

A large, stylized letter 'A' is formed using the characters 'S' and 'Y'. The left and right vertical strokes are composed of 'S' characters, while the central vertical stroke and the diagonal strokes are composed of 'Y' characters. The 'A' is symmetrical and has a bold, blocky appearance.

SYS
VO4

.....

[illegible]


```

0001 0 %TITLE 'SYSRDBRES - Rights database resident system services'
0002 0 MODULE SYSRDBRES (IDENT = 'V04-000') =
0003 1 BEGIN
0004 1
0005 1
0006 1 *****
0007 1 *
0008 1 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0009 1 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0010 1 * ALL RIGHTS RESERVED.
0011 1 *
0012 1 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0013 1 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0014 1 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0015 1 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0016 1 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0017 1 * TRANSFERRED.
0018 1 *
0019 1 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0020 1 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0021 1 * CORPORATION.
0022 1 *
0023 1 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0024 1 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0025 1 *
0026 1 *
0027 1 *****
0028 1
0029 1
0030 1 ++
0031 1 FACILITY: EXECUTIVE, SYSTEM SERVICES
0032 1
0033 1 ABSTRACT:
0034 1
0035 1 This module contains the $ASCTOID, $IDTOASC, and $FINISH_RDB
0036 1 rights database system services. In addition, it contains
0037 1 EX$OPEN_RDB, EX$CLOSE_RDB, and EX$SET_RDIPT which are
0038 1 subroutines used by the rights database system services.
0039 1 This module is contained in the exec, the remaining rights
0040 1 database system services are in a privileged shareable image.
0041 1
0042 1 ENVIRONMENT:
0043 1
0044 1 VAX/VMS native mode, user, supervisor, or exec modes.
0045 1
0046 1 --
0047 1
0048 1 AUTHOR: Andrew C. Goldstein, CREATION DATE: 16-Nov-1982 18:51
0049 1
0050 1 MODIFIED BY:
0051 1
0052 1 V03-011 LMP0218 L. Mark Pilant, 24-Mar-1984 22:56
0053 1 Fix a bug introduced in LMP0202. Make an underscore a legal
0054 1 identifier name character.
0055 1
0056 1 V03-010 LMP0202 L. Mark Pilant, 28-Feb-1984 14:37
0057 1 Add a translation table to upcase the identifier name.
  
```

```
58 0058 1
59 0059 1
60 0060 1
61 0061 1
62 0062 1
63 0063 1
64 0064 1
65 0065 1
66 0066 1
67 0067 1
68 0068 1
69 0069 1
70 0070 1
71 0071 1
72 0072 1
73 0073 1
74 0074 1
75 0075 1
76 0076 1
77 0077 1
78 0078 1
79 0079 1
80 0080 1
81 0081 1
82 0082 1
83 0083 1
84 0084 1
85 0085 1
86 0086 1
87 0087 1
88 0088 1
89 0089 1
90 0090 1
91 0091 1
92 0092 1
93 0093 1
94 0094 1
95 0095 1
96 0096 1
97 0097 1
98 0098 1
99 0099 1
100 0100 1
101 0101 1
102 0102 1
103 0103 1
104 0104 1
105 0105 1
106 0106 1
107 0107 1
108 0108 1
109 0109 1
110 0110 1
111 0111 1
112 0112 1
113 0113 1
114 0114 1

V03-009 RSH0089 R. Scott Hanna 05-Jan-1984
Change EXE$OPEN_RDB to return SSS_IVCHAN when the user
supplies a non-zero context which has an ISI vector entry
of zero.

V03-008 TMK0002 Todd M. Katz 29-Nov-1983
Change the name of EXE$ASCTOID to EXE$$ASCTOID. This change
was required because this system service could no longer be
reached from the change mode dispatcher.

V03-007 RSH0082 R. Scott Hanna 16-Nov-1983
$FINISH_RDB
- Return SSS_INSFARG when the CONTXT argument is
not supplied.
- Return SSS_IVCHAN if CONTXT points to a longword
that contains 0.

$IDTOASC
- Return SSS_INSFARG if CONTXT is not specified with
a wildcard ID.

V03-006 TMK0001 Todd M. Katz 22-Oct-1983
Change the name of EXE$FINISH_RDB to EXE$$FINISH_RDB
and the name of EXE$IDTOASC to EXE$$IDTOASC. This was
required because these system services could no longer be
reached from the change mode dispatcher.

V03-005 RSH0063 R. Scott Hanna 12-Sep-1983
Modify ID value check to return SSS_IVIDENT for an
ID of 0. Change wildcard ID value from 0 to -1 in
$IDTOASC.

V03-004 RSH0049 R. Scott Hanna 28-Jul-1983
Changed EXE$OPEN_RDB to use the new FAB bits
FAB$V_LNM_MODE instead of FAB$B_DSBMSK.

V03-003 RSH0040 R. Scott Hanna 21-Jun-1983
Add EXE$CLOSE_RDB. Provide additional ID name validation.
Open the rights database as a process permanent file when
there is no active image.

V03-002 RSH0029 R. Scott Hanna 25-May-1983
Modify FAB in EXE$OPEN_RDB so that logical name can be
used for the rights database file name.

V03-001 RSH0007 R. Scott Hanna 01-Mar-1983
Add routine EXE$SET_RDIPTTR and modify EXE$OPEN_RDB to
use it.

**
PSECT
CODE = YEXEPAGED (WRITE),
PLIT = YEXEPAGED (WRITE, EXECUTE);
LIBRARY 'SYSS$LIBRARY:LIB.L32';
```



```
115 0115 1
116 0116 1 FORWARD ROUTINE
117 0117 1 EXESSASCTOID,
118 0118 1 EXESSFINISH_RDB,
119 0119 1 EXESSIDTOASC,
120 0120 1 EXESOPEN_RDB,
121 0121 1 EXESCLOSE_RDB : NOVALUE,
122 0122 1 EXESSET_RDIPTR;
123 0123 1
124 0124 1 LINKAGE
125 0125 1 EXE_PROBE_DSC = JSB (REGISTER=1; REGISTER=1, REGISTER=2) :
126 0126 1 NOPRESERVE (3)
127 0127 1 NOTUSED (4,5,6,7,8,9,10,11),
128 0128 1 EXE_VAL_IDNAME = JSB (REGISTER=1; REGISTER=1, REGISTER=2) :
129 0129 1 NOPRESERVE (3)
130 0130 1 NOTUSED (4,5,6,7,8,9,10,11),
131 0131 1 EXE_ALOP1IMAG = JSB (REGISTER=1; REGISTER=1, REGISTER=2) :
132 0132 1 NOPRESERVE (3);
133 0133 1
134 0134 1 EXTERNAL ROUTINE
135 0135 1 EXESPROBEW_DSC : EXE_PROBE_DSC ADDRESSING_MODE (ABSOLUTE),
136 0136 1 EXESALOP1IMAG : EXE_ALOP1IMAG ADDRESSING_MODE (ABSOLUTE),
137 0137 1 EXESVAL_IDNAME : EXE_VAL_IDNAME ADDRESSING_MODE (ABSOLUTE),
138 0138 1 SYSSCMKRNL : ADDRESSING_MODE (ABSOLUTE);
139 0139 1
140 0140 1 EXTERNAL
141 0141 1 CTL$GL_RDIPTR : REF $BBLOCK ADDRESSING_MODE (ABSOLUTE),
142 0142 1 CTL$GL_IMGHDRBF : LONG ADDRESSING_MODE (ABSOLUTE);
143 0143 1
144 0144 1 BUILTIN
145 0145 1 PROBEW;
146 0146 1
147 0147 1
148 0148 1 LITERAL
149 0149 1 UIC$M_ID_FORM_FLAG = 1^31; ! mask for id form of identifier
```

```
151 0150 1 GLOBAL ROUTINE EXESSASCTOID (NAME, ID, ATTRIB) =
152 0151 1
153 0152 1 ++
154 0153 1
155 0154 1 FUNCTIONAL DESCRIPTION:
156 0155 1
157 0156 1 This routine searches the rights database for the specified
158 0157 1 name, and returns the matching identifier code and attributes.
159 0158 1
160 0159 1 CALLING SEQUENCE:
161 0160 1 SYSS$ASCTOID (NAME, ID, ATTRIB)
162 0161 1
163 0162 1 INPUT PARAMETERS:
164 0163 1 NAME: address of the identifier name character
165 0164 1 string descriptor to be converted
166 0165 1
167 0166 1 IMPLICIT INPUTS:
168 0167 1 NONE
169 0168 1
170 0169 1 OUTPUT PARAMETERS:
171 0170 1 ID: (optional) address to return the identifier longword
172 0171 1 ATTRIB: (optional) address to return the attributes longword
173 0172 1
174 0173 1 IMPLICIT OUTPUTS:
175 0174 1 NONE
176 0175 1
177 0176 1 ROUTINE VALUE:
178 0177 1 Status of operation
179 0178 1
180 0179 1 SIDE EFFECTS:
181 0180 1 NONE
182 0181 1
183 0182 1 --
184 0183 1
185 0184 2 BEGIN
186 0185 2
187 0186 2 LOCAL
188 0187 2 LOC_NAME : REF VECTOR, ! local copy of NAME
189 0188 2 LENGTH : LONG, ! output from EXESVAL_IDNAME
190 0189 2 ADDRESS : LONG, ! output from EXESVAL_IDNAME
191 0190 2 LOC_ID : LONG, ! local copy of ID
192 0191 2 LOC_ATTRIB : LONG, ! local copy of ATTRIB
193 0192 2 STATUS : LONG, ! general status value
194 0193 2 CLOSE : LONG, ! call EXESCLOSE_RDB flag
195 0194 2 RAB : $RAB_DECL, ! RAB for file I70
196 0195 2 REC_BUFFER : $BBLOCK [KGB$K_IDENT_RECORD], ! record buffer to read records
197 0196 2
198 0197 2 NAME_BUFFER : $BBLOCK [KGB$S_NAME]; ! name key buffer
199 0198 2
200 0199 2
201 0200 2 ! Define a global translation table to do the upcasing of the identifier
202 0201 2 name. This should parallel the validation table for the exec routine
203 0202 2 ! EXESVAL_IDNAME in the module [SYS.SRC]EXSUBROUT.MAR.
204 0203 2
205 0204 2 GLOBAL BIND
206 0205 2 EXEST_ID_UPCASE = CH$TRANSTABLE (
207 0206 2 REP 36 OF (%C' '), ! Null - #
```



```
208 0207 2
209 0208
210 0209
211 0210
212 0211
213 0212
214 0213
215 0214
216 0215
217 0216
218 0217
219 0218
220 0219
221 0220
222 0221
223 0222
224 0223
225 0224
226 0225
227 0226
228 0227
229 0228
230 0229
231 0230
232 0231
233 0232
234 0233
235 0234
236 0235
237 0236
238 0237
239 0238
240 0239
241 0240
242 0241
243 0242
244 0243
245 0244
246 0245
247 0246
248 0247
249 0248
250 0249
251 0250
252 0251
253 0252
254 0253
255 0254
256 0255
257 0256
258 0257
259 0258
260 0259
261 0260
262 0261
263 0262
264 0263

XC'S'
REP 11 OF (XC' ')
'0','1','2','3','4','5','6','7','8','9',
REP 7 OF (XC' ')
'A','B','C','D','E','F','G','H','I','J','K','L','M',
'N','O','P','Q','R','S','T','U','V','W','X','Y','Z',
REP 4 OF (XC' '),
REP 1 OF (XC' ')
'A','B','C','D','E','F','G','H','I','J','K','L','M',
'N','O','P','Q','R','S','T','U','V','W','X','Y','Z',
REP 5 OF (XC' ')
REP 128 OF (XC' ');
! %X80 - %XFF

LABEL
RDB_OPEN;
! rights database is open in this block

! Validate parameters

LOC_NAME = .NAME;
STATUS = EX$VAL_IDNAME( .LOC_NAME; LENGTH, ADDRESS);
IF NOT .STATUS THEN RETURN .STATUS;
CH$TRANSLATE (EX$T_ID_UPCASE, .LENGTH, .ADDRESS, ' ', KGB$S_NAME, NAME_BUFFER);

LOC_ID = .ID;
IF .LOC_ID NEQU 0 AND NOT PROBEW (%REF(0), %REF(4), .LOC_ID)
THEN
RETURN SSS_ACCVIO;

LOC_ATTRIB = .ATTRIB;
IF .LOC_ATTRIB NEQU 0 AND NOT PROBEW (%REF(0), %REF(4), .LOC_ATTRIB)
THEN
RETURN SSS_ACCVIO;

! Open the rights database for reading.

$RAB_INIT (RAB = RAB,
RAC = KEY,
KRF = 2,
KBF = NAME_BUFFER,
KSZ = KGB$S_NAME,
ROP = (WAT, -NLK),
USZ = KGB$K_IDENT_RECORD,
UBF = REC_BUFFER
);
STATUS = EX$OPEN RDB (0, 0, RAB[RAB$W_ISI], CLOSE);
IF NOT .STATUS THEN RETURN .STATUS;

RDB_OPEN:
BEGIN
! Read the record and return the data items.
!
STATUS = $GET (RAB = RAB);
```

```
: 265      0264      3      IF .STATUS EQLU RMSS_RNF THEN STATUS = SS$_NOSUCHID;  
: 266      0265      3      IF NOT .STATUS THEN LEAVE RDB_OPEN;  
: 267      0266      3  
: 268      0267      3      IF .LOC_ID NEQU 0 THEN .LOC_ID = .REC_BUFFER[KGB$$_IDENTIFIER];  
: 269      0268      3      IF .LOC_ATTRIB NEQU 0 THEN .LOC_ATTRIB = .REC_BUFFER[KGB$$_ATTRIBUTES];  
: 270      0269      3  
: 271      0270      3      STATUS = SS$_NORMAL;  
: 272      0271      3      END;  
: 273      0272      3  
: 274      0273      3      ! Close the rights database if there is no image  
: 275      0274      3      !  
: 276      0275      3  
: 277      0276      3      IF .CLOSE THEN EXE$CLOSE_RDB();  
: 278      0277      3      RETURN .STATUS  
: 279      0278      1      END;  
                                     ! End of routine EXE$$_ASCTOID
```

```
.TITLE  SYSRDBRES SYSRDBRES - Rights database resident  
                                system ser
```

```
.IDENT  \V04-000\
```

```
.PSECT  YEXEPAGED,2
```

```
20# 00000 P.AAA: .BYTE 32[36]  
24 00024 .BYTE 36  
20# 00025 .BYTE 32[11]  
30 00030 .ASCII \0\  
31 00031 .ASCII \1\  
32 00032 .ASCII \2\  
33 00033 .ASCII \3\  
34 00034 .ASCII \4\  
35 00035 .ASCII \5\  
36 00036 .ASCII \6\  
37 00037 .ASCII \7\  
38 00038 .ASCII \8\  
39 00039 .ASCII \9\  
20# 0003A .BYTE 32[7]  
41 00041 .ASCII \A\  
42 00042 .ASCII \B\  
43 00043 .ASCII \C\  
44 00044 .ASCII \D\  
45 00045 .ASCII \E\  
46 00046 .ASCII \F\  
47 00047 .ASCII \G\  
48 00048 .ASCII \H\  
49 00049 .ASCII \I\  
4A 0004A .ASCII \J\  
4B 0004B .ASCII \K\  
4C 0004C .ASCII \L\  
4D 0004D .ASCII \M\  
4E 0004E .ASCII \N\  
4F 0004F .ASCII \O\  
50 00050 .ASCII \P\  
51 00051 .ASCII \Q\  
52 00052 .ASCII \R\  
53 00053 .ASCII \S\  
54 00054 .ASCII \T\
```


