

FFFFFFFFFFFFFF	DDDDDDDDDDDD	LLL	
FFFFFFFFFFFFFF	DDDDDDDDDDDD	LLL	
FFFFFFFFFFFFFF	DDDDDDDDDDDD	LLL	
FFF	DDD	DDD	LLL
FFF	DDD	DDD	LLL
FFF	DDD	DDD	LLL
FFF	DDD	DDD	LLL
FFF	DDD	DDD	LLL
FFF	DDD	DDD	LLL
FFFFFFFFFFFFFF	DDD	DDD	LLL
FFFFFFFFFFFFFF	DDD	DDD	LLL
FFFFFFFFFFFFFF	DDD	DDD	LLL
FFF	DDD	DDD	LLL
FFF	DDD	DDD	LLL
FFF	DDD	DDD	LLL
FFF	DDD	DDD	LLL
FFF	DDD	DDD	LLL
FFF	DDD	DDD	LLL
FFF	DDDDDDDDDDDD	LLLLLLLLLLLLLLLL	
FFF	DDDDDDDDDDDD	LLLLLLLLLLLLLLLL	
FFF	DDDDDDDDDDDD	LLLLLLLLLLLLLLLL	

```

FFFFFFFFF DDDDDDD LL      PPPPPPP AAAA   RRRRRRR SSSSSSS EEEEEEEEE
FFFFFFFFF DDDDDDD LL      PPPPPPP AAAA   RRRRRRR SSSSSSS EEEEEEEEE
FF        DD      DD    LL      PP      PP  AA     AA  RR      RR  SS      EE
FF        DD      DD    LL      PP      PP  AA     AA  RR      RR  SS      EE
FF        DD      DD    LL      PP      PP  AA     AA  RR      RR  SS      EE
FF        DD      DD    LL      PP      PP  AA     AA  RR      RR  SS      EE
FFFFFFFFF DD      DD    LL      PPPPPPP AA     AA  RRRRRRR SSSSSSS EEEEEEEEE
FFFFFFFFF DD      DD    LL      PPPPPPP AA     AA  RRRRRRR SSSSSSS EEEEEEEEE
FF        DD      DD    LL      PP      PP  AA     AA  RR      RR  SS      EE
FF        DD      DD    LL      PP      PP  AA     AA  RR      RR  SS      EE
FF        DD      DD    LL      PP      PP  AA     AA  RR      RR  SS      EE
FF        DD      DD    LL      PP      PP  AA     AA  RR      RR  SS      EE
FF        DDDDDDD LL      PPPPPPP AA     AA  RRRRRRR SSSSSSS EEEEEEEEE
FF        DDDDDDD LL      PPPPPPP AA     AA  RRRRRRR SSSSSSS EEEEEEEEE

```

```

LL          IIIIII SSSSSSS
LL          IIIIII SSSSSSS
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
LLLLLLLLLL IIIIII SSSSSSS
LLLLLLLLLL IIIIII SSSSSSS

```

```

: R
...
...
...
...

```

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32

```

0001 0 %TITLE 'FDL$PARSE'
0002 0 %SBTTL 'FDL_Parse Action Routines'
0003 0 MODULE FDLPARSE ( IDENT='V04-000',
0004 0 ADDRESSING_MODE ( EXTERNAL = GENERAL ),
0005 0 ADDRESSING_MODE ( NONEXTERNAL = GENERAL ),
0006 0 OPTLEVEL=3
0007 0 ) =
0008 0
0009 1 BEGIN
0010 1
0011 1 :*****
0012 1 :*
0013 1 :* COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0014 1 :* DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0015 1 :* ALL RIGHTS RESERVED.
0016 1 :*
0017 1 :* THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0018 1 :* ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0019 1 :* INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0020 1 :* COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0021 1 :* OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0022 1 :* TRANSFERRED.
0023 1 :*
0024 1 :* THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0025 1 :* AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0026 1 :* CORPORATION.
0027 1 :*
0028 1 :* DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0029 1 :* SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0030 1 :*
0031 1 :*
0032 1 :*****

```

```
34 0033 1 |++
35 0034 1 |
36 0035 1 | Facility:
37 0036 1 |     RMS-32 FDL Utilities
38 0037 1 |
39 0038 1 | Environment:
40 0039 1 |     VAX/VMS Operating System
41 0040 1 |
42 0041 1 | Abstract:
43 0042 1 |     Routines which fill the rms control blocks
44 0043 1 |     for the FDL parser
45 0044 1 |
46 0045 1 | Contents:
47 0046 1 |     INIT_PARSE
48 0047 1 |     LINE_PARSED
49 0048 1 |     SET_AREA_P
50 0049 1 |     SET_DATE_P
51 0050 1 |     SET_JNL_P
52 0051 1 |     SET_ACL_P
53 0052 1 |     SET_FILE_P
54 0053 1 |     SET_KEY_P
55 0054 1 |     SET_RECORD_P
56 0055 1 |     SET_ACCESS_P
57 0056 1 |     SET_SHARING_P
58 0057 1 |     SET_CONNECT_P
59 0058 1 |     SET_PROT
60 0059 1 |     ALLOCATE_XAB
61 0060 1 |     FIND_ID
62 0061 1 |     GET_VM
63 0062 1 |     FREE_VM
64 0063 1 |
65 0064 1 | --
```

```

67 0065 1 |
68 0066 1 | Author:      Keith B Thompson      Creation date:  July-1981
69 0067 1 |
70 0068 1 |
71 0069 1 | Modified by:
72 0070 1 |
73 0071 1 |   V03-011 RRB0015      Rowland R. Bradley      29 Feb 1984
74 0072 1 |     Comment out references to ERASE_ON_DELETE and ACL support.
75 0073 1 |     Not supported for V4.0.
76 0074 1 |
77 0075 1 |   V03-010 RRB0008      Rowland R. Bradley      19 Jan 1984
78 0076 1 |     Support NULL strings in file name.
79 0077 1 |
80 0078 1 |   V03-009 KFH0007      Ken Henderson           10 Sep 1983
81 0079 1 |     Support for named UICs
82 0080 1 |
83 0081 1 |   V03-008 KFH0006      Ken Henderson           29 Jul 1983
84 0082 1 |     Check status of call to LIB$...
85 0083 1 |     Added DEFERRED_WRITE, ERASE_ON_DELETE
86 0084 1 |
87 0085 1 |   V03-007 KFH0005      Ken Henderson           6 Jan 1983
88 0086 1 |     Fixed allocation of keyname buffer
89 0087 1 |
90 0088 1 |   V03-006 KFH0004      Ken Henderson           21 Dec 1982
91 0089 1 |     Deleted unused ref to tpa_block
92 0090 1 |
93 0091 1 |   V03-005 KFH0003      Ken Henderson           22 Nov 1982
94 0092 1 |     Add support for default and main
95 0093 1 |     parses in FDL$PARSE
96 0094 1 |     Fix FDL$$FREE_VM to signal status
97 0095 1 |
98 0096 1 |   V03-004 KFH0002      Ken Henderson           6-Oct-1982
99 0097 1 |     Add support for Journal, Access,
100 0098 1 |     ACL, Sharing, Connect primaries
101 0099 1 |
102 0100 1 |   V03-003 KBT0069      Keith B. Thompson       24-Jun-1982
103 0101 1 |     Initialize the length in fdl$ab_item
104 0102 1 |
105 0103 1 |   V03-002 KBT0030      Keith Thompson          30-Mar-1982
106 0104 1 |     Fix error processing of the date & time stuff
107 0105 1 |
108 0106 1 |   V03-001 KFH0001      Ken Henderson          29 March 1982
109 0107 1 |     Fixed SET_AREA_P to set LBN
110 0108 1 |     instead of VBN for volume placement
111 0109 1 |
112 0110 1 | *****

```

```

114 0111 1
115 0112 1 PSECT
116 0113 1     OWN      =  FDL$OWN      (PIC),
117 0114 1     GLOBAL   =  FDL$GLOBAL  (PIC),
118 0115 1     PLIT     =  FDL$PLIT    (SHARE,PIC),
119 0116 1     CODE     =  FDL$CODE    (SHARE,PIC):
120 0117 1
121 0118 1 LIBRARY 'SYS$LIBRARY:STARLET';
122 0119 1 REQUIRE 'SRC$:FDLLTIL';
123 0304 1 REQUIRE 'LIB$:FDLPARDEF';
124 0843 1
125 0844 1 EXTERNAL ROUTINE
126 0845 1     LIB$GET_VM,
127 0846 1     LIB$FREE_VM,
128 0847 1     FDL$$RMS_ERROR           : NOVALUE;
129 0848 1
130 0849 1 DEFINE_ERROR_CODES;
131 0850 1
132 0851 1 FORWARD ROUTINE
133 0852 1     SET_AREA_P           : NOVALUE,
134 0853 1     SET_DATE_P           : NOVALUE,
135 0854 1     SET_JNL_P           : NOVALUE,
136 0855 1     SET_ACL_P           : NOVALUE,
137 0856 1     SET_FILE_P           : NOVALUE,
138 0857 1     SET_KEY_P           : NOVALUE,
139 0858 1     SET_RECORD_P        : NOVALUE,
140 0859 1     SET_ACCESS_P        : NOVALUE,
141 0860 1     SET_SHARING_P       : NOVALUE,
142 0861 1     SET_CONNECT_P        : NOVALUE,
143 0862 1     SET_PROT            : NOVALUE,
144 0863 1     ALLOCATE_XAB,
145 0864 1     FIND_ID              : NOVALUE,
146 0865 1     FDL$$GET_VM,
147 0866 1     FDL$$FREE_VM         : NOVALUE;
148 0867 1
149 0868 1 EXTERNAL
150 0869 1     FDL$AB_TPARSE_BLOCK   : BLOCK [ ,BYTE ],
151 0870 1     FDL$AB_ITEM           : DESC_BLK,
152 0871 1     FDL$AB_CTRL           : BLGCR [ ,BYTE ],
153 0872 1     FDL$GL_PCALL,
154 0873 1     FDL$GL_STMNTNUM,
155 0874 1     FDL$GL_PRIMARY,
156 0875 1     FDL$GL_PRINUM,
157 0876 1     FDL$AB_PRICTRL,
158 0877 1     FDL$GL_SECONDARY,
159 0878 1     FDL$GL_SECNUM,
160 0879 1     FDL$GL_QUALIFIER,
161 0880 1     FDL$GL_NUMBER,
162 0881 1     FDL$GL_SWITCH,
163 0882 1     FDL$GL_OWNER_UIC,
164 0883 1     FDL$GL_SPARET,
165 0884 1     FDL$GL_PROTECTION,
166 0885 1     FDL$GL_FID1,
167 0886 1     FDL$GL_FID2,
168 0887 1     FDL$GL_FID3,
169 0888 1     FDL$AB_AREA_BKZ      : REF VECTOR [ ,BYTE ],
170 0889 1     FDL$AL_DATE_TIME    : VECTOR [ ,LONG ],

```

```

: 171 0890 1          FDL$AB_STRING          : DESC_BLK,
: 172 0891 1
: 173 0892 1          FDL$AB_PARSED_FAB      : REF $FAB_DECL,
: 174 0893 1          FDL$AB_PARSED_RAB     : REF $RAB_DECL;
: 175 0894 1
: 176 0895 1 LITERAL SPACE = 32;
: 177 0896 1
: 178 0897 1
: 179 0898 1 OWN
: 180 0899 1          HIGHEST_AREA_NO : BYTE,
: 181 0900 1          CURRENT_XAB       : REF BLOCK [ ,BYTE ],
: 182 0901 1          END_XAB           : REF BLOCK [ ,BYTE ],
: 183 0902 1
: 184 0903 1          JNL_XAB           : REF $XABJNL_DECL,      ! Journal XAB
: 185 0904 1          DATE_XAB         : REF $XABDAT_DECL,     ! Date XAB
: 186 0905 1          REVISION_XAB     : REF $XABRDT_DECL,     ! Revision Date and Time XAB
: 187 0906 1          PROTECTION_XAB   : REF $XABPRO_DECL;    ! Protection XAB
: 188 0907 1

```

```
190 0908 1 %SBTTL 'INIT_PARSE'
191 0909 1 GLOBAL ROUTINE FDL$$INIT_PARSE : NOVALUE =
192 0910 1 ++
193 0911 1
194 0912 1 Functional Description:
195 0913 1
196 0914 1 Init variables and allocate a buffer for the area bucket sizes
197 0915 1
198 0916 1 Calling Sequence:
199 0917 1
200 0918 1 fdl$$init_parse()
201 0919 1
202 0920 1 Input Parameters:
203 0921 1 none
204 0922 1
205 0923 1 Implicit Inputs:
206 0924 1 none
207 0925 1
208 0926 1 Output Parameters:
209 0927 1 none
210 0928 1
211 0929 1 Implicit Outputs:
212 0930 1 none
213 0931 1
214 0932 1 Routine Value:
215 0933 1 none
216 0934 1
217 0935 1 Routines Called:
218 0936 1
219 0937 1 lib$get_vm
220 0938 1
221 0939 1 Side Effects:
222 0940 1
223 0941 1 Allocates a buffer pointed to by FDL$AB_AREA_BKZ
224 0942 1
225 0943 1 --
226 0944 1
227 0945 2 BEGIN
228 0946 2
229 0947 2 LOCAL
230 0948 2 BYTES;
231 0949 2
232 0950 2 ! Set the parse control bits
233 0951 2 !
234 0952 2 FDL$AB_CTRL [ FDL$V_STATUS ] = _SET;
235 0953 2 FDL$AB_CTRL [ FDL$V_INITIAL ] = _SET;
236 0954 2
237 0955 2 ! Clear the other CTRL bits except the following ones:
238 0956 2 ! PCALL
239 0957 2 ! DCL
240 0958 2 ! STRING_SPEC
241 0959 2 ! GCALL
242 0960 2
243 0961 2 FDL$AB_CTRL [ FDL$V_WARNING ] = _CLEAR;
244 0962 2 FDL$AB_CTRL [ FDL$V_PRIMARY ] = _CLEAR;
245 0963 2 FDL$AB_CTRL [ FDL$V_NEWPRI ] = _CLEAR;
246 0964 2 FDL$AB_CTRL [ FDL$V_SECONDARY ] = _CLEAR;
```



```

247 0965 2 FDL$AB_CTRL [ FDL$V_COMMENT ] = _CLEAR;
248 0966 2 FDL$AB_CTRL [ FDL$V_LINECMT ] = _CLEAR;
249 0967 2 FDL$AB_CTRL [ FDL$V_APOST_PRES ] = _CLEAR;
250 0968 2 FDL$AB_CTRL [ FDL$V_QUOTE_PRES ] = _CLEAR;
251 0969 2 FDL$AB_CTRL [ FDL$V_USED_STRING ] = _CLEAR;
252 0970 2
253 0971 2 ! Initialize the item length for fdl$get_line
254 0972 2
255 0973 2 FDL$AB_ITEM [ DSC$W_LENGTH ] = 0;
256 0974 2
257 0975 2 IF NOT .FDL$AB_CTRL [ FDL$V_REPARSE ]
258 0976 2 THEN
259 0977 2 BEGIN
260 0978 2
261 0979 2 ! Clear the pointers to xabs
262 0980 2
263 0981 2 JNL_XAB = _CLEAR;
264 0982 2 DATE_XAB = _CLEAR;
265 0983 2 REVISION_XAB = _CLEAR;
266 0984 2 PROTECTION_XAB = _CLEAR;
267 0985 2
268 0986 2 END;
269 0987 2
270 0988 2 ! Clear misc
271 0989 2
272 0990 2 FDL$GL_STMNTNUM = 0;
273 0991 2 FDL$AB_PRTCTRL = _CLEAR;
274 0992 2 CURRENT_XAB = _CLEAR;
275 0993 2 HIGHEST_AREA_NO = 0;
276 0994 2
277 0995 2 ! Allocate memory for the area bucket size array NOTE: Use lib$get_vm so
278 0996 2 ! we can return this in fdl$$finish_parse
279 0997 2
280 0998 2 BYTES = 256;
281 0999 2
282 1000 2 IF NOT LIB$GET_VM ( BYTES,FDL$AB_AREA_BKZ )
283 1001 2 THEN
284 1002 2 SIGNAL_STOP ( FDL$_INSVIRMEM );
285 1003 2
286 1004 2 ! Zero the values
287 1005 2
288 1006 2 CH$FILL( 0, .BYTES, .FDL$AB_AREA_BKZ );
289 1007 2
290 1008 2 RETURN
291 1009 2
292 1010 2 END;

```

.TITLE FDLPARSE VAX-11 FDL Utilities  
.IDENT \V04-000\

.PSECT \_FDL\$OWN,NOEXE, PIC,2

00000 HIGHEST\_AREA\_NO:  
          .BLK 1  
00001          .BLK 3  
00004 CURRENT\_XAB:

00008 END\_XAB:.BLKB 4  
0000C JNL\_XAB:.BLKB 4  
00010 DATE\_XAB:  
00014 REVISION\_XAB:  
00018 PROTECTION\_XAB:  
.BLKB 4  
.BLKB 4  
.BLKB 4

.EXTRN LIB\$GET\_VM, LIB\$FREE\_VM  
.EXTRN FDL\$SRMS\_ERROR, FDL\$FACILITY  
.EXTRN FDL\$FAO\_MAX, FDL\$ABKW  
.EXTRN FDL\$ABPRIKW, FDL\$CREATE  
.EXTRN FDL\$CREATED, FDL\$CREATEDSTM  
.EXTRN FDL\$FDLERROR, FDL\$ILL\_ARG  
.EXTRN FDL\$INSVIRMEM, FDL\$INVBLK  
.EXTRN FDL\$INVDATIM, FDL\$MULPRI  
.EXTRN FDL\$MULSEC, FDL\$NOQUAL  
.EXTRN FDL\$NULLPRI, FDL\$OPENFDL  
.EXTRN FDL\$OUTORDER, FDL\$OPENOUT  
.EXTRN FDL\$WRITEERR, FDL\$READERR  
.EXTRN FDL\$RFLOC, FDL\$TITLE  
.EXTRN FDL\$SYNTAX, FDL\$VALPRI  
.EXTRN FDL\$UNQUAKW, FDL\$UNPRIKW  
.EXTRN FDL\$UNSECKW, FDL\$WARNING  
.EXTRN FDL\$AB\_TPARSE\_BLOCK  
.EXTRN FDL\$AB\_ITEM, FDL\$AB\_CTRL  
.EXTRN FDL\$GL\_PCALL, FDL\$GL\_STMTNUM  
.EXTRN FDL\$GL\_PRIMARY, FDL\$GL\_PRINUM  
.EXTRN FDL\$AB\_PRICTRL, FDL\$GL\_SECONDARY  
.EXTRN FDL\$GL\_SECNUM, FDL\$GL\_QUALIFIER  
.EXTRN FDL\$GL\_NUMBER, FDL\$GL\_SWITCH  
.EXTRN FDL\$GL\_OWNER\_UIC  
.EXTRN FDL\$GL\_SPARET, FDL\$GL\_PROTECTION  
.EXTRN FDL\$GL\_FID1, FDL\$GL\_FID2  
.EXTRN FDL\$GL\_FID3, FDL\$AB\_AREA\_BKZ  
.EXTRN FDL\$AL\_DATE\_TIME  
.EXTRN FDL\$AB\_STRING, FDL\$AB\_PARSED\_FAB  
.EXTRN FDL\$AB\_PARSED\_RAB

.PSECT \_FDL\$CODE, NOWRT, SHR, PIC, 2

01FC 00000  
58 00000000G 00 9E 00002  
57 00000000G 00 9E 00009  
56 00000000' 00 9E 00010  
5E 04 C2 00017  
00 01 F0 0001A  
67 80 8F 88 0001F  
67 E378 8F AA 00023  
00000000G 00 B4 00028  
05 02 A7 E8 0002E  
66 7C 00032  
08 A6 7C 00034  
00000000G 00 D4 00037 1\$:  
00000000G 00 D4 0003D

