),XAB$S_CXRVBN(R3) ; copy nrp vbn
28 A3 44 A9 80 062C 1504 MOVW IRB$W_NRP_OFF(R9),XAB$W_CXROFF(R3) ; copy nrp offset
0082 31 0631 1505 BRW EX_NOIRB
0634 1506 :
0634 1507 : The following will copy all ISAM specific current and next record
0634 1508 : positioning information to the context XAB.
0634 1509 :
0634 1510 : Copy the buffer version number into the XAB for restart comparison.
0634 1511 : This version number indicates what the attached context buffer looks
0634 1512 : like. If the buffer changes in internal character, this version needs
0634 1513 : to be updated.
0634 1514 : Note: that the RFA's are copied from the IRAB to the context block.
0634 1515 : The high order word of the id field of the RFA is forced to Zero.
0634 1516 : This high order word is checked to be zero on restart.
0634 1517 :
0634 1518 : Probe the user's key buffer and write counted ascii key strings
0634 1519 : into it. The buffer will contain the size of the key followed by
0634 1520 : keybuffer 1, then, the size of the key followed by keybuffer 6. The
0634 1521 : size field will be verified on restart to all match with the key size
0634 1522 : kept in the XAB context block.
0634 1523 :
0634 1524 EX_ISAM: ; Indexed files

```

```

10 A3 04 1C 01 F0 0634 1525 INSV #XAB$C_CXB_VERN,#XAB$V_CXRBVER,#XAB$$_CXRBVER,XAB$L_CXRCOP(R3)
2C A3 00AC C9 D0 063A 1526 MOVL IRB$L_POS_VBN(R9),XAB$L_CXRPOS0(R3) ; Primary record NRP RFA
30 A3 00BA C9 3C 0640 1527 MOVZWL IRB$W_POS_ID(R9),XAB$W_CXRPOS4(R3) ; Primary record NRP RFA
34 A3 00AB C9 D0 0646 1528 MOVL IRB$L_CUR_VBN(R9),XAB$L_CXRCUR0(R3) ; Primary record RFA
38 A3 00B8 C9 3C 064C 1529 MOVZWL IRB$W_CUR_ID(R9),XAB$W_CXRCUR4(R3) ; Primary record RFA
3C A3 00B4 C9 D0 0652 1530 MOVL IRB$L_SIDR_VBN(R9),XAB$L_CXRSID0(R3) ; SIDR RFA
40 A3 00BE C9 3C 0658 1531 MOVZWL IRB$W_SIDR_ID(R9),XAB$W_CXRSID4(R3) ; SIDR RFA
44 A3 00C0 C9 B0 065E 1532 MOVW IRB$W_CUR_COUNT(R9),XAB$W_CXRCNT(R3) ; SIDR array count
46 A3 00C3 C9 90 0664 1533 MOVVB IRB$B_CUR_REF(R9),XAB$B_CXRKREF(R3) ; Current key of ref.
50 00B4 CA 3C 066A 1534 MOVZWL IFB$W_KBUFSZ(R10),R0 ; Key buffer size
00FF 8F 50 B1 066F 1535 CMPW R0,#255 ; find out if key size max
04 1B 0674 1536 BLEQU 50$ ; key sz less than 255
50 FF 8F 90 0676 1537 MOVVB #255,R0 ; key size is max.
47 A3 50 90 067A 1538 50$: MOVVB R0,XAB$B_CXRKLEN(R3) ; Key buffer size
51 48 A3 D0 067E 1539 MOVL XAB$L_CXRBUF(R3),R1 ; (r1) = user buffer
61 22 A3 0A A9 0D 0682 1540 PROBEW IRB$B_MODE(R9),XAB$W_CXRBFZ(R3),(R1) ; Probe writability
6F 13 0688 1541 BEQL XABERR ; buffer not okay for use
0200 8F 22 A3 B1 068A 1542 CMPW XAB$W_CXRBFZ(R3),#XAB$C_CXRBLEN ; Make sure long enough
67 1F 0690 1543 BLSSU XABERR ; must be at least that
0692 1544 ;
0692 1545 ; (R0) = key size
0692 1546 ; (R1) = User key buffer address.
0692 1547 ; Save the input registers for the MOVC#. First move keybuffer 1, put in
0692 1548 ; the key size, then after computing the size of keybuffer 6, move in that
0692 1549 ; key. Fill the remaining part of the user key buffer with zeros.
0692 1550 ;
0068 8F BB 0692 1551 PUSHR #*M<R3,R5,R6> ; save registers for MOVC
56 50 D0 0696 1552 MOVL R0,R6 ; save the length there
61 60 81 50 90 0699 1553 MOVVB R0,(R1)+ ; Move keybuf size, inc poin
83 56 28 069C 1554 MOVC3 R0,@IRB$L_KEYBUF(R9),(R1) ; Move keybuffer 1
50 56 05 90 06A1 1555 MOVVB R6,(R3)+ ; Move in key size
50 50 60 A9 C5 06A4 1556 MULL3 #5,R6,R0 ; compute addr of key buffer
63 56 00 60 56 C0 06A8 1557 ADDL2 IRB$L_KEYBUF(R9),R0 ; add in base address
0068 8F BA 06B2 1559 MOVCS R6,(R0),#0,R6,(R3) ; move key, fill with 0's
06B6 1560 POPR #*M<R3,R5,R6> ; restore registers
06B6 1561 EX_NOIRB: ;
05 06B6 1562 RSB ; Return
06B7 1563

```

```

06B7 1565      .SBTTL  RMSOPNCREXAB - ROUTINE TO RESTART OPEN OR CREATE
06B7 1566
06B7 1567      ;++
06B7 1568      ; RMSOPNCREXAB: Process restart operation for Open or Create.