55	00055	.ASCII	\U\
56	00056	.ASCII	\V\
57	00057	.ASCII	\W\
58	00058	.ASCII	\X\
59	00059	.ASCII	\Y\
5A	0005A	.ASCII	\Z\
20#	0005B	.BYTE	32[4]
5F	0005F	.ASCII	\
20	00060	.BYTE	32
41	00061	.ASCII	\A\
42	00062	.ASCII	\B\
43	00063	.ASCII	\C\
44	00064	.ASCII	\D\
45	00065	.ASCII	\E\
46	00066	.ASCII	\F\
47	00067	.ASCII	\G\
48	00068	.ASCII	\H\
49	00069	.ASCII	\I\
4A	0006A	.ASCII	\J\
4B	0006B	.ASCII	\K\
4C	0006C	.ASCII	\L\
4D	0006D	.ASCII	\M\
4E	0006E	.ASCII	\N\
4F	0006F	.ASCII	\O\
50	00070	.ASCII	\P\
51	00071	.ASCII	\Q\
52	00072	.ASCII	\R\
53	00073	.ASCII	\S\
54	00074	.ASCII	\T\
55	00075	.ASCII	\U\
56	00076	.ASCII	\V\
57	00077	.ASCII	\W\
58	00078	.ASCII	\X\
59	00079	.ASCII	\Y\
5A	0007A	.ASCII	\Z\
20#	0007B	.BYTE	32[5]
20#	00080	.BYTE	32[128]

EXEST_ID_UPCASE== P.AAA
.EXTRN EXESPROBEW DSC, EXESALOP1IMAG
.EXTRN EXESVAL_IDNAME, SYSSCMKRN
.EXTRN CTL\$GL_RDIPT, CTL\$GL_IMGHDRBF
.EXTRN SYSSGET

07FC 00000

.ENTRY	EXES\$ASCTOID, Save R2,R3,R4,R5,R6,R7,R8,R9,-;	0150
MOVAB	R10	
MOVL	-152(SP), SP	
JSB	NAME, LOC_NAME	0227
MOVL	@#EXESVAL_IDNAME	0228
BLBC	R0, STATUS	
MOVTC	STATUS, 4\$	0229
	LENGTH, (ADDRESS), #32, EXEST_ID_UPCASE, -	0230
	#32, NAME_BUFFER	
MOVL	ID, LOC_ID	0232
CLRL	R10	0233
TSTL	LOC_ID	
BEQL	1\$	

FEE2 CF

20

04

5E	FF68	CE	9E	00002
51	04	AC	D0	00007
	00000000G	9F	16	0000B
56		50	D0	00011
70		56	E9	00014
62		51	2E	00017
AE		20		0001E
58	08	AC	D0	00021
		5A	D4	00025
		58	D5	00027
		08	13	00029