.ENTRY FDL\$INIT\_PARSE, Save R2,R3,R4,R5,R6,R7,R8 ; 0909  
MOVAB FDL\$AB\_AREA\_BKZ, R8 ;  
MOVAB FDL\$AB\_CTRL, R7 ;  
MOVAB JNL\_XAB, R6 ;  
SUBL2 #4, SP ;  
INSV #1, #0, #3, FDL\$AB\_CTRL ; 0952  
BISB2 #128, FDL\$AB\_CTRL ; 0953  
BICW2 #58232, FDL\$AB\_CTRL ; 0969  
CLRW FDL\$AB\_ITEM ; 0973  
BLBS FDL\$AB\_CTRL+2, 1\$ ; 0975  
CLRQ JNL\_XAB ; 0981  
CLRQ REVISION\_XAB ; 0983  
CLRL FDL\$GL\_STMTNUM ; 0990  
CLRL FDL\$AB\_PRICTRL ; 0991

67

03

			F8	A6	D4	00043	CLRL	CURRENT_XAB	:	0992	
			F4	A6	94	00046	CLRB	HIGHEST_AREA_NO	:	0993	
	6E		0100	8F	3C	00049	MOVZWL	#256, BYTES	:	0998	
				58	DD	0004E	PUSHL	R8	:	1000	
				04	AE	00050	PUSHAB	BYTES	:		
		00000000G	00	02	FB	00053	CALLS	#2, LIB\$GET_VM	:		
			0D	50	E8	0005A	BLBS	R0, 2\$	:		
				00000000G	8F	DD	0005D	PUSHL	#FDL\$INSVIRMEM	:	1002
		00000000G	00	01	FB	00063	CALLS	#1, LIB\$STOP	:		
			50	68	D0	0006A	MOVL	FDL\$AB AREA BKZ, R0	:	1006	
6E		00	6E	00	2C	0006D	MOVCS	#0, (SP), #0, BYTES, (R0)	:		
						00072			:		
						04	00073	RET	:	1010	

; Routine Size: 116 bytes, Routine Base: \_FDL\$CODE + 0000

```

294 1011 1 %SBTTL 'FINISH_PARSE'
295 1012 1 GLOBAL ROUTINE "FDL$$FINISH_PARSE =
296 1013 1 ++
297 1014 1
298 1015 1 Functional Description:
299 1016 1
300 1017 1 Ties up any loose ends and returns with the final status value
301 1018 1
302 1019 1 Calling Sequence:
303 1020 1
304 1021 1 status = fdl$$finish_parse()
305 1022 1
306 1023 1 Input Parameters:
307 1024 1
308 1025 1 none
309 1026 1
310 1027 1 Implicit Inputs:
311 1028 1
312 1029 1 none
313 1030 1
314 1031 1 Output Parameters:
315 1032 1
316 1033 1 none
317 1034 1
318 1035 1 Implicit Outputs:
319 1036 1
320 1037 1 none
321 1038 1
322 1039 1 Routine Value:
323 1040 1
324 1041 1 S$$ NORMAL - If everything completed corectly
325 1042 1 FDL$ WARNING - If there were warnings duing processing
326 1043 1 FDL$ _FDLERROR - If there were real problems
327 1044 1
328 1045 1 Routines Called:
329 1046 1
330 1047 1 lib$free_vm
331 1048 1
332 1049 1 Side Effects:
333 1050 1 none
334 1051 1
335 1052 1 --
336 1053 1
337 1054 2 BEGIN
338 1055 2
339 1056 2 LOCAL
340 1057 2 STATUS,
341 1058 2 XAB : REF BLOCK [ ,BYTE ],
342 1059 2 BYTES;
343 1060 2
344 1061 2 ! If successful then continue and return ok
345 1062 2 !
346 1063 2 IF .FDL$AB_CTRL [ FDL$V_STATUS ]
347 1064 2 THEN
348 1065 2 STATUS = S$$_NORMAL
349 1066 2 ELSE
350 1067 2

```

```
351 1068 2      ! If the problem was a warning then continue and return fdl$_warning
352 1069 2      ! else return imeditaly
353 1070 2
354 1071 2      IF .FDLSAB_CTRL [ FDL$V_STATUS ] EQLU STS$K_WARNING
355 1072 2      THEN
356 1073 2          STATUS = FDL$_WARNING
357 1074 2      ELSE
358 1075 2          RETURN FDL$_FDLERROR;
359 1076 2
360 1077 2      ! Travel through the xabs and fix up random things
361 1078 2      ! UNLESS THIS IS JUST A DEFAULT PARSE
362 1079 2
363 1080 3      IF (
364 1081 4      ( NOT .FDLSAB_CTRL [ FDL$V_DFLT_PRES ] )
365 1082 3      OR
366 1083 4      ( .FDLSAB_CTRL [ FDL$V_REPARSE ] )
367 1084 2      ) THEN
368 1085 3          BEGIN
369 1086 3
370 1087 3          XAB = .FDLSAB_PARSED_FAB [ FAB$_XAB ];
371 1088 3
372 1089 3      WHILE .XAB NEQU 0
373 1090 3      DO
374 1091 4          BEGIN
375 1092 4
376 1093 4          ! If this is a key xab fix the fill factors if neccary
377 1094 4
378 1095 4          IF .XAB [ XAB$_COD ] EQLU XAB$_KEY
379 1096 4          THEN
380 1097 5              BEGIN
381 1098 5
382 1099 5          ! Make sure the area numbers are valid if not simply exit
383 1100 5          ! RMS will catch it during the create
384 1101 5
385 1102 5          IF ( .XAB [ XAB$_DAN ] GTRU .HIGHEST_AREA_NO ) OR
386 1103 6          ( .XAB [ XAB$_IAN ] GTRU .HIGHEST_AREA_NO )
387 1104 5          THEN
388 1105 5              EXITLOOP;
389 1106 5
390 1107 5          ! Data level fill
391 1108 5
392 1109 6          XAB [ XAB$_DFL ] = ( .FDLSAB_AREA_BKZ [ .XAB [ XAB$_DAN ] ] * BLOCK_SIZE *
393 1110 5          .XAB [ XAB$_DFL ] ) / 100;
394 1111 5
395 1112 5          ! Index level fill
396 1113 5
397 1114 6          XAB [ XAB$_IFL ] = ( .FDLSAB_AREA_BKZ [ .XAB [ XAB$_IAN ] ] * BLOCK_SIZE *
398 1115 5          .XAB [ XAB$_IFL ] ) / 100
399 1116 4          END;
400 1117 4
401 1118 4          XAB = .XAB [ XAB$_NXT ]
402 1119 4
403 1120 3      END;
404 1121 3
405 1122 2      END;
406 1123 2
407 1124 2      ! Deallocate memory for the area bucket size array
```

```

: 408 1125 2
: 409 1126 2
: 410 1127 2
: 411 1128 3
: 412 1129 3
: 413 1130 4
: 414 1131 3
: 415 1132 3
: 416 1133 2
: 417 1134 2
: 418 1135 2
: 419 1136 2
: 420 1137 1

```

```

!
BYTES = 256;
BEGIN
LOCAL STATUS;

IF NOT ( STATUS = LIB$FREE_VM ( BYTES,FDL$AB_AREA_BKZ ) )
THEN
SIGNAL_STOP ( .STATUS );
END;

RETURN .STATUS

END;

```

				007C 00000	.ENTRY	FDL\$\$FINISH_PARSE, Save R2,R3,R4,R5,R6	: 1012
		56	00000000G	00 9E 00002	MOVAB	FDL\$AB_AREA_BKZ, R6	
		55	00000000G	00 9E 00009	MOVAB	FDL\$AB_CTRL, R5	
50	65	5E		04 C2 00010	SUBL2	#4, SP-	
		03		00 EF 00013	EXTZV	#0, #3, FDL\$, _CTRL, R0	: 1063
		05		50 E9 00018	BLBC	R0, 1\$	
		53		01 D0 0001B	MOVL	#1, STATUS	: 1065
				13 11 0001E	BRB	3\$	
				09 12 00020	BNEQ	2\$	: 1071
		53	00000000G	8F D0 00022	MOVL	#FDL\$_WARNING, STATUS	: 1073
				08 11 00029	BRB	3\$	
		50	00000000G	8F D0 0002B	MOVL	#FDL\$_FDLError, R0	: 1075
				04 00032	RET		
	04	02	A5	01 E1 00033	BBC	#1, FDL\$AB_CTRL+2, 4\$	: 1081
			6B	02 A5 E9 00038	BLBC	FDL\$AB_CTRL+2, 7\$	: 1083
		50	00000000G	00 D0 0003C	MOVL	FDL\$AB_PARSED_FAB, R0	: 1087
		50		24 A0 D0 00043	MOVL	36(R0), XAB	
				5E 13 00047	BEQL	7\$	: 1089
		15		60 91 00049	CMPB	(XAB), #21	: 1095
				53 12 0004C	BNEQ	6\$	
		52		0A A0 9A 0004E	MOVZBL	10(XAB), R2	: 1102
		51	00000000'	00 9A 00052	MOVZBL	HIGHEST_AREA_NO, R1	
		51		52 01 00059	CMPB	R2, R1	
				49 1A 0005C	BGTRU	7\$	
		51		08 A0 91 0005E	CMPB	8(XAB), R1	: 1103
				43 1A 00062	BGTRU	7\$	
		51		66 D0 00064	MOVL	FDL\$AB_AREA_BKZ, R1	: 1109
		52		6241 9A 00067	MOVZBL	(R2)[RT], R2	: 1110
		54		1C A0 3C 0006B	MOVZWL	28(XAB), R4	
		52		54 C4 0006F	MULL2	R4, R2	
52		52		09 78 00072	ASHL	#9, R2, R2	: 1109
54		52	00000064	8F C7 00076	DIVL3	#100, R2, R4	: 1110
	1C	52		A0 B0 0007E	MOVW	R4, 28(XAB)	
		52		08 A0 9A 0G082	MOVZBL	8(XAB), R2	: 1114
		51		6241 9A 00086	MOVZBL	(R2)[R1], R1	: 1115
		54		1A A0 3C 0008A	MOVZWL	26(XAB), R4	
		51		54 C4 0008E	MULL2	R4, R1	
51		51		09 78 00091	ASHL	#9, R1, R1	: 1114
52		51	00000064	8F C7 00095	DIVL3	#100, R1, R2	: 1115

	1A	A0	52	B0	0009D		MOVW	R2, 26(XAB)		
		50	04	A0	D0 000A1	6\$:	MOVL	4(XAB), XAB	:	1118
				A0	11 000A5		BRB	5\$	:	
		6E	0100	8F	3C 000A7	7\$:	MOVZWL	#256, BYTES	:	1126
				56	DD 000AC		PUSHL	R6	:	1130
			04	AE	9F 000AE		PUSHAB	BYTES	:	
00000000G		00		02	FB 000B1		CALLS	#2, LIB\$FREE_VM	:	
		09		50	E8 000B8		BLBS	STATUS, 8\$	:	
				50	DD 000BB		PUSHL	STATUS	:	1132
00000000G		00		01	FB 000BD		CALLS	#1, LIB\$STOP	:	
		50		53	D0 000C4	8\$:	MOVL	STATUS, R0	:	1135
				04	000C7		RET		:	1137

; Routine Size: 200 bytes, Routine Base: \_FDL\$CODE + 0074

```
.. 422 1138 1 %SBTTL 'LINE PARSED'
423 1139 1 GLOBAL ROUTINE FDL$$LINE_PARSED =
424 1140 1 **
425 1141 1
426 1142 1 Functional Description:
427 1143 1
428 1144 1 Main parsing routine. Called by the parse tables it in turn
429 1145 1 calls the appropriate routines to parse the fdl line.
430 1146 1
431 1147 1 Calling Sequence:
432 1148 1
433 1149 1 Called from parse tables
434 1150 1
435 1151 1 Input Parameters:
436 1152 1
437 1153 1 fdl$gl_primary - Primary code
438 1154 1
439 1155 1 Implicit Inputs:
440 1156 1 none
441 1157 1
442 1158 1 Output Parameters:
443 1159 1 none
444 1160 1
445 1161 1 Implicit Outputs:
446 1162 1 none
447 1163 1
448 1164 1 Routine Value:
449 1165 1
450 1166 1 Values returned by called routines
451 1167 1
452 1168 1 Routines Called:
453 1169 1
454 1170 1 .fdl$gl_pcall
455 1171 1 set_area_p
456 1172 1 set_date_p
457 1173 1 set_jnl_p
458 1174 1 set_acl_p not supported V4.0
459 1175 1 set_file_p
460 1176 1 set_key_p
461 1177 1 set_record_p
462 1178 1 set_access_p
463 1179 1 set_sharing_p
464 1180 1 set_connect_p
465 1181 1
466 1182 1 Side Effects:
467 1183 1 none
468 1184 1
469 1185 1 --
470 1186 1
471 1187 2 BEGIN
472 1188 2
473 1189 2 TPARSE_ARGS;
474 1190 2
475 1191 2 LOCAL
476 1192 2 STATUS;
477 1193 2
478 1194 2 STATUS = SS$NORMAL;
```



```

479 1195 2
480 1196 2 ! If we have processed some really bad stuff then dont bother
481 1197 2
482 1198 2 IF .FDLSAB_CTRL [ FDLSV_STATUS ] EQLU STSSK_ERROR
483 1199 2 THEN
484 1200 2     RETURN .STATUS;
485 1201 2
486 1202 2 ! If this is an EDF call then let them process the command
487 1203 2
488 1204 2 IF .FDLSAB_CTRL [ FDLSV_PCALL ]
489 1205 2 THEN
490 1206 2     STATUS = (.FDLSGL_PCALL)()
491 1207 2 ELSE
492 1208 2
493 1209 2     ! If this is a primary only or line comment call ignore it
494 1210 2
495 1211 2     IF NOT ( .FDLSAB_CTRL [ FDLSV_NEWPRI ] OR .FDLSAB_CTRL [ FDLSV_LINECMT ] )
496 1212 2     THEN
497 1213 2         CASE .FDLSGL_PRIMARY FROM FDLSC_ACCESS TO FDLSC_TITLE OF
498 1214 2             SET
499 1215 2
500 1216 2                 [ FDLSC_ACCESS ] : SET_ACCESS_P();
501 1217 2
502 1218 2                 [ FDLSC_ACL ]   : SET_ACL_P();
503 1219 2
504 1220 2                 [ FDLSC_AREA ]  : SET_AREA_P();
505 1221 2
506 1222 2                 [ FDLSC_CONNECT ] : SET_CONNECT_P();
507 1223 2
508 1224 2                 [ FDLSC_DATE ]  : SET_DATE_P();
509 1225 2
510 1226 2                 [ FDLSC_FILE ]  : SET_FILE_P();
511 1227 2
512 1228 2                 [ FDLSC_JNL ]   : SET_JNL_P();
513 1229 2
514 1230 2                 [ FDLSC_KEY ]   : SET_KEY_P();
515 1231 2
516 1232 2                 [ FDLSC_RECORD ] : SET_RECORD_P();
517 1233 2
518 1234 2                 [ FDLSC_SHARING ] : SET_SHARING_P();
519 1235 2
520 1236 2                 [ INRANGE ]     : 0;      ! Catch all for non usefull
521 1237 2                                     ! primaries
522 1238 2
523 1239 2             TES;
524 1240 2
525 1241 2     ! Clear new primary in case it was set
526 1242 2
527 1243 2     FDL$AB_CTRL [ FDLSV_NEWPRI ] = _CLEAR;
528 1244 2
529 1245 2     RETURN .STATUS
530 1246 2
531 1247 1     END;

```

				000C	00000		.ENTRY	FDL\$LINE PARSED, Save R2,R3	:	1139
		53	00000000G	00	9E	00002	MOVAB	FDL\$AB_CTRL, R3	:	
		52		01	D0	00009	MOVL	#1, STATUS	:	1194
02	63	03		00	ED	0000C	CMPZV	#0, #3, FDL\$AB_CTRL, #2	:	1198
				03	12	00011	BNEQ	1\$	:	
				009E	31	00013	BRW	16\$	:	
	0F	01	A3	02	E1	00016	BBC	#2, FDL\$AB_CTRL+1, 2\$	:	1204
			50	00000000G	00	D0	MOVL	FDL\$GL_PCACL, R0	:	1206
			60		00	FB	CALLS	#0, (R0)	:	
			52		50	D0	MOVL	R0, STATUS	:	
					7E	11	BRB	13\$	:	
	7A		63		05	EO	BBS	#5, FDL\$AB_CTRL, 13\$	:	1211
	7E		A3		01	EO	BBS	#1, FDL\$AB_CTRL+1, 15\$	:	
	0E		01	00000000G	00	CF	CASEL	FDL\$GL_PRIMARY, #1, #14	:	1213
0076	0076	0027		001E			.WORD	4\$-3\$,-	:	
004B	0042	0039		0030				5\$-3\$,-	:	
0066	005D	0054		0076				15\$-3\$,-	:	
	0076	0076		006F				15\$-3\$,-	:	
								6\$-3\$,-	:	
								7\$-3\$,-	:	
								8\$-3\$,-	:	
								9\$-3\$,-	:	
								15\$-3\$,-	:	
								10\$-3\$,-	:	
								11\$-3\$,-	:	
								12\$-3\$,-	:	
								14\$-3\$,-	:	
								15\$-3\$,-	:	
								15\$-3\$	:	
		00000000V	00	00	FB	00059	CALLS	#0, SET_ACCESS_P	:	1216
		00000000V	00	4F	11	00060	BRB	15\$	:	
		00000000V	00	00	FB	00062	CALLS	#0, SET_ACL_P	:	1218
		00000000V	00	46	11	00069	BRB	15\$	:	
		00000000V	00	00	FB	0006B	CALLS	#0, SET_AREA_P	:	1220
		00000000V	00	3D	11	00072	BRB	15\$	:	
		00000000V	00	00	FB	00074	CALLS	#0, SET_CONNECT_P	:	1222
		00000000V	00	34	11	0007B	BRB	15\$	:	
		00000000V	00	00	FB	0007D	CALLS	#0, SET_DATE_P	:	1224
		00000000V	00	2B	11	00084	BRB	15\$	:	
		00000000V	00	00	FB	00086	CALLS	#0, SET_FILE_P	:	1226
		00000000V	00	22	11	0008D	BRB	15\$	:	
		00000000V	00	00	FB	0008F	CALLS	#0, SET_JNL_P	:	1228
		00000000V	00	19	11	00096	BRB	15\$	:	
		00000000V	00	00	FB	00098	CALLS	#0, SET_KEY_P	:	1230
		00000000V	00	10	11	0009F	BRB	15\$	:	
		00000000V	00	00	FB	000A1	CALLS	#0, SET_RECORD_P	:	1232
		00000000V	00	07	11	000A8	BRB	15\$	:	
		00000000V	00	00	FB	000AA	CALLS	#0, SET_SHARING_P	:	1234
		63		20	8A	000B1	BICB2	#32, FDL\$AB_CTRL	:	1243
		50		52	D0	000B4	MOVL	STATUS, R0	:	1245
				04	00	000B7	RET		:	1247

; Routine Size: 184 bytes, Routine Base: \_FDL\$CODE + 013C

```

: 533      1248 1 %SBTTL 'SET_AREA_P'
: 534      1249 1 ROUTINE SET_AREA_P : NOVALUE =
: 535      1250 1 **
: 536      1251 1
: 537      1252 1 Functional Description:
: 538      1253 1
: 539      1254 1     Fill in the blanks for the allocation xab
: 540      1255 1
: 541      1256 1 Calling Sequence:
: 542      1257 1
: 543      1258 1     set_area_p()
: 544      1259 1
: 545      1260 1 Input Parameters:
: 546      1261 1     none
: 547      1262 1
: 548      1263 1 Implicit Inputs:
: 549      1264 1
: 550      1265 1     fdl$secondary - Secondary code
: 551      1266 1
: 552      1267 1 Output Parameters:
: 553      1268 1     none
: 554      1269 1
: 555      1270 1 Implicit Outputs:
: 556      1271 1     none
: 557      1272 1
: 558      1273 1 Routine Value:
: 559      1274 1     none
: 560      1275 1
: 561      1276 1 Routines Called:
: 562      1277 1
: 563      1278 1     allocate_xab
: 564      1279 1
: 565      1280 1 Side Effects:
: 566      1281 1     none
: 567      1282 1
: 568      1283 1 --
: 569      1284 1
: 570      1285 2 BEGIN
: 571      1286 2
: 572      1287 2     ! To avoid some duplication of code ...
: 573      1288 2     ! Find out if there is a current xab if not then get one
: 574      1289 2     ! OR If the current xab is not the same type or number of what we want
: 575      1290 2     ! then get a new one
: 576      1291 2
: 577      1292 3     IF ( IF .CURRENT_XAB EQLU 0
: 578      1293 3         THEN 1
: 579      1294 3         ELSE
: 580      1295 3             IF ( .CURRENT_XAB [ XAB$B_COD ] NEQ XAB$C_ALL ) OR
: 581      1296 3                 ( .CURRENT_XAB [ XAB$B_AID ] NEQ .FDL$GL_PRINUM )
: 582      1297 3             THEN 1
: 583      1298 3             ELSE 0 )
: 584      1299 2     THEN
: 585      1300 3         BEGIN
: 586      1301 3
: 587      1302 3             ! Allocate memory for the new xab
: 588      1303 3             !
: 589      1304 3             ALLOCATE_XAB ( XAB$C_ALL, .FDL$GL_PRINUM );