06B7 1569      ;
06B7 1570      ;
06B7 1571      ; if the restart bit is set, use the information from the
06B7 1572      ; XABCXF to fill in the FAB or IFB as needed for restart.
06B7 1573      ;
06B7 1574      ; the entire xab chain is checked for validity by XAB_SCAN
06B7 1575      ;
06B7 1576      ;--
06B7 1577
06B7 1578      OPNCRE_ARGS:
18 3C 20 06B7 1579      .BYTE  XABSC_CXF,XABSC_CXFLFN,XBCSC_OPNCXF
00      06BA 1580      .BYTE  0
06BB 1581
06BB 1582      RMSOPNCREXAB::
5C  F9 AF 9E 06BB 1583      MOVAB  OPNCRE_ARGS,AP      ; Set arg list addr
F96E 31 06BF 1584      BRW    XAB_SCAN      ; Look for context XAB.
06C2 1585
06C2 1586      ;
06C2 1587      ; Check for Restart bit in XABCXF (FAB associated Context XAB).
06C2 1588      ; If restart (set), then verify input and place into FAB or IFB as
06C2 1589      ; needed for restart.
06C2 1590      ;
06C2 1591      XABOPNCXF:
2E 10 A3 00 E1 06C2 1592      BBC    #XABSV_CXFRST,XABSL_CXFCOP(R3),RET ;Check to see if restart
2E 54 18 E2 06C7 1593      BBSS   #XBCSC_OPNCXF,R4,XABERR      ; Check uniqueness of XABCXF
01 1A A3 91 06CB 1594      CMPB   XABSB_CXFVER(R3),#XABSC_CXT_VER1 ; Check version of context block
28 12 06CF 1595      BNEQ   XABERR      ; error if not version 1
02 A8 18 A3 B0 06D1 1596      SSB    #IFBSV_RESTART,(R9)      ; Remember restart in the IFAB
1D 13 06D5 1597      MOVW   XABSW_CXFIFI(R3),FABSW_IFI(R8) ; Move IFI to FAB
06DA 1598      BEQL   XABERR      ; Restart error if IFI = zero.
06DC 1599      ;
06DC 1600      ; Move saved context into FAB for restart operation.
06DC 1601      ;
14 A8 20 A3 B0 06DC 1602      MOVW   XABSW_CXFDEQ(R3),FABSW_DEQ(R8) ; Default extend quantity
16 A8 22 A3 90 06E1 1603      MOVB   XABSB_CXFFAC(R3),FABSB_FAC(R8) ; File access
17 A8 23 A3 90 06E6 1604      MOVB   XABSB_CXFSHR(R3),FABSB_SHR(R8) ; File Sharing bits
48 A8 28 A3 B0 06EB 1605      MOVW   XABSW_CXFGBC(R3),FABSW_GBC(R8) ; Global buffer count
1C A8 2A A3 90 06F0 1606      MOVB   XABSB_CXFRTV(R3),FABSB_RTV(R8) ; Retrieval window
06F5 1607      ;
06F5 1608      RET:  RMSSUC      ; Indicate success
05 06F8 1609      RSB    ; And return
06F9 1610
06F9 1611      XABERR:
06F9 1612      RMSERR  XAB      ; declare XAB error
FA05 31 06FE 1613      BRW    CLNSTK      ; remove return pc from stack and RSB
0701 1614

```

```

0701 1616      .SBTTL RMSCONNAB - ROUTINE TO RESTART CONNECT
0701 1617
0701 1618      :++
0701 1619      : RMSCONNAB:  process restart operation for connect
0701 1620      :
0701 1621      :
0701 1622      : if the restart bit is set, use the information from the
0701 1623      : XABCXR to fill in the RAB or IRB as needed for restart.
0701 1624      :
0701 1625      : the entire xab chain is checked for validity by XAB_SCAN
0701 1626      :
0701 1627      :--
0701 1628
0701 1629 CONNXAB_ARGS:
19 54 21 0701 1630      .BYTE  XAB$C_CXR,XAB$C_CXRLLEN,XBC$C_CONNCXR
0704 1631      .BYTE  0
0705 1632
0705 1633 RMSCONNAB::
5C  F9 AF 9E 0705 1634      MOVAB  CONNXAB_ARGS,AP      ; Set arg list addr
F924 31 0709 1635      BRW    XAB_SCAN      ; Look for context XAB.
070C 1636
070C 1637      :
070C 1638      : Check for Restart bit in XABCXR (RAB associated Context XAB).
070C 1639      : If restart (set), then verify input and place into RAB or IRB as
070C 1640      : needed for restart.
070C 1641      :
070C 1642 XABCONNCXR:
33 10 A3 00 E1 070C 1643      BBC    #XAB$V_CXRRST,XAB$L_CXRCOP(R3),50$ ; Check to see if restart
33 54 19 E2 0711 1644      BBSS   #XBC$C_CONNCXR,R4,60$ ; Check uniqueness of XABCXR
01 1A A3 91 0715 1645      CMPB   XAB$B_CXRVER(R3),#XAB$C_CXT_VER1 ; verify version
2D 12 0719 1646      BNEQ   60$ ; Error on version
071B 1647      SSB    #IRB$V_RESTART,(R9) ; Remember restart in the IRAB
00018001 8F 08 A3 D1 071F 1648      CMPL   XAB$L_CXRSTS(R3),#RMSS_STALL ; Verify that status is okay
1F 13 0727 1649      BEQL   60$ ; Restart error
00018009 8F 08 A3 D1 0729 1650      CMPL   XAB$L_CXRSTS(R3),#RMSS_PENDING ; Make sure Async I/O Finished
15 13 0731 1651      BEQL   60$ ; Restart error
02 A8 18 A3 B0 0733 1652      MOVW   XAB$W_CXRISI(R3),RAB$W_ISI(R8) ; Move ISI to RAB
0E 13 0738 1653      BEQL   60$ ; Restart error if ISI = zero.