68	04	5A	D6	0002B	INCL	R10		
		00	0D	0002D	PROBEW	#0, #4, (LOC_ID)		
	57	12	13	00031	BEQL	2\$		
		AC	D0	00033	1\$:	MOVL	ATTRIB, LOC_ATTRIB	0237
		59	D4	00037		CLRL	R9	0238
		57	D5	00039		TSTL	LOC_ATTRIB	
		0C	13	0003B		BEQL	3\$	
		59	D6	0003D		INCL	R9	
67	04	00	0D	0003F		PROBEW	#0, #4, (LOC_ATTRIB)	
		04	12	00043		BNEQ	3\$	
	50	0C	D0	00045	2\$:	MOVL	#12, R0	0240
			04	00048		RET		
0044	8F	00	2C	00049	3\$:	MOVC5	#0, (SP), #0, #68, \$RMS_PTR	0253
		54	AE	00050				
54	AE	4401	8F	B0	00052	MOVW	#17409, \$RMS_PTR	
58	AE	00120000	8F	D0	00058	MOVL	#1179648, \$RMS_PTR+4	
72	AE		01	90	00060	MOVB	#1, \$RMS_PTR+30	
74	AE		30	B0	00064	MOVW	#48, \$RMS_PTR+32	
78	AE	24	AE	9E	00068	MOVAB	REC_BUFFER, \$RMS_PTR+36	
EC	AD	04	AE	9E	0006D	MOVAB	NAME_BUFFER, \$RMS_PTR+48	
FO	AD	0220	8F	B0	00072	MOVW	#544, \$RMS_PTR+52	
		5A	5E	DD	00078	PUSHL	SP	0254
			AE	9F	0007A	PUSHAB	RAB+2	
			7E	7C	0007D	CLRQ	-(SP)	
0000V	CF		04	FB	0007F	CALLS	#4, EXESOPEN_RDB	
	56		50	D0	00084	MOVL	R0, STATUS	
	37		56	E9	00087	4\$:	BLBC	STATUS, 9\$
		54	AE	9F	0008A		PUSHAB	RAB
00000000G	00		01	FB	0008D	CALLS	#1, SYSSGET	
	56		50	D0	00094	MOVL	R0, STATUS	
000182B2	8F		56	D1	00097	CMPL	STATUS, #98994	0264
			05	12	0009E		BNEQ	5\$
	56	21EC	8F	3C	000A0	MOVZWL	#8684, STATUS	
	11		56	E9	000A5	5\$:	BLBC	STATUS, 8\$
	04		5A	E9	000A8		BLBC	R10, 6\$
	68	24	AE	D0	000AB	MOVL	REC_BUFFER, (LOC_ID)	0267
	04		59	E9	000AF	6\$:	BLBC	R9, 7\$
	67	28	AE	D0	000B2	MOVL	REC_BUFFER+4, (LOC_ATTRIB)	0268
	56		01	D0	000B6	7\$:	MOVL	#1, STATUS
	05		6E	E9	000B9	8\$:	BLBC	CLOSE, 9\$
0000V	CF		00	FB	000BC	CALLS	#0, EXESCLOSE_RDB	0270
	50		56	D0	000C1	9\$:	MOVL	STATUS, R0
			04	000C4		RET		0276
								0277
								0278

; Routine Size: 197 bytes, Routine Base: YEXEPAGED + 0100


```
281 0279 1 GLOBAL ROUTINE EXE$$FINISH_RDB (CONXT) =
282 0280 1
283 0281 1 ++
284 0282 1
285 0283 1 FUNCTIONAL DESCRIPTION:
286 0284 1
287 0285 1 This routine cleans up the RMS stream associated with the
288 0286 1 specified context.
289 0287 1
290 0288 1 CALLING SEQUENCE:
291 0289 1 SY$$FINISH_RDB (CONXT)
292 0290 1
293 0291 1 INPUT PARAMETERS:
294 0292 1 CONXT: address of a longword containing the record stream
295 0293 1 context.
296 0294 1
297 0295 1 IMPLICIT INPUTS:
298 0296 1 NONE
299 0297 1
300 0298 1 OUTPUT PARAMETERS:
301 0299 1 NONE
302 0300 1
303 0301 1 IMPLICIT OUTPUTS:
304 0302 1 NONE
305 0303 1
306 0304 1 ROUTINE VALUE:
307 0305 1 Status of operation
308 0306 1
309 0307 1 SIDE EFFECTS:
310 0308 1 Stream disconnected
311 0309 1
312 0310 1 --
313 0311 1
314 0312 2 BEGIN
315 0313 2
316 0314 2 LOCAL
317 0315 2 LOC_CONXT : LONG, : local copy of CONXT
318 0316 2 ISI_INDEX : LONG, : Index into ISI vector
319 0317 2 RAB : $RAB_DECL; : RAB for stream to close
320 0318 2
321 0319 2 : Validate parameters
322 0320 2 :
323 0321 2
324 0322 2 IF .CTL$GL_RDIPTER EQLU 0 THEN RETURN SS$_IVCHAN;
325 0323 2 LOC_CONXT = .CONXT;
326 0324 2 IF .LOC_CONXT EQLU 0 THEN RETURN SS$_INSFARG;
327 0325 2 IF NOT PROBEW (%REF(0), %REF(4), .LOC_CONXT) THEN RETURN SS$_ACCVIO;
328 0326 2
329 0327 2 : Find the stream open on the specified context.
330 0328 2 :
331 0329 2
332 0330 2 ISI_INDEX = .LOC_CONXT;
333 0331 2 IF (.ISI_INDEX EQLU 0) OR (.ISI_INDEX GTRU RDISK_ISI_MAX) THEN RETURN SS$_IVCHAN;
334 0332 2 .LOC_CONXT = 0;
335 0333 2 IF .VECTOR [CTL$GL_RDIPTER[RDISK_ISI_VEC], .ISI_INDEX] EQLU 0
336 0334 2 THEN
337 0335 2 RETURN SS$_NORMAL;
```

```

: 338
: 339
: 340
: 341
: 342
: 343
: 344
: 345
: 346
: 347
: 348

0336 2
0337 2 ! Set up a RAB and disconnect it.
0338 2
0339 2
0340 2 $RAB INIT (RAB = RAB);
0341 2 RAB[RAB$W_ISI] = .VECTOR [CTL$GL_RDIPTR[RDISL_ISI_VEC], .ISI_INDEX];
0342 2 VECTOR [CTL$GL_RDIPTR[RDISL_ISI_VEC], .ISI_INDEX] = 0;
0343 2 $DISCONNECT (RAB = RAB);
0344 2 RETURN SSS_NORMAL;
0345 2
0346 1 END;

! End of routine EXE$$FINISH_RDB
```

.EXTRN SYSS\$DISCONNECT

```

.ENTRY EXE$$FINISH_RDB, Save R2,R3,R4,R5,R6,R7,R8 : 0279
MOVAB -68(SP), SP
MOVL @#CTL$GL_RDIPTR, R7 : 0322
BEQL 3$
MOVL CONTXT, LOC_CONTXT : 0323
BNEQ 1$ : 0324
MOVZWL #276, R0
RET
PROBEW #0, #4, (LOC_CONTXT) : 0325
BNEQ 2$
MOVL #12, R0
RET
MOVL (LOC_CONTXT), ISI_INDEX : 0330
BEQL 3$ : 0331
CML ISI_INDEX, #10
BLEQU 4$
MOVZWL #316, R0
RET
CLRL (LOC_CONTXT) : 0332
MOVL 12(R7)[ISI_INDEX], R8 : 0333
BEQL 5$
MOVCS #0, (SP), #0, #68, $RMS_PTR : 0340
MOVW #17409, $RMS_PTR
MOVW R8, RAB+2 : 0341
CLRL 12(R7)[ISI_INDEX] : 0342
PUSHL SP : 0343
CALLS #1, SYSS$DISCONNECT
MOVL #1, R0 : 0344
RET : 0346
```