```

```

590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646

```

```

1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361

```

```

! Set the area number in the xab
CURRENT_XAB [ XAB$B_AID ] = .FDL$GL_PRINUM;

! If this is area 0 then copy the allocation etc. from the fab (this
! is because using areas override the fab allocation and this
! makes it look like it doesn't)
IF .CURRENT_XAB [ XAB$B_AID ] EQLU 0
THEN
BEGIN
! Copy Allocation, Bucket size and Extention
CURRENT_XAB [ XAB$S_ALQ ] = .FDL$AB_PARSED_FAB [ FAB$S_ALQ ];
CURRENT_XAB [ XAB$B_BKZ ] = .FDL$AB_PARSED_FAB [ FAB$B_BKS ];
CURRENT_XAB [ XAB$W_DEQ ] = .FDL$AB_PARSED_FAB [ FAB$W_DEQ ];
CURRENT_XAB [ XAB$S_ALQ ] = .FDL$AB_PARSED_FAB [ FAB$S_ALQ ];

IF .FDL$AB_PARSED_FAB [ FAB$B_BKS ] NEQU 0
THEN
FDL$AB_AREA_BKZ [ 0 ] = .FDL$AB_PARSED_FAB [ FAB$B_BKS ]
ELSE
FDL$AB_AREA_BKZ [ 0 ] = BUCKET_DEFAULT;

! Also get the duplicated contiguous options:
! Contiguous best try
IF .FDL$AB_PARSED_FAB [ FAB$V_CBT ]
THEN
CURRENT_XAB [ XAB$V_CBT ] = _SET;

! Contiguous
IF .FDL$AB_PARSED_FAB [ FAB$V_CTG ]
THEN
CURRENT_XAB [ XAB$V_CTG ] = _SET

END
ELSE
! Count this area
HIGHEST_AREA_NO = .HIGHEST_AREA_NO + 1

END;

! Set the fields in the area xab
CASE .FDL$GL_SECONDARY FROM FDL$C_ALLOC TO FDL$C_VOLU OF
SET
[ FDL$C_ALLOC ] : CURRENT_XAB [ XAB$S_ALQ ] = .FDL$GL_NUMBER;
[ FDL$C_BTCONT ] : CURRENT_XAB [ XAB$V_CBT ] = .FDL$GL_SWITCH;

```

```

647 1362 [ FDLSC_BKT ] : BEGIN
648 1363
649 1364 CURRENT_XAB [ XAB$B_BKZ ] = .FDL$GL_NUMBER;
650 1365
651 1366 ! Fill in the table for figuring fill numbers latter
652 1367 !
653 1368 FDL$AB_AREA_BKZ [ .FDL$GL_PRINUM ] = .FDL$GL_NUMBER
654 1369
655 1370 END;
656 1371
657 1372 [ FDLSC_CONTG ] : CURRENT_XAB [ XAB$V_CTG ] = .FDL$GL_SWITCH;
658 1373
659 1374 [ FDLSC_EXACT ] : CURRENT_XAB [ XAB$V_HRD ] = .FDL$GL_SWITCH;
660 1375
661 1376 [ FDLSC_EXTND ] : CURRENT_XAB [ XAB$W_DEQ ] = .FDL$GL_NUMBER;
662 1377
663 1378 [ FDLSC_POSI ] : CASE .FDL$GL_QUALIFIER FROM
664 1379 FDL$C_ANYPOS TO FDL$C_VIRPOS OF
665 1380 SET
666 1381 [ FDL$C_ANYPOS ] : CURRENT_XAB [ XAB$V_ONC ] = _SET;
667 1382
668 1383 [ FDL$C_CLUSPOS ] : CURRENT_XAB [ XAB$V_ONC ] = _SET;
669 1384
670 1385 [ FDL$C_CYLPOS ] : BEGIN
671 1386 CURRENT_XAB [ XAB$B_ALN ] = XAB$C_CYL;
672 1387 CURRENT_XAB [ XAB$L_LOC ] = .FDL$GL_NUMBER
673 1388 END;
674 1389
675 1390 [ FDL$C_FIDPOS ] : BEGIN
676 1391 CURRENT_XAB [ XAB$W_RF10 ] = .FDL$GL_FID1;
677 1392 CURRENT_XAB [ XAB$W_RF12 ] = .FDL$GL_FID2;
678 1393 CURRENT_XAB [ XAB$W_RF14 ] = .FDL$GL_FID3
679 1394 END;
680 1395
681 1396 [ FDL$C_FNMPOS ] : BEGIN
682 1397 FIND ID();
683 1398 CURRENT_XAB [ XAB$W_RF10 ] = .FDL$GL_FID1;
684 1399 CURRENT_XAB [ XAB$W_RF12 ] = .FDL$GL_FID2;
685 1400 CURRENT_XAB [ XAB$W_RF14 ] = .FDL$GL_FID3
686 1401 END;
687 1402
688 1403 [ FDL$C_LOGPOS ] : BEGIN
689 1404 CURRENT_XAB [ XAB$B_ALN ] = XAB$C_LBN;
690 1405 CURRENT_XAB [ XAB$L_LOC ] = .FDL$GL_NUMBER
691 1406 END;
692 1407
693 1408 [ FDL$C_NOPOS ] : CURRENT_XAB [ XAB$B_ALN ] = _CLEAR;
694 1409
695 1410 [ FDL$C_VIRPOS ] : BEGIN
696 1411 CURRENT_XAB [ XAB$B_ALN ] = XAB$C_VBN;
697 1412 CURRENT_XAB [ XAB$L_LOC ] = .FDL$GL_NUMBER
698 1413 END;
699 1414
700 1415 TES;
701 1416 [ FDL$C_VOLU ] : BEGIN
702 1417 CURRENT_XAB [ XAB$W_VOL ] = .FDL$GL_NUMBER;
703 1418

```





			04	0014E		RET		:	1378
0A	50		63	D0 0014F	24\$:	MOVL	CURRENT XAB, R0	:	1417
	A0		64	B0 00152		MOVW	FDL\$GL_NUMBER, 10(R0)	:	
		09	A0	95 00156		TSTB	9(R0)	:	1421
			04	12 00159		BNEQ	25\$	:	
09	A0		02	90 0015B		MOVB	#2, 9(R0)	:	1423
			04	0015F	25\$:	RET		:	1431

; Routine Size: 352 bytes, Routine Base: \_FDL\$CODE + 01F4



```

: 718 1432 1 %SBTTL 'SET_DATE_P'
: 719 1433 1 ROUTINE SET_DATE_P : NOVALUE =
: 720 1434 1 ++
: 721 1435 1
: 722 1436 1 Functional Description:
: 723 1437 1
: 724 1438 1     Fill in the blanks for the revision date and time xab
: 725 1439 1
: 726 1440 1 Calling Sequence:
: 727 1441 1
: 728 1442 1     set_date_p()
: 729 1443 1
: 730 1444 1 Input Parameters:
: 731 1445 1     none
: 732 1446 1
: 733 1447 1 Implicit Inputs:
: 734 1448 1
: 735 1449 1     fdl$secondary - Secondary code
: 736 1450 1
: 737 1451 1 Output Parameters:
: 738 1452 1     none
: 739 1453 1
: 740 1454 1 Implicit Outputs:
: 741 1455 1     none
: 742 1456 1
: 743 1457 1 Routine Value:
: 744 1458 1     none
: 745 1459 1
: 746 1460 1 Routines Called:
: 747 1461 1
: 748 1462 1     sys$bintim
: 749 1463 1
: 750 1464 1 Side Effects:
: 751 1465 1     none
: 752 1466 1
: 753 1467 1 --
: 754 1468 1
: 755 1469 2 BEGIN
: 756 1470 2
: 757 1471 2     ! See which xab we need
: 758 1472 2     !
: 759 1473 2     IF .FDL$GL_SECONDARY EQLU FDL$C_REV
: 760 1474 2     THEN
: 761 1475 3         BEGIN
: 762 1476 3
: 763 1477 3             ! If the revision xab has not been connected then connect it
: 764 1478 3             !
: 765 1479 3             IF .REVISION_XAB EQLU 0
: 766 1480 3             THEN
: 767 1481 3
: 768 1482 3                 ! Allocate the xab and enter it into the chain
: 769 1483 3                 !
: 770 1484 3                 REVISION_XAB = ALLOCATE_XAB ( XAB$C_RDT, 0 )
: 771 1485 3
: 772 1486 3             END
: 773 1487 2     ELSE
: 774 1488 2

```

```

775      1489 2      ! If the date xab has not been allocated then get one
776      1490 2      !
777      1491 2      IF .DATE_XAB EQLU 0
778      1492 2      THEN
779      1493 2      !
780      1494 2      ! Allocate the xab and enter it into the chain
781      1495 2      !
782      1496 2      DATE_XAB = ALLOCATE_XAB ( XAB$C_DAT, 0 );
783      1497 2      !
784      1498 2      ! Fill in the correct field
785      1499 2      !
786      1500 2      CASE .FDL$GL_SECONDARY FROM FDL$C_BACKUP TO FDL$C_REV OF
787      1501 2      SET
788      1502 3      [ FDL$C_BACKUP ] : BEGIN
789      1503 3          DATE_XAB [ XAB$L_BDT0 ] = .FDL$AL_DATE_TIME [ 0 ];
790      1504 3          DATE_XAB [ XAB$L_BDT4 ] = .FDL$AL_DATE_TIME [ 1 ];
791      1505 2          END;
792      1506 2      [ FDL$C_CREAT ] : BEGIN
793      1507 3          DATE_XAB [ XAB$L_CDT0 ] = .FDL$AL_DATE_TIME [ 0 ];
794      1508 3          DATE_XAB [ XAB$L_CDT4 ] = .FDL$AL_DATE_TIME [ 1 ];
795      1509 2          END;
796      1510 2      [ FDL$C_EXPR ] : BEGIN
797      1511 3          DATE_XAB [ XAB$L_EDT0 ] = .FDL$AL_DATE_TIME [ 0 ];
798      1512 3          DATE_XAB [ XAB$L_EDT4 ] = .FDL$AL_DATE_TIME [ 1 ];
799      1513 2          END;
800      1514 2      [ FDL$C_REV ] : BEGIN
801      1515 3          REVISION_XAB [ XAB$L_RDT0 ] = .FDL$AL_DATE_TIME [ 0 ];
802      1516 3          REVISION_XAB [ XAB$L_RDT4 ] = .FDL$AL_DATE_TIME [ 1 ];
803      1517 2          END;
804      1518 2      TES;
805      1519 2      RETURN
806      1520 2      END;
807      1521 2
808      1522 2
809      1523 2
810      1524 2
811      1525 2
812      1526 1
    
```

```

003C 00000 SET_DATE_P:
          55 00000000G 00 9E 00002      .WORD      Save R2,R3,R4,R5      : 1433
          54 00000000V 00 9E 00009      MOVAB      FDL$GL_SECONDARY, R5
          53 00000000' 00 9E 00010      MOVAB      ALLOCATE_XAB, R4
00000047 8F          65 D1 00017      MOVAB      DATE_XAB, R3
          11 12 0001E      CMPL      FDL$GL_SECONDARY, #71      : 1473
          04          A3 D5 00020      BNEQ      1$
          19 12 00023      TSTL      REVISION_XAB      : 1479
          7E          1E 7D 00025      BNEQ      2$
          64          02 FB 00028      MOVQ      #30, -(SP)      : 1484
          04 A3          50 D0 0002B      CALLS     #2, ALLOCATE_XAB
          0D 11 0002F      MOVL      R0, REVISION_XAB
          63 D5 00031 1$      BRB      2$      : 1475
          TSTL      DATE_XAB      : 1491
    
```

				09 12 00033	BNEQ	2\$		
	7E			12 7D 00035	MOVQ	#18, -(SP)	:	1496
	64			02 FB 00038	CALLS	#2, ALLOCATE_XAB	:	
	63			50 DO 0003B	MOVL	R0, DATE_XAB	:	
	52	00000000G		00 DO 0003E 2\$:	MOVL	FDL\$AL_DATE_TIME, R2	:	1503
	51	00000000G		00 DO 00045	MOVL	FDL\$AL_DATE_TIME+4, R1	:	1504
002C		03 00000044		65 CF 0004C	CASEL	FDL\$GL_SECONDARY, #68, #3	:	1500
	8F			0008 00054 3\$:	.WORD	4\$-3\$, -	:	
						5\$-3\$, -	:	
						6\$-3\$, -	:	
						7\$-3\$	:	
				63 DO 0005C 4\$:	MOVL	DATE_XAB, R0	:	1503
	24	A0		52 DO 0005F	MOVL	R2, 36(R0)	:	
	28	A0		51 DO 00063	MOVL	R1, 40(R0)	:	1504
				04 00067	RET		:	
				63 DO 00068 5\$:	MOVL	DATE_XAB, R0	:	1508
	14	A0		52 DO 0006B	MOVL	R2, 20(R0)	:	
	18	A0		51 DO 0006F	MOVL	R1, 24(R0)	:	1509
				04 00073	RET		:	
				63 DO 00074 6\$:	MOVL	DATE_XAB, R0	:	1513
	1C	A0		52 DO 00077	MOVL	R2, 28(R0)	:	
	20	A0		51 DO 0007B	MOVL	R1, 32(R0)	:	1514
				04 0007F	RET		:	
				04 A3 DO 00080 7\$:	MOVL	REVISION_XAB, R0	:	1518
	0C	A0		52 DO 00084	MOVL	R2, 12(R0)	:	
	10	A0		51 DO 00088	MOVL	R1, 16(R0)	:	1519
				04 0008C	RET		:	1526

; Routine Size: 141 bytes, Routine Base: \_FDL\$CODE + 0354

```

814 1527 1 %SBTTL 'SET_JNL_P'
815 1528 1 ROUTINE SET_JNL_P : NOVALUE =
816 1529 1 **
817 1530 1
818 1531 1 Functional Description:
819 1532 1
820 1533 1     Fill in the blanks for the journal xab
821 1534 1
822 1535 1 Calling Sequence:
823 1536 1
824 1537 1     set_jnl_p()
825 1538 1
826 1539 1 Input Parameters:
827 1540 1     none
828 1541 1
829 1542 1 Implicit Inputs:
830 1543 1
831 1544 1     fdl$secondary - Secondary code
832 1545 1
833 1546 1 Output Parameters:
834 1547 1     none
835 1548 1
836 1549 1 Implicit Outputs:
837 1550 1     none
838 1551 1
839 1552 1 Routine Value:
840 1553 1     none
841 1554 1
842 1555 1 Routines Called:
843 1556 1
844 1557 1     none
845 1558 1
846 1559 1 Side Effects:
847 1560 1     none
848 1561 1
849 1562 1 --
850 1563 1
851 1564 2 BEGIN
852 1565 2
853 1566 2     ! If the xab has not been connected, then connect it
854 1567 2     !
855 1568 2     IF .JNL_XAB EQLU 0
856 1569 2     THEN
857 1570 2         ! Allocate the xab and enter it into the chain
858 1571 2         !
859 1572 2         JNL_XAB = ALLOCATE_XAB ( XAB$C_JNL, 0 );
860 1573 2
861 1574 2     ! Fill in the correct field
862 1575 2     !
863 1576 2     CASE .FDL$GL_SECONDARY FROM FDL$C_AFTIM TO FDL$C_RU OF
864 1577 2     SET
865 1578 2         [ FDL$C_AFTIM ] : JNL_XAB [ XAB$V_AI ] = .FDL$GL_SWITCH;
866 1579 2
867 1580 2         [ FDL$C_AFTNAM ] : BEGIN
868 1581 2
869 1582 2         ! Allocate a buffer for the string and copy to it
870 1583 2

```

```

871 1584 JNL_XAB [ XABSL_AIA ] =
872 1585     -FDL$$GET_VMT .FDL$AB_STRING [ DSC$W_LENGTH ] );
873 1586
874 1587 CHSMOVE( .FDL$AB_STRING [ DSC$W_LENGTH ],
875 1588         .FDL$AB_STRING [ DSC$A_POINTER ],
876 1589         .JNL_XAB [ XABSL_AIA ] );
877 1590
878 1591 JNL_XAB [ XABSB_AIS ] =
879 1592     .FDL$AB_STRING [ DSC$W_LENGTH ]
880 1593 END;
881 1594
882 1595 [ FDL$C_AUDIT ] : JNL_XAB [ XABSV_AT ] = .FDL$GL_SWITCH;
883 1596
884 1597 [ FDL$C_AUDNAM ] : BEGIN
885 1598     ; Allocate a buffer for the string and copy to it
886 1599     ;
887 1600     JNL_XAB [ XABSL_ATA ] =
888 1601         -FDL$$GET_VMT .FDL$AB_STRING [ DSC$W_LENGTH ] );
889 1602
890 1603     CHSMOVE( .FDL$AB_STRING [ DSC$W_LENGTH ],
891 1604             .FDL$AB_STRING [ DSC$A_POINTER ],
892 1605             .JNL_XAB [ XABSL_ATA ] );
893 1606
894 1607     JNL_XAB [ XABSB_ATS ] =
895 1608         .FDL$AB_STRING [ DSC$W_LENGTH ]
896 1609     END;
897 1610
898 1611 [ FDL$C_BEFIG ] : JNL_XAB [ XABSV_BI ] = .FDL$GL_SWITCH;
899 1612
900 1613 [ FDL$C_BEFIGNAM ] : BEGIN
901 1614     ; Allocate a buffer for the string and copy to it
902 1615     ;
903 1616     JNL_XAB [ XABSL_BIA ] =
904 1617         -FDL$$GET_VMT .FDL$AB_STRING [ DSC$W_LENGTH ] );
905 1618
906 1619     CHSMOVE( .FDL$AB_STRING [ DSC$W_LENGTH ],
907 1620             .FDL$AB_STRING [ DSC$A_POINTER ],
908 1621             .JNL_XAB [ XABSL_BIA ] );
909 1622
910 1623     JNL_XAB [ XABSB_BIS ] =
911 1624         .FDL$AB_STRING [ DSC$W_LENGTH ]
912 1625     END;
913 1626
914 1627 [ FDL$C_RU ] : BEGIN
915 1628     ; Set the recovery unit bit according to what
916 1629     ; was specified
917 1630     ;
918 1631     JNL_XAB [ XABSV_RU ] = _CLEAR;
919 1632     JNL_XAB [ XABSV_ONLY_RU ] = _CLEAR;
920 1633     JNL_XAB [ XABSV_NEVER_RU ] = _CLEAR;
921 1634
922 1635     IF .FDL$GL_QUALIFIER EQLU FDL$C_IF_IN
923 1636     THEN
924 1637         JNL_XAB [ XABSV_RU ] = _SET
925 1638
926 1639     ELSE IF .FDL$GL_QUALIFIER EQLU FDL$C_NEC
927 1640     THEN