37 A8 21 A3 90 073A 1654      MOVB   XAB$B_CXRMBF(R3),RAB$B_MBF(R8) ; Copy back the Multi-block cnt
36 A8 20 A3 90 073F 1655      MOVB   XAB$B_CXRMBF(R3),RAB$B_MBF(R8) ; copy back the multi-buffer cnt
0744 1656
0744 1657 50$:  RMSSUC      ; Indicate success
05 0747 1658      RSB      ; And return
0748 1659
0748 1660 60$:  RMSERR  XAB      ; declare XAB error
F966 31 074D 1661      BRW    CLNSTK ; remove return pc from stack and RSB

```

```

0750 1663 :++
0750 1664 :
0750 1665 : XCONN03::
0750 1666 :
0750 1667 : This routine is used to recreate the ISAM stream NRP context.
0750 1668 :
0750 1669 : At this point, I KNOW that everything about the connect has gone okay.
0750 1670 : This routine will copy all ISAM specific current and next record
0750 1671 : positioning information from the context XAB into the IRAB and key
0750 1672 : buffers allocated on the Connect.
0750 1673 :
0750 1674 : A certain amount of sanity checking is done to verify that the NRP
0750 1675 : information is correct. On any error, return is immediate and no
0750 1676 : context restoration is done. First, a check is made to see if the
0750 1677 : version of the context block and the version number of the attached context
0750 1678 : buffer is VER1. Next, the high order word of each RFA is verified
0750 1679 : to be zero. The key size fields are checked to be the same. The user
0750 1680 : data record VBN/ID pair are zeroed to indicate that there is NO current
0750 1681 : record with respect to the operations such as Delete, and Update. These
0750 1682 : operations imply record locking which does not take place on process restart.
0750 1683 :
0750 1684 :--
0750 1685 :
0750 1686 XCONN03:
01 03 54 19 E5 0750 1687 BBCC #XBC$C_CONNCXR,R4,5$ ; Check uniqueness of XABCXR
008B 31 0754 1688 BRW 60$ ; error - multiple XAB's
01 01 1A A3 91 0757 1689 5$: CMPB XAB$B_CXRVER(R3),#XAB$C_CXT_VER1 ; verify version
03 13 0758 1690 BEQL 8$
0082 31 075D 1691 BRW 60$ ; error on version
01 10 A3 04 1C ED 0760 1692 8$: CMPZV #XAB$V_CXRBVER,#XAB$S_CXRBVER,XAB$L_CXRCOP(R3),#XAB$C_CXB_VER1
07A 12 0766 1693 BNEQ 60$ ; Error - not correct ver
00AC C9 2C A3 D0 0768 1694 MOVL XAB$L_CXRPOS0(R3),IRB$L_POS_VBN(R9) ; Primary record NRP RFA
00BA C9 30 A3 B0 076E 1695 MOVW XAB$W_CXRPOS4(R3),IRB$W_POS_ID(R9) ; Primary record NRP RFA
32 A3 B5 0774 1696 TSTW XAB$W_CXRPOS4+2(R3) ; Check the MBZ field of RFA
69 12 0777 1697 BNEQ 60$ ; Error out if not zero
00A8 C9 34 A3 D0 0779 1698 MOVL XAB$L_CXRCURO(R3),IRB$L_CUR_VBN(R9) ; Primary record RFA
00B8 C9 38 A3 B0 077F 1699 MOVW XAB$W_CXRCUR4(R3),IRB$W_CUR_ID(R9) ; Primary record RFA
3A A3 B5 0785 1700 TSTW XAB$W_CXRCUR4+2(R3) ; check MBZ field of RFA
58 12 0788 1701 BNEQ 60$ ; error out if not zero
00B4 C9 3C A3 D0 078A 1702 MOVL XAB$L_CXRSID0(R3),IRB$L_SIDR_VBN(R9) ; SIDR RFA
00BE C9 40 A3 B0 0790 1703 MOVW XAB$W_CXRSID4(R3),IRB$W_SIDR_ID(R9) ; SIDR RFA
42 A3 B5 0796 1704 TSTW XAB$W_CXRSID4+2(R3) ; Check MBZ field of RFA
47 12 0799 1705 BNEQ 60$ ; error out if not zero
00C0 C9 44 A3 B0 079B 1706 MOVW XAB$W_CXRCNT(R3),IRB$W_CUR_COUNT(R9) ; SIDR array count
00C3 C9 46 A3 90 07A1 1707 MOVW XAB$B_CXRKREF(R3),IRB$B_CUR_KREF(R9) ; Current key of ref.
04 A9 14 A3 D0 07A7 1708 MOVL XAB$L_CXRBKP(R3),IRB$L_BKPBITS(R9) ; copy bookkeeping bits
00B0 C9 D4 07AC 1709 CLRL IRB$L_UDR_VBN(R9) ; make sure User Data RFA is
00BC C9 B4 07B0 1710 CLRW IRB$W_UDR_ID(R9) ; And id too.
07B4 1711 :
07B4 1712 : Probe the user's key buffer and copy the counted ascii key strings
07B4 1713 : (minus the count) into the internal key buffers.