; Routine Size: 96 bytes, Routine Base: YEXEPAGED + 01C5


```
350 0347 1 GLOBAL ROUTINE EXE$IDTOASC (ID, NAMLEN, NAMBUF, RESID, ATTRIB, CONTXT) =
351 0348 1
352 0349 1 ++
353 0350 1
354 0351 1 FUNCTIONAL DESCRIPTION:
355 0352 1
356 0353 1 This routine searches the rights database for the specified
357 0354 1 identifier, and returns the matching name and attributes.
358 0355 1 If a wildcard identifier and a context is provided, the
359 0356 1 routine returns all identifiers in lexical order.
360 0357 1
361 0358 1 CALLING SEQUENCE:
362 0359 1 SY$IDTOASC (ID, NAMLEN, NAMBUF, RESID, ATTRIB, CONTXT)
363 0360 1
364 0361 1 INPUT PARAMETERS:
365 0362 1 ID: identifier longword to convert, or -1 to
366 0363 1 find all identifiers (wildcarding)
367 0364 1 CONTXT: (optional) address of a longword containing the record stream
368 0365 1 context. initially should be zero, value output
369 0366 1 on first call, value input on subsequent calls.
370 0367 1
371 0368 1 IMPLICIT INPUTS:
372 0369 1 NONE
373 0370 1
374 0371 1 OUTPUT PARAMETERS:
375 0372 1 NAMLEN: (optional) address of a word to store the length of the
376 0373 1 identifier name
377 0374 1 NAMBUF: (optional) address of a character string descriptor that
378 0375 1 describes the buffer to return the identifier name
379 0376 1 RESID: (optional) address of a longword to return the id found
380 0377 1 while wildcarding
381 0378 1 ATTRIB: (optional) address to return the attributes longword
382 0379 1
383 0380 1 IMPLICIT OUTPUTS:
384 0381 1 NONE
385 0382 1
386 0383 1 ROUTINE VALUE:
387 0384 1 Status of operation
388 0385 1
389 0386 1 SIDE EFFECTS:
390 0387 1 NONE
391 0388 1
392 0389 1 --
393 0390 1
394 0391 2 BEGIN
395 0392 2
396 0393 2 LOCAL
397 0394 2 LOC_ID : LONG, : local copy of ID
398 0395 2 LOC_NAMLEN : LONG, : local copy of NAMLEN
399 0396 2 LOC_NAMBUF : LONG, : local copy of NAMBUF
400 0397 2 LENGTH : LONG, : output from EXE$PROBEW_DSC
401 0398 2 ADDRESS : LONG, : output from EXE$PROBEW_DSC
402 0399 2 LOC_RESID : LONG, : local copy of RESID
403 0400 2 LOC_ATTRIB : LONG, : local copy of ATTRIB
404 0401 2 LOC_CONTXT : LONG, : local copy of CONTXT
405 0402 2 STATUS : LONG, : general status value
406 0403 2 P : LONG, : string pointer
```

```
407      0404      2      CLOSE      : LONG,      ! call EX$CLOSE_RDB flag
408      0405      2      RAB      : $RAB_DECL,      ! RAB for file 170
409      0406      2      REC_BUFFER      : $BBLOCK [KGB$K_IDENT_RECORD];
410      0407      2      ! record buffer to read records
411      0408
412      0409      2      LABEL
413      0410      2      RDB_OPEN;      ! rights database is open in this block
414      0411      2
415      0412      2      ! Validate parameters
416      0413      2      !
417      0414      2
418      0415      2      LOC_ID = .ID;
419      0416      2      IF .LOC_ID EQL -1
420      0417      2      THEN
421      0418      2      BEGIN
422      0419      2      IF .CONXT EQL 0 THEN RETURN SS$_INSFARG;
423      0420      2      LOC_ID = 0;
424      0421      2      END
425      0422      2      ELSE
426      0423      2      BEGIN
427      0424      2      IF (.LOC_ID AND UIC$M_ID_FORM_FLAG) NEQU 0
428      0425      2      THEN
429      0426      2      (IF (.LOC_ID GTRU UIC$K_LAST_ID) THEN RETURN SS$_IVIDENT)
430      0427      2      ELSE
431      0428      2      (IF (.LOC_ID GTRU UIC$K_MAX_UIC) OR (.LOC_ID EQL 0) THEN RETURN SS$_IVIDENT)
432      0429      2      END;
433      0430      2
434      0431      2      LOC_NAMLEN = .NAMLEN;
435      0432      2      IF .LOC_NAMLEN NEQU 0 AND NOT PROBEW (%REF(0), %REF(2), .LOC_NAMLEN)
436      0433      2      THEN
437      0434      2      RETURN SS$_ACCVIO;
438      0435      2
439      0436      2      LOC_NAMBUF = .NAMBUF;
440      0437      2      IF .LOC_NAMBUF NEQU 0
441      0438      2      THEN
442      0439      2      BEGIN
443      0440      2      IF NOT (STATUS = EX$PROBEW_DSC (.LOC_NAMBUF; LENGTH, ADDRESS))
444      0441      2      THEN
445      0442      2      RETURN .STATUS
446      0443      2      ELSE
447      0444      2      LENGTH = .LENGTH AND %X'FFFF'
448      0445      2      END;
449      0446      2
450      0447      2      LOC_RESID = .RESID;
451      0448      2      IF .LOC_RESID NEQU 0 AND NOT PROBEW (%REF(0), %REF(4), .LOC_RESID)
452      0449      2      THEN
453      0450      2      RETURN SS$_ACCVIO;
454      0451      2
455      0452      2      LOC_ATTRIB = .ATTRIB;
456      0453      2      IF .LOC_ATTRIB NEQU 0 AND NOT PROBEW (%REF(0), %REF(4), .LOC_ATTRIB)
457      0454      2      THEN
458      0455      2      RETURN SS$_ACCVIO;
459      0456      2
460      0457      2      LOC_CONXT = .CONXT;
461      0458      2      IF .LOC_CONXT NEQU 0 AND NOT PROBEW (%REF(0), %REF(4), .LOC_CONXT)
462      0459      2      THEN
463      0460      2      RETURN SS$_ACCVIO;
```



```

464 0461 2
465 0462 2
466 0463 2 ! Open the rights database for reading. Note that the open routine
467 0464 2 ! specifies KRF=2 so the stream will be correctly positioned for
468 0465 2 ! wildcarding by name if requested. If an explicit identifier is given,
469 0466 2 ! set up to read by identifier key.
470 0467 2
471 0468 2
472 P 0469 2 SRAB_INIT (RAB = RAB,
473 P 0470 2 RAC = SEQ,
474 P 0471 2 KRF = 2,
475 P 0472 2 ROP = (WAT, NLK),
476 P 0473 2 USZ = KGB$K_IDENT_RECORD,
477 P 0474 2 UBF = REC_BUFFER
478 0475 2 );
479 0476 2 IF .LOC_ID NEQU 0
480 0477 2 THEN
481 0478 2 BEGIN
482 0479 2 RAB[RAB$B_RAC] = RAB$C_KEY;
483 0480 2 RAB[RAB$B_KRF] = 0;
484 0481 2 RAB[RAB$B_KSZ] = 4;
485 0482 2 RAB[RAB$L_KBF] = LOC_ID;
486 0483 2 END;
487 0484 2
488 0485 2 STATUS = EX$OPEN RDB (.LOC_CONXT, 0, RAB[RAB$W_ISI], CLOSE);
489 0486 2 IF NOT .STATUS THEN RETURN .STATUS;
490 0487 2
491 0488 2 RDB_OPEN:
492 0489 2 BEGIN
493 0490 2 ! Read the record and return the data items. If wildcarding, we skip the
494 0491 2 ! maintenance record.
495 0492 2 !
496 0493 2
497 0494 2 WHILE 1 DO
498 0495 2 BEGIN
499 0496 2 STATUS = $GET (RAB = RAB);
500 0497 2 IF .STATUS EQLU RMSS$_RNF OR .STATUS EQLU RMSS$_EOF
501 0498 2 THEN
502 0499 2 STATUS = SS$ NOSUCHID;
503 0500 2 IF .STATUS EQLU RMSS$_RTB THEN STATUS = SS$_NORMAL;
504 0501 2 IF NOT .STATUS
505 0502 2 THEN
506 0503 2 BEGIN
507 0504 2 EX$FINISH RDB(.LOC_CONXT);
508 0505 2 LEAVE RDB_OPEN;
509 0506 2 END;
510 0507 2
511 0508 2 IF .LOC_ID NEQU 0 OR .REC_BUFFER[KGB$L_IDENTIFIER] NEQU 0
512 0509 2 THEN
513 0510 2 EXITLOOP;
514 0511 2
515 0512 2 END;
516 0513 2
517 0514 2 ! Return the requested data. Find the trailing spaces in the name
518 0515 2 ! string so it can be returned as a variable length string.
519 0516 2
520 0517 2
```

```

521 0518 3 P = CH$FIND_CH (KGB$S_NAME, REC_BUFFER[KGB$T_NAME], ' ');
522 0519 3 IF CH$FAIL T.P) THEN P = REC_BUFFER[KGB$T_NAME] + KGB$S_NAME;
523 0520 3
524 0521 3 IF .LOC_NAMLEN NEQU 0 THEN (.LOC_NAMLEN)<0,16> = .P - REC_BUFFER[KGB$T_NAME];
525 0522 3 IF .LOC_NAMBUF NEQU 0
526 0523 3 THEN
527 0524 3 CH$COPY (KGB$S_NAME, REC_BUFFER[KGB$T_NAME], ' ', .LENGTH, .ADDRESS);
528 0525 3
529 0526 3 IF .LOC_RESID NEQU 0 THEN .LOC_RESID = .REC_BUFFER[KGB$T_IDENTIFIER];
530 0527 3 IF .LOC_ATTRIB NEQU 0 THEN .LOC_ATTRIB = .REC_BUFFER[KGB$T_ATTRIBUTES];
531 0528 3
532 0529 3 STATUS = SSS_NORMAL;
533 0530 3 END;
534 0531 3
535 0532 3 ! Close the rights database if there is no image
536 0533 3 !
537 0534 3
538 0535 3 IF .CLOSE THEN EXE$CLOSE_RDB();
539 0536 3 RETURN .STATUS
540 0537 1 END;