```



	20	B6		60	57	28	0008C		MOV C3	R7, (R0), @32(R6)	:		
			1C	A6	57	90	00091		MOV B	R7, 28(R6)	:	1608	
						04	00095		RET		:	1607	
08	A2			02	6A	F0	00096	7\$:	INSV	FDL\$GL_SWITCH, #2, #1, 8(R2)	:	1611	
						04	0009C		RET		:		
				7E	68	3C	0009D	8\$:	MOVZWL	FDL\$AB_STRING, -(SP)	:	1617	
				6B	01	FB	000A0		CALLS	#1, FDL\$GET_VM	:		
			10	A2	50	D0	000A3		MOVL	R0, 16(R2)	:		
				57	68	3C	000A7		MOVZWL	FDL\$AB_STRING, R7	:	1619	
				50	04	A8	000AA		MOVL	FDL\$AB_STRING+4, R0	:	1620	
				56	69	D0	000AE		MOVL	JNL_XAB, R6	:	1621	
	10	B6		60	57	28	000B1		MOV C3	R7, (R0), @16(R6)	:		
			0C	A6	57	90	000B6		MOV B	R7, 12(R6)	:	1624	
						04	000BA		RET		:	1623	
				51	08	A2	9E	000BB	9\$:	MOVAB	8(R2), R1	:	1631
				61	23	8A	000BF		BICB2	#35, (R1)	:	1633	
				50	00000000G	00	D0	000C2		MOVL	FDL\$GL_QUALIFIER, R0	:	1635
				13		50	D1	000C9		CMPL	R0, #19	:	
						04	12	000CC		BNEQ	10\$	:	
				61		02	88	000CE		BISB2	#2, (R1)	:	1637
						04	000D1		RET		:		
				14		50	D1	000D2	10\$:	CMPL	R0, #20	:	1639
						04	12	000D5		BNEQ	11\$	:	
				61		01	88	000D7		BISB2	#1, (R1)	:	1641
						04	000DA		RET		:		
				15		50	D1	000DB	11\$:	CMPL	R0, #21	:	1643
						03	12	000DE		BNEQ	12\$	:	
				61		20	88	000E0		BISB2	#32, (R1)	:	1645
						04	000E3	12\$:	RET		:	1653	

; Routine Size: 228 bytes, Routine Base: \_FDL\$CODE + 03E1

```

: 942 1654 1 %SBTTL 'SET_ACL_P'
: 943 1655 1 ROUTINE SET_ACL_P : NOVALUE =
: 944 1656 1 **
: 945 1657 1
: 946 1658 1 Functional Description:
: 947 1659 1
: 948 1660 1     Fill in the blanks for the ACL xab
: 949 1661 1
: 950 1662 1 Calling Sequence:
: 951 1663 1
: 952 1664 1     set_acl_p()
: 953 1665 1
: 954 1666 1 Input Parameters:
: 955 1667 1     none
: 956 1668 1
: 957 1669 1 Implicit Inputs:
: 958 1670 1
: 959 1671 1     fdl$secondary - Secondary code
: 960 1672 1
: 961 1673 1 Output Parameters:
: 962 1674 1     none
: 963 1675 1
: 964 1676 1 Implicit Outputs:
: 965 1677 1     none
: 966 1678 1
: 967 1679 1 Routine Value:
: 968 1680 1     none
: 969 1681 1
: 970 1682 1 Routines Called:
: 971 1683 1
: 972 1684 1     none
: 973 1685 1
: 974 1686 1 Side Effects:
: 975 1687 1     none
: 976 1688 1
: 977 1689 1 --
: 978 1690 1
: 979 1691 2 BEGIN
: 980 1692 2
: 981 1693 2 : nop until there exists an ACLXAB
: 982 1694 2
: 983 1695 2 RETURN
: 984 1696 2
: 985 1697 1 END;

```

```

0000 00000 SET_ACL_P:
04 00002 .WORD Save nothing
RET

```

```

: 1655
: 1697

```

: Routine Size: 3 bytes, Routine Base: \_FDL\$CODE + 04C5



```

1698 1 %SBTTL 'SET_FILE_P'
1699 1 ROUTINE SET_FILE_P : NOVALUE =
1700 1 **
1701 1
1702 1 Functional Description:
1703 1
1704 1     Fill in the blanks for the fab
1705 1
1706 1 Calling Sequence:
1707 1
1708 1     set_file_p()
1709 1
1710 1 Input Parameters:
1711 1     none
1712 1
1713 1 Implicit Inputs:
1714 1
1715 1     fdl$secondary - Secondary code
1716 1
1717 1 Output Parameters:
1718 1     none
1719 1
1720 1 Implicit Outputs:
1721 1     none
1722 1
1723 1 Routine Value:
1724 1
1725 1     $$$_NORMAL or error from set_prot
1726 1
1727 1 Routines Called:
1728 1
1729 1     fdl$$get_vm
1730 1     set_prot
1731 1
1732 1 Side Effects:
1733 1     none
1734 1
1735 1 --
1736 1
1737 2 BEGIN
1738 2
1739 2 REGISTER
1740 2     PARSED_FAB : REF BLOCK [ .BYTE ];
1741 2
1742 2 PARSED_FAB = .FDL$AB_PARSED_FAB;
1743 2
1744 2 ! Set the fab according to the secondary parsed
1745 2 !
1746 2 SELECT .FDL$GL_SECONDARY OF
1747 2 SET
1748 2     [ FDL$C_ALL ] : PARSED_FAB [ FAB$ALQ ] = .FDL$GL_NUMBER;
1749 2
1750 2     [ FDL$C_BKTUP ] : 0;
1751 2
1752 2     [ FDL$C_BTC ] : PARSED_FAB [ FAB$V_CBT ] = .FDL$GL_SWITCH;
1753 2
1754 2     [ FDL$C_BKTSIZ ] : BEGIN

```

```

: 1044 1755
: 1045 1756 PARSED_FAB [ FAB$B_BKS ] = .FDL$GL_NUMBER;
: 1046 1757
: 1047 1758 ! Stuff the bucket size into the array for latter
: 1048 1759 !
: 1049 1760 FDL$AB_AREA_BKZ [ 0 ] = .FDL$GL_NUMBER
: 1050 1761
: 1051 1762 END;
: 1052 1763
: 1053 1764 [ FDL$C_CLUSIZ ] : 0;
: 1054 1765
: 1055 1766 [ FDL$C_FCTX ] : PARSED_FAB [ FAB$S_L_CTX ] = .FDL$GL_NUMBER;
: 1056 1767
: 1057 1768 [ FDL$C_CONT ] : PARSED_FAB [ FAB$V_CTG ] = .FDL$GL_SWITCH;
: 1058 1769
: 1059 1770 [ FDL$C_CIF ] : PARSED_FAB [ FAB$V_CIF ] = .FDL$GL_SWITCH;
: 1060 1771
: 1061 1772 [ FDL$C_DFNAM ] : BEGIN
: 1062 1773
: 1063 1774 ! Allocate a buffer for the string and copy it into it
: 1064 1775 !
: 1065 1776 PARSED_FAB [ FAB$S_L_DNA ] =
: 1066 1777 FDL$GET_VM( ".FDL$AB_STRING [ DSC$W_LENGTH ] );
: 1067 1778
: 1068 1779 CH$MOVE( .FDL$AB_STRING [ DSC$W_LENGTH ],
: 1069 1780 .FDL$AB_STRING [ DSC$A_POINTER ],
: 1070 1781 .PARSED_FAB [ FAB$S_L_DNA ] );
: 1071 1782
: 1072 1783 PARSED_FAB [ FAB$B_DNS ] =
: 1073 1784 .FDL$AB_STRING [ DSC$W_LENGTH ]
: 1074 1785
: 1075 1786 END;
: 1076 1787 [ FDL$C_DEFWRT ] : PARSED_FAB [ FAB$V_DFW ] = .FDL$GL_SWITCH;
: 1077 1788
: 1078 1789 [ FDL$C_DOC ] : PARSED_FAB [ FAB$V_DLT ] = .FDL$GL_SWITCH;
: 1079 1790
: 1080 1791 [ FDL$C_DIR ] : PARSED_FAB [ FAB$V_TMP ] = .FDL$GL_SWITCH;
: 1081 1792
: 1082 1793 ! not supported V4.0
: 1083 1794 ! [ FDL$C_EODEL ] : PARSED_FAB [ FAB$V_EDL ] = .FDL$GL_SWITCH;
: 1084 1795
: 1085 1796 [ FDL$C_EXTEN ] : PARSED_FAB [ FAB$W_DEQ ] = .FDL$GL_NUMBER;
: 1086 1797
: 1087 1798 [ FDL$C_GBC ] : PARSED_FAB [ FAB$W_GBC ] = .FDL$GL_NUMBER;
: 1088 1799
: 1089 1800 [ FDL$C_MTBLSIZ ] : PARSED_FAB [ FAB$W_BLS ] = .FDL$GL_NUMBER;
: 1090 1801
: 1091 1802 [ FDL$C_MTCP ] : PARSED_FAB [ FAB$V_POS ] = .FDL$GL_SWITCH;
: 1092 1803
: 1093 1804 [ FDL$C_MTNEF ] : PARSED_FAB [ FAB$V_NEF ] = .FDL$GL_SWITCH;
: 1094 1805
: 1095 1806 [ FDL$C_MTPRO ] : SET_PROT();
: 1096 1807
: 1097 1808 [ FDL$C_MTREW ] : PARSED_FAB [ FAB$V_RWO ] = .FDL$GL_SWITCH;
: 1098 1809
: 1099 1810 [ FDL$C_MTRWC ] : PARSED_FAB [ FAB$V_RWC ] = .FDL$GL_SWITCH;
: 1100 1811

```

```

: 1101 1812 2 [ FDL$C_MAXRECN]: PARSED_FAB [ FAB$L_MRN ] = .FDL$GL_NUMBER;
: 1102 1813 2
: 1103 1814 2 [ FDL$C_MAXVER] : PARSED_FAB [ FAB$V_MXV ] = .FDL$GL_SWITCH;
: 1104 1815 2
: 1105 1816 2 [ FDL$C_NAME ] : BEGIN
: 1106 1817 2 ! Check for non-null name string
: 1107 1818 2 !
: 1108 1819 2 IF .FDL$AB_STRING [ DSC$W_LENGTH] NEQ 0
: 1109 1820 2 THEN
: 1110 1821 2 BEGIN
: 1111 1822 2 ! Allocate a buffer for the string and copy it
: 1112 1823 2 !
: 1113 1824 2 ! PARSED_FAB [ FAB$L_FNA ] =
: 1114 1825 2 ! FDL$GET_VM( .FDL$AB_STRING [ DSC$W_LENGTH ] );
: 1115 1826 2 !
: 1116 1827 2 ! CH$MOVE( .FDL$AB_STRING [ DSC$W_LENGTH ],
: 1117 1828 2 ! .FDL$AB_STRING [ DSC$A_POINTER ],
: 1118 1829 2 ! .PARSED_FAB [ FAB$L_FNA ] );
: 1119 1830 2 END;
: 1120 1831 2 PARSED_FAB [ FAB$B_FNS ] =
: 1121 1832 2 ! .FDL$AB_STRING [ DSC$W_LENGTH ]
: 1122 1833 2 END;
: 1123 1834 2
: 1124 1835 2 [ FDL$C_NFS ] : PARSED_FAB [ FAB$V_NFS ] = .FDL$GL_SWITCH;
: 1125 1836 2
: 1126 1837 2 [ FDL$C_ORG ] : PARSED_FAB [ FAB$B_ORG ] = .FDL$GL_QUALIFIER;
: 1127 1838 2
: 1128 1839 2 [ FDL$C_OFP ] : PARSED_FAB [ FAB$V_OFP ] = .FDL$GL_SWITCH;
: 1129 1840 2
: 1130 1841 2 [ FDL$C_OWNER ] : SET_PROT();
: 1131 1842 2
: 1132 1843 2 [ FDL$C_POC ] : PARSED_FAB [ FAB$V_SPL ] = .FDL$GL_SWITCH;
: 1133 1844 2
: 1134 1845 2 [ FDL$C_PROT ] : SET_PROT();
: 1135 1846 2
: 1136 1847 2 [ FDL$C_READC ] : PARSED_FAB [ FAB$V_RCK ] = .FDL$GL_SWITCH;
: 1137 1848 2
: 1138 1849 2 [ FDL$C_REVISN ]: BEGIN
: 1139 1850 2 !
: 1140 1851 2 ! If the revision xab has not been connected then connect it
: 1141 1852 2 !
: 1142 1853 2 ! IF .REVISION_XAB EQLU 0
: 1143 1854 2 ! THEN
: 1144 1855 2 !
: 1145 1856 2 ! ! Allocate the xab and enter it into the chain
: 1146 1857 2 ! !
: 1147 1858 2 ! REVISION_XAB = ALLOCATE_XAB ( XAB$C_RDT, 0 );
: 1148 1859 2 !
: 1149 1860 2 ! REVISION_XAB [ XAB$W_RVN ] = .FDL$GL_NUMBER
: 1150 1861 2 !
: 1151 1862 2 ! END;
: 1152 1863 2
: 1153 1864 2 [ FDL$C_SQO ] : PARSED_FAB [ FAB$V_SQO ] = .FDL$GL_SWITCH;
: 1154 1865 2
: 1155 1866 2 [ FDL$C_SOC ] : PARSED_FAB [ FAB$V_SCF ] = .FDL$GL_SWITCH;
: 1156 1867 2
: 1157 1868 2 [ FDL$C_SUPER ] : PARSED_FAB [ FAB$V_SUP ] = .FDL$GL_SWITCH;

```



			35	A6	58	90	000A9	MOVW	R8, 53(PARSED_FAB)	1784
			00000050	8F	57	D1	000AD	7\$: CMPL	R7, #80	1787
04	A6	01		05	06	12	000B4	BNEQ	8\$	
			00000051	8F	69	F0	000B6	INSV	FDL\$GL_SWITCH, #5, #1, 4(PARSED_FAB)	
					57	D1	000BC	8\$: CMPL	R7, #8T	1789
					06	12	000C3	BNEQ	9\$	
05	A6	01		07	69	F0	000C5	INSV	FDL\$GL_SWITCH, #7, #1, 5(PARSED_FAB)	
			00000052	8F	57	D1	000CB	9\$: CMPL	R7, #82	1791
					06	12	000D2	BNEQ	10\$	
04	A6	01		03	69	F0	000D4	INSV	FDL\$GL_SWITCH, #3, #1, 4(PARSED_FAB)	
			00000054	8F	57	D1	000DA	10\$: CMPL	R7, #8Z	1796
					04	12	000E1	BNEQ	11\$	
			14	A6	6A	B0	000E3	MOVW	FDL\$GL_NUMBER, 20(PARSED_FAB)	
			00000055	8F	57	D1	000E7	11\$: CMPL	R7, #85	1798
					04	12	000EE	BNEQ	12\$	
			48	A6	6A	B0	000F0	MOVW	FDL\$GL_NUMBER, 72(PARSED_FAB)	
			00000056	8F	57	D1	000F4	12\$: CMPL	R7, #86	1800
					04	12	000FB	BNEQ	13\$	
			3C	A6	6A	B0	000FD	MOVW	FDL\$GL_NUMBER, 60(PARSED_FAB)	
			00000057	8F	57	D1	00101	13\$: CMPL	R7, #87	1802
					06	12	00108	BNEQ	14\$	
05	A6	01		00	69	F0	0010A	INSV	FDL\$GL_SWITCH, #0, #1, 5(PARSED_FAB)	
			00000058	8F	57	D1	00110	14\$: CMPL	R7, #88	1804
					06	12	00117	BNEQ	15\$	
05	A6	01		02	69	F0	00119	INSV	FDL\$GL_SWITCH, #2, #1, 5(PARSED_FAB)	
			00000059	8F	57	D1	0011F	15\$: CMPL	R7, #89	1806
					07	12	00126	BNEQ	16\$	
			00000000V	00	00	FB	00128	CALLS	#0, SET_PROT	
			0000005A	8F	57	D1	0012F	16\$: CMPL	R7, #90	1808
					06	12	00136	BNEQ	17\$	
04	A6	01		07	69	F0	00138	INSV	FDL\$GL_SWITCH, #7, #1, 4(PARSED_FAB)	
			0000005B	8F	57	D1	0013E	17\$: CMPL	R7, #9T	1810
					06	12	00145	BNEQ	18\$	
05	A6	01		03	69	F0	00147	INSV	FDL\$GL_SWITCH, #3, #1, 5(PARSED_FAB)	
			0000005C	8F	57	D1	0014D	18\$: CMPL	R7, #92	1812
					04	12	00154	BNEQ	19\$	
			38	A6	6A	D0	00156	MOVL	FDL\$GL_NUMBER, 56(PARSED_FAB)	
			0000005D	8F	57	D1	0015A	19\$: CMPL	R7, #93	1814
					06	12	00161	BNEQ	20\$	
04	A6	01		01	69	F0	00163	INSV	FDL\$GL_SWITCH, #1, #1, 4(PARSED_FAB)	
			0000005E	8F	57	D1	00169	20\$: CMPL	R7, #9Z	1816
					1F	12	00170	BNEQ	22\$	
				50	6B	3C	00172	MOVZWL	FDL\$AB_STRING, R0	1819
					16	13	00175	BEQL	R0	
			00000000V	00	50	DD	00177	PUSHL	R0	1825
			2C	A6	01	FB	00179	CALLS	#1, FDL\$\$GET_VM	
					50	D0	00180	MOVL	R0, 44(PARSED_FAB)	
				04	AB	D0	00184	MOVL	FDL\$AB_STRING+4, R0	1828
				60	6B	28	00188	MOV3	FDL\$AB_STRING, (R0), @44(PARSED_FAB)	1829
			34	A6	6B	90	0018D	21\$: MOVW	FDL\$AB_STRING, 52(PARSED_FAB)	1832
			00000060	8F	57	D1	00191	22\$: CMPL	R7, #96	1835
					06	12	00198	BNEQ	23\$	
06	A6	01		00	69	F0	0019A	INSV	FDL\$GL_SWITCH, #0, #1, 6(PARSED_FAB)	
			00000062	8F	57	D1	001A0	23\$: CMPL	R7, #98	1837
					08	12	001A7	BNEQ	24\$	
			1D	A6	00	90	001A9	MOVW	FDL\$GL_QUALIFIER, 29(PARSED_FAB)	
			00000061	8F	57	D1	001B1	24\$: CMPL	R7, #97	1839