07B4 1714 :
50 00B4 CA 3C 07B4 1715 MOVZWL IFB$W_KBUFSZ(R10),R0 ; Key buffer size
00FF 8F 50 B1 07B9 1716 CMPW R0,#255 ; find out if key size max
04 1B 07BE 1717 BLEQU 50$ ; key sz less than 255
50 FF 8F 90 07C0 1718 MOVW #255,R0 ; key size is max.
47 A3 50 91 07C4 1719 50$: CMPB R0,XAB$B_CXRKLEN(R3) ; compare that key size same

```


| | | | | | | | | | | | |
|----|----------|----|------|----|------|------|--------------|--|-----------------------------------|------------------------------|------------------------------|
| | | | 18 | 12 | 07C8 | 1720 | BNEQ | 60\$ | | : buffer not okay for use | |
| | | | A3 | D0 | 07CA | 1721 | MOVL | XAB\$L_CXRBUF(R3),R1 | | : (r1) = user buffer | |
| 61 | 22 | A3 | 0A | A9 | 07CE | 1722 | PROBER | IRB\$B_MODE(R9),XAB\$W_CXRBFZ(R3),(R1) | | : Probe readability | |
| | | | 08 | 13 | 07D4 | 1723 | BEQL | 58\$ | | : Buffer not okay for use | |
| | 0200 | 8F | 22 | A3 | B1 | 07D6 | 1724 | CMPW | XAB\$W_CXRBFZ(R3),#XAB\$C_CXRBLEN | : Make sure long enough | |
| | | | 0C | 1E | 07DC | 1725 | BGEQU | 70\$ | | : must be at least that | |
| | | | | | 07DE | 1726 | | | | | |
| | | | 00E8 | 8F | BA | 07DE | 1727 | 58\$: POPR | #^M<R3,R5,R6,R7> | : restore registers | |
| | | | | | | 07E2 | 1728 | 60\$: RMSERR | XAB | | |
| | | | F91C | 31 | 07E7 | 1729 | BRW | CLNSTK | | | |
| | | | | | 07EA | 1730 | | | | | |
| | | | | | 07EA | 1731 | | | | | |
| | | | | | 07EA | 1732 | | | | | |
| | | | | | 07EA | 1733 | | | | | |
| | | | | | 07EA | 1734 | | | | | |
| | | | | | 07EA | 1735 | | | | | |
| | | | | | 07EA | 1736 | | | | | |
| | | | 00E8 | 8F | BB | 07EA | 1737 | 70\$: PUSHR | #^M<R3,R5,R6,R7> | : save registers for MOVC | |
| | | | 56 | 50 | D0 | 07EE | 1738 | MOVL | R0,R6 | : save the length there | |
| | | | 50 | 81 | 90 | 07F1 | 1739 | MOVB | (R1)+,R0 | : Move keybuf size, inc poin | |
| | | | 56 | 50 | 91 | 07F4 | 1740 | CMPB | R0,R6 | : compare size fields | |
| | | | | E5 | 12 | 07F7 | 1741 | BNEQ | 58\$ | : No match - invalid buffer | |
| 60 | B9 | | 61 | 50 | 28 | 07F9 | 1742 | MOVCL | R0,(R1),@IRB\$L_KEYBUF(R9) | : Move keybuffer 1 | |
| | | | 56 | 81 | 91 | 07FE | 1743 | CMPB | (R1)+,R6 | : Compare key size fields | |
| | | | | DB | 12 | 0801 | 1744 | BNEQ | 58\$ | : No match - invalid buffer | |
| | | | | | | 0803 | 1745 | | | | |
| | | | | | | 0803 | 1746 | | | | |
| | | | | | | 0803 | 1747 | | | | |
| | | | 50 | 56 | 05 | C5 | 0803 | 1748 | MULL3 | #5,R6,R0 | : compute addr of key buffer |
| | | | 50 | 60 | A9 | C0 | 0807 | 1749 | ADDL2 | IRB\$L_KEYBUF(R9),R0 | : add in base address |
| | | | 60 | 61 | 56 | 28 | 080B | 1750 | MOVCL | R6,(R1),(R0) | : move key |
| | | | | | | 080F | 1751 | | | | |
| | | | | | | 080F | 1752 | | | | |
| | | | | | | 080F | 1753 | | | | |
| | | | 56 | 56 | C0 | 080F | 1754 | ADDL | R6,R6 | : (r6) size of key | |
| | | | 56 | 02 | C0 | 0812 | 1755 | ADDL | #2,R6 | : add the counted ascii part | |
| 56 | 00000200 | | 8F | 56 | C3 | 0815 | 1756 | SUBL3 | R6,#XAB\$C_CXRBLEN,R6 | : subtract from total length | |
| | | | 61 | 56 | 3B | 081D | 1757 | SKPC | #0,R6,(R1) | : make sure they are zero | |
| | | | | BB | 12 | 0821 | 1758 | BNEQ | 58\$ | : not valid buffer.... | |
| | | | 00E8 | 8F | BA | 0823 | 1759 | POPR | #^M<R3,R5,R6,R7> | : restore registers | |
| | | | | | | 0827 | 1760 | | | | |
| | | | | | | 0827 | 1761 | | | | |
| | | | | 05 | 082A | 1762 | 80\$: RMSSUC | RSB | | | |

```

082B 1764      .SBTTL Terminal XAB Routines
082B 1765
082B 1766      :++
082B 1767      : XABTRM routine called if a terminal XAB is found for a $GET/$PUT.