```

! End of routine EXE\$\$IDTOASC

			OFFC 00000		.ENTRY	EXE\$\$IDTOASC, Save R2,R3,R4,R5,R6,R7,R8,R9,-	
						R10,R11	0347
					MOVAB	-144(SP), SP	
14	5E	FF70	CE 9E 00002		MOVL	ID, LOC_ID	0415
	AE	04	AC D0 00007		MOVL	LOC_ID, -R0	0416
FFFFFFF	50	14	AE D0 0000C		CMPL	R0, #-1	
	8F		50 D1 00010		BNEQ	2\$	
			10 12 00017		TSTL	CONXT	0419
		18	AC D5 00019		BNEQ	1\$	
			06 12 0001C		MOVZWL	#276, R0	
	50	0114	8F 3C 0001E		RET		
			04 00023		CLRL	LOC_ID	0420
		14	AE D4 00024 1\$:		BRB	5\$	0416
			22 11 00027		TSTL	R0	0424
			50 D5 00029 2\$:		BGEQ	3\$	
8FFFFFFF	8F		0B 18 0002B		CMPL	R0, #-1879048193	0426
			50 D1 0002D		BLEQU	5\$	
			15 1B 00034		BRB	4\$	
3FFFFFFF	8F		0D 11 00036		CMPL	R0, #1073741823	0428
			50 D1 00038 3\$:		BGTRU	4\$	
			04 1A 0003F		TSTL	R0	
			50 D5 00041		BNEQ	5\$	
			06 12 00043		MOVZWL	#8740, R0	
	50	2224	8F 3C 00045 4\$:		RET		
			04 0004A		MOVL	NAMLEN, LOC_NAMLEN	0431
	58	08	AC D0 0004B 5\$:		CLRL	8(SP)	0432
		08	AE D4 0004F		TSTL	LOC_NAMLEN	
			58 D5 00052		BEQL	6\$	
			09 13 00054		INCL	8(SP)	
68	02	08	AE D6 00056		PROBEW	#0, #2, (LOC_NAMLEN)	
			00 0D 00059		BEQL	11\$	
			5A 13 0005D		MOVL	NAMBUF, LOC_NAMBUF	0436
	51	0C	AC D0 0005F 6\$:				

			04	AE	D4	00063	CLRL	4(SP)	0437
				51	D5	00066	TSTL	LOC_NAMBUF	
				1F	13	00068	BEQL	8\$	
			04	AE	D6	0006A	INCL	4(SP)	
				9F	16	0006D	JSB	@EXES\$PROBEW_DSC	0440
				50	D0	00073	MOVL	R0, STATUS	
				51	D0	00076	MOVL	R1, 16(SP)	
				52	D0	0007A	MOVL	R2, 12(SP)	
				56	E8	0007E	BLBS	STATUS, 7\$	
			0116	31	00081	BRW	27\$		
				AE	3C	00084	MOVZWL	LENGTH, LENGTH	0444
			10	AC	D0	00089	MOVL	RESID, LOC_RESID	0447
				6E	D4	0008D	CLRL	(SP)	0448
				5A	D5	0008F	TSTL	LOC_RESID	
				08	13	00091	BEQL	9\$	
				6E	D6	00093	INCL	(SP)	
				00	0D	00095	PROBEW	#0, #4, (LOC_RESID)	
				1E	13	00099	BEQL	11\$	
			59	AC	D0	0009B	MOVL	ATTRIB, LOC_ATTRIB	0452
				5B	D4	0009F	CLRL	R11	0453
				59	D5	000A1	TSTL	LOC_ATTRIB	
				08	13	000A3	BEQL	10\$	
				5B	D6	000A5	INCL	R11	
				00	0D	000A7	PROBEW	#0, #4, (LOC_ATTRIB)	
				0C	13	000AB	BEQL	11\$	
				AC	D0	000AD	MOVL	CONXT, LOC_CONXT	0457
				0A	13	000B1	BEQL	12\$	0458
				00	0D	000B3	PROBEW	#0, #4, (LOC_CONXT)	
				04	12	000B7	BNEQ	12\$	
				0C	D0	000B9	MOVL	#12, R0	0460
				04	000BC	RET			
				00	2C	000BD	MOVCS	#0, (SP), #0, #68, \$RMS_PTR	0475
				AE	000C4				
			4C	AE	8F	B0	MOVW	#17409, \$RMS_PTR	
				8F	D0	000C6	MOVL	#1179648, \$RMS_PTR+4	
				AE	94	000CC	CLRB	\$RMS_PTR+30	
				30	B0	000D4	MOVW	#48, \$RMS_PTR+32	
				AE	9E	000D7	MOVAB	REC_BUFFER, \$RMS_PTR+36	
				02	90	000DB	MOVB	#2, \$RMS_PTR+53	
				52	D4	000E0	CLRL	R2	0476
				AE	D5	000E4	TSTL	LOC_ID	
				0F	13	000E6	BEQL	13\$	
				52	D6	000E9	INCL	R2	
				01	90	000EB	MOVB	#1, RAB+30	0479
				04	B0	000ED	MOVW	#4, RAB+52	0481
				AE	9E	000F1	MOVAB	LOC_ID, RAB+48	0482
				AE	9F	000F5	PUSHAB	CLOSE	0485
				AE	9F	000FA	PUSHAB	RAB+2	
				7E	D4	000FD	CLRL	-(SP)	
				57	DD	00100	PUSHL	LOC_CONXT	
				04	FB	00102	CALLS	#4, EXES\$OPEN_RDB	
				50	D0	00104	MOVL	R0, STATUS	
				56	E8	00109	BLBS	STATUS, 14\$	0486
				31	0010C	BRW	27\$		
				AE	9F	0010F	PUSHAB	RAB	0497
				01	FB	00112	CALLS	#1, SYSS\$GET	
				50	D0	00115	MOVL	R0, STATUS	
				50	D0	0011C			

SYSRDBRES
V04-000

SYSRDBRES - Rights database resident system ser

M 14

16-Sep-1984 02:38:54
14-Sep-1984 13:14:20

VAX-11 Bliss-32 V4.0-742
[SYS.SRC]SYSRDBRES.B32;1

Page 16
(4)

000182B2	8F		56	D1	0011F	CMPL	STATUS, #98994	0498
			09	13	00126	BEQL	15\$	
0001827A	8F		56	D1	00128	CMPL	STATUS, #98938	
			05	12	0012F	BNEQ	16\$	
	56	21EC	8F	3C	00131	MOVZWL	#8684, STATUS	0500
000181A8	8F		56	D1	00136	CMPL	STATUS, #98728	0501
			03	12	0013D	BNEQ	17\$	
	56		01	D0	0013F	MOVL	#1, STATUS	
	09		56	E8	00142	BLBS	STATUS, 18\$	0502
			57	DD	00145	PUSHL	LOC_CONXT	0505
FE54	CF		01	FB	00147	CALLS	#1, EXE\$FINISH_RDB	
			43	11	0014C	BRB	26\$	0506
	05		52	E8	0014E	BLBS	R2, 19\$	0509
		1C	AE	D5	00151	TSTL	REC_BUFFER	
			BC	13	00154	BEQL	14\$	
			20	3A	00156	LOCC	#32, #32, REC_BUFFER+16	0518
	2C	AE	02	12	0015B	BNEQ	20\$	
			51	D4	0015D	CLRL	R1	
			51	D5	0015F	TSTL	P	0519
			04	12	00161	BNEQ	21\$	
	51	4C	AE	9E	00163	MOVAB	REC_BUFFER+48, P	
	08	08	AE	E9	00167	BLBC	8(SP), 22\$	0521
	50	2C	AE	9E	0016B	MOVAB	REC_BUFFER+16, R0	
	51		50	A3	0016F	SUBW3	R0, P, (LOC_NAMLEN)	
	09	04	AE	E9	00173	BLBC	4(SP), 23\$	0522
10	AE	20	2C	AE	00177	MOVCS	#32, REC_BUFFER+16, #32, LENGTH, @ADDRESS	0524
			0C	BE	0017E			
	04		6E	E9	00180	BLBC	(SP), 24\$	0526
	6A	1C	AE	D0	00183	MOVL	REC_BUFFER, (LOC_RESID)	
	04		5B	E9	00187	BLBC	R11, 25\$	0527
	69	20	AE	D0	0018A	MOVL	REC_BUFFER+4, (LOC_ATTRIB)	
	56		01	D0	0018E	MOVL	#1, STATUS	0529
	05	18	AE	E9	00191	BLBC	CLOSE, 27\$	0535
0000V	CF		00	FB	00195	CALLS	#0, EXE\$CLOSE_RDB	
	50		56	D0	0019A	MOVL	STATUS, R0	0536
			04	0019D	RET			0537

