

DDDDDDDDDDDDDD		CCCCCCCCCCCC	LLL
DDDDDDDDDDDDDD		CCCCCCCCCCCC	LLL
DDDDDDDDDDDDDD		CCCCCCCCCCCC	LLL
DDD	DDD	CCC	LLL
DDD	DDD	CCC	LLL
DDD	DDD	CCC	LLL
DDD	DDD	CCC	LLL
DDD	DDD	CCC	LLL
DDD	DDD	CCC	LLL
DDD	DDD	CCC	LLL
DDD	DDD	CCC	LLL
DDD	DDD	CCC	LLL
DDD	DDD	CCC	LLL
DDD	DDD	CCC	LLL
DDD	DDD	CCC	LLL
DDD	DDD	CCC	LLL
DDD	DDD	CCC	LLL
DDDDDDDDDDDDDD		CCCCCCCCCCCC	LLLLLLLLLLLLLLLL
DDDDDDDDDDDDDD		CCCCCCCCCCCC	LLLLLLLLLLLLLLLL
DDDDDDDDDDDDDD		CCCCCCCCCCCC	LLLLLLLLLLLLLLLL

[illegible]

LEXICON
Table of contents

I 6
- LEXICAL FUNCTION PARSE AND EXECUTION R 16-SEP-1984 00:00:36 VAX/VMS Macro V04-00

Page 0

(4)	513	CONTROL PARAMETERS
(4)	513	SYSTEM MESSAGE PARAMETERS
(4)	513	SYSTEM LOADABLE CODE PARAMETERS
(4)	513	TERMINAL DRIVER SYSTEM PARAMETERS
(4)	513	RMS DEFAULT PARAMETERS
(4)	513	FILE ACP CONFIGURATION DATA
(4)	513	Job Controller Parameters
(4)	513	Login Security Parameters
(4)	513	Cluster Parameters
(6)	1056	EVALUATE LEXICAL FUNCTION STRING
(7)	1101	LEXICAL FUNCTION DISPATCHER
(9)	1384	CONVERT SIGNED AND UNSIGNED INTEGER FIELD
(10)	1433	CONVERT TO COMPARABLE DATE/TIME STRING
(11)	1616	RETURN DEFAULT DIRECTORY NAME STRING
(12)	1640	EDIT A STRING
(13)	1697	FIND A STRING ELEMENT
(14)	1749	GET DCL INFORMATION
(15)	2013	EXTRACT SUBSTRING FROM STRING
(16)	2046	FAO SYSTEM SERVICE
(17)	2093	GET FILE ATTRIBUTES
(18)	2310	GETDVI SYSTEM SERVICE
(19)	2508	GETJPI SYSTEM SERVICE
(20)	2701	GETSYI SYSTEM SERVICE
(21)	2807	CONVERT IDENTIFIER
(22)	2864	CONVERT TO INTEGER
(23)	2889	RETURN LENGTH OF STRING
(24)	2912	LOCATE SUBSTRING IN STRING
(25)	2941	LOGICAL NAME TRANSLATION
(26)	3005	LOGICAL NAME TRANSLATION
(27)	3239	GET MESSAGE TEXT
(28)	3264	RETURN JOB MODE
(29)	3312	PARSE A FILE SPECIFICATION
(30)	3459	GET NEXT PROCESS ID
(31)	3522	CHECK IF SPECIFIED PRIVS ARE SET OR CLEAR
(32)	3583	RETURN PROCESS NAME
(33)	3625	GET NEXT WILDCARD FILESPEC
(34)	3774	SET/CLEAR SPECIFIED PRIVILEGES
(35)	3856	APPEND PRIVILEGE STATE TO STRING
(36)	3904	CONVERT EXPRESSION TO ASCII STRING
(37)	3929	RETURN CURRENT TIME AND DATE
(38)	3961	GET SYMBOL TYPE
(39)	4008	RETURN USER IDENTIFICATION CODE
(40)	4047	RETURN VERIFY MODE
(41)	4087	ERROR RETURNS
(42)	4139	CONVERT UIC TO ASCII STRING
(43)	4173	CONVERT PROTECTION MASK TO ASCII STRING
(44)	4239	CONVERT FILE ID TO ASCII STRING
(45)	4271	CONVERT BINARY TO BOOLEAN STRING

```
0000 1 .TITLE LEXICON - LEXICAL FUNCTION PARSE AND EXECUTION ROUTINES
0000 2 .IDENT 'V04-000'
0000 3
0000 4
0000 5 *****
0000 6
0000 7 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0000 8 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0000 9 * ALL RIGHTS RESERVED.
0000 10
0000 11 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0000 12 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0000 13 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0000 14 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0000 15 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0000 16 * TRANSFERRED.
0000 17
0000 18 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000 19 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0000 20 * CORPORATION.
0000 21
0000 22 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0000 23 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0000 24
0000 25
0000 26 *****
0000 27
0000 28 LEXICAL FUNCTION PARSE AND EXECUTION SUPPORT ROUTINES
0000 29
0000 30 D. N. CUTLER 3-FEB-78
0000 31
0000 32 MODIFIED BY:
0000 33
0000 34 V03-034 HWS0103 Harold Schultz 15-Aug-1984
0000 35 When unable to find lexical function, undo search
0000 36 optimization in function name before outputting
0000 37 error message (Clear high bit in first character
0000 38 of function name).
0000 39
0000 40 V03-033 PCG0024 Peter George 31-May-1984
0000 41 Preclear XAB used by FSFILE to avoid RMS accvio.
0000 42
0000 43 V03-032 HWS0062 Harold Schultz 27-Apr-1984
0000 44 In FSPARSE, return requested portion of file spec.
0000 45 even if parse error. (undo PCG0018). Return null only
0000 46 if no specific portion of file spec requested.
0000 47
0000 48 V03-031 HWS0051 Harold Schultz 05-Apr-1984
0000 49 When doing an FSPARSE, cleanup any remaining internal
0000 50 RMS context.
0000 51
0000 52 V03-030 HWS0046 Harold Schultz 02-Apr-1984
0000 53 Translate concealed logical names when checking if a
0000 54 file is "known" with FSFILE.
0000 55
0000 56 V03-029 PCG0023 Peter George 02-Mar-1984
0000 57 Clean up FSSEARCH if input is "...".
```

```
0000 58 :
0000 59 :
0000 60 :
0000 61 :
0000 62 :
0000 63 :
0000 64 :
0000 65 :
0000 66 :
0000 67 :
0000 68 :
0000 69 :
0000 70 :
0000 71 :
0000 72 :
0000 73 :
0000 74 :
0000 75 :
0000 76 :
0000 77 :
0000 78 :
0000 79 :
0000 80 :
0000 81 :
0000 82 :
0000 83 :
0000 84 :
0000 85 :
0000 86 :
0000 87 :
0000 88 :
0000 89 :
0000 90 :
0000 91 :
0000 92 :
0000 93 :
0000 94 :
0000 95 :
0000 96 :
0000 97 :
0000 98 :
0000 99 :
0000 100 :
0000 101 :
0000 102 :
0000 103 :
0000 104 :
0000 105 :
0000 106 :
0000 107 :
0000 108 :
0000 109 :
0000 110 :
0000 111 :
0000 112 :
0000 113 :
0000 114 :

V03-028 HWS0010 Harold Schultz 13-Feb-1984
Use PRC V CARRCNTL instead of contents of prompt string
to check for presence/absence of CR/LF
Enable a parameter list of parse types to be passed to
F$PARSE. Add new parse type NO CONCEAL.
In F$PARSE, check validity of name of filespec field parameter
before performing the parse.

V03-027 PCG0022 Peter George 02-Feb-1984
Add SYNTAX_ONLY argument to F$PARSE.
Have F$ID return a null string on all system service errors.

V03-026 RAS0247 Ron Schaefer 27-Jan-1984
Fix F$PARSE/F$SEARCH to release RMS context correctly
when doing the null parse.

V03-025 PCG0021 Peter George 16-Jan-1984
Fix broken branch.

V03-024 PCG0020 Peter George 12-Jan-1984
Fix broken branch.

V03-023 PCG0019 Peter George 11-Jan-1984
Have F$PARSE do a syntax-only parse.
Add F$EDIT lowercase.

V03-022 PCG0018 Peter George 28-Nov-1983
Return a null string no matter what part of an
non-existent file is requested in F$PARSE.

V03-021 TMK0001 Todd M. Katz 12-Oct-1983
Translate logical names using LNMSDCL_LOGICAL instead of
LNMSDEFAULT_SEARCH as the table name.

V03-020 PCG0017 Peter George 19-Sep-1983
Fix bug in GETJPI table expansion.

V03-019 PCG0016 Peter George 12-Sep-1983
Init NAM block ESS field in F$FILE.
Use $GETJPI definition of MODE.

V03-018 PCG0015 Peter George 17-Aug-1983
Change case of weekday strings.
Change the way that default protection is retrieved.
Specify that a NAM block is need for known file opens.
Have F$USER return new format UIC's.
Increase accuracy of F$GETJPI error reporting.
Add F$IDENTIFIER.

V03-017 PCG0014 Peter George 27-Jul-1983
Support both F$LOGICAL and F$TRNLNM.
Add LENGTH item to F$TRNLNM.

V03-016 PCG0013 Peter George 23-Jun-1983
Finish adding new arguments to F$CVTIME.
Use event flags more intelligently.
```

```
0000 115 : Use !XI FAO directive in FORMAT_UIC.
0000 116 : Add support for JNL ACP.
0000 117 : Skip blanks between lexical function name and open paren.
0000 118 : Make FSXLOGICAL FSLOGICAL. Add TABLE item to FSLOGICAL.
0000 119 : Add 'OTHER' process mode type.
0000 120 :
0000 121 : V03-015 PCG0012 Peter George 13-Jun-1983
0000 122 : Fix bug in F$ENV('CAPTIVE').
0000 123 : Fix broken branch.
0000 124 : Signal P2 error for NOSUCHNODE in F$GETSYI.
0000 125 :
0000 126 : V03-014 PCG0011 Peter George 01-Jun-1983
0000 127 : Add second and third arguments to F$CVTIME.
0000 128 : Add F$ENV('OUTPUT_RATE').
0000 129 :
0000 130 : V03-013 PCG0010 Peter George 27-May-1983
0000 131 : Add STDTIM data type.
0000 132 : Make modifications to F$GETSYI.
0000 133 :
0000 134 : V03-012 PCG0009 Peter George 30-Apr-1983
0000 135 : Add FSXLOGICAL.
0000 136 : Add F$ENV('CAPTIVE','INTERACTIVE').
0000 137 : Add second argument to F$VERIFY.
0000 138 : Add second argument to F$GETSYI.
0000 139 : Add HEXSTR data type.
0000 140 :
0000 141 : V03-011 PCG0008 Peter George 30-Mar-1983
0000 142 : Add F$ENV('KEY_STATE','VERIFY_PROCEDURE','VERIFY_IMAGE').
0000 143 : Return standardized protection string everywhere.
0000 144 : Remove all references to PRC_V_GOTO.
0000 145 :
0000 146 : V03-010 PCG0007 Peter George 03-Mar-1983
0000 147 : Move value conversion and editing routines to CONVERT.
0000 148 : Set UPD and DEL share options in F$FILE.
0000 149 : Add F$FILE ('KNOWN').
0000 150 :
0000 151 : V03-009 KFH0003 Ken Henderson 2 Mar 1983
0000 152 : Modified JPI_ITEM_CODE macro to conform to new
0000 153 : JPITABLE.MAR.
0000 154 :
0000 155 : V03-008 KFH0002 Ken Henderson 25 Feb 1983
0000 156 : Changed GETSYI macro to SYI_ITEM_CODE to inherit
0000 157 : all 217 SYSBOOT parameters and other new GETSYI
0000 158 : item-codes. Also changed GETDVI macro to DVI_ITEM_CODE
0000 159 : to inherit all new GETDVI item-codes. Also added
0000 160 : the new PSECT DCL$ZDATA to contain all the tables.
0000 161 :
0000 162 : V03-007 PCG0006 Peter George 15-Feb-1983
0000 163 : Cleanup some F$SEARCH code.
0000 164 : Change F$EDIT COMMENT to UNCOMMENT.
0000 165 : Change F$ENV FILENAME to PROCEDURE.
0000 166 : Remove F$ENV MESSAGE.
0000 167 : Change F$ENV MESSAGE_QUALIFIERS to MESSAGE.
0000 168 :
0000 169 : V03-006 KFH0001 Ken Henderson 10 Feb 1983
0000 170 : Changed GETJPI macro into JPI_ITEM_CODE
0000 171 : and added call to JPI_GENERATE_TABLE macro.
```