07	A6	01	05	06 12 001B8	BNEQ	25\$	:	:
			8F	69 F0 001BA	INSV	FDL\$GL_SWITCH, #5, #1, 7(PARSED_FAB)	:	:
			00000063	57 D1 001C0	CMPL	R7, #99	:	1841
				07 12 001C7	BNEQ	26\$	:	:
			00000000V	00 FB 001C9	CALLS	#0, SET_PROT	:	:
			8F	57 D1 001D0	CMPL	R7, #100	:	1843
05	A6	01	05	06 12 001D7	BNEQ	27\$	:	:
			8F	69 F0 001D9	INSV	FDL\$GL_SWITCH, #5, #1, 5(PARSED_FAB)	:	:
			00000065	57 D1 001DF	CMPL	R7, #101	:	1845
				07 12 001E6	BNEQ	28\$	:	:
			00000000V	00 FB 001E8	CALLS	#0, SET_PROT	:	:
			8F	57 D1 001EF	CMPL	R7, #102	:	1847
06	A6	01	07	06 12 001F6	BNEQ	29\$	:	:
			8F	69 F0 001F8	INSV	FDL\$GL_SWITCH, #7, #1, 6(PARSED_FAB)	:	:
			00000067	57 D1 001FE	CMPL	R7, #103	:	1849
				24 12 00205	BNEQ	31\$	:	:
			00000000'	00 D5 00207	TSTL	REVISION_XAB	:	1853
				11 12 0020D	BNEQ	30\$	:	:
			7E	1E 7D 0020F	MOVQ	#30, -(SP)	:	1858
			00000000V	02 FB 00212	CALLS	#2, ALLOCATE_XAB	:	:
			00000000'	50 D0 00219	MOVL	R0, REVISION_XAB	:	:
				00 D0 00220	MOVL	REVISION_XAB, R0	:	1860
			08	6A B0 00227	MOVW	FDL\$GL_NUMBER, 8(R0)	:	:
			8F	57 D1 0022B	CMPL	R7, #104	:	1864
04	A6	01	06	06 12 00232	BNEQ	32\$	:	:
			8F	69 F0 00234	INSV	FDL\$GL_SWITCH, #6, #1, 4(PARSED_FAB)	:	:
			00000069	57 D1 0023A	CMPL	R7, #105	:	1866
05	A6	01	06	06 12 00241	BNEQ	33\$	:	:
			8F	69 F0 00243	INSV	FDL\$GL_SWITCH, #6, #1, 5(PARSED_FAB)	:	:
			0000006A	57 D1 00249	CMPL	R7, #106	:	1868
04	A6	01	02	06 12 00250	BNEQ	34\$	:	:
			8F	69 F0 00252	INSV	FDL\$GL_SWITCH, #2, #1, 4(PARSED_FAB)	:	:
			0000006B	57 D1 00258	CMPL	R7, #107	:	1870
04	A6	01	04	06 12 0025F	BNEQ	35\$	:	:
			8F	69 F0 00261	INSV	FDL\$GL_SWITCH, #4, #1, 4(PARSED_FAB)	:	:
			0000006C	57 D1 00267	CMPL	R7, #108	:	1872
07	A6	01	04	06 12 0026E	BNEQ	36\$	:	:
			8F	69 F0 00270	INSV	FDL\$GL_SWITCH, #4, #1, 7(PARSED_FAB)	:	:
			0000006D	57 D1 00276	CMPL	R7, #109	:	1874
06	A6	01	01	06 12 0027D	BNEQ	37\$	:	:
			8F	69 F0 0027F	INSV	FDL\$GL_SWITCH, #1, #1, 6(PARSED_FAB)	:	:
			0000006E	57 D1 00285	CMPL	R7, #110	:	1876
			1C	04 12 0028C	BNEQ	38\$	:	:
			A6	6A 90 0028E	MOVB	FDL\$GL_NUMBER, 28(PARSED_FAB)	:	:
			8F	57 D1 00292	CMPL	R7, #111	:	1878
05	A6	01	01	06 12 00299	BNEQ	39\$	:	:
			01	69 F0 0029B	INSV	FDL\$GL_SWITCH, #1, #1, 5(PARSED_FAB)	:	:
				04 002A1	RET		:	1884

; Routine Size: 674 bytes, Routine Base: \_FDL\$CODE + 04C8

```

1175 1885 1 %SBTTL 'SET_KEY_P'
1176 1886 1 ROUTINE SET_KEY_P : NOVALUE =
1177 1887 1 ++
1178 1888 1
1179 1889 1 Functional Description:
1180 1890 1
1181 1891 1     Fill in the blanks for the key xab
1182 1892 1
1183 1893 1 Calling Sequence:
1184 1894 1
1185 1895 1     set_key_p()
1186 1896 1
1187 1897 1 Input Parameters:
1188 1898 1     none
1189 1899 1
1190 1900 1 Implicit Inputs:
1191 1901 1
1192 1902 1     fdl$secondary - Secondary code
1193 1903 1
1194 1904 1 Output Parameters:
1195 1905 1     none
1196 1906 1
1197 1907 1 Implicit Outputs:
1198 1908 1     none
1199 1909 1
1200 1910 1 Routine Value:
1201 1911 1     none
1202 1912 1
1203 1913 1 Routines Called:
1204 1914 1
1205 1915 1     allocate_xab
1206 1916 1
1207 1917 1 Side Effects:
1208 1918 1     none
1209 1919 1
1210 1920 1 --
1211 1921 1
1212 1922 2 BEGIN
1213 1923 2
1214 1924 2 ! Find out if there is a current xab if not then get one
1215 1925 2 !
1216 1926 2 IF .CURRENT_XAB EQL 0
1217 1927 2 THEN
1218 1928 2     BEGIN
1219 1929 2
1220 1930 2     ALLOCATE_XAB ( XAB$C_KEY, .FDL$GL_PRINUM );
1221 1931 2
1222 1932 2     CURRENT_XAB [ XAB$B_REF ] = .FDL$GL_PRINUM
1223 1933 2
1224 1934 2     END
1225 1935 2 ELSE
1226 1936 2
1227 1937 2 ! If the current xab is not the same type or number of what we want
1228 1938 2 ! then get a new one
1229 1939 2 !
1230 1940 2 IF ( .CURRENT_XAB [ XAB$B_COD ] NEQ XAB$C_KEY ) OR
1231 1941 3 ( .CURRENT_XAB [ XAB$B_REF ] NEQ .FDL$GL_PRINUM )

```

```

: 1232 1942 2 THEN
: 1233 1943 3 BEGIN
: 1234 1944 3
: 1235 1945 3 ALLOCATE_XAB ( XAB$C_KEY, .FDL$GL_PRINUM );
: 1236 1946 3
: 1237 1947 3 CURRENT_XAB [ XAB$B_REF ] = .FDL$GL_PRINUM
: 1238 1948 3
: 1239 1949 2 END;
: 1240 1950 2
: 1241 1951 2 ! Set the key xab fields
: 1242 1952 2 !
: 1243 1953 2 CASE .FDL$GL_SECONDARY FROM FDL$C_CHANGE TO FDL$C_SEGTYPE OF
: 1244 1954 2 SET
: 1245 1955 2 [ FDL$C_CHANGE ] : CURRENT_XAB [ XAB$V_CHG ] = .FDL$GL_SWITCH;
: 1246 1956 2
: 1247 1957 2 [ FDL$C_DAREA ] : CURRENT_XAB [ XAB$B_DAN ] = .FDL$GL_NUMBER;
: 1248 1958 2
: 1249 1959 2 [ FDL$C_DFILL ] : CURRENT_XAB [ XAB$W_DFL ] = .FDL$GL_NUMBER;
: 1250 1960 2
: 1251 1961 2 [ FDL$C_DATKC ] : CURRENT_XAB [ XAB$V_KEY_NCMPR ] = NOT .FDL$GL_SWITCH;
: 1252 1962 2
: 1253 1963 2 [ FDL$C_DATRC ] : CURRENT_XAB [ XAB$V_DAT_NCMPR ] = NOT .FDL$GL_SWITCH;
: 1254 1964 2
: 1255 1965 2 [ FDL$C_DUPS ] : CURRENT_XAB [ XAB$V_DUP ] = .FDL$GL_SWITCH;
: 1256 1966 2
: 1257 1967 2 [ FDL$C_IAREA ] : CURRENT_XAB [ XAB$B_IAN ] = .FDL$GL_NUMBER;
: 1258 1968 2
: 1259 1969 2 [ FDL$C_IDXC ] : CURRENT_XAB [ XAB$V_IDX_NCMPR ] = NOT .FDL$GL_SWITCH;
: 1260 1970 2
: 1261 1971 2 [ FDL$C_IFILL ] : CURRENT_XAB [ XAB$W_IFL ] = .FDL$GL_NUMBER;
: 1262 1972 2
: 1263 1973 3 [ FDL$C_KYNAME ] : BEGIN
: 1264 1974 3 CURRENT_XAB [ XAB$L_KNM ] = FDL$$GET_VM ( 32 );
: 1265 1975 3 CH$COPY( .FDL$AB_STRING [ DSC$W_LENGTH ],
: 1266 1976 3 .FDL$AB_STRING [ DSC$A_POINTER ],
: 1267 1977 3 SPACE, 32,
: 1268 1978 3 .CURRENT_XAB [ XAB$L_KNM ] )
: 1269 1979 2 END;
: 1270 1980 2
: 1271 1981 2 [ FDL$C_LAREA ] : CURRENT_XAB [ XAB$B_LAN ] = .FDL$GL_NUMBER;
: 1272 1982 2
: 1273 1983 2 [ FDL$C_NULL ] : CURRENT_XAB [ XAB$V_NUL ] = .FDL$GL_SWITCH;
: 1274 1984 2
: 1275 1985 2 [ FDL$C_NULLVAL ] : CURRENT_XAB [ XAB$B_NUL ] = .FDL$GL_QUALIFIER;
: 1276 1986 2
: 1277 1987 2 [ FDL$C_PROL ] : IF .CURRENT_XAB [ XAB$B_REF ] EQLU 0
: 1278 1988 2 THEN
: 1279 1989 2 CURRENT_XAB [ XAB$B_PROLOG ] = .FDL$GL_NUMBER;
: 1280 1990 2
: 1281 1991 2 [ FDL$C_SEGLEN ] : CASE .FDL$GL_SECNUM FROM 0 TO 7 OF
: 1282 1992 2 SET
: 1283 1993 2 [ 0 ] : CURRENT_XAB [ XAB$B_SIZ0 ] = .FDL$GL_NUMBER;
: 1284 1994 2 [ 1 ] : CURRENT_XAB [ XAB$B_SIZ1 ] = .FDL$GL_NUMBER;
: 1285 1995 2 [ 2 ] : CURRENT_XAB [ XAB$B_SIZ2 ] = .FDL$GL_NUMBER;
: 1286 1996 2 [ 3 ] : CURRENT_XAB [ XAB$B_SIZ3 ] = .FDL$GL_NUMBER;
: 1287 1997 2 [ 4 ] : CURRENT_XAB [ XAB$B_SIZ4 ] = .FDL$GL_NUMBER;
: 1288 1998 2 [ 5 ] : CURRENT_XAB [ XAB$B_SIZ5 ] = .FDL$GL_NUMBER;

```



: 1289 1999 2  
: 1290 2000 2  
: 1291 2001 2  
: 1292 2002 2  
: 1293 2003 2  
: 1294 2004 2  
: 1295 2005 2  
: 1296 2006 2  
: 1297 2007 2  
: 1298 2008 2  
: 1299 2009 2  
: 1300 2010 2  
: 1301 2011 2  
: 1302 2012 2  
: 1303 2013 2  
: 1304 2014 2  
: 1305 2015 2  
: 1306 2016 2  
: 1307 2017 2  
: 1308 2018 2  
: 1309 2019 2  
: 1310 2020 2  
: 1311 2021 2  
: 1312 2022 2  
: 1313 2023 2  
: 1314 2024 2  
: 1315 2025 2  
: 1316 2026 2  
: 1317 2027 2  
: 1318 2028 2  
: 1319 2029 2  
: 1320 2030 2  
: 1321 2031 2  
: 1322 2032 2  
: 1323 2033 2  
: 1324 2034 1

```

[ 6 ] : CURRENT_XAB [ XAB$B_SIZE6 ] = .FDL$GL_NUMBER;
[ 7 ] : CURRENT_XAB [ XAB$B_SIZE7 ] = .FDL$GL_NUMBER;
TES;

[ FDL$C_SEGPOS ]: CASE .FDL$GL_SECNUM FROM 0 TO 7 OF
SET
[ 0 ] : CURRENT_XAB [ XAB$W_POS0 ] = .FDL$GL_NUMBER;
[ 1 ] : CURRENT_XAB [ XAB$W_POS1 ] = .FDL$GL_NUMBER;
[ 2 ] : CURRENT_XAB [ XAB$W_POS2 ] = .FDL$GL_NUMBER;
[ 3 ] : CURRENT_XAB [ XAB$W_POS3 ] = .FDL$GL_NUMBER;
[ 4 ] : CURRENT_XAB [ XAB$W_POS4 ] = .FDL$GL_NUMBER;
[ 5 ] : CURRENT_XAB [ XAB$W_POS5 ] = .FDL$GL_NUMBER;
[ 6 ] : CURRENT_XAB [ XAB$W_POS6 ] = .FDL$GL_NUMBER;
[ 7 ] : CURRENT_XAB [ XAB$W_POS7 ] = .FDL$GL_NUMBER;
TES;

[ FDL$C_SEGTP ]: CASE .FDL$GL_SECNUM FROM 0 TO 7 OF
SET
[ 0 ] : BEGIN
CURRENT_XAB [ XAB$B_DTP ] = .FDL$GL_QUALIFIER;
CURRENT_XAB [ XAB$B_TYPO ] = .FDL$GL_QUALIFIER;
END;
[ 1 ] : CURRENT_XAB [ XAB$B_TYP1 ] = .FDL$GL_QUALIFIER;
[ 2 ] : CURRENT_XAB [ XAB$B_TYP2 ] = .FDL$GL_QUALIFIER;
[ 3 ] : CURRENT_XAB [ XAB$B_TYP3 ] = .FDL$GL_QUALIFIER;
[ 4 ] : CURRENT_XAB [ XAB$B_TYP4 ] = .FDL$GL_QUALIFIER;
[ 5 ] : CURRENT_XAB [ XAB$B_TYP5 ] = .FDL$GL_QUALIFIER;
[ 6 ] : CURRENT_XAB [ XAB$B_TYP6 ] = .FDL$GL_QUALIFIER;
[ 7 ] : CURRENT_XAB [ XAB$B_TYP7 ] = .FDL$GL_QUALIFIER;
TES;

TES;
RETURN
END;

```

69 17 A2

```

OFFC 00000 SET_KEY_P:
.SB 00000000G 00 9F 00002 .WORD Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11 : 1886
5A 00000000G 00 9E 00009 MOVAB FDL$GL_QUALIFIER, R11
59 00000000G 00 9E 00010 MOVAB FDL$GL_SECNUM, R10
58 00000000' 00 9E 00017 MOVAB FDL$GL_PRINUM, R9
57 00000000G 00 9E 0001E MOVAB CURRENT_XAB, R8
56 00000000G 00 9E 00025 MOVAB FDL$GL_SWITCH, R7
52 00000000G 68 D0 0002C MOVAB FDL$GL_NUMBER, R6
0D 13 0002F MOVAB CURRENT_XAB, R2 : 1926
15 62 91 00031 BEQL 1$
08 12 00034 CMPB (R2), #21 : 1940
08 00 ED 00036 BNEQ 1$
12 13 0003C CMPZV #0, #8, 23(R2), FDL$GL_PRINUM : 1941
69 DD 0003E 1$: BEQL 2$
15 DD 00040 PUSHL FDL$GL_PRINUM : 1945
PUSHL #21

```

			00000000v	00	02	FB	00042		CALLS	#2, ALLOCATE_XAB		
				50	68	D0	00049		MOVL	CURRENT_XAB, R0		1947
			17	A0	69	90	0004C		MOVB	FDLSGL_PRINUM, 23(R0)		
				52	68	D0	00050	2\$:	MOVL	CURRENT_XAB, R2		1955
			10	BF	00	CF	00053		CASEL	FDLSGL_SECONDARY, #119, #16		1953
0033			002E	0029	0022		0005F	3\$:	.WORD	4\$-3\$, -		
0053			004E	0047	003D		00067			5\$-3\$, -		
008A			0062	0085	005D		0006F			6\$-3\$, -		
00E0			00A1	0096	0091		00077			7\$-3\$, -		
					011F		0007F			8\$-3\$, -		
										9\$-3\$, -		
										10\$-3\$, -		
										11\$-3\$, -		
										12\$-3\$, -		
										13\$-3\$, -		
										14\$-3\$, -		
										15\$-3\$, -		
										16\$-3\$, -		
										17\$-3\$, -		
										18\$-3\$, -		
										19\$-3\$, -		
										20\$-3\$, -		
										21\$-3\$, -		
										22\$-3\$, -		
										23\$-3\$, -		
										24\$-3\$, -		
										25\$-3\$, -		
										26\$-3\$, -		
										27\$-3\$, -		
										28\$-3\$, -		
										29\$-3\$, -		
										30\$-3\$, -		
										31\$-3\$, -		
										32\$-3\$, -		
12	A2			01	01	67	F0	00081	4\$:	INSV	FDLSGL_SWITCH, #1, #1, 18(R2)	1955
								04		RET		
				0A	A2	66	90	00088	5\$:	MOVB	FDLSGL_NUMBER, 10(R2)	1957
								04		RET		
				1C	A2	66	B0	0008D	6\$:	MOVW	FDLSGL_NUMBER, 28(R2)	1959
								04		RET		
					50	67	D2	00092	7\$:	MCOML	FDLSGL_SWITCH, R0	1961
12	A2			01	06	50	F0	00095		INSV	R0, #6, #1, 18(R2)	
								04		RET		
					50	67	D2	0009C	8\$:	MCOML	FDLSGL_SWITCH, R0	1963
12	A2			01	07	50	F0	0009F		INSV	R0, #7, #1, 18(R2)	
								04		RET		
					00	67	F0	000A5	9\$:	INSV	FDLSGL_SWITCH, #0, #1, 18(R2)	1965
								04		RET		
				08	A2	66	90	000AD	10\$:	MOVB	FDLSGL_NUMBER, 8(R2)	1967
								04		RET		
					50	67	D2	000B2	11\$:	MCOML	FDLSGL_SWITCH, R0	1969
12	A2			01	03	50	F0	000B5		INSV	R0, #3, #1, 18(R2)	
								04		RET		
					1A	66	B0	000BC	12\$:	MOVW	FDLSGL_NUMBER, 26(R2)	1971
								04		RET		
						20	DD	000C1	13\$:	PUSHL	#32	1974
			00000000v	00	01	01	FB	000C3		CALLS	#1, FDL\$GET_VM	
			38	A2	50	D0	000CA		MOVL	R0, 56(R2)		
				51	00000000G	00	D0	000CE		MOVL	FDLSAB_STRING+4, R1	1976
				50		68	D0	000D5		MOVL	CURRENT_XAB, R0	1978
				20	00000000G	00	2C	000D8		MOVCS	FDLSAB_STRING, (R1), #32, #32, @56(R0)	
					38	B0		000E1				
								04		RET		1975
				09	A2	66	90	000E4	14\$:	MOVB	FDLSGL_NUMBER, 9(R2)	1981
								04		RET		
					02	67	F0	000E9	15\$:	INSV	FDLSGL_SWITCH, #2, #1, 18(R2)	1983
12	A2			01				04		RET		
					15	68	90	000F0	16\$:	MOVB	FDLSGL_QUALIFIER, 21(R2)	1985
								04		RET		



0023	001E	0019	50	6B	DO	0017E	39\$:	MOVL	FDL\$GL_QUALIFIER, R0	2018
0037	0032	002D	00	6A	CF	00181		CASEL	FDL\$GL_SECNUM, #0, #7	2015
				0010		00185	40\$:	.WORD	41\$-40\$,-	
				0028		0018D			42\$-40\$,-	
									43\$-40\$,-	
									44\$-40\$,-	
									45\$-40\$,-	
									46\$-40\$,-	
									47\$-40\$,-	
									48\$-40\$	
		13	A2	50	90	00195	41\$:	MOVB	R0, 19(R2)	2018
		40	A2	50	90	00199		MOVB	R0, 64(R2)	2019
					04	0019D		RET		
		41	A2	50	90	0019E	42\$:	MOVB	R0, 65(R2)	2021
					04	001A2		RET		
		42	A2	50	90	001A3	43\$:	MOVB	R0, 66(R2)	2022
					04	001A7		RET		
		43	A2	50	90	001A8	44\$:	MOVB	R0, 67(R2)	2023
					04	001AC		RET		
		44	A2	50	90	001AD	45\$:	MOVB	R0, 68(R2)	2024
					04	001B1		RET		
		45	A2	50	90	001B2	46\$:	MOVB	R0, 69(R2)	2025
					04	001B6		RET		
		46	A2	50	90	001B7	47\$:	MOVB	R0, 70(R2)	2026
					04	001BB		RET		
		47	A2	50	90	001BC	48\$:	MOVB	R0, 71(R2)	2027
					04	001C0		RET		2034