; Routine Size: 414 bytes, Routine Base: YEXEPAGED + 0225


```
542 0538 1 GLOBAL ROUTINE EX$OPEN_RDB (CONXT, WRITE, ISI, CLOSE) =
543 0539 1
544 0540 1 ++
545 0541 1
546 0542 1 FUNCTIONAL DESCRIPTION:
547 0543 1
548 0544 1 This routine sees to it that the rights database is open
549 0545 1 as needed. If a context is specified, it locates the specified
550 0546 1 stream or opens it. If the database needs to be re-opened
551 0547 1 for write, it does so.
552 0548 1
553 0549 1 CALLING SEQUENCE:
554 0550 1 EX$OPEN_RDB (CONXT, WRITE, ISI, CLOSE)
555 0551 1
556 0552 1 INPUT PARAMETERS:
557 0553 1 CONXT: address of a longword containing the record stream
558 0554 1 context. (IFI vector index) initially should be zero,
559 0555 1 value output on first call, value input on subsequent calls.
560 0556 1 WRITE: 0 to open read-only
561 0557 1 1 to open read/write
562 0558 1
563 0559 1 IMPLICIT INPUTS:
564 0560 1 NONE
565 0561 1
566 0562 1 OUTPUT PARAMETERS:
567 0563 1 ISI: address to return the internal stream identifier (IFI)
568 0564 1 CLOSE: address to return a flag which indicates whether a call of
569 0565 1 EX$CLOSE_RDB is required.
570 0566 1 0 = an image is active and a call is not required
571 0567 1 1 = no active image. EX$CLOSE_RDB must be called.
572 0568 1
573 0569 1 IMPLICIT OUTPUTS:
574 0570 1 NONE
575 0571 1
576 0572 1 ROUTINE VALUE:
577 0573 1 Status of RMS open
578 0574 1
579 0575 1 SIDE EFFECTS:
580 0576 1 Rights database opened, ISI vector modified
581 0577 1
582 0578 1 --
583 0579 1
584 0580 2 BEGIN
585 0581 2
586 0582 2 REGISTER
587 0583 2 SIZE = 1, ! size returned from EX$ALOP1IMAG
588 0584 2 ADDRESS = 2; ! address returned from EX$ALOP1IMAG
589 0585 2
590 0586 2 LOCAL
591 0587 2 STATUS : LONG, ! general status value
592 0588 2 ISI_INDEX : LONG, ! index into ISI vector
593 0589 2 LOCAL_ISI : LONG, ! local copy of ISI
594 0590 2 FAB : $FAB_DECL, ! FAB to open rights database
595 0591 2 RAB : $RAB_DECL, ! RAB to connect stream
596 0592 2 ARGLIST : VECTOR [2] ! argument list for EX$SET_RDIPTR
597 0593 2 INITIAL (1,0);
598 0594 2
```

```

599 0595 2 .CLOSE = 0;
600 0596
601 0597 ! Allocate RDI if it has not been allocated already
602 0598 !
603 0599
604 0600 IF .CTL$GL_RDIPTTR EQLU 0
605 0601 THEN
606 0602 BEGIN
607 0603 STATUS = EXESALOP1IMAG (RDISS_RDIDEF; SIZE, ADDRESS);
608 0604 IF NOT .STATUS THEN RETURN SSS_INSMEM;
609 0605 .ADDRESS = .SIZE;
610 0606 ARGLIST[1] = .ADDRESS;
611 0607 STATUS = SYSSCMKRNLC (EXESSET_RDIPTTR, ARGLIST);
612 0608 IF NOT .STATUS THEN RETURN .STATUS;
613 0609 CH$FILL (0, RDISS_RDIDEF-4, .CTL$GL_RDIPTTR+4);
614 0610 END;
615 0611
616 0612 ! Locate the ISI implied by the context. If there is no context, we
617 0613 use ISI vector zero. If the context is zero, we find a free vector
618 0614 entry.
619 0615 !
620 0616
621 0617 IF .CONXT EQLU 0
622 0618 THEN
623 0619 ISI_INDEX = 0
624 0620
625 0621 ELSE
626 0622 BEGIN
627 0623 ISI_INDEX = ..CONXT;
628 0624 IF .ISI_INDEX GTRU RDISK_ISI_MAX THEN RETURN SSS_IVCHAN;
629 0625 IF .ISI_INDEX EQLU 0
630 0626 THEN
631 0627 BEGIN
632 0628 IF
633 0629 BEGIN
634 0630 INCR J FROM 1 TO RDISK_ISI_MAX
635 0631 DO
636 0632 BEGIN
637 0633 IF .VECTOR [CTL$GL_RDIPTTR[RDISL_ISI_VEC], .J] EQLU 0
638 0634 THEN
639 0635 BEGIN
640 0636 ISI_INDEX = .J;
641 0637 EXITLOOP 0;
642 0638 END;
643 0639 END
644 0640 END
645 0641 THEN RETURN SSS_NOIOCHAN;
646 0642 END
647 0643 ELSE
648 0644 IF .VECTOR [CTL$GL_RDIPTTR[RDISL_ISI_VEC], .ISI_INDEX] EQLU 0 THEN RETURN SSS_IVCHAN;
649 0645 IF .WRITE THEN ISI_INDEX = 0;
650 0646 .CONXT = .ISI_INDEX;
651 0647 END;
652 0648
653 0649 ! Get the ISI for the selected stream. If the ISI is zero, it must
654 0650 be opened.
655 0651 !

```



```

656 0652 2
657 0653 2 LOCAL_ISI = VECTOR [CTL$GL_RDIPTTR[RDISL_ISI_VEC], .ISI_INDEX];
658 0654 2 IF .LOCAL_ISI NEQU 0
659 0655 2 AND (IF .WRITE THEN .CTL$GL_RDIPTTR[RDISL_IFI_WRITE] NEQU 0 ELSE 1)
660 0656 2 THEN
661 0657 2 BEGIN
662 0658 2 (.ISI)<0,16> = .LOCAL_ISI;
663 0659 2 RETURN SSS_NORMAL;
664 0660 2 END;
665 0661 2
666 0662 2 ! Get a FAB and RAB set up for whatever now needs to be done. Note -
667 0663 2 ! the KRF = 2 is assumed by EXESSIDTOASC.
668 0664 2
669 0665 2
670 P 0666 2 $FAB_INIT (FAB = FAB,
671 P 0667 2 FNM = 'RIGHTSLIST',
672 P 0668 2 DNM = 'SYSS$SYSTEM:.DAT',
673 P 0669 2 FAC = GET,
674 P 0670 2 SHR = (GET, PUT, UPD, DEL, MSE)
675 0671 2 );
676 0672 2 FAB[FAB$V_LNM_MODE] = PSL$C_EXEC;
677 0673 2 FAB[FAB$W_IFI] = .CTL$GL_RDIPTTR[RDISL_IFI_READ];
678 0674 2
679 P 0675 2 $RAB_INIT (RAB = RAB,
680 P 0676 2 FAB = FAB,
681 P 0677 2 KRF = 2
682 0678 2 );
683 0679 2 RAB[RAB$W_ISI] = .LOCAL_ISI;
684 0680 2
685 0681 2 ! If LOCAL_ISI is nonzero at this point, it is because we need the database
686 0682 2 ! reopened for write. Disconnect ISI entry 0 and set up to open for
687 0683 2 ! write.
688 0684 2
689 0685 2
690 0686 2 IF .LOCAL_ISI NEQU 0
691 0687 2 THEN
692 0688 2 BEGIN
693 0689 2 VECTOR [CTL$GL_RDIPTTR[RDISL_ISI_VEC], 0] = 0;
694 0690 2 STATUS = $DISCONNECT (RAB = RAB);
695 0691 2 IF NOT .STATUS THEN RETURN .STATUS;
696 0692 2 END;
697 0693 2 IF .WRITE
698 0694 2 THEN
699 0695 2 BEGIN
700 0696 2 FAB[FAB$V_PUT] = 1;
701 0697 2 FAB[FAB$V_UPD] = 1;
702 0698 2 FAB[FAB$V_DEL] = 1;
703 0699 2 FAB[FAB$V_MSE] = 0;
704 0700 2 END;
705 0701 2
706 0702 2 ! Now open the file if it is not already suitably open.
707 0703 2
708 0704 2
709 0705 2 IF (IF .WRITE
710 0706 2 THEN
711 0707 2 .CTL$GL_RDIPTTR[RDISL_IFI_WRITE] EQLU 0
712 0708 2 ELSE
```

```

: 713 0709 3      .CTL$GL_RDIPTTR[RDISL_IFI_READ] EQLU 0
: 714 0710 3      )
: 715 0711 3      THEN
: 716 0712 3      BEGIN
: 717 0713 3      FAB[FAB$W_IFI] = 0;
: 718 0714 3      IF .CTL$GL_IMGHDRBF EQLU 0
: 719 0715 3      THEN
: 720 0716 4      BEGIN
: 721 0717 4      .CLOSE = 1;
: 722 0718 4      FAB[FAB$V_PPF] = 1;
: 723 0719 4      END;
: 724 0720 3      STATUS = $OPEN (FAB = FAB);
: 725 0721 3      IF NOT .STATUS THEN RETURN .STATUS;
: 726 0722 3      IF .WRITE
: 727 0723 3      THEN
: 728 0724 3      CTL$GL_RDIPTTR[RDISL_IFI_WRITE] = .FAB[FAB$W_IFI]
: 729 0725 3      ELSE
: 730 0726 3      CTL$GL_RDIPTTR[RDISL_IFI_READ] = .FAB[FAB$W_IFI];
: 731 0727 3      END;
: 732 0728 3      ! Now set up the RAB suitably and connect it.
: 733 0729 3      !
: 734 0730 3      !
: 735 0731 2      STATUS = $CONNECT (RAB = RAB);
: 736 0732 2      IF NOT .STATUS THEN RETURN .STATUS;
: 737 0733 2      VECTOR [CTL$GL_RDIPTTR[RDISL_ISI_VEC], .ISI_INDEX] = .RAB[RAB$W_ISI];
: 738 0734 2      (.ISI)<0,16> = .RAB[RAB$W_ISI];
: 739 0735 2      IF .CONXT NEQU 0 THEN .CONXT = .ISI_INDEX;
: 740 0736 2      SS$NORMAL
: 741 0737 2      ! End of routine EXE$OPEN_RDB
: 742 0738 2      END;
: 743 0739 2
: 744 0740 1
```

```

54 41 44 2E 3A 4D 53 49 4C 53 54 48 47 49 52 003C3 P.AAB: .ASCII \RIGHTSLIST\
54 41 44 2E 3A 4D 45 54 53 59 53 24 53 59 53 003CD P.AAC: .ASCII \SYS$SYSTEM:.DATA
                                     .EXTRN  SYS$OPEN, SYS$CONNECT
                                     OFFC 00000
                                     .ENTRY  EXE$OPEN_RDB, Save R2,R3,R4,R5,R6,R7,R8,R9,-; 0538
                                     R10,R11
                                     MOVAB  -152(SP), SP
                                     PUSHL  #1 0580
                                     CLRL   ARGLIST+4
                                     CLRL   @CLOSE 0595
                                     TSTL   @#CTL$GL_RDIPTTR 0600
                                     BNEQ   3$
                                     MOVL   #56, R1 0603
                                     JSB    @#EXE$ALOP1IMAG
                                     MOVL   R0, STATUS
                                     BLBS   STATUS, 1$ 0604
                                     MOVZWL #292, R0
                                     RET
                                     MOVL   SIZE, (ADDRESS) 0605
                                     MOVL   ADDRESS, ARGLIST+4 0606
                                     PUSHL  SP 0607
```