082B 1768      :
082B 1769      : Simply save address of XABTRM in R5 for use by caller of RMSXAB_SCAN.
082B 1770      :--
082B 1771
082B 1772      XABTRM:
07 54 1B E2 082B 1773      BBSS      #XBCSC_GETPUTTRM,R4,ERRIMX TRM
          55 53 D0 082F 1774      : Flag XAB seen; if seen, error.
          082F 1775      MOVL      R3, R5      : Save XAB address in R5 for later use.
          0832 1776      RMSSJUC      : Indicate success.
          05 0835 1777      RSB        : All done.
          0836 1778
          F8C8 31 0836 1779      ERRIMX_TRM:
          0836 1780      BRW      ERRIMX      : Need extended branch.
          0839 1781
          0839 1782      .END

```

RMOXAB
Symbol table

XAB PROCESSING ROUTINES

J 15

16-SEP-1984 00:41:40 VAX/VMS Macro V04-00
5-SEP-1984 16:22:47 [RMS.SRC]RMOXAB.MAR;1

Page 40
(27)

| | | | | | | | |
|-----------------|------------|---|----|------------------|------------|---|----|
| \$\$PSECT EP | = 00000000 | | | FABSC_STMCR | = 00000006 | | |
| \$\$RMSTEST | = 0000001A | | | FABSC_STMLF | = 00000005 | | |
| \$\$RMS_PBUGCHK | = 00000010 | | | FABSL_ALQ | = 00000010 | | |
| \$\$RMS_TBUGCHK | = 00000008 | | | FABSL_FOP | = 00000004 | | |
| \$\$RMS_UMODE | = 00000004 | | | FABSL_STS | = 00000008 | | |
| ACESB_SIZE | = 00000000 | | | FABSL_STV | = 0000000C | | |
| ACESL_ACCESS | = 00000004 | | | FABSL_XAB | = 00000024 | | |
| ATRSC_ACLEVEL | = 00000018 | | | FABSM_CR | = 00000002 | | |
| ATRSC_ACLENGTH | = 00000026 | | | FABSM_FTN | = 00000001 | | |
| ATRSC_ADDAENT | = 0000001F | | | FABSM_PRN | = 00000004 | | |
| ATRSC_ASCDATES | = 0000000D | | | FABSS_ORG | = 00000004 | | |
| ATRSC_BAKDATE | = 00000014 | | | FABSV_CIF | = 00000019 | | |
| ATRSC_CREDATE | = 00000011 | | | FABSV_ORG | = 00000004 | | |
| ATRSC_EXPDATE | = 00000013 | | | FABSW_DEQ | = 00000014 | | |
| ATRSC_FPRO | = 00000016 | | | FABSW_GBC | = 00000048 | | |
| ATRSC_HDR1_ACC | = 0000001E | | | FABSW_IFI | = 00000002 | | |
| ATRSC_JOURNAL | = 0000001D | | | FCHSV_CONTIG | = 00000007 | | |
| ATRSC_READACL | = 00000025 | | | FCHSV_CONTIGB | = 00000005 | | |
| ATRSC_RECATTR | = 00000004 | | | FIBSB_WSIZE | = 00000003 | | |
| ATRSC_REVDATE | = 00000012 | | | FIBSL_ACLCTX | = 00000030 | | |
| ATRSC_STATBLK | = 00000009 | | | FIBSL_ACL_STATUS | = 00000034 | | |
| ATRSC_UCHAR | = 00000003 | | | FIBSL_STATUS | = 00000038 | | |
| ATRSC_UIC | = 00000015 | | | FIBSV_ALCON | = 00000000 | | |
| ATRSC_UIC_RO | = 0000001A | | | FIBSV_ALCONB | = 00000001 | | |
| BKTSC_MAXBKTSIZ | = 0000003F | | | FIBSV_PROPAGATE | = 00000003 | | |
| CHKXAB | 00000041 | R | 01 | FIBSW_EXCTL | = 00000016 | | |
| CLNSTK | 00000106 | R | 01 | FIBSW_VERLIMIT | = 0000002C | | |
| CONNXAB_ARGS | 00000701 | R | 01 | FOP | = 00000020 | | |
| CREXABO_ARGS | 0000037C | R | 01 | FWASL_SBN | = 000001A8 | | |
| ERRAID | 000003C9 | R | 01 | FWASL_UIC | = 00000028 | | |
| ERRBKZ | 000003D0 | R | 01 | FWASQ_AJNL | = 000008D0 | | |
| ERRCOD | 000000FA | R | 01 | FWASQ_ATJNL | = 000008D8 | | |
| ERRIMX | 00000101 | R | 01 | FWASQ_BIJNL | = 000008C8 | | |
| ERRIMX_BR | 000003C6 | R | 01 | FWASQ_FIB | = 00000010 | | |
| ERRIMX_BR1 | 000002C0 | R | 01 | FWASS_AIJNLN | = 00000010 | | |
| ERRIMX_BRJ | 000004A0 | R | 01 | FWASS_ATJNLN | = 00000010 | | |
| ERRIMX_TRM | 00000836 | R | 01 | FWASS_BIJNLN | = 00000010 | | |
| ERRXAB | 000000F0 | R | 01 | FWAST_AIACE | = 000008F4 | | |