; Routine Size: 449 bytes. Routine Base: \_FDL\$CODE + 076A

```

1326 2035 1 %SBTTL 'SET_RECORD_P'
1327 2036 1 ROUTINE SET_RECORD_P : NOVALUE =
1328 2037 1 **
1329 2038 1
1330 2039 1 Functional Description:
1331 2040 1
1332 2041 1     Fill in the blanks for the fab fields concerning the record
1333 2042 1
1334 2043 1 Calling Sequence:
1335 2044 1
1336 2045 1     set_record_p()
1337 2046 1
1338 2047 1 Input Parameters:
1339 2048 1     none
1340 2049 1
1341 2050 1 Implicit Inputs:
1342 2051 1
1343 2052 1     fdl$secondary - Secondary code
1344 2053 1
1345 2054 1 Output Parameters:
1346 2055 1     none
1347 2056 1
1348 2057 1 Implicit Outputs:
1349 2058 1     none
1350 2059 1
1351 2060 1 Routine Value:
1352 2061 1     none
1353 2062 1
1354 2063 1 Routines Called:
1355 2064 1     none
1356 2065 1
1357 2066 1 Side Effects:
1358 2067 1     none
1359 2068 1
1360 2069 1 --
1361 2070 1
1362 2071 2 BEGIN
1363 2072 2
1364 2073 2 REGISTER
1365 2074 2     PARSED_FAB      : REF BLOCK [ ,BYTE ];
1366 2075 2
1367 2076 2 PARSED_FAB = .FDL$AB_PARSED_FAB;
1368 2077 2
1369 2078 2 ! Set em up
1370 2079 2 !
1371 2080 2 CASE .FDL$GL_SECONDARY FROM FDL$C_BLKSPN TO FDL$C_SIZE OF
1372 2081 2 SET
1373 2082 2     [ FDL$C_BLKSPN ]: PARSED_FAB [ FAB$V_BLK ] = NOT .FDL$GL_SWITCH;
1374 2083 2
1375 2084 2     [ FDL$C_CARCTRL]: CASE .FDL$GL_QUALIFIER FROM FDL$C_NONE TO FDL$C_PRINT OF
1376 2085 2 SET
1377 2086 2         ! We must clear the other flags while setting the one
1378 2087 2         ! we want (without clearing BLK if set)
1379 2088 2         !
1380 2089 2     [ FDL$C_NONE ] : PARSED_FAB [ FAB$B_RAT ] =
1381 2090 2         .PARSED_FAB [ FAB$B_RAT ] AND
1382 2091 2         FAB$M_BLK;

```

```

: 1383 2092 2
: 1384 2093 2
: 1385 2094 2
: 1386 2095 2
: 1387 2096 2
: 1388 2097 2
: 1389 2098 2
: 1390 2099 2
: 1391 2100 2
: 1392 2101 2
: 1393 2102 2
: 1394 2103 2
: 1395 2104 2
: 1396 2105 2
: 1397 2106 2
: 1398 2107 2
: 1399 2108 2
: 1400 2109 2
: 1401 2110 2
: 1402 2111 2
: 1403 2112 1

```

```

[ FDL$C_CR ] : PARSED FAB [ FAB$B_RAT ] =
( .PARSED FAB [ FAB$B_RAT ] AND
FAB$M_BLK ) OR FAB$M_CR;
[ FDL$C_FTN ] : PARSED FAB [ FAB$B_RAT ] =
( .PARSED FAB [ FAB$B_RAT ] AND
FAB$M_BLK ) OR FAB$M_FTN;
[ FDL$C_PRINT ] : PARSED FAB [ FAB$B_RAT ] =
( .PARSED FAB [ FAB$B_RAT ] AND
FAB$M_BLK ) OR FAB$M_PRN;

TES;
[ FDL$C_VFCSIZ ]: PARSED_FAB [ FAB$B_FSZ ] = .FDL$GL_NUMBER;
[ FDL$C_FMT ] : PARSED_FAB [ FAB$B_RFM ] = .FDL$GL_QUALIFIER;
[ FDL$C_SIZE ] : PARSED_FAB [ FAB$W_MRS ] = .FDL$GL_NUMBER;

TES;
RETURN
END;

```

```

                                000C 00000 SET_RECORD_P:
                                .WORD Save R2,R3
                                53 00000000G 00 9E 00002 MOVAB FDL$GL_NUMBER, R3
                                52 00000000G 00 9E 00009 MOVAB FDL$GL_QUALIFIER, R2
                                50 00000000G 00 D0 00010 MOVL FDL$AB_PARSED_FAB, PARSED_FAB
                                04 00000088 0018 8F 00000000G 00 CF 00017 CASEL FDL$GL_SECONDARY, #136, #4
                                005F 005A 000A 00023 1$: .WORD 2$-1$, -
                                0064 0002B 3$-1$, -
                                9$-1$, -
                                10$-1$, -
                                11$-1$
                                51 00000000G 00 D2 0002D 2$: MCOML FDL$GL_SWITCH, R1
                                03 51 F0 00034 INSV R1, #3, #1, 30(PARSED_FAB)
                                04 0003A RET
                                51 1E A0 9E 0003B 3$: MOVAB 30(PARSED_FAB), R1
                                03 08 62 CF 0003F CASEL FDL$GL_QUALIFIER, #8, #3
                                002B 001C 000D 0008 00043 4$: .WORD 5$-4$, -
                                6$-4$, -
                                7$-4$, -
                                8$-4$
                                61 F7 8F 8A 0004B 5$: BICB2 #9, (R1)
                                04 0004F RET
                                50 61 9A 00050 6$: MOVZBL (R1), R0
                                50 FFFFFFF7 8F CA 00053 BICL2 #9, R0
                                61 50 02 89 0005A BISB3 #2, R0, (R1)
                                04 0005E RET
                                50 61 9A 0005F 7$: MOVZBL (R1), R0
                                50 FFFFFFF7 8F CA 00062 BICL2 #9, R0
                                61 50 01 89 00069 BISB3 #1, R0, (R1)
                                04 0006D RET
                                50 61 9A 0006E 8$: MOVZBL (R1), R0

```

```

: 2036
: 2076
: 2080
: 2082
: 2089
: 2084
: 2090
: 2089
: 2093
: 2094
: 2092
: 2096
: 2097
: 2095
: 2099

```

61	50	FFFFFFF7	8F	CA	00071		BICL2	#-9, R0	:	2100
	50		04	89	00078		BISB3	#4, R0, (R1)	:	2084
				04	0007C		RET		:	2103
	3F	A0	63	90	0007D	9\$:	MOVB	FDLSGL_NUMBER, 63(PARSED_FAB)	:	
				04	00081		RET		:	
	1F	A0	62	90	00082	10\$:	MOVB	FDLSGL_QUALIFIER, 31(PARSED_FAB)	:	2105
				04	00086		RET		:	
	36	A0	63	80	00087	11\$:	MOVW	FDLSGL_NUMBER, 54(PARSED_FAB)	:	2107
				04	0008B		RET		:	2112

; Routine Size: 140 bytes, Routine Base: \_FDL\$CODE + 092B

```

1405 2113 1 %SBTTL 'SET_ACCESS_P'
1406 2114 1 ROUTINE SET_ACCESS_P : NOVALUE =
1407 2115 1 **
1408 2116 1
1409 2117 1 Functional Description:
1410 2118 1
1411 2119 1 Fill in the blanks for the fab fields concerning access mode
1412 2120 1
1413 2121 1 Calling Sequence:
1414 2122 1
1415 2123 1 set_access_p()
1416 2124 1
1417 2125 1 Input Parameters:
1418 2126 1 none
1419 2127 1
1420 2128 1 Implicit Inputs:
1421 2129 1
1422 2130 1 fdl$secondary - Secondary code
1423 2131 1
1424 2132 1 Output Parameters:
1425 2133 1 none
1426 2134 1
1427 2135 1 Implicit Outputs:
1428 2136 1 none
1429 2137 1
1430 2138 1 Routine Value:
1431 2139 1 none
1432 2140 1
1433 2141 1 Routines Called:
1434 2142 1 none
1435 2143 1
1436 2144 1 Side Effects:
1437 2145 1 none
1438 2146 1
1439 2147 1 --
1440 2148 1
1441 2149 2 BEGIN
1442 2150 2
1443 2151 2 REGISTER
1444 2152 2 PARSED_FAB : REF BLOCK [ ,BYTE ];
1445 2153 2
1446 2154 2 PARSED_FAB = .FDL$AB_PARSED_FAB;
1447 2155 2
1448 2156 2 ! Set em up
1449 2157 2 !
1450 2158 2 CASE .FDL$GL_SECONDARY FROM FDL$C_FACBIO TO FDL$C_FACUPD OF
1451 2159 2 SET
1452 2160 2 [ FDL$C_FACBIO ] : PARSED_FAB [ FABS$V_BIO ] = .FDL$GL_SWITCH;
1453 2161 2
1454 2162 2 [ FDL$C_FACDEL ] : PARSED_FAB [ FABS$V_DEL ] = .FDL$GL_SWITCH;
1455 2163 2
1456 2164 2 [ FDL$C_FACGET ] : PARSED_FAB [ FABS$V_GET ] = .FDL$GL_SWITCH;
1457 2165 2
1458 2166 2 [ FDL$C_FACPUT ] : PARSED_FAB [ FABS$V_PUT ] = .FDL$GL_SWITCH;
1459 2167 2
1460 2168 2 [ FDL$C_FACBRO ] : PARSED_FAB [ FABS$V_BRO ] = .FDL$GL_SWITCH;
1461 2169 2

```



```

: 1462      2170 2      [ FDL$C_FACTRN ] : PARSED_FAB [ FAB$V_TRN ] = .FDL$GL_SWITCH;
: 1463      2171 2
: 1464      2172 2      [ FDL$C_FACUPD ] : PARSED_FAB [ FAB$V_UPD ] = .FDL$GL_SWITCH;
: 1465      2173 2      TES;
: 1466      2174 2
: 1467      2175 2      RETURN
: 1468      2176 2
: 1469      2177 1      END;
    
```

```

                                0000 00000 SET_ACCESS_P:
                                .WORD
                                Save nothing
                                FDL$AB PARSED_FAB, PARSED_FAB      : 2114
                                22(PARSED_FAB), R1                  : 2154
                                FDL$GL_SWITCH, R0                    : 2160
                                CASEL FDL$GL_SECONDARY, #1, #6
                                2$-1$, -                             : 2158
                                3$-1$, -
                                4$-1$, -
                                5$-1$, -
                                6$-1$, -
                                7$-1$, -
                                8$-1$, -
0020      06
           001A      0014 000E 0001C 1$:
           0032      002C 0026 00024
61      01      05      50 FO 0002A 2$:  INSV  R0, #5, #1, (R1)      : 2160
           04 0002F      RET
61      01      02      50 FO 00030 3$:  INSV  R0, #2, #1, (R1)      : 2162
           04 00035      RET
61      01      01      50 FO 00036 4$:  INSV  R0, #1, #1, (R1)      : 2164
           04 0003B      RET
61      01      00      50 FO 0003C 5$:  INSV  R0, #0, #1, (R1)      : 2166
           04 00041      RET
61      01      06      50 FO 00042 6$:  INSV  R0, #6, #1, (R1)      : 2168
           04 00047      RET
61      01      04      50 FO 00048 7$:  INSV  R0, #4, #1, (R1)      : 2170
           04 0004D      RET
61      01      03      50 FO 0004E 8$:  INSV  R0, #3, #1, (R1)      : 2172
           04 00053      RET
    
```

; Routine Size: 84 bytes, Routine Base: \_FDL\$CODE + 09B7

```

: 1471 2178 1 %SBTTL 'SET_SHARING_P'
: 1472 2179 1 ROUTINE SET_SHARING_P : NOVALUE =
: 1473 2180 1 ++
: 1474 2181 1
: 1475 2182 1 Functional Description:
: 1476 2183 1
: 1477 2184 1 Fill in the blanks for the fab fields concerning sharing
: 1478 2185 1
: 1479 2186 1 Calling Sequence:
: 1480 2187 1
: 1481 2188 1 set_sharing_p()
: 1482 2189 1
: 1483 2190 1 Input Parameters:
: 1484 2191 1 none
: 1485 2192 1
: 1486 2193 1 Implicit Inputs:
: 1487 2194 1
: 1488 2195 1 fdl$secondary - Secondary code
: 1489 2196 1
: 1490 2197 1 Output Parameters:
: 1491 2198 1 none
: 1492 2199 1
: 1493 2200 1 Implicit Outputs:
: 1494 2201 1 none
: 1495 2202 1
: 1496 2203 1 Routine Value:
: 1497 2204 1 none
: 1498 2205 1
: 1499 2206 1 Routines Called:
: 1500 2207 1 none
: 1501 2208 1
: 1502 2209 1 Side Effects:
: 1503 2210 1 none
: 1504 2211 1
: 1505 2212 1 --
: 1506 2213 1
: 1507 2214 2 BEGIN
: 1508 2215 2
: 1509 2216 2 REGISTER
: 1510 2217 2 PARSED_FAB : REF BLOCK [ ,BYTE ];
: 1511 2218 2
: 1512 2219 2 PARSED_FAB = .FDL$AB_PARSED_FAB;
: 1513 2220 2
: 1514 2221 2 ! Set em up
: 1515 2222 2 !
: 1516 2223 2 CASE .FDL$GL_SECONDARY FROM FDL$C_SHRDEL TO FDL$C_SHRUP1 OF
: 1517 2224 2 SET
: 1518 2225 2 [ FDL$C_SHRDEL ] : PARSED_FAB [ FABS$V_SHRDEL ] = .FDL$GL_SWITCH;
: 1519 2226 2
: 1520 2227 2 [ FDL$C_SHRGET ] : PARSED_FAB [ FABS$V_SHRGET ] = .FDL$GL_SWITCH;
: 1521 2228 2
: 1522 2229 2 [ FDL$C_SHRMSE ] : PARSED_FAB [ FABS$V_MSE ] = .FDL$GL_SWITCH;
: 1523 2230 2
: 1524 2231 2 [ FDL$C_SHRNIL ] : PARSED_FAB [ FABS$V_NIL ] = .FDL$GL_SWITCH;
: 1525 2232 2
: 1526 2233 2 [ FDL$C_SHRPUT ] : PARSED_FAB [ FABS$V_SHRPUT ] = .FDL$GL_SWITCH;
: 1527 2234 2

```

```

: 1528      2235 2      [ FDL$C_SHRUPD ] : PARSED_FAB [ FAB$V_SHRUPD ] = .FDL$GL_SWITCH;
: 1529      2236 2
: 1530      2237 2      [ FDL$C_SHRUPI ] : PARSED_FAB [ FAB$V_UPI ] = .FDL$GL_SWITCH;
: 1531      2238 2      TES;
: 1532      2239 2
: 1533      2240 2      RETURN
: 1534      2241 2
: 1535      2242 1      END;

```

```

                                0000 00000 SET_SHARING_P:
                                .WORD Save nothing
                                50 00000000G 00 D0 00002      MOVL FDL$AB PARSED_FAB, PARSED_FAB
                                51          17  A0 9E 00009      MOVAB 23(PARSED_FAB), R1
                                50 00000000G 00 D0 0000D      MOVL FDL$GL_SWITCH, R0
0020      06 0000008D      8F 00000000G 00 CF 00014      CASEL FDL$GL_SECONDARY, #141, #6
                                0014      000E 00020 1$:      .WORD 2$-1$, -
                                0032      002C      0026      00028      3$-1$, -
                                4$-1$, -
                                5$-1$, -
                                6$-1$, -
                                7$-1$, -
                                8$-1$
                                61      01      02      50 F0 0002E 2$:      INSV R0, #2, #1, (R1)
                                04 00033      RET
                                61      01      01      50 F0 00034 3$:      INSV R0, #1, #1, (R1)
                                04 00039      RET
                                61      01      04      50 F0 0003A 4$:      INSV R0, #4, #1, (R1)
                                04 0003F      RET
                                61      01      05      50 F0 00040 5$:      INSV R0, #5, #1, (R1)
                                04 00045      RET
                                61      01      00      50 F0 00046 6$:      INSV R0, #0, #1, (R1)
                                04 0004B      RET
                                61      01      03      50 F0 0004C 7$:      INSV R0, #3, #1, (R1)
                                04 00051      RET
                                61      01      06      50 F0 00052 8$:      INSV R0, #6, #1, (R1)
                                04 00057      RET

```

; Routine Size: 88 bytes, Routine Base: \_FDL\$CODE + 0A0B

```

: 1537 2243 1 %SBTTL 'SET_CONNECT_P'
: 1538 2244 1 ROUTINE SET_CONNECT_P : NOVALUE =
: 1539 2245 1 ++
: 1540 2246 1
: 1541 2247 1 Functional Description:
: 1542 2248 1
: 1543 2249 1 Fill in the blanks for the Rab fields
: 1544 2250 1
: 1545 2251 1 Calling Sequence:
: 1546 2252 1
: 1547 2253 1 set_connect_p()
: 1548 2254 1
: 1549 2255 1 Input Parameters:
: 1550 2256 1 none
: 1551 2257 1
: 1552 2258 1 Implicit Inputs:
: 1553 2259 1
: 1554 2260 1 fdl$secondary - Secondary code
: 1555 2261 1
: 1556 2262 1 Output Parameters:
: 1557 2263 1 none
: 1558 2264 1
: 1559 2265 1 Implicit Outputs:
: 1560 2266 1 none
: 1561 2267 1
: 1562 2268 1 Routine Value:
: 1563 2269 1 none
: 1564 2270 1
: 1565 2271 1 Routines Called:
: 1566 2272 1 none
: 1567 2273 1
: 1568 2274 1 Side Effects:
: 1569 2275 1 none
: 1570 2276 1
: 1571 2277 1 --
: 1572 2278 1
: 1573 2279 2 BEGIN
: 1574 2280 2
: 1575 2281 2 REGISTER
: 1576 2282 2 PARSED_RAB : REF BLOCK [ ,BYTE ];
: 1577 2283 2
: 1578 2284 2 PARSED_RAB = .FDL$AB_PARSED_RAB;
: 1579 2285 2
: 1580 2286 2 ! Set em up
: 1581 2287 2 !
: 1582 2288 2 CASE .FDL$GL_SECONDARY FROM FDL$C_ASY TO FDL$C_WBH OF
: 1583 2289 2 SET
: 1584 2290 2 [ FDL$C_ASY ] : PARSED_RAB [ RAB$V_ASY ] = .FDL$GL_SWITCH;
: 1585 2291 2
: 1586 2292 2 [ FDL$C_BIO ] : PARSED_RAB [ RAB$V_BIO ] = .FDL$GL_SWITCH;
: 1587 2293 2
: 1588 2294 2 [ FDL$C_BUCODE ] : PARSED_RAB [ RAB$L_BKT ] = .FDL$GL_NUMBER;
: 1589 2295 2
: 1590 2296 2 [ FDL$C_RCTX ] : PARSED_RAB [ RAB$L_CTX ] = .FDL$GL_NUMBER;
: 1591 2297 2
: 1592 2298 2 [ FDL$C_EOF ] : PARSED_RAB [ RAB$V_EOF ] = .FDL$GL_SWITCH;
: 1593 2299 2