0000	172	:	
0000	173	:	V03-005 PCG0005 Peter George 07-Jan-1983
0000	174	:	Add F\$ENVIRONMENT.
0000	175	:	
0000	176	:	V03-004 PCG0004 Peter George 15-Dec-1982
0000	177	:	Add AUTHPRI, PAGEFILLLOC, SWPFILLLOC F\$GETJPI fields.
0000	178	:	Do not report F\$SEARCH errors.
0000	179	:	
0000	180	:	V03-003 PCG0003 Peter George 14-Nov-1982
0000	181	:	Fold DCL\$REMBLANKS and DCL\$UPCASE into DCL\$EDIT.
0000	182	:	
0000	183	:	V03-002 PCG0002 Peter George 07-Oct-1982
0000	184	:	Add F\$TYPE, F\$EDIT, and F\$ELEMENT.
0000	185	:	Add DCL\$UPCASE. Tighten up bounds checking in
0000	186	:	F\$CVSI and F\$CVUI.
0000	187	:	
0000	188	:	V03-001 PHL0101 Peter H. Lipman 28-Jun-1982
0000	189	:	Use the new services, \$GETDVIW, \$GETJPIW, \$GETSYIW
0000	190	:	
0000	191	:---	

```
0000 193 :  
0000 194 : MACRO LIBRARY CALLS  
0000 195 :  
0000 196 :  
0000 197 PRCDEF ;DEFINE PROCESS WORK AREA  
0000 198 WRKDEF ;DEFINE COMMAND WORK AREA  
0000 199 SYMDEF ;DEFINE TYPES OF SYMBOLS  
0000 200 IDFDEF ;DEFINE INDIRECT FRAME BLOCK  
0000 201 EDITDEF ;DEFINE EDIT CODES  
0000 202 $PPDDEF ;DEFINE LOGINOUT/DCL COMMUNICATION AREA  
0000 203 $CLMSGDEF ;DEFINE ERROR/STATUS VALUES  
0000 204 $DIBDEF ;DEFINE DEVICE INFORMATION FIELDS  
0000 205 $DEVDEF ;DEFINE DEVICE CHARACTERISTICS BITS  
0000 206 $DVIDEF ;DEFINE GETDVI INFORMATION FIELDS  
0000 207 $JPIDEF ;DEFINE JOB/PROCESS INFORMATION ID'S  
0000 208 $LNMDEF ;DEFINE LOGICAL NAME CODES  
0000 209 $PSLDEF ;DEFINE PSL CODES  
0000 210 $SYIDEF ;DEFINE GETSYI INFORMATION ID'S  
0000 211 $FABDEF ;FAB DEFINITIONS  
0000 212 $NAMDEF ;NAM DEFINITIONS  
0000 213 $XABDEF ;XAB DEFINITIONS  
0000 214 $XABALLDEF ;ALLOCATION XAB DEFINITIONS  
0000 215 $XABDATDEF ;DATE/TIME XAB DEFINITIONS  
0000 216 $XABFHCDDEF ;FILE HEADER XAB DEFINITIONS  
0000 217 $XABPRODEF ;PROTECTION XAB DEFINITIONS  
0000 218 $XABSUMDEF ;SUMMARY ISAM XAB DEFINITIONS  
0000 219 $STSDEF ;STATUS DEFINITIONS  
0000 220 :  
0000 221 :  
0000 222 : DEFINE THIS SO SYSPARAM WILL BE CONDITIONALIZED CORRECTLY  
0000 223 :  
00000000 0000 224 GETSYISW = 0  
0000 225 :  
0000 226 :  
0000 227 : FUNCTION ARGUMENT DEFINITIONS  
0000 228 :  
0000 229 :  
00000010 0000 230 MAX ARGS = 16 ;MAXIMUM NUMBER OF ARGUMENTS ALLOWED  
00000000 0000 231 NPARM=0 ;NUMBER OF PARAMETERS  
00000004 0000 232 P1=4 ;STACK OFFSET TO PARAMETER 1  
0000000C 0000 233 P2=12 ;STACK OFFSET TO PARAMETER 2  
00000014 0000 234 P3=20 ;STACK OFFSET TO PARAMETER 3  
0000001C 0000 235 P4=28 ;STACK OFFSET TO PARAMETER 4  
00000024 0000 236 P5=36 ;STACK OFFSET TO PARAMETER 5  
0000002C 0000 237 P6=44 ;STACK OFFSET TO PARAMETER 6  
0000 238 :  
0000 239 :  
0000 240 : DEFINE FORMAT OF LONGWORD DESCRIBING FUNCTION ARGUMENTS  
0000 241 :  
0000 242 : BYTE 0: NUMBER OF ARGUMENTS ALLOWED WITH FUNCTION  
0000 243 : BYTES 1-3: ARRAY OF 4 NIBBLES, EACH NIBBLE DESCRIBING AN ARGUMENT  
0000 244 :  
0000 245 : BITS 0-1: TYPE OF VALUE ALLOWED  
0000 246 : 0 = BINARY  
0000 247 : 1 = STRING  
0000 248 : 2 = REFERENCE TO SYMBOL NAME  
0000 249 : 3 = EITHER STRING OR BINARY
```

```
0000 250 : BIT 2: TRUE IF ARGUMENT OPTIONAL, ELSE REQUIRED
0000 251 : BIT 3: UNUSED
0000 252 :
0000 253 : IF THE NUMBER OF ARGUMENTS EXCEEDS 4, THEN ALL REMAINING ARGUMENTS
0000 254 : ARE TREATED AS 'EITHER STRING OR BINARY'.
0000 255 :
0000 256 :
00000000 0000 257 ARG_V_TYPE = 0 ; STARTING POSITION OF TYPE FIELD
00000002 0000 258 ARG_S_TYPE = 2 ; SIZE OF TYPE FIELD
00000000 0000 259 ARG_C_BINARY = 0 ; BINARY
00000001 0000 260 ARG_C_STRING = 1 ; STRING
00000002 0000 261 ARG_C_SYMREF = 2 ; REFERENCE TO SYMBOL NAME
00000003 0000 262 ARG_C_ANY = 3 ; EITHER STRING OR BINARY
00000002 0000 263 ARG_V_OPTIONAL = 2 ; TRUE IF ARGUMENT OPTIONAL
00000003 0000 264 ARG_V_KEYWORD = 3 ; TRUE IF KEYWORD ARGUMENT
00000008 0000 265 ARG_C_ENTSIZE = 8 ; SIZE OF ENTRY (NAME PLUS ARGS)
0000 266 :
0000 267 :
0000 268 ; MACRO TO GENERATE LEXICAL FUNCTION DEFINITION TABLE
0000 269 :
0000 270 :
0000 271 .MACRO FIRST4 NAME
0000 272 .NCHR $ZZ$,NAME
0000 273 $XX$ = 128
0000 274 .IRPC $YYS$,NAME
0000 275 .BYTE ^A'$YYS'$!$XX$
0000 276 $XX$ = 0
0000 277 .ENDR
0000 278 .REPT 4-$ZZ$
0000 279 .ASCII ..
0000 280 .ENDR
0000 281 .ENDM FIRST4
0000 282 :
0000 283 .MACRO FUNCARG TYPE,PRESNCE=REQUIRED,FORMAT=VALUE
0000 284 : RETURNS $XX$ WITH NIBBLE DESCRIBING ARGUMENT
0000 285 $XX$ = -1
0000 286 .IIF IDN %EXTRACT(0,1,TYPE),<N>, $XX$ = ARG_C_BINARY
0000 287 .IIF IDN %EXTRACT(0,1,TYPE),<S>, $XX$ = ARG_C_STRING
0000 288 .IIF IDN %EXTRACT(0,1,TYPE),<R>, $XX$ = ARG_C_SYMREF
0000 289 .IIF IDN %EXTRACT(0,1,TYPE),<A>, $XX$ = ARG_C_ANY
0000 290 .IF LT $XX$
0000 291 .ERROR ;INVALID ARGUMENT TYPE. (STRING, NUMERIC, REFERENCE OR ANY)
0000 292 .ENDC
0000 293 .IIF IDN <PRESNCE><OPTIONAL>, $XX$ = $XX$ + <1@ARG V OPTIONAL>
0000 294 .IIF IDN <FORMAT><KEYWORD>, $XX$ = $XX$ + <1@ARG_V_KEYWORD>
0000 295 .ENDM FUNCARG
0000 296 :
0000 297 .MACRO FUNCT NAME,ARGTYPES,FLAGS
0000 298 FIRST4 %EXTRACT(0,4,NAME)
0000 299 NARGS = 0
0000 300 .IRP $ZZ$,<ARGTYPES>
0000 301 NARGS = NARGS + 1
0000 302 .ENDR
0000 303 .IF IDN <FLAGS><VARIABLE>
0000 304 NARGS = MAX_ARGS
0000 305 .ENDC
0000 306 $ARG$ = 0
```

```

0000 307      $BITS = 0
0000 308      .IRP      $YYS,<ARGTYPES>
0000 309      FUNCARG $YYS
0000 310      $ARGSS = $ARGSS + <$XSS @ $BITS>
0000 311      $BITS = $BITS + 4
0000 312      .IIF GT $BITS-24, .ERROR ;TOO MANY EXPLICITLY DEFINED ARGUMENTS
0000 313      .ENDR
0000 314      .LONG      NARGS + <$ARGSS @ 3>
0000 315      .ENDM      FUNCT