| ERRXAB_BRJ | 00000494 | R | 01 | FWAST_AIJNLN | = 000008F8 | | |
| ERR CONFRU | 000000E9 | R | 01 | FWAST_ATACE | = 00000908 | | |
| EXTXABR_ARGS | 0000054E | R | 01 | FWAST_ATJNLN | = 0000090C | | |
| EXTXAB_ARGS | 0000054A | R | 01 | FWAST_BIACE | = 000008E0 | | |
| EX_ISAM | 00000634 | R | 01 | FWAST_BIJNLN | = 000008E4 | | |
| EX_NOIFB | 000005D5 | R | 01 | FWAST_FIBBUF | = 000001F4 | | |
| EX_NOIRB | 000006B6 | R | 01 | FWASW_PRO | = 0000002C | | |
| EX_REL | 00000627 | R | 01 | FWASW_UCHAR | = 00000044 | | |
| EX_SEQ | 00000627 | R | 01 | GOTXAB | 00000084 | R | 01 |
| FABSB_BID | = 00000000 | | | IDXRET | 0000033B | R | 01 |
| FABSB_BKS | = 0000003E | | | IFBSB_BKS | = 0000005E | | |
| FABSB_FAC | = 00000016 | | | IFBSB_FAC | = 00000022 | | |
| FABSB_RFM | = 0000001F | | | IFBSB_JNLFLG | = 000000A0 | | |
| FABSB_RTV | = 0000001C | | | IFBSB_MODE | = 0000000A | | |
| FABSB_SHR | = 00000017 | | | IFBSB_ORGCASE | = 00000023 | | |
| FABSC_BID | = 00000003 | | | IFBSB_SHR | = 0000004E | | |
| FABSC_FIX | = 00000001 | | | IFBSC_IDX | = 00000002 | | |
| FABSC_SEQ | = 00000000 | | | IFBSL_BKPBITS | = 00000004 | | |
| FABSC_STM | = 00000004 | | | IFBSL_FWA_PTR | = 00000038 | | |

RMOXAB
Symbol table

XAB PROCESSING ROUTINES

K 15

16-SEP-1984 00:41:40 VAX/VMS Macro V04-00
5-SEP-1984 16:22:47 [RMS.SRC]RMOXAB.MAR;1

Page 41
(27)

| | | | | | | | |
|-----------------|------------|----|----|------------------|------------|----|----|
| IFBSL_HBK | = 00000070 | | | RMS\$SETALLOC | 00000380 | RG | 01 |
| IFBSV_AI | = 00000003 | | | RMS\$SETEXTRMS | 00000552 | RG | 01 |
| IFBSV_AT | = 00000004 | | | RMS\$SET_XABALL | ***** | X | 01 |
| IFBSV_BI | = 00000002 | | | RMS\$XAB_SCAN | 00000030 | RG | 01 |
| IFBSV_CREATE | = 00000032 | | | RMS\$_AID | = 000183F4 | | |
| IFBSV_NEVER_RU | = 00000005 | | | RMS\$_BKZ | = 00018424 | | |
| IFBSV_ONLY_RU | = 00000000 | | | RMS\$_COD | = 000184AC | | |
| IFBSV_RESTART | = 0000003B | | | RMS\$_ENV | = 00018724 | | |
| IFBSV_RU | = 00000001 | | | RMS\$_IMX | = 0001856C | | |
| IFBSW_DEQ | = 00000062 | | | RMS\$_ORG | = 0001860C | | |
| IFBSW_GBC | = 00000064 | | | RMS\$_PENDING | = 00018009 | | |
| IFBSW_KBUFSZ | = 000000B4 | | | RMS\$_STALL | = 00018001 | | |
| IFBSW_LRL | = 00000052 | | | RMS\$_XAB | = 0001870C | | |
| IFBSW_RTDEQ | = 0000004C | | | RMS\$_XCR | = 00018302 | | |
| IRBSB_CUR_KREF | = 000000C3 | | | RXABTBL | 0000000A | R | 01 |
| IRBSB_MBC | = 00000055 | | | RXABTBLLEN | = 00000002 | | |
| IRBSB_MBF | = 0000005C | | | SETALQ | 00000336 | R | 01 |
| IRBSB_MODE | = 0000000A | | | SETSTV | 000000F5 | R | 01 |
| IRBSL_BKPBITS | = 00000004 | | | STVRSB | 000003D5 | R | 01 |
| IRBSL_CUR_VBN | = 000000A8 | | | SUCXIT | 000000E5 | R | 01 |
| IRBSL_KEYBUF | = 00000060 | | | SYS\$GETTIM | ***** | GX | 01 |
| IRBSL_NRP_VBN | = 00000040 | | | UIC | 00000251 | R | 01 |
| IRBSL_POS_VBN | = 000000AC | | | XAB\$_AID | = 00000017 | | |
| IRBSL_SIDR_VBN | = 000000B4 | | | XAB\$_AIL | = 00000015 | | |
| IRBSL_UDR_VBN | = 000000B0 | | | XAB\$_AIS | = 00000014 | | |
| IRBSV_RESTART | = 0000003B | | | XAB\$_AOP | = 00000008 | | |
| IRBSW_CUR_COUNT | = 000000C0 | | | XAB\$_ATL | = 0000001D | | |
| IRBSW_CUR_ID | = 000000B8 | | | XAB\$_ATR | = 00000009 | | |
| IRBSW_NRP_OFF | = 00000044 | | | XAB\$_ATS | = 0000001C | | |
| IRBSW_POS_ID | = 000000BA | | | XAB\$_BIL | = 0000000D | | |
| IRBSW_SIDR_ID | = 000000BE | | | XAB\$_BIS | = 0000000C | | |