```

```

: 1594      2300      2      [ FDL$C_FLOA ] : PARSED_RAB [ RAB$V_LOA ] = .FDL$GL_SWITCH;
: 1595      2301      2
: 1596      2302      2      [ FDL$C_FDEL ] : PARSED_RAB [ RAB$V_FDL ] = .FDL$GL_SWITCH;
: 1597      2303      2
: 1598      2304      2      [ FDL$C_KGE ]  : PARSED_RAB [ RAB$V_KGE ] = .FDL$GL_SWITCH;
: 1599      2305      2
: 1600      2306      2      [ FDL$C_KGT ]  : PARSED_RAB [ RAB$V_KGT ] = .FDL$GL_SWITCH;
: 1601      2307      2
: 1602      2308      2      [ FDL$C_KLIM ] : PARSED_RAB [ RAB$V_LIM ] = .FDL$GL_SWITCH;
: 1603      2309      2
: 1604      2310      2      [ FDL$C_KRF ]  : PARSED_RAB [ RAB$B_KRF ] = .FDL$GL_NUMBER;
: 1605      2311      2
: 1606      2312      2      [ FDL$C_LOCMODE ] : PARSED_RAB [ RAB$V_LOC ] = .FDL$GL_SWITCH;
: 1607      2313      2
: 1608      2314      2      [ FDL$C_REA ]  : PARSED_RAB [ RAB$V_REA ] = .FDL$GL_SWITCH;
: 1609      2315      2
: 1610      2316      2      [ FDL$C_RLK ]  : PARSED_RAB [ RAB$V_RLK ] = .FDL$GL_SWITCH;
: 1611      2317      2
: 1612      2318      2      [ FDL$C_ULK ]  : PARSED_RAB [ RAB$V_ULK ] = .FDL$GL_SWITCH;
: 1613      2319      2
: 1614      2320      2      [ FDL$C_MBC ]  : PARSED_RAB [ RAB$B_MBC ] = .FDL$GL_NUMBER;
: 1615      2321      2
: 1616      2322      2      [ FDL$C_MBF ]  : PARSED_RAB [ RAB$B_MBF ] = .FDL$GL_NUMBER;
: 1617      2323      2
: 1618      2324      2      [ FDL$C_NLK ]  : PARSED_RAB [ RAB$V_NLK ] = .FDL$GL_SWITCH;
: 1619      2325      2
: 1620      2326      2      [ FDL$C_NXR ]  : PARSED_RAB [ RAB$V_NXR ] = .FDL$GL_SWITCH;
: 1621      2327      2
: 1622      2328      2      [ FDL$C_RAH ]  : PARSED_RAB [ RAB$V_RAH ] = .FDL$GL_SWITCH;
: 1623      2329      2
: 1624      2330      2      [ FDL$C_RRL ]  : PARSED_RAB [ RAB$V_RRL ] = .FDL$GL_SWITCH;
: 1625      2331      2
: 1626      2332      2      [ FDL$C_TMO ]  : PARSED_RAB [ RAB$B_TMO ] = .FDL$GL_NUMBER;
: 1627      2333      2
: 1628      2334      2      [ FDL$C_TMENB ] : PARSED_RAB [ RAB$V_TMO ] = .FDL$GL_SWITCH;
: 1629      2335      2
: 1630      2336      2      [ FDL$C_TPT ]  : PARSED_RAB [ RAB$V_TPT ] = .FDL$GL_SWITCH;
: 1631      2337      2
: 1632      2338      2      [ FDL$C_TTCCO ] : PARSED_RAB [ RAB$V_CCO ] = .FDL$GL_SWITCH;
: 1633      2339      2
: 1634      2340      2      [ FDL$C_TTCVT ] : PARSED_RAB [ RAB$V_CVT ] = .FDL$GL_SWITCH;
: 1635      2341      2
: 1636      2342      2      [ FDL$C_TTPMT ] : PARSED_RAB [ RAB$V_PMT ] = .FDL$GL_SWITCH;
: 1637      2343      2
: 1638      2344      2      [ FDL$C_TTPTA ] : PARSED_RAB [ RAB$V_PTA ] = .FDL$GL_SWITCH;
: 1639      2345      2
: 1640      2346      2      [ FDL$C_TTRNE ] : PARSED_RAB [ RAB$V_RNE ] = .FDL$GL_SWITCH;
: 1641      2347      2
: 1642      2348      2      [ FDL$C_TTRNF ] : PARSED_RAB [ RAB$V_RNF ] = .FDL$GL_SWITCH;
: 1643      2349      2
: 1644      2350      2      [ FDL$C_UIF ]  : PARSED_RAB [ RAB$V_UIF ] = .FDL$GL_SWITCH;
: 1645      2351      2
: 1646      2352      2      [ FDL$C_WAT ]  : PARSED_RAB [ RAB$V_WAT ] = .FDL$GL_SWITCH;
: 1647      2353      2
: 1648      2354      2      [ FDL$C_WBH ]  : PARSED_RAB [ RAB$V_WBH ] = .FDL$GL_SWITCH;
: 1649      2355      2
: 1650      2356      2

```

TES;



05	A0	01	05	62	F0 00080	78:	INSV	FDLSGL_SWITCH, #5, #1, 5(PARSED_RAB)	2300
					04 00086		RET		
04	A0	01	06	62	F0 00087	88:	INSV	FDLSGL_SWITCH, #6, #1, 4(PARSED_RAB)	2302
					04 0008D		RET		
06	A0	01	05	62	F0 0008E	98:	INSV	FDLSGL_SWITCH, #5, #1, 6(PARSED_RAB)	2304
					04 00094		RET		
06	A0	01	06	62	F0 00095	108:	INSV	FDLSGL_SWITCH, #6, #1, 6(PARSED_RAB)	2306
					04 0009B		RET		
05	A0	01	06	62	F0 0009C	118:	INSV	FDLSGL_SWITCH, #6, #1, 5(PARSED_RAB)	2308
					04 000A2		RET		
		35	A0	63	90 000A3	128:	MOVB	FDLSGL_NUMBER, 53(PARSED_RAB)	2310
					04 000A7		RET		
06	A0	01	00	62	F0 000AB	138:	INSV	FDLSGL_SWITCH, #0, #1, 6(PARSED_RAB)	2312
					04 000AE		RET		
04	A0	01	02	62	F0 000AF	148:	INSV	FDLSGL_SWITCH, #2, #1, 4(PARSED_RAB)	2314
					04 000B5		RET		
06	A0	01	03	62	F0 000B6	158:	INSV	FDLSGL_SWITCH, #3, #1, 6(PARSED_RAB)	2316
					04 000BC		RET		
06	A0	01	02	62	F0 000BD	168:	INSV	FDLSGL_SWITCH, #2, #1, 6(PARSED_RAB)	2318
					04 000C3		RET		
		37	A0	63	90 000C4	178:	MOVB	FDLSGL_NUMBER, 55(PARSED_RAB)	2320
					04 000C8		RET		
		36	A0	63	90 000C9	188:	MOVB	FDLSGL_NUMBER, 54(PARSED_RAB)	2322
					04 000CD		RET		
06	A0	01	04	62	F0 000CE	198:	INSV	FDLSGL_SWITCH, #4, #1, 6(PARSED_RAB)	2324
					04 000D4		RET		
06	A0	01	07	62	F0 000D5	208:	INSV	FDLSGL_SWITCH, #7, #1, 6(PARSED_RAB)	2326
					04 000DB		RET		
05	A0	01	01	62	F0 000DC	218:	INSV	FDLSGL_SWITCH, #1, #1, 5(PARSED_RAB)	2328
					04 000E2		RET		
04	A0	01	03	62	F0 000E3	228:	INSV	FDLSGL_SWITCH, #3, #1, 4(PARSED_RAB)	2330
					04 000E9		RET		
		1F	A0	63	90 000EA	238:	MOVB	FDLSGL_NUMBER, 31(PARSED_RAB)	2332
					04 000EE		RET		
07	A0	01	01	62	F0 000EF	248:	INSV	FDLSGL_SWITCH, #1, #1, 7(PARSED_RAB)	2334
					04 000F5		RET		
04	A0	01	01	62	F0 000F6	258:	INSV	FDLSGL_SWITCH, #1, #1, 4(PARSED_RAB)	2336
					04 000FC		RET		
07	A0	01	07	62	F0 000FD	268:	INSV	FDLSGL_SWITCH, #7, #1, 7(PARSED_RAB)	2338
					04 00103		RET		
07	A0	01	02	62	F0 00104	278:	INSV	FDLSGL_SWITCH, #2, #1, 7(PARSED_RAB)	2340
					04 0010A		RET		
07	A0	01	06	62	F0 0010B	288:	INSV	FDLSGL_SWITCH, #6, #1, 7(PARSED_RAB)	2342
					04 00111		RET		
07	A0	01	05	62	F0 00112	298:	INSV	FDLSGL_SWITCH, #5, #1, 7(PARSED_RAB)	2344
					04 00118		RET		
07	A0	01	00	62	F0 00119	308:	INSV	FDLSGL_SWITCH, #0, #1, 7(PARSED_RAB)	2346
					04 0011F		RET		
07	A0	01	03	62	F0 00120	318:	INSV	FDLSGL_SWITCH, #3, #1, 7(PARSED_RAB)	2348
					04 00126		RET		
04	A0	01	04	62	F0 00127	328:	INSV	FDLSGL_SWITCH, #4, #1, 4(PARSED_RAB)	2350
					04 0012D		RET		
06	A0	01	01	62	F0 0012E	338:	INSV	FDLSGL_SWITCH, #1, #1, 6(PARSED_RAB)	2352
					04 00134		RET		
05	A0	01	02	62	F0 00135	348:	INSV	FDLSGL_SWITCH, #2, #1, 5(PARSED_RAB)	2354
					04 0013B		RET		2359





```
1655 2360 1 %SBTTL 'SET_PROT'
1656 2361 1 ROUTINE SET_PROT : JOVALUE =
1657 2362 1 **
1658 2363 1
1659 2364 1 Functional Description:
1660 2365 1
1661 2366 1     Fill in the blanks for the protection xab
1662 2367 1
1663 2368 1 Calling Sequence:
1664 2369 1
1665 2370 1     set_prot()
1666 2371 1
1667 2372 1 Input Parameters:
1668 2373 1     none
1669 2374 1
1670 2375 1 Implicit Inputs:
1671 2376 1
1672 2377 1     fdl$secondary - Secondary code
1673 2378 1
1674 2379 1 Output Parameters:
1675 2380 1     none
1676 2381 1
1677 2382 1 Implicit Outputs:
1678 2383 1     none
1679 2384 1
1680 2385 1 Routine Value:
1681 2386 1     none
1682 2387 1
1683 2388 1 Routines Called:
1684 2389 1     none
1685 2390 1
1686 2391 1 Side Effects.
1687 2392 1     none
1688 2393 1
1689 2394 1 --
1690 2395 1
1691 2396 2 BEGIN
1692 2397 2
1693 2398 2     ! See if the protection xab has been allocated yet
1694 2399 2     !
1695 2400 2     IF .PROTECTION_XAB EQLU 0
1696 2401 2     THEN
1697 2402 2
1698 2403 2         ! Allocate the xab and enter it into the chain
1699 2404 2         !
1700 2405 2         PROTECTION_XAB = ALLOCATE_XAB ( XAB$C_PRO, 0 );
1701 2406 2
1702 2407 2     ! Set the fields according to the secondary
1703 2408 2     !
1704 2409 2     SELECTONEU .FDL$GL_SECONDARY OF
1705 2410 2     SET
1706 2411 2         [ FDL$C_MTPRO ] : PROTECTION_XAB [ XAB$B_MTACC ] = .FDL$GL_QUALIFIER;
1707 2412 2
1708 2413 2         [ FDL$C_PROT ] : PROTECTION_XAB [ XAB$W_PRO ] = NOT .FDL$GL_PROTECTION;
1709 2414 2
1710 2415 2         [ FDL$C_OWNER ] : PROTECTION_XAB [ XAB$L_UIC ] = .FDL$GL_OWNER_UIC;
1711 2416 2     TES;
```

:	1712	2417	2	
:	1713	2418	2	RETURN
:	1714	2419	2	
:	1715	2420	1	END;

```

                                0004 00000 SET_PROT:
                                .WORD      Save R2
00000000V 52 00000000' 00 9E 00002  MOVAB  PROTECTION_XAB, R2      : 2361
                                                62 D5 00009  TSTL  PROTECTION_XAB      : 2400
                                                0D 12 0000B  BNEQ  1$
                                                7E 13 7D 0000D  MOVQ  #19, -(SP)      : 2405
000000059 00 00000000G 00 02 FB 00010  CALLS #2, ALLOCATE_XAB
                                                62 50 D0 00017  MOVL  R0, PROTECTION_XAB
                                                50 00000000G 00 D0 0001A 1$: MOVL  FDL$GL_SECONDARY, R0      : 2409
000000059 8F 00000000G 50 D1 00021  CMPL  R0, #89      : 2411
                                                0C 12 00028  BNEQ  2$
                                                50 62 D0 0002A  MOVL  PROTECTION_XAB, R0
0A A0 00000000G 00 90 0002D  MOVB  FDL$GL_QUALIFIER, 10(R0)
                                                04 00035  RET
000000065 8F 00000000G 50 D1 00036 2$: CMPL  R0, #101      : 2413
                                                0C 12 0003D  BNEQ  3$
                                                50 62 D0 0003F  MOVL  PROTECTION_XAB, R0
0B A0 00000000G 00 B2 00042  MCOMW FDL$GL_PROTECTION, 8(R0)
                                                04 0004A  RET
000000063 8F 00000000G 50 D1 0004B 3$: CMPL  R0, #99      : 2415
                                                0B 12 00052  BNEQ  4$
                                                50 62 D0 00054  MOVL  PROTECTION_XAB, R0
0C A0 00000000G 00 D0 00057  MOVL  FDL$GL_OWNER_UIC, 12(R0)
                                                04 0005F 4$: RET      : 2420

```

; Routine Size: 96 bytes. Routine Base: \_FDL\$CODE + 0B9F

```
: 1717 2421 1 %SBTTL 'ALLOCATE_XAB'  
: 1718 2422 1 ROUTINE ALLOCATE_XAB ( XAB_TYPE, XAB_NUM ) =  
: 1719 2423 1 **  
: 1720 2424 1  
: 1721 2425 1 Functional Description:  
: 1722 2426 1  
: 1723 2427 1 Allocates an RMS extended attribute block from virtual memory  
: 1724 2428 1  
: 1725 2429 1 *****  
: 1726 2430 1  
: 1727 2431 1 NOTE: THIS ROUTINE ASSUMES XABs ARE CONNECTED TO THE $FAB !!!  
: 1728 2432 1 IT WILL NOT WORK WITH XABs THAT ARE CONNECTED TO THE $RAB !!!  
: 1729 2433 1 *****  
: 1730 2434 1  
: 1731 2435 1  
: 1732 2436 1 Calling Sequence:  
: 1733 2437 1  
: 1734 2438 1 allocate_xab( xab_type, xab_num )  
: 1735 2439 1  
: 1736 2440 1 Input Parameters:  
: 1737 2441 1  
: 1738 2442 1 xab_type - The RMS code for the type of xab wanted ie. XAB$C_xab  
: 1739 2443 1 xab_num - Which xab is desired (for key and area xabs)  
: 1740 2444 1  
: 1741 2445 1 Implicit Inputs:  
: 1742 2446 1 none  
: 1743 2447 1  
: 1744 2448 1 Output Parameters:  
: 1745 2449 1 none  
: 1746 2450 1  
: 1747 2451 1 Implicit Outputs:  
: 1748 2452 1 none  
: 1749 2453 1  
: 1750 2454 1 Routine Value:  
: 1751 2455 1  
: 1752 2456 1 Pointer to the new xab (also pointed to by current xab)  
: 1753 2457 1  
: 1754 2458 1 Routines Called:  
: 1755 2459 1  
: 1756 2460 1 fdl$$get_vm  
: 1757 2461 1  
: 1758 2462 1 Side Effects:  
: 1759 2463 1  
: 1760 2464 1 current_xab pointes to the new xab  
: 1761 2465 1  
: 1762 2466 1 --  
: 1763 2467 1  
: 1764 2468 2 BEGIN  
: 1765 2469 2  
: 1766 2470 2 LOCAL  
: 1767 2471 2 XAB : REF BLOCK [ ,BYTE ],  
: 1768 2472 2 FOUND,  
: 1769 2473 2 XAB_LEN,  
: 1770 2474 2 NEW_XAB;  
: 1771 2475 2  
: 1772 2476 2 ! Find the size of the type of xab we want.  
: 1773 2477 2
```

```

: 1774      2478      3      XAB_LEN = ( SELECTONEU .XAB_TYPE OF
: 1775      2479      3      SET
: 1776      2480      3      [ XABSC_ALL ] : XABSC_ALLLEN;
: 1777      2481      3      [ XABSC_DAT ] : XABSC_DATLEN;
: 1778      2482      3      [ XABSC_JNL ] : XABSC_JNLLEN;
: 1779      2483      3      [ XABSC_KEY ] : XABSC_KEYLEN;
: 1780      2484      3      [ XABSC_PRO ] : XABSC_PROLEN;
: 1781      2485      3      [ XABSC_RDT ] : XABSC_RDTLEN;
: 1782      2486      3      TES );
: 1783      2487      2
: 1784      2488      2      FOUND = _CLEAR;
: 1785      2489      2
: 1786      2490      2      ! See if the xab we need already exists
: 1787      2491      2      ! (if we're in the second parse)
: 1788      2492      2      !
: 1789      2493      3      IF (
: 1790      2494      4      ( .FDLSAB_CTRL [ FDL$V_REPARSE ] )
: 1791      2495      3      AND
: 1792      2496      4      ( ( .XAB_TYPE EQLU XABSC_ALL ) OR ( .XAB_TYPE EQLU XABSC_KEY ) )
: 1793      2497      2      ) THEN
: 1794      2498      2      BEGIN
: 1795      2499      2
: 1796      2500      2      XAB = .FDLSAB_PARSED_FAB [ FABS$L_XAB ];
: 1797      2501      2
: 1798      2502      2      WHILE .XAB NEQU 0
: 1799      2503      2      DO
: 1800      2504      4      BEGIN
: 1801      2505      4
: 1802      2506      5      IF (
: 1803      2507      7      (( .XAB_TYPE EQLU XABSC_ALL )
: 1804      2508      6      AND
: 1805      2509      7      ( .XAB [ XAB$B_COD ] EQLU XABSC_ALL )
: 1806      2510      6      AND
: 1807      2511      6      ( .XAB [ XAB$B_AID ] EQLU .XAB_NUM ))
: 1808      2512      5      OR
: 1809      2513      7      (( .XAB_TYPE EQLU XABSC_KEY )
: 1810      2514      6      AND
: 1811      2515      7      ( .XAB [ XAB$B_COD ] EQLU XABSC_KEY )
: 1812      2516      6      AND
: 1813      2517      6      ( .XAB [ XAB$B_REF ] EQLU .XAB_NUM ))
: 1814      2518      4      ) THEN
: 1815      2519      5      BEGIN
: 1816      2520      5
: 1817      2521      5      NEW XAB = .XAB;
: 1818      2522      5      FOUND = _SET;
: 1819      2523      5      EXITLOOP;
: 1820      2524      5
: 1821      2525      4      END;
: 1822      2526      4
: 1823      2527      4      XAB = .XAB [ XAB$NXT ];
: 1824      2528      4
: 1825      2529      3      END;
: 1826      2530      3
: 1827      2531      2      END;
: 1828      2532      2
: 1829      2533      2      IF NOT .FOUND
: 1830      2534      2      THEN

```

```

: 1831      2535      3      BEGIN
: 1832      2536      3      :
: 1833      2537      3      : Allocate a buffer for the new xab
: 1834      2538      3      :
: 1835      2539      3      NEW_XAB = FDL$$GET_VM( .XAB_LEN );
: 1836      2540      3      :
: 1837      2541      3      : If this is the first xab link it to the fab else just connect it to
: 1838      2542      3      : the last xab in the chain
: 1839      2543      3      :
: 1840      2544      3      IF .FDL$AB_PARSED_FAB [ FAB$SL_XAB ] EQL 0
: 1841      2545      3      THEN
: 1842      2546      3      FDL$AB_PARSED_FAB [ FAB$SL_XAB ] = .NEW_XAB
: 1843      2547      3      ELSE
: 1844      2548      3      END_XAB [ XAB$SL_NXT ] = .NEW_XAB;
: 1845      2549      3      :
: 1846      2550      3      END_XAB = .NEW_XAB;
: 1847      2551      3      :
: 1848      2552      3      END;
: 1849      2553      3      :
: 1850      2554      3      ! Make this xab the current one
: 1851      2555      3      :
: 1852      2556      3      CURRENT_XAB = .NEW_XAB;
: 1853      2557      3      :
: 1854      2558      3      IF NOT .FOUND
: 1855      2559      3      THEN
: 1856      2560      3      BEGIN
: 1857      2561      3      :
: 1858      2562      3      : Init. some stuff in it
: 1859      2563      3      :
: 1860      2564      3      CURRENT_XAB [ XAB$B_COD ] = .XAB_TYPE;
: 1861      2565      3      CURRENT_XAB [ XAB$B_BLN ] = .XAB_LEN;
: 1862      2566      3      CURRENT_XAB [ XAB$SL_NXT ] = 0;
: 1863      2567      3      :
: 1864      2568      3      END;
: 1865      2569      3      :
: 1866      2570      3      RETURN .CURRENT_XAB
: 1867      2571      3      :
: 1868      2572      3      END;