PC	Op	OpC	OpD	OpI	OpR	OpS	OpT	OpV	OpW	OpX	OpY	OpZ	OpAA	OpAB	OpAC	OpAD	OpAE	OpAF	OpAG	OpAH	OpAI	OpAJ	OpAK	OpAL	OpAM	OpAN	OpAO	OpAP	OpAQ	OpAR	OpAS	OpAT	OpAU	OpAV	OpAW	OpAX	OpAY	OpAZ	OpBA	OpBB	OpBC	OpBD	OpBE	OpBF	OpBG	OpBH	OpBI	OpBJ	OpBK	OpBL	OpBM	OpBN	OpBO	OpBP	OpBQ	OpBR	OpBS	OpBT	OpBU	OpBV	OpBW	OpBX	OpBY	OpBZ	OpCA	OpCB	OpCC	OpCD	OpCE	OpCF	OpCG	OpCH	OpCI	OpCJ	OpCK	OpCL	OpCM	OpCN	OpCO	OpCP	OpCQ	OpCR	OpCS	OpCT	OpCU	OpCV	OpCW	OpCX	OpCY	OpCZ	OpDA	OpDB	OpDC	OpDD	OpDE	OpDF	OpDG	OpDH	OpDI	OpDJ	OpDK	OpDL	OpDM	OpDN	OpDO	OpDP	OpDQ	OpDR	OpDS	OpDT	OpDU	OpDV	OpDW	OpDX	OpDY	OpDZ	OpEA	OpEB	OpEC	OpED	OpEE	OpEF	OpEG	OpEH	OpEI	OpEJ	OpEK	OpEL	OpEM	OpEN	OpEO	OpEP	OpEQ	OpER	OpES	OpET	OpEU	OpEV	OpEW	OpEX	OpEY	OpEZ	OpFA	OpFB	OpFC	OpFD	OpFE	OpFF	OpFG	OpFH	OpFI	OpFJ	OpFK	OpFL	OpFM	OpFN	OpFO	OpFP	OpFQ	OpFR	OpFS	OpFT	OpFU	OpFV	OpFW	OpFX	OpFY	OpFZ	OpGA	OpGB	OpGC	OpGD	OpGE	OpGF	OpGG	OpGH	OpGI	OpGJ	OpGK	OpGL	OpGM	OpGN	OpGO	OpGP	OpGQ	OpGR	OpGS	OpGT	OpGU	OpGV	OpGW	OpGX	OpGY	OpGZ	OpHA	OpHB	OpHC	OpHD	OpHE	OpHF	OpHG	OpHH	OpHI	OpHJ	OpHK	OpHL	OpHM	OpHN	OpHO	OpHP	OpHQ	OpHR	OpHS	OpHT	OpHU	OpHV	OpHW	OpHX	OpHY	OpHZ	OpIA	OpIB	OpIC	OpID	OpIE	OpIF	OpIG	OpIH	OpII	OpIJ	OpIK	OpIL	OpIM	OpIN	OpIO	OpIP	OpIQ	OpIR	OpIS	OpIT	OpIU	OpIV	OpIW	OpIX	OpIY	OpIZ	OpJA	OpJB	OpJC	OpJD	OpJE	OpJF	OpJG	OpJH	OpJI	OpJJ	OpJK	OpJL	OpJM	OpJN	OpJO	OpJP	OpJQ	OpJR	OpJS	OpJT	OpJU	OpJV	OpJW	OpJX	OpJY	OpJZ	OpKA	OpKB	OpKC	OpKD	OpKE	OpKF	OpKG	OpKH	OpKI	OpKJ	OpKK	OpKL	OpKM	OpKN	OpKO	OpKP	OpKQ	OpKR	OpKS	OpKT	OpKU	OpKV	OpKW	OpKX	OpKY	OpKZ	OpLA	OpLB	OpLC	OpLD	OpLE	OpLF	OpLG	OpLH	OpLI	OpLJ	OpLK	OpLL	OpLM	OpLN	OpLO	OpLP	OpLQ	OpLR	OpLS	OpLT	OpLU	OpLV	OpLW	OpLX	OpLY	OpLZ	OpMA	OpMB	OpMC	OpMD	OpME	OpMF	OpMG	OpMH	OpMI	OpMJ	OpMK	OpML	OpMM	OpMN	OpMO	OpMP	OpMQ	OpMR	OpMS	OpMT	OpMU	OpMV	OpMW	OpMX	OpMY	OpMZ	OpNA	OpNB	OpNC	OpND	OpNE	OpNF	OpNG	OpNH	OpNI	OpNJ	OpNK	OpNL	OpNM	OpNN	OpNO	OpNP	OpNQ	OpNR	OpNS	OpNT	OpNU	OpNV	OpNW	OpNX	OpNY	OpNZ	OpOA	OpOB	OpOC	OpOD	OpOE	OpOF	OpOG	OpOH	OpOI	OpOJ	OpOK	OpOL	OpOM	OpON	OpOO	OpOP	OpOQ	OpOR	OpOS	OpOT	OpOU	OpOV	OpOW	OpOX	OpOY	OpOZ	OpPA	OpPB	OpPC	OpPD	OpPE	OpPF	OpPG	OpPH	OpPI	OpPJ	OpPK	OpPL	OpPM	OpPN	OpPO	OpPP	OpPQ	OpPR	OpPS	OpPT	OpPU	OpPV	OpPW	OpPX	OpPY	OpPZ	OpQA	OpQB	OpQC	OpQD	OpQE	OpQF	OpQG	OpQH	OpQI	OpQJ	OpQK	OpQL	OpQM	OpQN	OpQO	OpQP	OpQQ	OpQR	OpQS	OpQT	OpQU	OpQV	OpQW	OpQX	OpQY	OpQZ	OpRA	OpRB	OpRC	OpRD	OpRE	OpRF	OpRG	OpRH	OpRI	OpRJ	OpRK	OpRL	OpRM	OpRN	OpRO	OpRP	OpRQ	OpRR	OpRS	OpRT	OpRU	OpRV	OpRW	OpRX	OpRY	OpRZ	OpSA	OpSB	OpSC	OpSD	OpSE	OpSF	OpSG	OpSH	OpSI	OpSJ	OpSK	OpSL	OpSM	OpSN	OpSO	OpSP	OpSQ	OpSR	OpSS	OpST	OpSU	OpSV	OpSW
----	----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------