```

```
0000 317 :  
0000 318 : LEXICAL FUNCTION DEFINITION TABLE  
0000 319 :  
0000 320 :  
00000000 321 .PSECT DCL$ZDATA,BYTE,RD,NOWRT  
0000 322 :  
0000 323 LXFUNTABLE:  
0000 324 FUNCT CVSINTEGER, <N,N,S> ;CONVERT SIGNED INTEGER  
0008 325 FUNCT CVUINTEGER, <N,N,S> ;CONVERT UNSIGNED INTEGER  
0010 326 FUNCT CVTIME, <<S,OPTIONAL>,- ;CONVERT TO COMPARABLE TIME STRING  
0010 327 <S,OPTIONAL,KEYWORD>,-  
0010 328 <S,OPTIONAL,KEYWORD>>  
0018 329 FUNCT DIRECTORY ;DEFAULT DIRECTORY STRING  
0020 330 FUNCT EDIT, <S,- ;EDIT A STRING  
0020 331 <S,REQUIRED,KEYWORD>>  
0028 332 FUNCT ELEMENT, <N,S,S> ;FIND THE SPECIFIED ELEMENT  
0030 333 FUNCT ENVIRONMENT, <<S,REQUIRED,KEYWORD>> ;GET DCL CONTEXT INFO  
0038 334 FUNCT EXTRACT, <N,N,S> ;EXTRACT SUBSTRING FROM STRING  
0040 335 FUNCT FAO, <S,ANY,ANY,- ;FAO SERVICE  
0040 336 ANY,ANY,ANY>,VARIABLE  
0048 337 FUNCT FILE_ATTRIBUTES, <S,- ;GET FILE ATTRIBUTES  
0048 338 <S,REQUIRED,KEYWORD>>  
0050 339 FUNCT GETDVI, <S,- ;GETDVI SYSTEM SERVICE  
0050 340 <S,REQUIRED,KEYWORD>>  
0058 341 FUNCT GETJPI, <S,- ;GETJPI SYSTEM SERVICE  
0058 342 <S,REQUIRED,KEYWORD>>  
0060 343 FUNCT GETSYI, <<S,REQUIRED,KEYWORD>,- ;GETSYI SYSTEM SERVICE  
0060 344 <S,OPTIONAL,KEYWORD>>  
0068 345 FUNCT IDENTIFIER, <ANY,- ;TRANSLATE IDENTIFIER  
0068 346 <S,REQUIRED,KEYWORD>>  
0070 347 FUNCT INTEGER, <ANY> ;CONVERT EXPRESSION TO INTEGER  
0078 348 FUNCT LENGTH, <S> ;STRING LENGTH IN BINARY  
0080 349 FUNCT LOCATE, <S,S> ;FIND OFFSET TO SUBSTRING  
0088 350 FUNCT LOGICAL, <S> ;LOGICAL NAME TRANSLATION  
0090 351 FUNCT MESSAGE, <N> ;GET MESSAGE STRING  
0098 352 FUNCT MODE ;KEYWORD DESCRIBING TYPE OF JOB  
00A0 353 FUNCT PARSE, <- ;PARSE A FILESPEC  
00A0 354 <S>,- ;PRIMARY FILESPEC  
00A0 355 <S,OPTIONAL>,- ;DEFAULT FILESPEC  
00A0 356 <S,OPTIONAL>,- ;RELATED (STICKY) FILESPEC  
00A0 357 <S,OPTIONAL,KEYWORD>,- ;FILESPEC FIELD DESIRED  
00A0 358 <S,OPTIONAL,KEYWORD>> ;TYPE OF PARSE  
00A8 359 FUNCT PID, <REF> ;GET NEXT PROCESS ID STRING  
00B0 360 FUNCT PRIVILEGE, <<S,REQUIRED,KEYWORD>> ;TRUE IF PRIVS ARE SET/CLEAR  
00B8 361 FUNCT PROCESS ;PROCESS NAME STRING  
00C0 362 FUNCT SEARCH, <S,<N,OPTIONAL>> ;GET NEXT WILDCARD FILESPEC  
00C8 363 FUNCT SETPRV, <<S,REQUIRED,KEYWORD>> ;SETPRV SYSTEM SERVICE  
00D0 364 FUNCT STRING, <ANY> ;CONVERT EXPRESSION TO STRING  
00D8 365 FUNCT TIME ;TIME OF DAY STRING  
00E0 366 FUNCT TRNLNM, <<S>,- ;LOGICAL NAME TRANSLATION  
00E0 367 <S,OPTIONAL>,- ;TABLE NAME  
00E0 368 <N,OPTIONAL>,- ;TRANSLATION NUMBER  
00E0 369 <S,OPTIONAL,KEYWORD>,- ;ACCESS MODE  
00E0 370 <S,OPTIONAL,KEYWORD>,- ;ATTRIBUTE  
00E0 371 <S,OPTIONAL,KEYWORD>> ;ITEM  
00E8 372 FUNCT TYPE <REF> ;TYPE OF SYMBOL  
00F0 373 FUNCT USER ;UIC STRING
```

```
00F8 374      FUNCT  VERIFY,      <<N,OPTIONAL>>,- ;VERIFY MODE IN BINARY, AND
00F8 375      <<N,OPTIONAL>> ;OPTIONALLY CHANGE VERIFY MODE
0100 376
00000100 0100 377 LXFUNTABLEN=.-LXFUNTABLE      ;LENGTH OF LEXICAL FUNCTION TABLE
0100 378
0100 379
0100 380 : DEFINE THE DTYPES THAT LEXICON USES (SUBSET OF ALL POSSIBLE DTYPES)
0100 381 :
00000000 0100 382      BIN = 0
00000001 0100 383      STR = 1
00000002 0100 384      HEX = 2
00000003 0100 385      PRV = 3
00000004 0100 386      BIT = 4
00000005 0100 387      HEXSTR = 5
00000006 0100 388      STDIM = 6
0100 389
0100 390 :
0100 391 : MACRO TO DEFINE VALID $GETJPI ITEMS WHICH MAY BE USED IN F$GETJPI
0100 392 : THIS MACRO IS CALLED BY THE COMMONLY DEFINED MACRO: JPI_GENERATE_TABLE
0100 393 :
0100 394      .MACRO  JPI_ITEM_CODE  BASE,-      ; of EXE$GETJPI internal tables
0100 395      NAME,-      ; of the item-code
0100 396      SOURCE,-      ; of the data
0100 397      DTYPE,-      ; data type and usage
0100 398      BITPOS,-      ; of the 'bitval' bit
0100 399      BITSIZ,-      ; of the 'bitval' bit
0100 400      OUTLEN,-      ; of the data
0100 401      STRUCT      ; structure containing the data
0100 402
0100 403      .WORD  JPI$_NAME
0100 404
0100 405      $XX$ = BIN
0100 406      .IIF IDN <DTYPE><DECNUM>,,      $XX$=BIN
0100 407      .IIF IDN <DTYPE><PADSTR>,,      $XX$=STR
0100 408      .IIF IDN <DTYPE><CNTSTR>,,      $XX$=STR
0100 409      .IIF IDN <DTYPE><HEXNUM>,,      $XX$=HEX
0100 410      .IIF IDN <DTYPE><PRVMSK>,,      $XX$=PRV
0100 411      .IIF IDN <DTYPE><STDUIC>,,      $XX$=BIN
0100 412      .IIF IDN <DTYPE><STDIM>,,      $XX$=STR
0100 413      .IIF IDN <DTYPE><STRDSC>,,      $XX$=STR
0100 414      .IIF IDN <DTYPE><BITVEC>,,      $XX$=BIN
0100 415      .IIF IDN <DTYPE><BITVAL>,,      $XX$=BIT
0100 416      .IIF IDN <DTYPE><PKTMSK>,,      $XX$=BIN
0100 417      .IIF IDN <DTYPE><ACPTYP>,,      $XX$=BIN
0100 418      .IIF IDN <DTYPE><HEX.TR>,,      $XX$=HEXSTR
0100 419
0100 420      .BYTE  $XX$
0100 421      .ASCII 'NAME'
0100 422      .ENDM  JPI_ITEM_CODE
```

```
0100 424 :
0100 425 :*****
0100 426 :
0100 427 : DEFINE VALID $ ITEMS WHICH MAY BE USED IN F$GETJPI
0100 428 :
0100 429 :*****
0100 430 :
0100 431 :
0100 432 GETJPI_TABLE:
0100 433 :
0100 434 : Each element of GETJPI_TABLE is:
0100 435 :
0100 436 :-----
0100 437 : .WORD JPI$ 'NAME'
0100 438 : .BYTE DTYPE
0100 439 : .ASCII 'NAME'
0100 440 :-----
0100 441 :
0100 442 JPI_GENERATE_TABLE
047C 443
00000000 444 .LONG 0
0480 445
0480 446 :
0480 447 : MACRO TO DEFINE VALID $GETSYI ITEMS WHICH MAY BE USED IN F$GETSYI
0480 448 : THIS MACRO IS CALLED BY THE COMMONLY DEFINED MACRO: SYI_GENERATE_TABLE
0480 449 :
0480 450 .MACRO SYI_ITEM_CODE BASE,- ; of EXE$GETSYI internal tables
0480 451 NAME,- ; of the item-code
0480 452 SOURCE,- ; of the data
0480 453 DTYPE,- ; data type and usage
0480 454 BITPOS,- ; of FLD data items
0480 455 BITSIZ,- ; of FLD data items
0480 456 OUTLEN ; of the returned data
0480 457
0480 458 .WORD SYI$ 'NAME'
0480 459
0480 460 $XX$ = BIN
0480 461 : .IIF IDN <DTYPE><DECNUM>, $XX$=BIN
0480 462 : .IIF IDN <DTYPE><PADSTR>, $XX$=STR
0480 463 : .IIF IDN <DTYPE><CNTSTR>, $XX$=STR
0480 464 : .IIF IDN <DTYPE><HEXNUM>, $XX$=HEX
0480 465 : .IIF IDN <DTYPE><PRVMSK>, $XX$=PRV
0480 466 : .IIF IDN <DTYPE><STDUI>, $XX$=BIN
0480 467 : .IIF IDN <DTYPE><STDIM>, $XX$=STR
0480 468 : .IIF IDN <DTYPE><STRDSC>, $XX$=STR
0480 469 : .IIF IDN <DTYPE><BITVEC>, $XX$=BIN
0480 470 : .IIF IDN <DTYPE><BITVAL>, $XX$=BIT
0480 471 : .IIF IDN <DTYPE><HEXSTR>, $XX$=HEXSTR
0480 472 :
0480 473 :
0480 474 : The above mapping is identical to GETJPI, but the GETSYI code can't
0480 475 : yet handle DTYPEs other than HEX, BIN, STR, BIT, and HEXSTR
0480 476 : so the following table is used in the meantime until everything is done.
0480 477 :
0480 478 :
0480 479 .IIF IDN <DTYPE><DECNUM>, $XX$=BIN
0480 480 .IIF IDN <DTYPE><PADSTR>, $XX$=STR
```

```
0480 481      .IIF IDN <DTYPE><CNTSTR>,      $XX$=STR
0480 482      .IIF IDN <DTYPE><HEXNUM>,      $XX$=HEX
0480 483      .IIF IDN <DTYPE><PRVMSK>,      $XX$=BIN
0480 484      .IIF IDN <DTYPE><STDUIC>,      $XX$=BIN
0480 485      .IIF IDN <DTYPE><STDTIM>,      $XX$=STDTIM
0480 486      .IIF IDN <DTYPE><STRDSC>,      $XX$=STR
0480 487      .IIF IDN <DTYPE><BITVEC>,      $XX$=BIN
0480 488      .IIF IDN <DTYPE><BITVAL>,      $XX$=BIT
0480 489      .IIF IDN <DTYPE><HEXSTR>,      $XX$=HEXSTR
0480 490
0480 491      .BYTE      $XX$
0480 492      .ASCIC  'NAME'
0480 493      .ENDM      SYI_ITEM_CODE
0480 494
0480 495      :
0480 496      :*****
0480 497      :
0480 498      : DEFINE VALID $GETSYI ITEMS WHICH MAY BE USED IN F$GETSYI
0480 499      :
0480 500      :*****
0480 501      :
0480 502
0480 503 GETSYI_TABLE:
0480 504
0480 505 ; Each element of GETSYI_TABLE is:
0480 506 ;
0480 507      :-----:
0480 508      : .WORD  SYIS 'NAME' :
0480 509      : .BYTE  DTYPE :
0480 510      : .ASCIC 'NAME' :
0480 511      :-----:
0480 512
0480 513      SYI_GENERATE_TABLE
```

```

0C9C      .NLIST CND
0C9C      PARAMETER          ADDRESS=EXESGL_DEFFLAGS,-
                                DEFAULT=1,-
                                MAX=1,-
                                MIN=0,-
                                NAME=BUGREBOOT,-
                                BIT=EXESV_BUGREBOOT,-
                                TYPE=<DYNAMIC,SYS>,-
                                UNIT=Boolean
00000004   OUTLEN = 4
                                SYI_ITEM_CODE FLD,-
                                                <BUGREBOOT>-
                                                <EXESGL_DEFFLAGS>,-
                                                BITVAL=-
                                                <EXESV_BUGREBOOT>,-
                                                1,-
                                                1
                                2001   .WORD    SYS$_BUGREBOOT
00000000   $XX$ = BIN
                                :       .IIF IDN <BITVAL><DECNUM>,        $XX$=BIN
                                :       .IIF IDN <BITVAL><PADSTR>,         $XX$=STR
                                :       .IIF IDN <BITVAL><CNTSTR>,         $XX$=STR
                                :       .IIF IDN <BITVAL><HEXNUM>,         $XX$=HEX
                                :       .IIF IDN <BITVAL><PRVMASK>,        $XX$=PRV
                                :       .IIF IDN <BITVAL><STDUICL>,        $XX$=BIN
                                :       .IIF IDN <BITVAL><STDTIM>,         $XX$=STR
                                :       .IIF IDN <BITVAL><STRDSC>,         $XX$=STR
                                :       .IIF IDN <BITVAL><BITVEC>,         $XX$=BIN
                                :       .IIF IDN <BITVAL><BITVAL>,         $XX$=BIT
                                :       .IIF IDN <BITVAL><HEXSTR>,         $XX$=HEXSTR
                                :
                                : The above mapping is identical to GETJPI, but the
                                : yet handle DTYPES other than HEX, BIN, STR, BIT,
                                : so the following table is used in the meantime un
                                :
0C9E
0C9E
0C9E
0C9E
0C9E
0C9E
0C9E
0C9E
0C9E
0C9E
0C9E
0C9E
0C9E
0C9E
0C9E
0C9E
0C9E
0C9E
0C9E
0C9F
0C9F
0CA9
0CA9
55 42 00' .BYTE    $XX$
09         .ASCII  "BUGREBOOT"