| IRBSW_UDR_ID | = 000000BC | | | XAB\$_BKZ | = 00000016 | | |
| MTACC | 0000025D | R | 01 | XAB\$_BLN | = 00000001 | | |
| NXTXAB | 0000007E | R | 01 | XAB\$_COD | = 00000000 | | |
| OPNCRE_ARGS | 000006B7 | R | 01 | XAB\$_CXFFAC | = 00000022 | | |
| OPNXABT_ARGS | 00000019 | R | 01 | XAB\$_CXFORG | = 00000027 | | |
| OPNXAB_ARGS | 0000000C | R | 01 | XAB\$_CXFRTV | = 0000002A | | |
| PRO | 00000245 | R | 01 | XAB\$_CXF SHR | = 00000023 | | |
| PROT_MODE | 00000269 | R | 01 | XAB\$_CXFVER | = 0000001A | | |
| RAB\$_BID | = 00000000 | | | XAB\$_CXRKLEN | = 00000047 | | |
| RAB\$_MBC | = 00000037 | | | XAB\$_CXRKREF | = 00000046 | | |
| RAB\$_MBF | = 00000036 | | | XAB\$_CXRMBC | = 00000021 | | |
| RAB\$_STS | = 00000008 | | | XAB\$_CXRMBF | = 00000020 | | |
| RAB\$_XAB | = 00000040 | | | XAB\$_CXRVER | = 0000001A | | |
| RAB\$_ISI | = 00000002 | | | XAB\$_MTACC | = 0000000A | | |
| RET | 000006F5 | R | 01 | XAB\$_PROT_MODE | = 00000010 | | |
| REV DATE COUNT | 000001BF | R | 01 | XAB\$_PROT_OPT | = 0000000B | | |
| RMS\$CONNXAB | 00000705 | RG | 01 | XAB\$_RFO | = 00000008 | | |
| RMSERRAID | 000003C9 | RG | 01 | XAB\$_C_ALL | = 00000014 | | |
| RMSERRIMX | 00000101 | RG | 01 | XAB\$_C_ALLLEN | = 00000020 | | |
| RMS\$EXTEND_XAB | ***** | X | 01 | XAB\$_C_CXB_VER1 | = 00000001 | | |
| RMS\$IALLO3 | ***** | X | 01 | XAB\$_C_CXF | = 00000020 | | |
| RMS\$IKEYO3 | ***** | X | 01 | XAB\$_C_CXFLEN | = 0000003C | | |
| RMS\$IUMO3 | ***** | X | 01 | XAB\$_C_CXR | = 00000021 | | |
| RMS\$OPEN_XAB | 0000002C | RG | 01 | XAB\$_C_CXRBLN | = 00000200 | | |
| RMS\$OPEN_XAB1 | 00000026 | RG | 01 | XAB\$_C_CXRLEN | = 00000054 | | |
| RMS\$OPNCREXAB | C000068B | RG | 01 | XAB\$_C_CXT_VER1 | = 00000001 | | |

RMOXAB
Symbol table

XAB PROCESSING ROUTINES

L 15

16-SEP-1984 00:41:40 VAX/VMS Macro V04-00
5-SEP-1984 16:22:47 [RMS.SRC]RMOXAB.MAR;1

Page 42
(27)

| | | | | | |
|-----------------|------------|-----------------|------------|---|----|
| XABSC_DAT | = 00000012 | XABSV_AI | = 00000003 | | |
| XABSC_DATLEN | = 0000002C | XABSV_AT | = 00000004 | | |
| XABSC_DATLEN_V2 | = 00000024 | XABSV_BI | = 00000002 | | |
| XABSC_FHC | = 0000001D | XABSV_CBT | = 00000005 | | |
| XABSC_FHCLEN | = 0000002C | XABSV_CTG | = 00000007 | | |
| XABSC_JNL | = 00000022 | XABSV_CXFRST | = 00000000 | | |
| XABSC_JNLLEN | = 0000003C | XABSV_CXRBVER | = 0000001C | | |
| XABSC_KEY | = 00000015 | XABSV_CXRRST | = 00000000 | | |
| XABSC_PRO | = 00000013 | XABSV_NEVER_RU | = 00000005 | | |
| XABSC_PROLEN_V3 | = 00000010 | XABSV_ONLY_RU | = 00000000 | | |
| XABSC_RDT | = 0000001E | XABSV_PROPAGATE | = 00000000 | | |
| XABSC_RDTLEN | = 00000014 | XABSV_RU | = 00000001 | | |
| XABSC_SUM | = 00000016 | XABSW_ACLLEN | = 0000001E | | |
| XABSC_TRM | = 0000001F | XABSW_ACLSIZ | = 0000001C | | |
| XABSK_PROLEN | = 00000058 | XABSW_CXFDEQ | = 00000020 | | |
| XABSL_ACLBUF | = 00000018 | XABSW_CXFGBC | = 00000028 | | |
| XABSL_ACLCTX | = 00000020 | XABSW_CXFIFI | = 00000018 | | |
| XABSL_ACLSTS | = 00000024 | XABSW_CXRBFZ | = 00000022 | | |
| XABSL_AIA | = 00000018 | XABSW_CXRCNT | = 00000044 | | |
| XABSL_ALQ | = 00000010 | XABSW_CXRRCUR4 | = 00000038 | | |
| XABSL_ATA | = 00000020 | XABSW_CXRISI | = 00000018 | | |
| XABSL_BIA | = 00000010 | XABSW_CXROFF | = 00000028 | | |
| XABSL_CDT0 | = 00000014 | XABSW_CXRPOS4 | = 00000030 | | |
| XABSL_CDT4 | = 00000018 | XABSW_CXRSID4 | = 00000040 | | |