```

007C 00000 ALLOCATE_XAB:					
				Save R2,R3,R4,R5,R6	: 2422
56	00000000G	00	9E 00002	MOVAB FDL\$AB_PARSED_FAB, R6	
55	00000000'	00	9E 00009	MOVAB CURRENT_XAB, R5	
52	04	AC	D0 00010	MOVL XAB_TYPE, R2	: 2478
14		52	D1 00014	CML R2, #20	: 2480
		05	12 00017	BNEQ 1\$	
53		20	D0 00019	MOVL #32, XAB_LEN	
		37	11 0001C	BRB 7\$	
12		52	D1 0001E 1\$:	CML R2, #18	: 2481
		05	12 00021	BNEQ 2\$	
53		2C	D0 00023	MOVL #44, XAB_LEN	
		2D	11 00026	BRB 7\$	
22		52	D1 00028 2\$:	CML R2, #34	: 2482

				05	12	0002B	BNEQ	3\$		
				3C	D0	0002D	MOVL	#60, XAB_LEN		
				23	11	00030	BRB	7\$		
				52	D1	00032	CMPL	R2, #21		2483
				06	12	00035	BNEQ	4\$		
				8F	9A	00037	MOVZBL	#76, XAB_LEN		
				18	11	0003B	BRB	7\$		
				52	D1	0003D	CMPL	R2, #19		2484
				06	12	00040	BNEQ	5\$		
				8F	9A	00042	MOVZBL	#88, XAB_LEN		
				0D	11	00046	BRB	7\$		
				52	D1	00048	CMPL	R2, #30		2485
				05	13	0004B	BEQL	6\$		
				01	CE	0004D	MNEGL	#1, XAB_LEN		
				03	11	00050	BRB	7\$		
				14	D0	00052	MOVL	#20, XAB_LEN		
				54	D4	00055	CLRL	FOUND		2488
				00	E9	00057	BLBC	FDL\$AB_CTRL+2, 13\$		2494
				52	D1	0005E	CMPL	R2, #20		2496
				05	13	00061	BEQL	8\$		
				52	D1	00063	CMPL	R2, #21		
				3D	12	00066	BNEQ	13\$		
				66	D0	00068	MOVL	FDL\$AB_PARSED_FAB, R0		2500
				A0	D0	0006B	MOVL	36(R0), XAB		
				34	13	0006F	B^QL	13\$		2502
				52	D1	00071	CMPL	R2, #20		2507
				0E	12	00074	BNEQ	10\$		
				61	91	00076	CMPB	(XAB), #20		2509
				09	12	00079	BNEQ	10\$		
08	AC	17	A1	08	00	ED 0007B	CMPZV	#0, #8, 23(XAB), XAB_NUM		2511
				13	13	00082	BEQL	11\$		
				52	D1	00084	CMPL	R2, #21		2513
				16	12	00087	BNEQ	12\$		
				61	91	00089	CMPB	(XAB), #21		2515
				11	12	0008C	BNEQ	12\$		
08	AC	17	A1	08	00	ED 0008E	CMPZV	#0, #8, 23(XAB), XAB_NUM		2517
				08	12	00095	BNEQ	12\$		
				50	51	D0 00097	MOVL	XAB, NEW XAB		2521
				54	01	D0 0009A	MOVL	#1, FOUND		2522
				06	11	0009D	BRB	13\$		2519
				51	A1	D0 0009F	MOVL	4(XAB), XAB		2527
				CA	11	000A3	BRB	9\$		2502
				54	E8	000A5	BLBS	FOUND, 16\$		2533
				53	DD	000A8	PUSHL	XAB_LEN		2539
				00	01	FB 000AA	CALLS	#1, FDL\$\$GET_VM		
				51	66	D0 000B1	MOVL	FDL\$AB_PARSED_FAB, R1		2544
				24	A1	D5 000B4	TSTL	36(R1)		
				06	12	000B7	BNEQ	14\$		
				24	A1	D0 000B9	MOVL	NEW_XAB, 36(R1)		2546
				08	11	000BD	BRB	15\$		
				51	A5	D0 000BF	MOVL	END_XAB, R1		2548
				04	A1	D0 000C3	MOVL	NEW_XAB, 4(R1)		
				04	A5	D0 000C7	MOVL	NEW_XAB, END_XAB		2550
				65	50	D0 000CB	MOVL	NEW_XAB, CURRENT_XAB		2556
				0D	54	E8 000CE	BLBS	FOUND, 17\$		2558
				50	65	D0 000D1	MOVL	CURRENT_XAB, R0		2564
				60	52	90 000D4	MOVB	R2, (R0)		

FDLPARSE  
V04-000

VAX-11 FDL Utilities  
ALLOCATE\_XAB

K 9  
16-Sep-1984 01:50:08  
14-Sep-1984 12:31:19

VAX-11 Bliss-32 V4.0-742  
DISKSVMSMASTER:[FDL.SRC]FDLPARSE.B32;1 Page 61  
(19)

01	A0	53	90	000D7	MOVW	XAB_LEN, 1(R0)	:	2565
	50	A0	D4	000DB	CLRL	4(R0)	:	2566
		65	D0	000DE	MOVL	CURRENT_XAB, R0	:	2570
		04	04	000E1	RET		:	2572

; Routine Size: 226 bytes, Routine Base: \_FDL\$CODE + 0BFF

FL  
VC

```

: 1870 2573 1 %SBTTL 'FIND_ID'
: 1871 2574 1 ROUTINE FIND_ID : NOVALUE =
: 1872 2575 1 ++
: 1873 2576 1
: 1874 2577 1 Functional Description:
: 1875 2578 1
: 1876 2579 1 Finds a file ID of a file specified by the FDL$STRING descriptor
: 1877 2580 1
: 1878 2581 1 Calling Sequence:
: 1879 2582 1
: 1880 2583 1 find_id()
: 1881 2584 1
: 1882 2585 1 Input Parameters:
: 1883 2586 1 none
: 1884 2587 1
: 1885 2588 1 Implicit Inputs:
: 1886 2589 1 none
: 1887 2590 1
: 1888 2591 1 Output Parameters:
: 1889 2592 1 none
: 1890 2593 1
: 1891 2594 1 Implicit Outputs:
: 1892 2595 1 none
: 1893 2596 1
: 1894 2597 1 Routine Value:
: 1895 2598 1 none
: 1896 2599 1
: 1897 2600 1 Routines Called:
: 1898 2601 1
: 1899 2602 1 fdl$$get_vm
: 1900 2603 1
: 1901 2604 1 Side Effects:
: 1902 2605 1 none
: 1903 2606 1
: 1904 2607 1 --
: 1905 2608 1
: 1906 2609 2 BEGIN
: 1907 2610 2
: 1908 2611 2 LOCAL
: 1909 2612 2 FAB : REF BLOCK [ ,BYTE ],
: 1910 2613 2 NAM : REF BLOCK [ ,BYTE ];
: 1911 2614 2
: 1912 2615 2 ! Get the address space for the FAB and the Name block
: 1913 2616 2 !
: 1914 2617 2 FAB = FDL$$GET_VM( FAB$K_BLN );
: 1915 2618 2
: 1916 2619 2 NAM = FDL$$GET_VM( NAM$K_BLN + ESA_BUF_SIZ );
: 1917 2620 2
: 1918 2621 2 |
: 1919 2622 2 | +-----+
: 1920 2623 2 | | nam blk |
: 1921 2624 2 | +-----+
: 1922 2625 2 | | exp str buf |
: 1923 2626 2 | +-----+
: 1924 2627 2 |
: 1925 2628 2 ! Init the blocks and fill in all of the good stuff
: 1926 P 2629 2 $FAB_INIT ( FAB = .FAB,

```



```

: 1927 P 2630 2          FNA = .FDL$AB_STRING [ DSC$A_POINTER ],
: 1928 P 2631 2          FNS = .FDL$AB_STRING [ DSC$W_LENGTH ],
: 1929 2632 2          NAM = .NAM );
: 1930 2633 2
: 1931 P 2634 2          $NAM_INIT ( ESA = .NAM + NAM$K_BLN,
: 1932 P 2635 2          ESS = ESA_BUF_SIZ,
: 1933 2636 2          NAM = .NAM );
: 1934 2637 2
: 1935 2638 2          ! Parse and search for the file
: 1936 2639 2          !
: 1937 2640 2          IF $PARSE( FAB=.FAB )
: 1938 2641 2          THEN
: 1939 2642 2
: 1940 2643 2          IF $SEARCH( FAB=.FAB )
: 1941 2644 2          THEN
: 1942 2645 2          BEGIN
: 1943 2646 2
: 1944 2647 2          ! Get the old file ID
: 1945 2648 2          !
: 1946 2649 2          FDL$GL_FID1 = .NAM [ NAM$W_FID_NUM ];
: 1947 2650 2          FDL$GL_FID2 = .NAM [ NAM$W_FID_SEQ ];
: 1948 2651 2          FDL$GL_FID3 = .NAM [ NAM$W_FID_RVN ];
: 1949 2652 2
: 1950 2653 2          END
: 1951 2654 2          ELSE
: 1952 2655 2          SIGNAL( FDL$_RFLOC )
: 1953 2656 2          ELSE
: 1954 2657 2          SIGNAL( FDL$_RFLOC );
: 1955 2658 2
: 1956 2659 2          ! Deallocate the space we used
: 1957 2660 2          !
: 1958 2661 2          FDL$$FREE_VM( FAB$K_BLN, .FAB );
: 1959 2662 2          FDL$$FREE_VM( NAM$K_BLN+ESA_BUF_SIZ, .NAM );
: 1960 2663 2
: 1961 2664 2          RETURN
: 1962 2665 2
: 1963 2666 1          END;

```

.EXTRN SYS\$PARSE, SYS\$SEARCH

			03FC 0000	FIND_ID:	.WORD	Save R2,R3,R4,R5,R6,R7,R8,R9	: 2574
		59	00000000V	00	9E 00002	MOVAB	FDL\$\$GET_VM, R9
		58	00000000V	00	9E 00009	MOVAB	FDL\$\$FREE_VM, R8
		7E	50	8F	9A 00010	MOVZBL	#80, -(SP)
		69		01	FB 00014	CALLS	#1, FDL\$\$GET_VM
		57		50	DO 00017	MOVL	R0, FAB
		7E	015F	8F	3C 0001A	MOVZWL	#351, -(SP)
		69		01	FB 0001F	CALLS	#1, FDL\$\$GET_VM
		56		50	DO 00022	MOVL	R0, NAM
0050	8F			00	2C 00025	MOVCS	#0, (SP), #0, #80, (FAB)
				67	0002C		
		67	5003	8F	80 0002D	MOVW	#20483, (FAB)
		16	A7	02	90 00032	MOVB	#2, 22(FAB)
		1F	A7	02	90 00036	MOVB	#2, 31(FAB)
		28	A7	56	DO 0003A	MOVL	NAM, 40(FAB)
							: 2617
							: 2619
							: 2632

0060	8F	00	2C 34	A7 A7 6E	00000000G 00000000G	00 00 00	D0 90 2C	0003E 00046 0004E	MOV MOVB MOVCS	FDL\$AB_STRING+4, 44(FAB) FDL\$AB_STRING, 52(FAB) #0, (SP), #0, #96, (NAM)	:	2636	:
				66 0A 0C	6002 60	8F 01 A6	B0 8E A6	00056 0005B 0005F	MOVW MNEGB MOVAB	#24578, (NAM) #1, 10(NAM) 96(R6), 12(NAM)	:		:
		00000000G				57 00 26	DD FB E9	00064 00066 0006D	PUSHL CALLS BLBC	FAB #1, SYSSPARSE R0, 1\$	:	2640	:
		00000000G				57 00 1A	DD FB E9	00070 00072 00079	PUSHL CALLS BLBC	FAB #1, SYSSSEARCH R0, 1\$	:	2643	:
		00000000G			24	A6	3C	0007C	MOVZWL	36(NAM), FDL\$GL_FID1	:	2649	:
		00000000G			26	A6	3C	00084	MOVZWL	38(NAM), FDL\$GL_FID2	:	2650	:
		00000000G			28	A6	3C	0008C	MOVZWL	40(NAM), FDL\$GL_FID3	:	2651	:
						0D	11	00094	BRB	2\$	:		:
		00000000G			00000000G	8F	DD	00096	1\$: PUSHL	#FDL\$ RFLOC	:	2657	:
						01	FB	0009C	CALLS	#1, LIBSSIGNAL	:		:
						57	DD	000A3	2\$: PUSHL	FAB	:	2661	:
						7E	8F	9A	MOVZBL	#80, -(SP)	:		:
					50	02	FB	000A9	CALLS	#2, FDL\$\$FREE_VM	:		:
						56	DD	000AC	PUSHL	NAM	:	2662	:
						7E	8F	3C	MOVZWL	#351, -(SP)	:		:
					015F	02	FB	000B3	CALLS	#2, FDL\$\$FREE_VM	:		:
						04	00	000B6	RET		:	2666	:

; Routine Size: 183 bytes, Routine Base: \_FDL\$CODE + 0CE1

```

: 1965 2667 1 %SBTTL 'GET VM'
: 1966 2668 1 GLOBAL ROUTINE FDL$$GET_VM( BYTES ) =
: 1967 2669 1  **
: 1968 2670 1
: 1969 2671 1 Functional Description:
: 1970 2672 1
: 1971 2673 1 Allocate virtual memory and zeros it
: 1972 2674 1
: 1973 2675 1 Calling Sequence:
: 1974 2676 1
: 1975 2677 1 fdl$$get_vm( bytes )
: 1976 2678 1
: 1977 2679 1 Input Parameters:
: 1978 2680 1
: 1979 2681 1 bytes - number of bytes to allocate
: 1980 2682 1
: 1981 2683 1 Implicit Inputs:
: 1982 2684 1 none
: 1983 2685 1
: 1984 2686 1 Output Parameters:
: 1985 2687 1 none
: 1986 2688 1
: 1987 2689 1 Implicit Outputs:
: 1988 2690 1 none
: 1989 2691 1
: 1990 2692 1 Routine Value:
: 1991 2693 1
: 1992 2694 1 address of the start of the buffer
: 1993 2695 1
: 1994 2696 1 Routine Called:
: 1995 2697 1 lib$get_vm
: 1996 2698 1
: 1997 2699 1 Side Effects:
: 1998 2700 1 none
: 1999 2701 1
: 2000 2702 1 --
: 2001 2703 1
: 2002 2704 1
: 2003 2705 2 BEGIN
: 2004 2706 2
: 2005 2707 2 LOCAL
: 2006 2708 2 VM_POINTER;
: 2007 2709 2
: 2008 2710 2 ! If we don't succede signal an error and stop
: 2009 2711 2 !
: 2010 2712 2 IF NOT LIB$GET_VM ( BYTES,VM_POINTER )
: 2011 2713 2 THEN
: 2012 2714 2 SIGNAL_STOP ( FDL$_!NSVIRMEM );
: 2013 2715 2
: 2014 2716 2 ! Zero this address space
: 2015 2717 2 !
: 2016 2718 2 CH$FILL ( 0, .BYTES, .VM_POINTER );
: 2017 2719 2
: 2018 2720 2 RETURN .VM_POINTER
: 2019 2721 2
: 2020 2722 1 END;

```

				003C 00000	.ENTRY	FDL\$\$GET_VM, Save R2,R3,R4,R5	:	2668
		5E		04 C2 00002	SUBL2	#4, SP	:	
				5E DD 00005	PUSHL	SP	:	2712
			04	AC 9F 00007	PUSHAB	BYTES	:	
		00000000G	00	02 FB 0000A	CALLS	#2, LIB\$GET_VM	:	
			0D	50 E8 00011	BLBS	R0, 1\$	:	
			00000000G	8F DD 00014	PUSHL	#FDL\$ INSVIRMEM	:	2714
		00000000G	00	01 FB 0001A	CALLS	#1, LIB\$STOP	:	
04	AC		6E	00 2C 00021 1\$:	MOVCS	#0, (SP), #0, BYTES, @VM_POINTER	:	2718
				BE 00027			:	
			50	6E D0 00029	MOVL	VM_POINTER, R0	:	2720
				04 0002C	RET		:	2722

: Routine Size: 45 bytes, Routine Base: \_FDL\$CODE + 0D98

```

: 2022 2723 1 %SBTTL 'FREE_VM'
: 2023 2724 1 GLOBAL ROUTINE FDL$$FREE_VM( BYTES,ADDR ) : NOVALUE =
: 2024 2725 1 **
: 2025 2726 1
: 2026 2727 1 Functional Description:
: 2027 2728 1
: 2028 2729 1 Deallocate virtual memory
: 2029 2730 1
: 2030 2731 1 Calling Sequence:
: 2031 2732 1
: 2032 2733 1 fdl$$free_vm( bytes,addr )
: 2033 2734 1
: 2034 2735 1 Input Parameters:
: 2035 2736 1
: 2036 2737 1 bytes - number of bytes to deallocate
: 2037 2738 1 addr - address of block
: 2038 2739 1
: 2039 2740 1 Implicit Inputs:
: 2040 2741 1 none
: 2041 2742 1
: 2042 2743 1 Output Parameters:
: 2043 2744 1 none
: 2044 2745 1
: 2045 2746 1 Implicit Outputs:
: 2046 2747 1 none
: 2047 2748 1
: 2048 2749 1 Routine Value:
: 2049 2750 1 none
: 2050 2751 1
: 2051 2752 1 Routine Called:
: 2052 2753 1
: 2053 2754 1 lib$free_vm
: 2054 2755 1
: 2055 2756 1 Side Effects:
: 2056 2757 1 none
: 2057 2758 1
: 2058 2759 1 --
: 2059 2760 1
: 2060 2761 2 BEGIN
: 2061 2762 2
: 2062 2763 2 LOCAL
: 2063 2764 2 STATUS;
: 2064 2765 2
: 2065 2766 2 ! If we don't succede signal an error and stop
: 2066 2767 2 !
: 2067 2768 3 IF NOT ( STATUS = LIB$FREE_VM ( BYTES,ADDR ) )
: 2068 2769 2 THEN
: 2069 2770 2 SIGNAL_STOP ( .STATUS );
: 2070 2771 2
: 2071 2772 2 RETURN
: 2072 2773 2
: 2073 2774 1 END;

```

```

0000 00000 .ENTRY FDL$$FREE_VM, Save nothing ; 2724
08 AC 9F 00002 PUSHAB ADDR ; 2768
04 AC 9F 00005 PUSHAB BYTES
00000000G 00 02 FB 00008 CALLS #2, LIB$FREE_VM
09 50 E8 0000F BLBS STATUS, 1$
00000000G 00 50 DD 00012 PUSHL STATUS ; 2770
01 FB 00014 CALLS #1, LIB$STOP
04 0001B 1$: RET ; 2774

```

; Routine Size: 28 bytes, Routine Base: \_FDL\$CODE + 0DC5

```

; 2074 2775 1
; 2075 2776 0 END ELUDOM

```

.EXTRN LIB\$SIGNAL, LIB\$STOP

PSECT SUMMARY

Name	Bytes	Attributes
_FDL\$DOWN	28	NOVEC, WRT, RD, NOEXE, NOSHR, LCL, REL, CON, PIC, ALIGN(2)
_FDL\$CODE	3553	NOVEC, NOWRT, RD, EXE, SHR, LCL, REL, CON, PIC, ALIGN(2)

Library Statistics

File	Total	Symbols Loaded	Percent	Pages Mapped	Processing Time
_\$255\$DUA28:[SYSLIB]STARLET.L32;1	9776	244	2	581	00:01.0

COMMAND QUALIFIERS

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

```

; Size: 3553 code + 28 data bytes
; Run Time: 00:59.3
; Elapsed Time: 03:08.7
; Lines/CPU Min: 2809
; Lexemes/CPU-Min: 21493
; Memory Used: 276 pages
; Compilation Complete

```

0177 AH-BT13A-SE  
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION  
CONFIDENTIAL AND PROPRIETARY

FDL\_PARSE LIS

FDL\_PARSE LIS

FDL\_SD\_MSG LIS

FDL\_TABES LIS

FDL\_GEN\_TAB LIS

FDL\_MSG LIS

The image displays a grid of 100 small, overlapping window-like panels arranged in a 10x10 pattern. Each panel contains technical data, including text, tables, and diagrams. The panels are semi-transparent and overlap, creating a dense, multi-layered visual effect. The overall color scheme is dark blue and black, with white and light blue text and graphics. The panels appear to be screenshots of various system utilities or data processing tools, consistent with the VAX/VMS operating system mentioned in the header.