```

```
00000000 148D 514
148D 515 .LONG 0
1491 516
1491 517 :
1491 518 : PRODUCE A TABLE OF ASCIC NAMES DESCRIBING EACH OF THE PROCESS STATES
1491 519 :
1491 520
1491 521 .MACRO $DEFINI NAME,P1,P2
1491 522 .ENDM
1491 523 .MACRO $DEFEND NAME,P1,P2
1491 524 .ENDM
1491 525 .MACRO $EQU SYMBOL,VALUE
1491 526 S = %LOCATE(<>,SYMBOL)+1
1491 527 L = %LENGTH(SYMBOL)-S
1491 528 .ASCIC /%EXTRACT(S,L,SYMBOL)/
1491 529 .BLKB 8-<L+1>
1491 530 .ENDM $EQU
1491 531
1491 532 STATE_TABLE:
1491 533 .ASCIC 'ILLEGAL' ; FOR ZERO CASE
1491 534
1499 534 $STATEDEF ; TABLE OF PROCESS STATE NAMES
1509 535
00000003 1509 536 .MDELETE $DEFINI,$DEFEND,$EQU
1509 537
1509 538
1509 539 :
1509 540 : MACRO TO DEFINE VALID DATA ITEMS WHICH MAY BE USED IN FSFILE_ATTRIBUTES
1509 541 :
1509 542
1509 543 .MACRO FILITEM NAME,BLOCK,FIELD,SIZE,BITNUM,TYPE=BINARY
1509 544 $XX$ = 0
1509 545 .IIF IDN <BLOCK><FAB>, $XX$=0
1509 546 .IIF IDN <BLOCK><NAM>, $XX$=1
1509 547 .IIF IDN <BLOCK><XABALL>, $XX$=2
1509 548 .IIF IDN <BLOCK><XABDAT>, $XX$=3
1509 549 .IIF IDN <BLOCK><XABFHC>, $XX$=4
1509 550 .IIF IDN <BLOCK><XABPRO>, $XX$=5
1509 551 .IIF IDN <BLOCK><XABSUM>, $XX$=6
1509 552 .BYTE $XX$ ; RMS BLOCK ID
1509 553 $XX$ = 0
1509 554 .IIF NB <FIELD>,$XX$ = 'FIELD'
1509 555 .BYTE $XX$ ; OFFSET TO FIELD IN RMS BLOCK
1509 556 $XX$ = 0
1509 557 .IIF IDN <SIZE><BIT>, $XX$=1
1509 558 .IIF IDN <SIZE><BYTE>, $XX$=8
1509 559 .IIF IDN <SIZE><WORD>, $XX$=16
1509 560 .IIF IDN <SIZE><LONG>, $XX$=32
1509 561 .BYTE $XX$ ; SIZE OF FIELD
1509 562 $XX$ = 0
1509 563 .IIF NB <BITNUM>,$XX$ = 'BITNUM'
1509 564 .BYTE $XX$ ; BIT NUMBER
1509 565 $XX$ = 0
1509 566 .IIF IDN <TYPE><BINARY>, $XX$=0
1509 567 .IIF IDN <TYPE><DATE>, $XX$=1
1509 568 .IIF IDN <TYPE><ASCIC>, $XX$=2
1509 569 .IIF IDN <TYPE><FILEID>, $XX$=3
```

```
1509 570 .IIF IDN <TYPE><ORG>, $XX$=4
1509 571 .IIF IDN <TYPE><RAT>, $XX$=5
1509 572 .IIF IDN <TYPE><RFM>, $XX$=6
1509 573 .IIF IDN <TYPE><UIC>, $XX$=7
1509 574 .IIF IDN <TYPE><PROT>, $XX$=8
1509 575 .IIF IDN <TYPE><EOF>, $XX$=9
1509 576 .IIF IDN <TYPE><BOOLEAN>, $XX$=10
1509 577 .IIF IDN <TYPE><KNOWN>, $XX$=11 ; FSFILE DEPENDS ON THIS BEI
1509 578 .BYTE $XX$ ; TYPE OF DATA ITEM
1509 579 .ASCIC 'NAME' ; ASCII NAME OF DATA ITEM
1509 580 .ENDM FILITEM
1509 581
1509 582 :
1509 583 : DEFINE XAB BLOCK ID'S AND LENGTHS
1509 584 :
1509 585
1509 586 XAB_INFO:
20 14 1509 587 .BYTE XABSC_ALL,XABSC_ALLLEN
2C 12 150B 588 .BYTE XABSC_DAT,XABSC_DATLEN
2C 1D 150D 589 .BYTE XABSC_FHC,XABSC_FHLEN
58 13 150F 590 .BYTE XABSC_PRO,XABSC_PROLEN
0C 16 1511 591 .BYTE XABSC_SUM,XABSC_SUMLEN
1513 592
1513 593 :
1513 594 : DEFINE VALID DATA ITEMS WHICH MAY BE USED IN FSFILE_ATTRIBUTES
1513 595 :
1513 596
1513 597 FILE_TABLE:
1513 598 FILITEM ALQ,FAB,FABSL_ALQ,SIZE=LONG ;ALLOCATION QUANTITY
151C 599 FILITEM BKS,FAB,FABSB_BKS,SIZE=BYTE ;BUCKET SIZE
1525 600 FILITEM BLS,FAB,FABSW_BLS,SIZE=WORD ;BLOCK SIZE
152E 601 FILITEM DEQ,FAB,FABSW_DEQ,SIZE=WORD ;DEFAULT EXTENSION QUANTITY
1537 602 FILITEM FSZ,FAB,FABSB_FSZ,SIZE=BYTE ;FIXED CONTROL AREA SIZE
1540 603 FILITEM MRN,FAB,FABSL_MRN,SIZE=LONG ;MAXIMUM RECORD NUMBER
1549 604 FILITEM MRS,FAB,FABSW_MRS,SIZE=WORD ;MAXIMUM RECORD SIZE
1552 605 FILITEM ORG,FAB,FABSB_ORG,SIZE=BYTE,TYPE=ORG ;FILE ORGANIZATION
155B 606 FILITEM RAT,FAB,FABSB_RAT,SIZE=BYTE,TYPE=RAT ;RECORD ATTRIBUTES
1564 607 FILITEM RFM,FAB,FABSB_RFM,SIZE=BYTE,TYPE=RFM ;RECORD FORMAT
156D 608 FILITEM KNOWN,NAM,FABSL_CTX,SIZE=LONG,TYPE=KNOWN;KNOWN FILE CHECK
1578 609 FILITEM CTG,FAB,FABSL_FOP,SIZE=BIT,BITNUM=FABSV_CTG,TYPE=BOOLEAN ;CONTIGUOUS
1581 610 FILITEM CBT,FAB,FABSL_FOP,SIZE=BIT,BITNUM=FABSV_CBT,TYPE=BOOLEAN ;CONTIGUOUS
158A 611 FILITEM RCK,FAB,FABSL_FOP,SIZE=BIT,BITNUM=FABSV_RCK,TYPE=BOOLEAN ;READ CHECK
1593 612 FILITEM WCK,FAB,FABSL_FOP,SIZE=BIT,BITNUM=FABSV_WCK,TYPE=BOOLEAN ;WRITE CHEC
159C 613 FILITEM CDT,XABDAT,XABSQ_CDT,TYPE=DATE ;CREATION DATE/TIME
15A5 614 FILITEM EDT,XABDAT,XABSQ_EDT,TYPE=DATE ;EXPIRATION DATE/TIME
15AE 615 FILITEM RDT,XABDAT,XABSQ_RDT,TYPE=DATE ;REVISION DATE/TIME
15B7 616 FILITEM BDT,XABDAT,XABSQ_BDT,TYPE=DATE ;BACKUP DATE/TIME
15C0 617 FILITEM RVN,XABDAT,XABSW_RVN,SIZE=WORD ;REVISION NUMBER
15C9 618 FILITEM UIC,XABPRO,XABSW_MBM,SIZE=LONG,TYPE=UIC ;OWNER UIC STRING
15D2 619 FILITEM GRP,XABPRO,XABSW_GRP,SIZE=WORD ;OWNER GROUP NUMBER
15DB 620 FILITEM MBM,XABPRO,XABSW_MBM,SIZE=WORD ;OWNER MEMBER NUMBER
15E4 621 FILITEM PRO,XABPRO,XABSW_PRO,SIZE=WORD,TYPE=PROT ;FILE PROTECTION BITMASK
15ED 622 FILITEM EOF,XABFHC,XABSL_EBK,SIZE=LONG,TYPE=EOF ;END-OF-FILE BLOCK NUMBER
15F6 623 FILITEM NOA,XABSUM,XABSB_NOA,SIZE=BYTE ;NUMBER OF AREAS
15FF 624 FILITEM NOK,XABSUM,XABSB_NOK,SIZE=BYTE ;NUMBER OF KEYS
1608 625 FILITEM PVN,XABSUM,XABSW_PVN,SIZE=WORD ;PROLOGUE VERSION NUMBER
1611 626 FILITEM DID,NAM,NAMSW_DID,TYPE=FILEID ;DIRECTORY ID NUMBER
```

```
00000000 00000000 161A 627      FILITEM FID,NAM,NAM$W_FID,TYPE=FILEID      ;FILE ID NUMBER
1623 628      FILITEM DVI,NAM,NAM$T_DVI,TYPE=ASCIC      ;DEVICE NAME STRING
162C 629      .QUAD 0
1634 630
1634 631      :
1634 632      : MACRO TO DEFINE VALID F$PARSE FILESPEC FIELDS
1634 633      :
1634 634      :
1634 635      .MACRO FILFLD NAME,LENGTH,ADDRESS
1634 636      .ASCIC 'NAME'
1634 637      .BYTE LENGTH      ;OFFSET TO LENGTH BYTE IN NAM
1634 638      .BYTE ADDRESS      ;OFFSET TO ADDRESS LONGWORD IN NAM
1634 639      .ENDM
1634 640
1634 641      :
1634 642      : DEFINE VALID FILESPEC FIELDS WHICH MAY BE RETURNED BY F$PARSE
1634 643      :
1634 644      :
1634 645      PARSE_FIELDS:
1634 646      FILFLD NODE,NAM$B_NODE,NAM$L_NODE
1638 647      FILFLD DEVICE,NAM$B_DEV,NAM$C_DEV
1644 648      FILFLD DIRECTORY,NAM$B_DIR,NAM$L_DIR
1650 649      FILFLD NAME,NAM$B_NAME,NAM$L_NAME
1657 650      FILFLD TYPE,NAM$B_TYPE,NAM$L_TYPE
165E 651      FILFLD VERSION,NAM$B_VER,NAM$L_VER
00 1668 652      .BYTE 0
1669 653
1669 654      .MACRO PRSTYP NAME,FIELD
1669 655      .ASCIC 'NAME'
1669 656      .BYTE NAM$M_'FIELD'
1669 657      .ENDM
1669 658
1669 659      PARSE_TYPES:
1669 660      PRSTYP SYNTAX_ONLY,SYNCHK
00 1676 661      PRSTYP NO_CONCEAL,NOCONCEAL
1682 662      .BYTE 0
1683 663
1683 664      :
1683 665      : MACRO TO DEFINE VALID $GETDVI ITEMS WHICH MAY BE USED IN F$GETDVI
1683 666      : THIS MACRO IS CALLED BY THE COMMONLY DEFINED MACRO: DVI_GENERATE_TABLE
1683 667      :
1683 668      .MACRO DVI_ITEM_CODE NAME,-      ; of the item-code
1683 669      SPECIAL,-      ; flag
1683 670      SOURCE,-      ; of the data
1683 671      DTYPE,-      ; data type and usage
1683 672      BITPOS,-      ; of the 'bitval' bit
1683 673      OUTLEN,-      ; of the returned data
1683 674      STRUCT,-      ; structure containing the data
1683 675      DEVTYP      ; flag
1683 676
1683 677      :
1683 678      : SPLDEVNAM and EXISTS are special cases
1683 679      :
1683 680      .IF IDN <NAME><SPLDEVNAM>
1683 681      .WORD DVI$_DEVNAM
1683 682      .IFF
1683 683
```

```
1683 684 .IF IDN <NAME><EXISTS>
1683 685 .WORD DVIS_DEVCHAR
1683 686 .IFF
1683 687 .WORD DVIS_'NAME'
1683 688 .ENDC
1683 689
1683 690 .ENDC
1683 691
1683 692 $XXS = 0
1683 693 .IIF IDN <DTYPE><DECNUM>, $XXS=0
1683 694 .IIF IDN <DTYPE><PADSTR>, $XXS=1
1683 695 .IIF IDN <DTYPE><CNTSTR>, $XXS=1
1683 696 .IIF IDN <DTYPE><HEXNUM>, $XXS=4
1683 697 .IIF IDN <DTYPE><PRVMSK>, $XXS=0
1683 698 .IIF IDN <DTYPE><STDUIC>, $XXS=2
1683 699 .IIF IDN <DTYPE><STDTIM>, $XXS=0
1683 700 .IIF IDN <DTYPE><STRDSC>, $XXS=1
1683 701 .IIF IDN <DTYPE><BITVEC>, $XXS=0
1683 702 .IIF IDN <DTYPE><BITVAL>, $XXS=3
1683 703 .IIF IDN <DTYPE><PRTMSK>, $XXS=5
1683 704 .IIF IDN <DTYPE><ACPTYP>, $XXS=6
1683 705 .IIF IDN <DTYPE><HEXSTR>, $XXS=7
1683 706
1683 707 .BYTE $XXS
1683 708
1683 709 .IF IDN <NAME><EXISTS>
1683 710 .BYTE DEV$V_EXISTS
1683 711 .IFF
1683 712 .BYTE 0
1683 713 .ENDC
1683 714
1683 715 .ASCIC 'NAME'
1683 716 .ENDM DVI_ITEM_CODE
1683 717
FFFFFFFF 1683 718 DEV$V_EXISTS = -1
```

```
1683 720 :
1683 721 :*****
1683 722 :
1683 723 : DEFINE VALID $GETDVI ITEMS WHICH MAY BE USED IN F$GETDVI
1683 724 :
1683 725 :*****
1683 726 :
1683 727 :
1683 728 GETDVI_TABLE:
1683 729
1683 730 : Each element of GETDVI_TABLE is:
1683 731 :
1683 732 :-----:
1683 733 :.WORD   DVIS 'NAME'
1683 734 :.BYTE   DTYPE
1683 735 :.BYTE   DEVSV_EXISTS
1683 736 :.ASCIC  'NAME'
1683 737 :-----:
1683 738
1683 739 DVI_GENERATE_TABLE
1E3D 740
1E3D 741
1E3D 742 : These codes are special to LEXICON, they aren't 'real' item-codes
1E3D 743 :
1E3D 744 DVI_ITEM_CODE SPLDEVNAM, F, DEVNAM, PADSTR, 0, 4, UCB, ANY
1E4B 745 DVI_ITEM_CODE EXISTS, F, DEVCHAR, BITVAL, 0, 1, UCB, ANY
1E56 746
00000000 00000000 1E56 747 .QUAD 0 ; END = ZERO LENGTH NAME
1E5E 748
1E5E 749 :
1E5E 750 : BUILD TABLE OF ACP TYPES
1E5E 751 :
00000001 1E5E 752 $XXS = 1
1E5E 753 .MACRO ACP NAME
1E5E 754 ASSUME DVIS_C_ACP_'NAME EQ $XXS
1E5E 755 $XXS = $XXS + 1
1E5E 756 .ASCIC 'NAME'
1E5E 757 .BLKB 8 - <%LENGTH(NAME)+1>
1E5E 758 .ENDM ACP
1E5E 759
1E5E 760 ACP_TABLE:
4C 41 47 45 4C 4C 49 00' 1E5E 761 .ASCIC 'ILLEGAL'
07 1E5E
1E66 762 ACP F11V1
1E6E 763 ACP F11V2
1E76 764 ACP MTA
1E7E 765 ACP NET
1E86 766 ACP REM
1E8E 767 ACP JNL
4C 41 47 45 4C 4C 49 00' 1E96 768 .ASCIC 'ILLEGAL'
07 1E96
1E9E 769
1E9E 770 :
1E9E 771 : MACRO TO DEFINE VALID F$EDIT ITEM NAMES
1E9E 772 :
1E9E 773 .MACRO EDIT NAME
1E9E 774 .ASCIC 'NAME'
```

```
1E9E 775 .BYTE EDIT_M_'NAME
1E9E 776 .ENDM
1E9E 777
1E9E 778 ;
1E9E 779 : DEFINE VALID F$EDIT ITEM NAMES
1E9E 780 :
1E9E 781 EDIT_TABLE:
1E9E 782 ASSUME EDIT_V_UNCOMMENT EQ 0
1E9E 783 ASSUME EDIT_V_COMPRESS EQ 1
1E9E 784 ASSUME EDIT_V_COLLAPSE EQ 2
1E9E 785 ASSUME EDIT_V_TRIM EQ 3
1E9E 786 ASSUME EDIT_V_UPCASE EQ 4
1E9E 787 ASSUME EDIT_V_LOWERCASE EQ 5
1E9E 788 ASSUME EDIT_V_EOL EQ 6
1E9E 789 EDIT UNCOMMENT
1EA9 790 EDIT COMPRESS
1EB3 791 EDIT COLLAPSE
1EBD 792 EDIT TRIM
1EC3 793 EDIT UPCASE
1ECB 794 EDIT LOWERCASE
00 1ED6 795 .BYTE 0
1ED7 796
1ED7 797 ;
1ED7 798 : MACRO TO DEFINE VALID F$ENVIRONMENT ITEM NAMES
1ED7 799 :
00000000 1ED7 800 $XX$ = 0
1ED7 801 .MACRO ENVIRON NAME
1ED7 802 .ASCIC 'NAME'
1ED7 803 .BYTE $XX$
1ED7 804 $XX$ = $XX$ + 1
1ED7 805 .ENDM
1ED7 806
1ED7 807 ;
1ED7 808 : DEFINE VALID F$ENVIRONMENT ITEM NAMES
1ED7 809 :
1ED7 810 ENVIRON_TABLE:
1ED7 811 ENVIRON CONTROL
1EE0 812 ENVIRON NOCONTROL
1EEB 813 ENVIRON DEFAULT
1EF4 814 ENVIRON DEPTH
1EFB 815 ENVIRON ON_SEVERITY
1F08 816 ENVIRON PROMPT
1F10 817 ENVIRON PROMPT_CONTROL
1F20 818 ENVIRON PROTECTION
1F2C 819 ENVIRON MESSAGE
1F35 820 ENVIRON MAX_DEPTH
1F40 821 ENVIRON ON_CONTROL_Y
1F4E 822 ENVIRON PROCEDURE
1F59 823 ENVIRON KEY_STATE
1F64 824 ENVIRON VERIFY_PROCEDURE
1F76 825 ENVIRON VERIFY_IMAGE
1F84 826 ENVIRON CAPTIVE
1F8D 827 ENVIRON INTERACTIVE
00 1F9A 828 ENVIRON OUTPUT_RATE
1FA7 829 .BYTE 0
1FA8 830
1FA8 831 ;
```

```
1FA8 832 : MACRO TO DEFINE VALID FSTRNLNM ITEM NAMES
1FA8 833 :
1FA8 834 .MACRO LOGICAL NAME,ITEM
1FA8 835 .ASCIC 'NAME'
1FA8 836 .WORD LNMS_'ITEM'
1FA8 837 $XX$ = 0
1FA8 838 .IIF IDN <ITEM><ATTRIBUTES>,$XX$ = LNMSV_'NAME'
1FA8 839 .BYTE $XX$
1FA8 840 .ENDM
1FA8 841
1FA8 842 :
1FA8 843 : DEFINE VALID FSTRNLNM ITEM NAMES
1FA8 844 :
1FA8 845 LOGICAL_TABLE:
1FA8 846 LOGICAL ACCESS MODE,ACMODE
1FB7 847 LOGICAL CONCEALED,ATTRIBUTES
1FC4 848 LOGICAL CONFINED,ATTRIBUTES
1FCF 849 LOGICAL CRELOG,ATTRIBUTES
1FD9 850 LOGICAL LENGTH,LENGTH
1FE3 851 LOGICAL MAX_INDEX,MAX_INDEX
1FF0 852 LOGICAL NO_ALIAS,ATTRIBUTES
1FFC 853 LOGICAL TABLE,ATTRIBUTES
2005 854 LOGICAL TABLE_NAME,TABLE
2013 855 LOGICAL TERMINAL,ATTRIBUTES
00 201F 856 LOGICAL VALUE,STRING
2028 857 .BYTE 0
2029 858
2029 859 :
2029 860 : MACRO TO DEFINE VALID FSTRNLNM ATTRIBUTE TYPES
2029 861 :
2029 862 .MACRO LOGATTR NAME
2029 863 .ASCIC 'NAME'
2029 864 $XX$ = 0
2029 865 .IIF IDN <NAME><CASE_BLIND>,$XX$ = LNMSM_'NAME'
2029 866 .LONG $XX$
2029 867 .ENDM
2029 868
2029 869 :
2029 870 : DEFINE VALID FSTRNLNM ATTRIBUTE TYPES
2029 871 :
2029 872 LOGATTR_TABLE:
2029 873 LOGATTR CASE_BLIND
2038 874 LOGATTR CASE_SENSITIVE
00 20'B 875 .BYTE 0
204C 876
204C 877 :
204C 878 : MACRO TO DEFINE VALID ACCESS MODES
204C 879 :
00000000 204C 880 $XX$ = 0
204C 881 .MACRO ACCESS NAME
204C 882 .ASCIC 'NAME'
204C 883 .BYTE $XX$
204C 884 $XX$ = $XX$ + 1
204C 885 .ENDM
204C 886
204C 887 :
204C 888 : DEFINE VALID ACCESS MODES (MUST BE IN PSL$C_XXX ORDER)
```

```
204C 889 :  
204C 890 ACCESS_TABLE:  
204C 891 ACCESS KERNEL  
2054 892 ACCESS EXECUTIVE  
205F 893 ACCESS SUPERVISOR  
206B 894 ACCESS USER  
00 2071 895 .BYTE 0  
2072 896  
2072 897 :  
2072 898 : BUILD TABLE OF MESSAGE SEVERITIES  
2072 899 :  
2072 900 .MACRO SEVERITY NAME  
2072 901 .ASCIC 'NAME'  
2072 902 .BLKB 8 - <%LENGTH(NAME)+1>  
2072 903 .ENDM SEVERITY  
2072 904  
2072 905 SEVERITY TABLE:  
2072 906 ASSUME STSSK_WARNING EQ 0  
2072 907 SEVERITY WARNING  
207A 908 ASSUME STSSK_ERROR EQ 2  
207A 909 SEVERITY ERROR  
2082 910 ASSUME STSSK_SEVERE EQ 4  
2082 911 SEVERITY SEVERE  
208A 912 SEVERITY NONE ;PLACE HOLDER FOR NEXT 'NONE'  
2092 913 SEVERITY NONE  
209A 914  
209A 915 :  
209A 916 : MACRO TO DEFINE VALID F$CVTIME TYPE NAMES  
209A 917 :  
00000000 209A 918 $XXS = 0  
209A 919 .MACRO CVTTYPE NAME  
209A 920 .ASCIC 'NAME'  
209A 921 .BYTE $XXS  
209A 922 $XXS = $XXS + 1  
209A 923 .ENDM  
209A 924  
209A 925 :  
209A 926 : DEFINE VALID F$CVTIME TYPE NAMES  
209A 927 :  
209A 928 CVTIME_TYPES:  
209A 929 CVTTYPE COMPARISON  
20A6 930 CVTTYPE ABSOLUTE  
00 20B0 931 CVTTYPE DELTA  
20B7 932 .BYTE 0  
20B8 933  
20B8 934 :  
20B8 935 : MACRO TO DEFINE VALID F$CVTIME ITEM NAMES  
20B8 936 :  
20B8 937 .MACRO CVTIME NAME,CSTART,CCOUNT,ASTART,ACOUNT,DSTART,DCOUNT  
20B8 938 .ASCIC 'NAME'  
20B8 939 .BYTE CSTART  
20B8 940 .BYTE CCOUNT  
20B8 941 .BYTE ASTART  
20B8 942 .BYTE ACOUNT  
20B8 943 .BYTE DSTART  
20B8 944 .BYTE DCOUNT  
20B8 945 .ENDM
```

```
20B8 946
20B8 947 :
20B8 948 : DEFINE VALID FSCVTIME ITEM NAMES
20B8 949 : (0,0 MEANS UNSUPPORTED FOR DELTA TIMES, WEEKDAY FOR OTHERS)
20B8 950 :
20B8 951 CVTIME_ITEMS:
20B8 952 CVTIME DATETIME,0,7,0,7,0,5
20C7 953 CVTIME DATE,0,3,0,3,0,1
20D2 954 CVTIME TIME,3,4,3,4,1,4
20DD 955 CVTIME DAY,2,1,0,1,0,1
20E7 956 CVTIME MONTH,1,1,1,1,0,0
20F3 957 CVTIME YEAR,0,1,2,1,0,0
20FE 958 CVTIME HOUR,3,1,3,1,1,1
2109 959 CVTIME MINUTE,4,1,4,1,2,1
2116 960 CVTIME SECOND,5,1,5,1,3,1
2123 961 CVTIME HUNDREDTH,6,1,6,1,4,1
2133 962 CVTIME WEEKDAY,0,0,0,0,0,0
00 2141 963 .BYTE 0
2142 964
2142 965 MONTH_TABLE:
2142 966 .ASCII 'JAN-FEB-MAR-APR-MAY-JUN-JUL-AUG-SEP-OCT-NOV-DEC-'
215A
2166
2172 967 MONTH_NO_TABLE:
2172 968 .ASCII '010203040506070809101112??'
217E
218A
218C 969
218C 970 :
218C 971 : BUILD TABLE OF WEEKDAYS
218C 972 :
218C 973 .MACRO DAY NAME
218C 974 .ASCIC 'NAME'
218C 975 .BLKB 10 - <%LENGTH(NAME)+1>
218C 976 .ENDM DAY
218C 977
218C 978 WEEKDAY_TABLE:
218C 979 DAY Monday
2196 980 DAY Tuesday
21A0 981 DAY Wednesday
21AA 982 DAY Thursday
21B4 983 DAY Friday
21BE 984 DAY Saturday
21C8 985 DAY Sunday
21D2 986
21D2 987 :
21D2 988 : MACRO TO DEFINE VALID F$IDENTIFIER KEYWORDS
21D2 989 :
00000000 21D2 990 $XXS = 0
21D2 991 .MACRO IDENT NAME
21D2 992 .ASCIC 'NAME'
21D2 993 .BYTE $XXS
21D2 994 $XXS = $XXS + 1
21D2 995 .ENDM
21D2 996
21D2 997 :
```

```
21D2 998 : DEFINE VALID F$IDENTIFIER KEYWORDS
21D2 999 :
21D2 1000 IDENT_TABLE:
21D2 1001 IDENT NAME TO NUMBER
21E2 1002 IDENT NUMBER_TO_NAME
00 21F2 1003 .BYTE 0
21F3 1004 :
21F3 1005 :
21F3 1006 : MACRO TO DEFINE VALID PROCESS MODES
21F3 1007 :
00000000 21F3 1008 $XXS = 0
21F3 1009 .MACRO PRCMODE NAME
21F3 1010 .ASCIC 'NAME'
21F3 1011 .BYTE $XXS
21F3 1012 $XXS = $XXS + 1
21F3 1013 .ENDM
21F3 1014 :
21F3 1015 :
21F3 1016 : DEFINE VALID PROCESS MODES (MUST BE IN JPISK_XXX ORDER)
21F3 1017 :
21F3 1018 MODE_TABLE:
21F3 1019 PRCMODE OTHER
21FA 1020 PRCMODE NETWORK
2203 1021 PRCMODE BATCH
220A 1022 PRCMODE INTERACTIVE
00 2217 1023 .BYTE 0
2218 1024 :
48 43 54 41 42 00' 2218 1025 BATCH:
05 2218 1026 .ASCIC 'BATCH' ;BATCH JOB
221E 1027 INTERACTIVE:
45 56 49 54 43 41 52 45 54 4E 49 00' 221E 1028 .ASCIC 'INTERACTIVE' ;INTERACTIVE JOB
08 221E 1029 NETWORK:
4B 52 4F 57 54 45 4E 00' 222A 1030 .ASCIC 'NETWORK' ;NETWORK JOB
07 222A 1031 OTHER:
52 45 48 54 4F 00' 2232 1032 .ASCIC 'OTHER' ;OTHER NON-INTERACTIVE JOB
05 2232 1033 :
2238 1034 : LIST MESSAGE FIELDS
2238 1035 :
2238 1036 :
54 58 45 54 00' 2238 1037 TEXT: .ASCIC 'TEXT'
04 2238 1038 IDENT: .ASCIC 'IDENTIFICATION'
54 41 43 49 46 49 54 4E 45 44 49 00' 223D 1039 SEVERITY:
4E 4F 49 2249 1040 .ASCIC 'SEVERITY'
0E 223D 1041 FACILITY:
59 54 49 52 45 56 45 53 00' 224C 1042 .ASCIC 'FACILITY'
08 224C 1043 :
59 54 49 4C 49 43 41 46 00' 2255 1044 :
08 2255 1045 NOALL: .ASCII 'NOALL'
225E 1044 :
4C 4C 41 4F 4E 225E 1045 NOALL: .ASCII 'NOALL' ;KEYWORD FOR [NO]ALL PRIVILEGES
```