| XABSL_CXFBKP | = 00000014 | XABSW_DEQ | = 00000014 | | |
| XABSL_CXFCOP | = 00000010 | XABSW_JOP | = 00000008 | | |
| XABSL_CXFSTS | = 00000008 | XABSW_LRL | = 0000000A | | |
| XABSL_CXFSTV | = 0000000C | XABSW_PRO | = 00000008 | | |
| XABSL_CXRBKP | = 00000014 | XABSW_RVN | = 00000008 | | |
| XABSL_CXRBUF | = 00000048 | XABSW_VERLIMIT | = 00000026 | | |
| XABSL_CXRCOP | = 00000010 | XABCLSPRO | 000002D0 | R | 01 |
| XABSL_CXRUCURO | = 00000034 | XABCLSPDT | 000001E6 | R | 01 |
| XABSL_CXRPOSO | = 0000002C | XABCONNCXR | 0000070C | R | 01 |
| XABSL_CXRSID0 | = 0000003C | XABCREALL0 | 00000392 | R | 01 |
| XABSL_CXRSTS | = 00000008 | XABCREALL1 | 0000033C | R | 01 |
| XABSL_CXRSTV | = 0000000C | XABCREALL2 | 0000036E | R | 01 |
| XABSL_CXRVBN | = 00000024 | XABCREJNL | 000003DD | R | 01 |
| XABSL_EBK | = 00000010 | XABCREPRO | 00000275 | R | 01 |
| XABSL_HBK | = 0000000C | XABCREPRO1 | 000002C3 | R | 01 |
| XABSL_NXT | = 00000004 | XABDSPALL | 00000302 | R | 01 |
| XABSL_RDT0 | = 0000000C | XABDSPALL1 | 00000321 | R | 01 |
| XABSL_RDT4 | = 00000010 | XABDSPFHC | 0000010A | R | 01 |
| XABSL_SBN | = 00000028 | XABDSPFHC1 | 00000148 | R | 01 |
| XABSL_UIC | = 0000000C | XABENTPRO | 000002F3 | R | 01 |
| XABSM_AI | = 00000008 | XABERR | 000006F9 | R | 01 |
| XABSM_AT | = 00000010 | XABEXTCXF | 00000585 | R | 01 |
| XABSM_BI | = 00000004 | XABEXTCXR | 000005D6 | R | 01 |
| XABSM_CBT | = 00000020 | XABOPNALL | 0000031C | R | 01 |
| XABSM_CTG | = 00000080 | XABOPNCXF | 000006C2 | R | 01 |
| XABSM_NEVER_RU | = 00000020 | XABOPNDAT | 0000017C | R | 01 |
| XABSM_ONLY_RU | = 00000001 | XABOPNFHC | 00000114 | R | 01 |
| XABSM_RU | = 00000002 | XABOPNFHC1 | 0000013D | R | 01 |
| XABSQ_BDT | = 00000024 | XABOPNJNL | 000004A3 | R | 01 |
| XABSQ_CDT | = 00000014 | XABOPNPRO | 000001F9 | R | 01 |
| XABSQ_EDT | = 0000001C | XABOPNPRO1 | 00000233 | R | 01 |
| XABSQ_RDT | = 0000000C | XABOPNRDT | 000001B8 | R | 01 |
| XABSS_CXRBVER | = 00000004 | XABTBL | 00000000 | R | 01 |

```

XABTBLEN      = 0000000A
XABTRM        = 0000082B R    01
XAB_SCAN      = 00000030 R    01
XBCSC_CLSPRO  = 0000000A G
XBCSC_CLSRDT  = 0000000B G
XBCSC_CONNCXR = 00000019 G
XBCSC_CREALLO = 00000004 G
XBCSC_CREALL1 = 00000006 G
XBCSC_CREALL2 = 00000010 G
XBCSC_CREJNL  = 00000015 G
XBCSC_CREPRO  = 00000007 G
XBCSC_CREPRO1 = 0000001E G
XBCSC_DSPALL  = 0000000E G
XBCSC_DSPALL1 = 0000000F G
XBCSC_DSPFHC  = 0000000C G
XBCSC_DSPFHC1 = 0000000D G
XBCSC_ENTPRO  = 0000001D G
XBCSC_EXTALL  = 00000005 G
XBCSC_EXTCXF  = 00000016 G
XBCSC_EXTCXR  = 00000017 G
XBCSC_GETPUTRM = 0000001B G
XBCSC_OPNALL  = 00000009 G
XBCSC_OPNALL3 = 00000012 G
XBCSC_OPNCXF  = 00000018 G
XBCSC_OPNDAT  = 00000003 G
XBCSC_OPNFHC  = 00000000 G
XBCSC_OPNFHC1 = 00000001 G
XBCSC_OPNJNL  = 00000014 G
XBCSC_OPNKEY3 = 00000013 G
XBCSC_OPNPRO  = 00000002 G
XBCSC_OPNPRO1 = 0000001C G
XBCSC_OPNRDT  = 00000008 G
XBCSC_OPNSUM3 = 00000011 G
XBCSC_XCONN03 = 0000001A G
XCONN03      = 00000750 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 |
| RMSRMSO | 00000839 (2105.) | 01 (1.) | PIC USR CON REL GBL NOSHR EXE RD NOWRT NOVEC BYTE |
| SABSS | 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 | 00:00:00.08 | 00:00:00.64 |
| Command processing | 138 | 00:00:00.84 | 00:00:07.52 |
| Pass 1 | 750 | 00:00:23.87 | 00:00:49.22 |
| Symbol table sort | 0 | 00:00:03.25 | 00:00:03.56 |
| Pass 2 | 350 | 00:00:06.01 | 00:00:16.94 |