08	AE	4401	8F	B0	00106	MOVW	#17409, SRMS_PTR	:	
3D	AE		02	90	0010C	MOVB	#2, SRMS_PTR+53	:	
44	AE	4C	AE	9E	00110	MOVAB	FAB, SRMS_PTR+60	:	
0A	AE		59	B0	00115	MOVW	LOCAL_ISI, RAB+2	:	0679
	16		5B	E9	00119	BLBC	R11, T4\$	:	0686
	50		57	D0	0011C	MOVL	R7, R0	:	0689
		0C	A0	D4	0011F	CLRL	12(R0)	:	
		08	AE	9F	00122	PUSHAB	RAB	:	0690
00000000G	00		01	FB	00125	CALLS	#1, SYSSDISCONNECT	:	
	58		50	D0	0012C	MOVL	R0, STATUS	:	
	6B		58	E9	0012F	BLBC	STATUS, 21\$	:	0691
	08	08	AC	E9	00132	BLBC	WRITE, 15\$	:	0693
62	AE		0D	88	00136	BISB2	#13, FAB+22	:	0698
63	AE		10	8A	0013A	BICB2	#16, FAB+23	:	0699
	50	00000000G	9F	D0	0013E	MOVL	@#CTL\$GL_RDIPTR, R0	:	0707
	05		08	AC	E9	BLBC	WRITE, 16\$	:	
			08	A0	D5	TSTL	8(R0)	:	
			03	11	0014C	BRB	17\$	:	
		04	A0	D5	0014E	TSTL	4(R0)	:	0709
			3A	12	00151	BNEQ	20\$	:	
		4E	AE	B4	00153	CLRW	FAB+2	:	0713
		00000000G	9F	D5	00156	TSTL	@#CTL\$GL_IMGHDRBF	:	0714
			08	12	0015C	BNEQ	18\$	:	
10	BC		01	D0	0015E	MOVL	#1, @CLOSE	:	0717
52	AE		04	88	00162	BISB2	#4, FAB+6	:	0718
		4C	AE	9F	00166	PUSHAB	FAB	:	0720
00000000G	00		01	FB	00169	CALLS	#1, SYSSOPEN	:	
	58		50	D0	00170	MOVL	R0, STATUS	:	
	27		58	E9	00173	BLBC	STATUS, 21\$	:	0721
	50	00000000G	9F	D0	00176	MOVL	@#CTL\$GL_RDIPTR, R0	:	0724
	07		08	AC	E9	BLBC	WRITE, 19\$	:	
08	A0	4E	AE	3C	00181	MOVZWL	FAB+2, 8(R0)	:	
			05	11	00186	BRB	20\$	:	
04	A0	4E	AE	3C	00188	MOVZWL	FAB+2, 4(R0)	:	0726
		08	AE	9F	0018D	PUSHAB	RAB	:	0732
00000000G	00		01	FB	00190	CALLS	#1, SYSSCONNECT	:	
	58		50	D0	00197	MOVL	R0, STATUS	:	
	04		58	E8	0019A	BLBS	STATUS, 22\$	:	0733
	50		58	D0	0019D	MOVL	STATUS, R0	:	
			04	001A0	RET			:	
	50	00000000G	9F	D0	001A1	MOVL	@#CTL\$GL_RDIPTR, R0	:	0735
0C	A046	0A	AE	3C	001A8	MOVZWL	RAB+2, 12(R0)[ISI_INDEX]	:	
0C	BC	0A	AE	B0	001AE	MOVW	RAB+2, @ISI	:	0736
			5A	D5	001B3	TSTL	R10	:	0737
			03	13	001B5	BEQL	23\$	:	
	6A		56	D0	001B7	MOVL	ISI_INDEX, (R10)	:	
	50		01	D0	001BA	MOVL	#1, -R0	:	0740
			04	001BD	RET			:	

; Routine Size: 446 bytes, Routine Base: YEXEPAGED + 03DC


```
746 0741 1 GLOBAL ROUTINE EX$CLOSE_RDB : NOVALUE =
747 0742 1
748 0743 1 ++
749 0744 1
750 0745 1 FUNCTIONAL DESCRIPTION:
751 0746 1
752 0747 1 This routine closes the rights database file and zeros the Rights
753 0748 1 Database Identifier (RDI) table.
754 0749 1
755 0750 1 CALLING SEQUENCE:
756 0751 1 EX$CLOSE_RDB()
757 0752 1
758 0753 1 INPUT PARAMETERS:
759 0754 1 NONE
760 0755 1
761 0756 1 IMPLICIT INPUTS:
762 0757 1 NONE
763 0758 1
764 0759 1 OUTPUT PARAMETERS:
765 0760 1 NONE
766 0761 1
767 0762 1 IMPLICIT OUTPUTS:
768 0763 1 Rights database file closed and the RDI table zeroed.
769 0764 1
770 0765 1 ROUTINE VALUE:
771 0766 1 NONE
772 0767 1
773 0768 1 SIDE EFFECTS:
774 0769 1 NONE
775 0770 1
776 0771 1 --
777 0772 1
778 0773 2 BEGIN
779 0774 2
780 0775 2 LOCAL
781 0776 2 FAB : $FAB_DECL; ! FAB to close rights database
782 0777 2
783 0778 2 $FAB_INIT (FAB = FAB);
784 0779 2
785 0780 2 ! Close out read channels
786 0781 2 !
787 0782 2
788 0783 2 IF .CTL$GL_RDIPTTR[RDISL_IFI_READ] NEQU 0
789 0784 2 THEN
790 0785 3 BEGIN
791 0786 3 FAB[FAB$W_IFI] = .CTL$GL_RDIPTTR[RDISL_IFI_READ];
792 0787 3 $CLOSE (FAB = FAB);
793 0788 3 END;
794 0789 2
795 0790 2 ! Close out write channels
796 0791 2 !
797 0792 2
798 0793 2 IF .CTL$GL_RDIPTTR[RDISL_IFI_WRITE] NEQU 0
799 0794 2 THEN
800 0795 3 BEGIN
801 0796 3 FAB[FAB$W_IFI] = .CTL$GL_RDIPTTR[RDISL_IFI_WRITE];
802 0797 3 $CLOSE (FAB = FAB);
```

```
: 803      0798 2      END;
: 804      0799 2
: 805      0800 2      ! Zero RDI table
: 806      0801 2      !
: 807      0802 2
: 808      0803 2      CH$FILL (0, RDISS_RDIDEF-4, .CTL$GL_RDIPTR+4);
: 809      0804 1      END;
```

.EXTRN SYS\$CLOSE

.ENTRY EXE\$CLOSE_RDB, Save R2,R3,R4,R5,R6,R7

```
MOVAB SYS$CLOSE, R7
MOVAB @#CTL$GL_RDIPTR, R6
MOVAB -80(SP), SP
MOVCS #0, (SP), #0, #80, $RMS_PTR
```

```
MOVW #20483, $RMS_PTR
MOVB #2, $RMS_PTR+22
MOVB #2, $RMS_PTR+31
MOVL CTL$GL_RDIPTR, R0
TSTL 4(R0)
```

```
BEQL 1$
MOVW 4(R0), FAB+2
PUSHL SP
CALLS #1, SYS$CLOSE
MOVL CTL$GL_RDIPTR, R0
TSTL 8(R0)
```

```
BEQL 2$
MOVW 8(R0), FAB+2
PUSHL SP
CALLS #1, SYS$CLOSE
MOVL CTL$GL_RDIPTR, R0
MOVCS #0, (SP), #0, #52, 4(R0)
```

RET

: 0741

: 0778

: 0783

: 0786

: 0787

: 0793

: 0796

: 0797

: 0803

: 0804

; Routine Size: 88 bytes, Routine Base: YEXEPAGED + 059A


```
: 811 0805 1 GLOBAL ROUTINE EXESSET_RDIPTR (ADDRESS) =
: 812 0806 1
: 813 0807 1 ++
: 814 0808 1
: 815 0809 1 FUNCTIONAL DESCRIPTION:
: 816 0810 1
: 817 0811 1 This routine stores the address of the RDI block in CTL$GL_RDIPTR.
: 818 0812 1 It is called from exec mode with the change to kernel mode system
: 819 0813 1 service.
: 820 0814 1
: 821 0815 1 CALLING SEQUENCE:
: 822 0816 1 SYSSCMKRN (EXESSET_RDIPTR, ARGLIST)
: 823 0817 1
: 824 0818 1 INPUT PARAMETERS:
: 825 0819 1 ADDRESS: The address of the allocated RDI block
: 826 0820 1
: 827 0821 1 IMPLICIT INPUTS:
: 828 0822 1 NONE
: 829 0823 1
: 830 0824 1 OUTPUT PARAMETERS:
: 831 0825 1 NONE
: 832 0826 1
: 833 0827 1 IMPLICIT OUTPUTS:
: 834 0828 1 NONE
: 835 0829 1
: 836 0830 1 ROUTINE VALUE:
: 837 0831 1 SSS_NORMAL
: 838 0832 1
: 839 0833 1 SIDE EFFECTS:
: 840 0834 1 NONE
: 841 0835 1
: 842 0836 1 --
: 843 0837 1
: 844 0838 2 BEGIN
: 845 0839 2
: 846 0840 2 CTL$GL_RDIPTR = .ADDRESS;
: 847 0841 2 RETURN SSS_NORMAL;
: 848 0842 1 END;
```

```
00000000G 9F 04 AC 0000 0000
50 01 D0 00002
04 0000A
04 0000D
```

```
.ENTRY EXESSET_RDIPTR, Save nothing
MOVL ADDRESS, @#CTL$GL_RDIPTR
MOVL #1, R0
RET
```

```
: 0805
: 0840
: 0841
: 0842
```

; Routine Size: 14 bytes, Routine Base: YEXEPAGED + 05F2

```
: 849 0843 1
: 850 0844 1 END
: 851 0845 0 ELUDOM
```

SYSRDBRES
V04-000

SYSRDBRES - Rights database resident system ser J 15
16-Sep-1984 02:38:54
14-Sep-1984 13:14:20

VAX-11 Bliss-32 V4.0-742
[SYS.SRC]SYSRDBRES.B32;1

Page 26
(7)

PSECT SUMMARY

Name	Bytes	Attributes
YEXEPAGED	1536	NOVEC, WRT, RD , EXE,NOSHR, LCL, REL, CON,NOPI,ALIGN(2)

Library Statistics

File	----- Total	Symbols Loaded	----- Percent	Pages Mapped	Processing Time
_\$255\$DUA28:[SYSLIB]LIB.L32;1	18619	110	0	1000	00:01.8

COMMAND QUALIFIERS

BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LIS\$:SYSRDBRES/OBJ=OBJ\$:SYSRDBRES MSRC\$:SYSRDBRES/UPDATE=(ENH\$:SYSRDBRES)

: Size: 1255 code + 281 data bytes
: Run Time: 00:30.5
: Elapsed Time: 00:34.2
: Lines/CPU Min: 1660
: Lexemes/CPU-Min: 27785
: Memory Used: 203 pages
: Compilation Complete

0387 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