45 53 4C 41 46 00'	2263	1046 FALSE: .ASCIC 'FALSE'	;BOOLEAN FALSE VALUE
05 2263			
45 55 52 54 00'	2269	1047 TRUE: .ASCIC 'TRUE'	;BOOLEAN TRUE VALUE
04 2269			
47 4E 49 52 54 53 00'	226E	1048	
06 226E		1049 STRING: .ASCIC 'STRING'	;STRING SYMBOL
52 45 47 45 54 4E 49 00'	2275	1050 INTEGER:	
07 2275		1051 .ASCIC 'INTEGER'	;INTEGER SYMBOL
47 4F 4C 5F 4C 43 44 24 4D 4E 4C 00'	227D	1052	
4C 41 43 49 2289		1053 TABDEF: .ASCIC 'LNMSDCL_LOGICAL'	;DEFAULT LOGICAL NAME TABLE
0F 227D			
3A 30 41 4C 4E 5F 00'	228D	1054 NLAO: .ASCIC '_NLAO:'	;NULL DEVICE
06 228D			

```
2294 1056 .SBTTL EVALUATE LEXICAL FUNCTION STRING
2294 1057
0000 1058 .PSECT DCL$ZCODE,BYTE,RD,NOWRT
0000 1059
0000 1060 :---
0000 1061 : DCL$FUNCTION - EVALUATE A LEXICAL FUNCTION STRING
0000 1062 :
0000 1063 : THIS ROUTINE CAN BE CALLED TO EVALUATE A STRING WHICH CONTAINS
0000 1064 : A LEXICAL FUNCTION AND ITS ARGUMENTS, AND RETURNS ITS VALUE.
0000 1065 :
0000 1066 : INPUTS:
0000 1067 :
0000 1068 : R1/R2 = DESCRIPTOR OF FUNCTION/SYMBOL STRING
0000 1069 : R3/R4 = DESCRIPTOR OF SCRATCH STORAGE
0000 1070 :
0000 1071 : OUTPUTS:
0000 1072 :
0000 1073 : R0 = STATUS
0000 1074 : R1/R2 = DESCRIPTOR OF VALUE STRING (NULL IF ERROR)
0000 1075 :---
0000 1076 : .IF DF SUPRESS_CODE ;NOBODY CALLS THIS AT THE MOMENT
0000 1077
0000 1078 DCL$FUNCTION::
0000 1079 PUSH R3,R4,R5,R6,R7 ;SAVE REGISTERS
0000 1080 PUSH L WRK_L_EXPANDPTR(R10) ;SAVE EXPANSION BUFFER POINTER
0000 1081 PUSH L WRK_L_CHARPTR(R10) ;SAVE INPUT BUFFER POINTER
0000 1082 STATUS BUFOVF ;ASSUME BUFFER OVERFLOW
0000 1083 CMPL R1,R3 ;CHECK IF STRING TOO BIG FOR BUFFER
0000 1084 BGEQ 90$ ;SKIP IT IF ITS TOO LONG
0000 1085 MOVL R4,WRK_L_EXPANDPTR(R10) ;PUT EXPANSION IN FRONT OF BUFFER
0000 1086 MOVAB -2(R4)[R3],R0 ;CALCULATE ENDING ADDRESS OF BUFFER
0000 1087 SUBL R1,R0 ;SUBTRACT LENGTH OF INPUT STRING
0000 1088 MOVL R0,WRK_L_CHARPTR(R10) ;SET INPUT CHARACTER POINTER TO STRING
0000 1089 MOVC R1,(R2),T(R0) ;STORE INPUT STRING ON STACK
0000 1090 CLRB (R3)+ ;TERMINATE INPUT STRING WITH EOL CHAR
0000 1091 BSBW DCL$GETOKEN ;GET FUNCTION NAME STRING
0000 1092 MOVQ R1,R6 ;AND PASS TO DCL$LEXIF
0000 1093 BSBW DCL$LEXIF ;EVALUATE LEXICAL FUNCTION
0000 1094 90$: POPL WRK_L_CHARPTR(R10) ;RESTORE INPUT BUFFER POINTER
0000 1095 POPL WRK_L_EXPANDPTR(R10) ;RESTORE EXPANSION BUFFER POINTER
0000 1096 POP R3,R4,R5,R6,R7 ;RESTORE REGISTERS
0000 1097 RSB
0000 1098
0000 1099 .ENDC
```

```
0000 1101 .SBTTL LEXICAL FUNCTION DISPATCHER
0000 1102 :+
0000 1103 : DCL$LEXIF - LEXICAL FUNCTION DISPATCH ROUTINE
0000 1104 :
0000 1105 : THIS ROUTINE IS CALLED WHEN A STRING SUBSTITUTION IS SPECIFIED AND THE STRING
0000 1106 : IS CURRENTLY UNDEFINED.
0000 1107 :
0000 1108 : INPUTS:
0000 1109 :
0000 1110 : R6 = LENGTH OF SYMBOL NAME.
0000 1111 : R7 = ADDRESS OF SYMBOL NAME.
0000 1112 : R10 = ADDRESS OF COMMAND WORK AREA
0000 1113 : R11 = ADDRESS OF PROCESS WORK AREA
0000 1114 :
0000 1115 : OUTPUTS:
0000 1116 :
0000 1117 : THE SYMBOL NAME IS MATCHED WITH THE LEXICAL FUNCTION TABLE. IF NO MATCH
0000 1118 : OCCURS, THEN A NULL STRING IS RETURNED FOR SUBSTITUTION IN THE ORIGINAL
0000 1119 : COMMAND LINE. IF A MATCH OCCURS, THEN THE APPROPRIATE LEXICAL FUNCTION
0000 1120 : IS EVALUATED AND THE RESULTANT STRING IS RETURNED.
0000 1121 :
0000 1122 : R1/R2 = A QUADWORD DESCRIBING THE FUNCTION RESULT
0000 1123 : IF R2 NONZERO, THEN R1/R2 IS A STRING DESCRIPTOR
0000 1124 : IF R2 ZERO, THEN R1 IS A BINARY LONGWORD VALUE.
0000 1125 :
0000 1126 : INPUT AND EXPANSION POINTERS ARE UPDATED AS APPROPRIATE.
0000 1127 :-
0000 1128 :
0000 1129 DCL$LEXIF::
0000 1130 PUSH R3,R4,R5,R6,R7,R8,R9,AP ; LEXICAL FUNCTION DISPATCH
0000 1131 MOV SP,AP ; SAVE REGISTERS
0000 1132 CMPL #6,R6 ; MARK STACK ADDRESS
0000 1133 BGTRU 10$ ; SPECIFIED NAME LONGER THAN 6 CHARACTERS?
0000 1134 MOV #6,R6 ; IF GTRU NO
0000 1135 CMPW #A/F$/,(R7)+ ; REDUCE LENGTH TO 6 CHARACTERS
0000 1136 BNEQ 60$ ; POSSIBLY LEXICAL FUNCTION NAME?
0000 1137 BSBW DCL$SETNBLK ; IF NEQ NO
0000 1138 CMPB #A/(/,R0 ; CHECK CHARACTER AFTER FUNCTION NAME
0000 1139 BNEQ 60$ ; POSSIBLY LEXICAL FUNCTION?
0000 1140 SUBL #2,R6 ; IF NEQ NO
0000 1141 BGTR 18$ ; REDUCE LENGTH OF FUNCTION NAME
0000 1142 CLRQ R0 ; IF LEQ NOT LEXICAL FUNCTION NAME
0000 1143 MOV SP,R2 ; FAILURE, NO SUCH FUNCTION, NULL VALUE
0000 1144 BRW 90$ ; INDICATE RESULT IS A STRING
0000 1145 STATUS IVFNAM ; EXIT
0000 1146 BRB 85$ ; SET INVALID FUNCTION NAME STATUS
0000 1147 STATUS ABFNAM ; SET AMBIGUOUS FUNCTION NAME STATUS
0000 1148 BICB #X80,(R7) ; UNDO MATCH OPTIMIZATION
0000 1149 MOVAW -(R7),WRK_L_MARKPTR(R10) ; SET STARTING ADDRESS OF FUNCTION NAME
0000 1150 BRW DCL$PARSERR
0000 1151 :
0000 1152 : SEARCH LEXICAL FUNCTION NAME TABLE FOR A MATCH
0000 1153 :
0000 1154 :
0000 1155 18$: BSBW DCL$MOVCHAR ; COPY ( CHARACTER
0000 1156 BISB #X80,(R7) ; SET FORCE MATCH FLAG
0000 1157 MATCHC R6,(R7),#LXFUNTABLE,LXFUNTABLE ; SEARCH FOR FUNCTION NAME MATCH
```

13F8 8F BB 0000 1130
5C 5E DO 0004 1131
56 06 D1 0007 1132
56 03 1A 000A 1133
87 56 06 DO 000C 1134
2446 8F B1 000F 1135 10\$:
0D 12 0014 1136
FFE7' 30 0016 1137
50 28 91 0019 1138
05 12 001C 1139
56 02 C2 001E 1140
24 14 0021 1141
50 7C 0023 1142 60\$:
52 5E DO 0025 1143
015F 31 0028 1144
002B 1145 70\$:
07 11 0032 1146
0034 1147 80\$:
67 80 8F 8A 003B 1148 85\$:
F48A CA 77 3E 003F 1149
160A 31 0044 1150
0047 1151
0047 1152
0047 1153
0047 1154
0047 1155 18\$:
67 80 8F 88 004A 1156
0100 8F 67 56 39 004E 1157

```
00000000'EF 0054
58 52 56 12 0059 1158 BNEQ 70$ ;BRANCH IF NO MATCH FOUND
58 52 08 C0 005B 1159 ADDL R6,R2 ;CALCULATE BYTES REMAINING INCLUDING SUBSTR1
63 0A 67 56 39 005E 1160 DIVL3 #ARG_C_ENTSIZE,R2,R8 ;CALCULATE COMPLETE TABLE ENTRY NUMBER
CB 13 0062 1161 MATCHC R6,(R7),#10,(R3) ;SCAN FOR DUPLICATE MATCH
0067 1162 BEQL 80$ ;IF EQL DUPLICATE MATCH FOUND
0069 1163 :
0069 1164 : DETERMINE THE FUNCTION INDEX (R8) AND RETRIEVE THE NUMBER OF ARGUMENTS
0069 1165 : THAT THE FUNCTION TAKES AS WELL AS THE ARGUMENT TYPES (NUMERIC OR STRING).
0069 1166 :
58 20 58 C3 0069 1167 SUBL3 R8,#LXFUNTABLE/ARG_C_ENTSIZE,R8 ;CALCULATE REAL FUNCTION INDEX
006D 1168 ASSUME ARG_C_ENTSIZE EQ 8
57 00000004'EF48 7E 006D 1169 MOVAQ LXFUNTABLE+4(R8),R7 ;POINT TO ARGUMENT INFORMATION LONGWORD
57 67 D0 0075 1170 MOVL (R7),R7 ;SAVE IT IN REGISTER
0078 1171 :
0078 1172 : PARSE EACH FUNCTION ARGUMENT, STORING EITHER THE STRING DESCRIPTOR OR
0078 1173 : THE NUMERIC VALUE IN THE ARGUMENT DESCRIPTORS. STRINGS ARE SAVED ON
0078 1174 : THE STACK IN ORDER TO MAKE FUNCTIONS RECURSIVE.
0078 1175 :
0080 8F 5E FF7C CE 9E 0078 1176 MOVAB -<MAX_ARGS*8+4>(SP),SP ;ALLOCATE ARGUMENT DESCRIPTORS + COUNT
FF 8F 6E 00 2C 007D 1177 MOVCS #0,(SP),#*XFF,#MAX_ARGS*8,4(SP) ;INIT ALL ARGS TO MISSING (-1)
04 AE 0085 :
5C 04 AE 9E 0087 1178 PUSHL AP ;SAVE STARTING STACK ADDRESS IN -4(AP)
55 5C D0 0089 1179 MOVAB 4(SP),AP ;POINT TO START OF ARGUMENT LIST
85 D4 008D 1180 MOVL AP,R5 ;ADDRESS OF NEXT DESCRIPTOR TO FILL
0090 1181 CLRL (R5)+ ;INITIALIZE # ARGUMENTS SPECIFIED
0092 1182 :
FF68' 30 0092 1183 20$: BSBW DCL$SETNBLK ;SKIP LEADING BLANKS AND TABS
FF68' 30 0095 1184 BSBW DCL$MARK ;MARK CURRENT PARSE POSITION
57 6C 91 0098 1185 CMPB NPARM(AP),R7 ;ANY MORE ARGUMENTS TO PROCESS?
03 19 009B 1186 BLSS 21$ ;IF GEQ YES
0085 31 009D 1187 BRW 50$ ;END OF ARGUMENTS
56 03 D0 00A0 1188 21$: MOVL #ARG_C_ANY,R6 ;ASSUME EITHER BINARY OR STRING
06 6C 91 00A3 1189 CMPB NPARM(AP),#6 ;BEYOND EXPLICITLY DEFINED ARGS?
0C 18 00A6 1190 BGEQ 23$ ;IF SO, ASSUME 'ANY'
56 6C 02 78 00A8 1191 ASHL #2,NPARM(AP),R6 ;CALCULATE OFFSET TO NIBBLE
56 56 08 C0 00AC 1192 ADDL #8,R6 ;BIT POSITION OF NIBBLE IN R7
56 57 08 56 EF 00AF 1193 EXTZV R6,#8,R7,R6 ;EXTRACT NIBBLE DESCRIBING ARGUMENT
0084 1194 ;AND NIBBLE DESCRIBING NEXT ARGUMENT
0084 1195 ;AS WELL, FOR END-OF-LIST CHECKS
0084 1196 :
0084 1197 : CHECK IF ARGUMENT MISSING
0084 1198 :
2C 50 91 0084 1199 23$: CMPB R0,#^A', ' ;CHECK IF ARGUMENT NULL
44 13 0087 1200 BEQL 28$ ;BRANCH IF NULL
29 50 91 0089 1201 CMPB R0,#^A')' ;CHECK IF ARGUMENT NULL
3F 13 008C 1202 BEQL 28$ ;BRANCH IF NULL
008E 1203 :
008E 1204 : PARSE ARGUMENT EXPRESSION
008E 1205 :
02 56 02 00 ED 008E 1206 CMPZV #ARG_V_TYPE,#ARG_S_TYPE,R6,#ARG_C_SYMREF ;REFERENCE TO SYMBOL?
42 13 00C3 1207 BEQL 30$ ;BRANCH IF SO
FF38' 30 00C5 1208 BSBW DCL$EXPRESS ;GET VALUE OF EXPRESSION
26 50 E9 00C8 1209 BLBC R0,22$ ;BRANCH IF ERROR DETECTED
00CB 1210 :
00CB 1211 : VERIFY THAT THE EXPRESSION ARGUMENT WAS THE EXPECTED TYPE (STRING OR BINARY)
00CB 1212 : IF IT WASN'T, THEN CONVERT THE EXPRESSION TO THE PROPER MODE, IF POSSIBLE.
```

```
03 56 02 00 ED 00CB 1213 :
3D 13 00CB 1214 : CMPZV #ARG_V_TYPE,#ARG_S_TYPE,R6,#ARG_C_ANY ;ANY TYPE ALLOWED?
52 D5 00D0 1215 : BEQL 40$ ;BRANCH-IF SO
00D4 1216 : TSTL R2 ;CHECK IF STRING OR BINARY LONGWORD
00D4 1217 : ASSUME ARG_C_BINARY EQ 0
00D4 1218 : ASSUME ARG_C_STRING EQ 1
0D 56 E8 00D4 1219 : BLBS R6,27$ ;BRANCH IF EXPECTED STRING ARGUMENT
36 13 00D7 1220 : BEQL 40$ ;OK IF EXPECTED NUMERIC AND GOT IT
00000000'EF 16 00D9 1221 : JSB DCL$CVT_BINARY ;CONVERT STRING TO BINARY
OF 50 E9 00DF 1222 : BLBC R0,22$ ;BRANCH IF ERROR DETECTED
2B 11 00E2 1223 : BRB 40$
29 12 00E4 1224 27$: BNEQ 40$ ;OK IF EXPECTED STRING AND GOT IT
50 51 D0 00E6 1225 : MOVL R1,R0 ;SET BINARY VALUE TO CONVERT
00000000'EF 16 00E9 1226 : JSB DCL$CBTA_DEC ;CONVERT BINARY VALUE TO DECIMAL STRING
1E 11 00EF 1227 : BRB 40$ ;USE CONVERTED STRING AS ARGUMENT
00F1 1228 :
00F1 1229 : AN ERROR HAS OCCURRED. IF WE ARE IN A COMMENT, JUST STORE
00F1 1230 : AN ARGUMENT OF NULL OR ZERO AND CONTINUE.
00F1 1231 :
03 FO AA 0C E0 00F1 1232 22$: BBS #WRK_V_COMMENT,WRK_W_FLAGS(R10),24$ ;SKIP IF IN COMMENT
1558 31 00F6 1233 : BRW DCL$PARSERR ;OTHERWISE, REPORT THE ERROR
51 7C 00F9 1234 24$: CLRQ R1 ;IF IN COMMENT, CONTINUE WITH A ZERO
12 11 00FB 1235 : BRB 40$
00FD 1236 :
00FD 1237 : PROCESS AN EMPTY ARGUMENT. IF ALLOWED, -1 WILL APPEAR IN THE QUADWORD
00FD 1238 : TO INDICATE MISSING ARGUMENT.
00FD 1239 :
31 56 02 E0 00FD 1240 28$: BBS #ARG_V_OPTIONAL,R6,43$ ;BRANCH IF OPTIONAL VALUES ALLOWED
FEFC' 30 0101 1241 : BSBW DCL$MOVCHAR ;MOVE DELIMITER TO EXPANSION BUFFER
0088 31 0104 1242 29$: BRW 95$ ;AND REPORT MISSING ARGUMENT
0107 1243 :
0107 1244 : PARSE ARGUMENT TO BE STORED BY REFERENCE RATHER THAN BY VALUE
0107 1245 :
22 FEF6' 30 0107 1246 30$: BSBW DCL$GETOKEN ;GET ARGUMENT SYMBOL OR LITERAL
62 91 010A 1247 : CMPB (R2),#^A' ;QUOTED STRING?
F5 13 010D 1248 : BEQL 29$ ;IF SO, ERROR - NOT BY REFERENCE
010F 1249 :
010F 1250 : STORE STRING ARGUMENT ON STACK TO PRESERVE IT OVER RECURSION
010F 1251 :
52 D5 010F 1252 40$: TSTL R2 ;STRING ARGUMENT?
1C 13 0111 1253 : BEQL 42$ ;IF NOT, JUST STORE LONGWORD
03 56 03 E1 0113 1254 : BBC #ARG_V_KEYWORD,R6,41$ ;KEYWORD ARGUMENT?
FEE6' 30 0117 1255 : BSBW DCL$SQOEEZE ;EDIT THE KEYWORD LIST
5E 51 C2 011A 1256 41$: SUBL R1,SP ;ALLOCATE SPACE FOR STRING ON STACK
85 51 D0 011D 1257 : MOVL R1,(R5)+ ;SAVE STRING LENGTH
85 5E D0 0120 1258 : MOVL SP,(R5)+ ;SAVE (COPIED) STRING ADDRESS
55 DD 0123 1259 : PUSHL R5 ;SAVE REGISTERS
04 AE 62 51 28 0125 1260 : MOVC R1,(R2),4(SP) ;SAVE ARGUMENT STRING ON STACK
55 8ED0 012A 1261 : POPL R5 ;RESTORE REGISTERS
06 11 012D 1262 : BRB 45$
012F 1263 :
65 51 7D 012F 1264 42$: MOVQ R1,(R5) ;STORE VALUE IN QUADWORD
55 08 C0 0132 1265 43$: ADDL #8,R5 ;SKIP TO NEXT ARGUMENT DESCRIPTOR
6C D6 0135 1266 45$: INCL NPARM(AP) ;INCREMENT # ARGUMENTS FOUND
FEC6' 30 0137 1267 : BSBW DCL$SETNBLK ;SKIP TRAILING SPACES
57 6C 91 013A 1268 : CMPB NPARM(AP),R7 ;LAST PARAMETER TO PROCESS?
16 13 013D 1269 : BEQL 50$ ;IF EQL YES
```

```

50  FEBE' 30 013F 1270      BSBW  DCL$MOVCHAR      ;GET NEXT INPUT CHARACTER
    2C  91 0142 1271      CMPB  #'A/,/,R0          ;TRAILING COMMA?
    03  12 0145 1272      BNEQ  48$              ;BRANCH IF NOT
    FF48 31 0147 1273      BRW   20$              ;LOOP THRU ALL ARGUMENTS
06  57  91 014A 1274 48$:  CMPB  R7,#6            ;VARIABLE # ARGUMENTS?
    09  1A 014D 1275      BGTRU 55$              ;BRANCH IF SO
05 56  06  E0 014F 1276      BBS   #4+ARG_V_OPTIONAL,R6,55$ ;OK IF NEXT ARGUMENT IS OPTIONAL
    3A  11 0153 1277      BRB   95$              ;ERROR IF REQUIRED ARGUMENT NOT PRESENT
    0155 1278
    FEAB' 30 0155 1279 50$: BSBW  DCL$MOVCHAR      ;GET NEXT INPUT CHARACTER
50  29  91 0158 1280 55$:  CMPB  #'A/)/,R0          ;TRAILING RIGHT PARENTHESIS?
    3B  12 015B 1281      BNEQ  100$             ;IF NEQ NO
    FEA0' 30 015D 1282      BSBW  DCL$MARK          ;MARK CURRENT PARSE POSITION
    0160 1283
    0160 1284 : CONSTRUCT A DESCRIPTOR OF SCRATCH BUFFER SPACE AND CALL THE APPROPRIATE
    0160 1285 : LEXICAL FUNCTION ROUTINE TO OBTAIN THE RESULT STRING.
    0160 1286 :
52  53  F486 CA D0 0160 1287      MOVL  WRK_L_EXPANDPTR(R10),R3 ;BUILD DESCRIPTOR OF SCRATCH AREA
    F48E CA  F486 CA C3 0165 1288      SUBL3 WRK_L_EXPANDPTR(R10),WRK_L_CHARPTR(R10),R2 ;SET UNUSED LENGTH
    4C  10 016D 1289      BSBB  DISPATCH          ;DISPATCH FUNCTION
    016F 1290
    016F 1291 : R1/R2 NOW DESCRIBE THE FUNCTION RESULT:
    016F 1292 : IF R2 NONZERO, R1/R2 ARE A STRING DESCRIPTOR
    016F 1293 : IF R2 ZERO, R1 IS A BINARY LONGWORD VALUE
    016F 1294 :
    016F 1295 : RESULTANT STRING MIGHT NOW RESIDE ON THE STACK (E.G. AN EXTRACT
    016F 1296 : OF AN ARGUMENT STORED ON THE STACK). WE MUST COPY IT TO THE EXPANSION
    016F 1297 : BUFFER BEFORE RETURNING IT TO THE CALLER.
    016F 1298 :
    52  D5 016F 1299      TSTL  R2                ;CHARACTER STRING?
    11  13 0171 1300      BEQL  58$              ;BRANCH IF NOT
    F486 DA 56 51 D0 0173 1301      MOVL  R1,R6            ;SAVE LENGTH OF STRING
    62  51 28 0176 1302      MOVC  R1,(R2),@WRK_L_EXPANDPTR(R10) ;COPY IT TO UNUSED EXPANSION AREA
    51  56 D0 017C 1303      MOVL  R6,R1            ;RETURN LENGTH OF RESULT STRING
52  F486 CA D0 017F 1304      MOVL  WRK_L_EXPANDPTR(R10),R2 ;AND ADDRESS OF RESULT STRING
    50  01 D0 0184 1305 58$: MOVL  #1,R0            ;SUCCESS
    5E  7C D0 0187 1306      MOVL  -(AP),SP          ;REMOVE ALL TEMPORARIES FROM STACK
    13F8 8F BA 018A 1307 90$: POPR  #^M<R3,R4,R5,R6,R7,R8,R9,AP> ;RESTORE REGISTERS
    05 018E 1308      RSB
    018F 1309
    018F 1310 :
    018F 1311 : INVALID SYMBOL DELIMITER
    018F 1312 :
    50  00038268 8F D0 018F 1313 95$: MOVL  #CLIS_ARGREQ,R0      ;SET ARGUMENT REQUIRED STATUS
    07  11 0196 1314      BRB   105$              ;
    0198 1315 100$:      STATUS SYMDEL              ;SET INVALID SYMBOL DELIMITER STATUS
    F486 CA  F48E CA D0 019F 1316 105$: MOVL  WRK_L_CHARPTR(R10),WRK_L_EXPANDPTR(R10) ;SET ADDRESS OF NEXT CHARACT
    F48A CA  F486 CA 01 C3 01A6 1317      SUBL3 #1,WRK_L_EXPANDPTR(R10),WRK_L_MARKPTR(R10) ;SET ADDRESS OF INVALID D
    F48E DA 95 01AE 1318      TSTB  @WRK_L_CHARPTR(R10)        ;AT END OF LINE?
    04  12 01B2 1319      BNEQ  110$             ;IF NEQ NO
    F48E CA D7 01B4 1320      DECL  WRK_L_CHARPTR(R10)          ;BACK UP TO END OF LINE
    1496 31 01B8 1321 110$: BRW   DCL$PARSERR      ;

```

```
0188 1323 :  
0188 1324 : DISPATCH TO PROPER LEXICAL FUNCTION ROUTINE  
0188 1325 :  
0188 1326 : ON INPUT TO ROUTINE:  
0188 1327 :  
0188 1328 :     R2/R3 = DESCRIPTOR OF SCRATCH AREA  
0188 1329 :     R10 = ADDRESS OF PARSING WORK AREA (WRK)  
0188 1330 :     R11 = ADDRESS OF PROCESS WORK AREA (PRC)  
0188 1331 :  
0188 1332 :     P1(AP) = DESCRIPTOR OF FIRST ARGUMENT VALUE  
0188 1333 :     P2(AP) = DESCRIPTOR OF SECOND ARGUMENT VALUE  
0188 1334 :     P3(AP) = DESCRIPTOR OF THIRD ARGUMENT VALUE  
0188 1335 :     ETC. (UP TO MAX_ARGS)  
0188 1336 :  
0188 1337 : ON OUTPUT FROM ROUTINE:  
0188 1338 :  
0188 1339 :     R1/R2 = A QUADWORD DESCRIBING THE FUNCTION RESULT  
0188 1340 :     IF R2 NONZERO, THEN R1/R2 IS A STRING DESCRIPTOR  
0188 1341 :     IF R2 ZERO, THEN R1 IS A BINARY LONGWORD VALUE.  
0188 1342 :  
0188 1343 :     R3-R9 DESTROYED.  
0188 1344 :  
0188 1345 :  
0188 1346 : DISPATCH:  
0188 1347 : CASE  
0188 1348 :     R8,<-  
0188 1349 :     F$CVSINTEGER,- :CONVERT SIGNED INTEGER  
0188 1350 :     F$CVUIINTEGER,- :CONVERT UNSIGNED INTEGER  
0188 1351 :     F$CVTIME,- :CONVERT TO COMPARABLE DATE/TIME  
0188 1352 :     F$DIRECTORY,- :DEFAULT DIRECTORY  
0188 1353 :     F$EDIT,- :EDIT A STRING  
0188 1354 :     F$ELEMENT,- :FIND A STRING ELEMENT  
0188 1355 :     F$ENVIRONMENT,- :GET DCL INFORMATION  
0188 1356 :     F$EXTRACT,- :EXTRACT SUBSTRING FROM STRING  
0188 1357 :     F$FAO,- :FORMATTED ASCII OUTPUT  
0188 1358 :     F$FILE,- :GET FILE ATTRIBUTES  
0188 1359 :     F$GETDVI,- :GETDVI SYSTEM SERVICE  
0188 1360 :     F$GETJPI,- :GETJPI SYSTEM SERVICE  
0188 1361 :     F$GETSYI,- :GETSYI SYSTEM SERVICE  
0188 1362 :     F$IDENTIFIER,- :CONVERT IDENTIFIER  
0188 1363 :     F$INTEGER,- :CONVERT TO INTEGER  
0188 1364 :     F$LENGTH,- :STRING LENGTH  
0188 1365 :     F$LOCATE,- :LOCATE SUBSTRING  
0188 1366 :     F$LOGICAL,- :LOGICAL NAME TRANSLATION  
0188 1367 :     F$MESSAGE,- :GET MESSAGE  
0188 1368 :     F$MODE,- :JOB MODE  
0188 1369 :     F$PARSE,- :PARSE FILE SPECIFICATION  
0188 1370 :     F$PID,- :GET NEXT PROCESS ID  
0188 1371 :     F$PRIVILEGE,- :TRUE IF PRIVS SET/CLEAR  
0188 1372 :     F$PROCESS,- :PROCESS NAME  
0188 1373 :     F$SEARCH,- :SEARCH FOR NEXT WILDCARD FILESPEC  
0188 1374 :     F$SETPRV,- :SETPRV SYSTEM SERVICE  
0188 1375 :     F$STRING,- :CONVERT TO ASCII STRING  
0188 1376 :     F$TIME,- :CURRENT TIME AND DATE  
0188 1377 :     F$TRNLNM,- :LOGICAL NAME TRANSLATION  
0188 1378 :     F$TYPE,- :TYPE OF SYMBOL  
0188 1379 :     F$USER,- :USER IDENTIFICATION  
0188 1380 :     F$VERIFY,- :VERIFY MODE
```

LEXICON
V04-000

- LEXICAL FUNCTION PARSE AND EXECUTION R 16-SEP-1984 00:00:36 VAX/VMS Macro V04-00
LEXICAL FUNCTION DISPATCHER 4-SEP-1984 23:41:41 [DCL.SRC]LEXICON.MAR;1

Page 30
(8)

51	D4	01BB	1380		>	:
	05	01FF	1381	CLRL	R1	:RETURN NULL STRING
		0201	1382	RSB		:

```
0202 1384 .SBTTL CONVERT SIGNED AND UNSIGNED INTEGER FIELD
0202 1385 :+
0202 1386 : F$CVSINTEGER - CONVERT SIGNED INTEGER
0202 1387 : F$CVUINTEGER - CONVERT UNSIGNED INTEGER
0202 1388 :
0202 1389 : THIS LEXICAL FUNCTION RETURNS THE BINARY DATA EXTRACTED FROM AN ARBITRARY
0202 1390 : FIELD IN A STRING.
0202 1391 :
0202 1392 : F$CVSINTEGER(POSITION, SIZE, STRING)
0202 1393 : F$CVUINTEGER(POSITION, SIZE, STRING)
0202 1394 :
0202 1395 : INPUTS:
0202 1396 :
0202 1397 : P1(AP) = STARTING BIT NUMBER
0202 1398 : P2(AP) = NUMBER OF BITS
0202 1399 : P3(AP) = DESCRIPTOR OF TEXT STRING.
0202 1400 :
0202 1401 : OUTPUTS:
0202 1402 :
0202 1403 : R1 = CONVERTED BINARY LONGWORD
0202 1404 : R2 = 0 TO INDICATE BINARY DATA TYPE
0202 1405 :-
0202 1406 :
0202 1407 : .ENABL LSB
0202 1408 F$CVSINTEGER: :CONVERT SIGNED INTEGER
0202 1409 : MOVL #1,R5 :SET SIGNED CONVERSION INDICATOR
0205 1410 : BRB 10$
0207 1411 F$CVUINTEGER: :CONVERT UNSIGNED INTEGER
0207 1412 : CLRL R5 :SET UNSIGNED CONVERSION INDICATOR
0209 1413 10$: :CMPL #32,P2(AP) :SIZE VALUE GREATER THAN 1 LONGWORD?
020D 1414 : BLSSU 90$ :IF GEQU NO
020F 1415 : CLRL R1 :ASSUME FIELD IS NOT DEFINED
0211 1416 : ADDL3 P2(AP),P1(AP),R3 :CALCULATE ENDING BIT POSITION
0217 1417 : MULL3 #8,P3(AP),R2 :CALCULATE SIZE OF STRING IN BITS
021C 1418 : CMPL P1(AP),R2 :FIELD POSITION BEYOND END OF STRING?
0220 1419 : BGEQU 90$ :IF GEQU YES
0222 1420 : SUBL R3,R2 :FIELD POSITION TOTALLY WITHIN STRING?
0225 1421 : BLSS 90$ :IF GEQ YES
0227 1422 : BLBS R5,50$ :IF LBS SIGNED CONVERSION
022A 1423 : EXTZV P1(AP),P2(AP),@P3+4(AP),R1 :EXTRACT FIELD
0232 1424 : BRB 60$
0234 1425 50$: :EXTV P1(AP),P2(AP),@P3+4(AP),R1 :EXTRACT SIGNED FIELD
023C 1426 60$: :CLRL R2 :MARK RESULT IS BINARY DATA
023E 1427 : RSB
023F 1428 :
023F 1429 90$: :MOVL #CLIS_INVRANGE,R0 :SET INVALID RANGE MESSAGE
0246 1430 : BRW ERROR :SIGNAL ERROR
0249 1431 : .DSABL LSB
```

55 01 DO 02 11 55 D4 0207 1411 10\$ 0209 1413 020D 1414 020F 1415 0211 1416 0217 1417 021C 1418 0220 1419 0222 1420 0225 1421 0227 1422 022A 1423 0232 1424 0234 1425 023C 1426 023E 1427 023F 1428 023F 1429 0246 1430 0249 1431

53 04 AC 0C AC 51 18 BC 0C AC 04 AC 52 14 AC 04 AC 52 0A 55 E8 51 18 BC 0C AC 04 AC 52 05 50 00038228 8F 13CA 31

```
0249 1433 .SBTTL CONVERT TO COMPARABLE DATE/TIME STRING
0249 1434 :+
0249 1435 : F$CVTIME - CONVERT TO COMPARABLE DATE/TIME STRING
0249 1436 :
0249 1437 : THIS FUNCTION CONVERTS A STANDARD VMS DATE/TIME STRING INTO A
0249 1438 : STRING OF THE FORM 'YYYY-MM-DD HH:MM:SS.NN' WHICH CAN BE COMPARED
0249 1439 : TO ANOTHER STRING OF THE SAME FORM USING THE STRING COMPARISON
0249 1440 : OPERATORS. IT CAN ALSO BE USED TO GENERATE A STANDARDIZED DATE/TIME
0249 1441 : STRING OR TO EXTRACT ITEMS FROM A STARDIZED DATE/TIME STRING.
0249 1442 :
0249 1443 : F$CVTIME(TIME, ITEM)
0249 1444 :
0249 1445 : INPUTS:
0249 1446 :
0249 1447 : P1(AP) = DESCRIPTOR OF DATE/TIME STRING
0249 1448 : P2(AP) = DESCRIPTOR OF TIME TYPE
0249 1449 : P3(AP) = DESCRIPTOR OF ITEM KEYWORD
0249 1450 :
0249 1451 : OUTPUTS:
0249 1452 :
0249 1453 : R1/R2 = DESCRIPTOR OF REQUESTED DATE/TIME STRING
0249 1454 :
0249 1455 : TIME_DELIMS:
20 2E 2D 3A 0249 1456 : .ASCII /:-. /
024D 1457 :
024D 1458 F$CVTIME:
024D 1459 :
024D 1460 : Build the result buffer descriptor.
024D 1461 :
17E 53 18 C1 024D 1462 : ADDL3 #24,R3,-(SP) :BUILD TIME BUFFER DESCRIPTOR
59 53 D0 0251 1463 : MOVL R3,R9 :SAVE ADDR OF SECOND DESCR
7E 52 D0 0254 1464 : MOVL R2,-(SP) :
6E 2F D1 0257 1465 : CMPL #24+23,(SP) :TIME BUFFER LARGE ENOUGH?
06 1B 025A 1466 : BLEQU 10$ :IF LEQU YES
5E 08 C0 025C 1467 : ADDL #2*4,SP :RESTORE THE STACK
13AA 31 025F 1468 : BRW BUFOVF :REPORT BUFFER OVERFLOW
0262 1469 :
0262 1470 :
0262 1471 : Get the type information.
0262 1472 :
58 D4 0262 1473 10$: CLRL R8 :ASSUME NO TYPE SPECIFIED
OC AC D5 0264 1474 : TSTL P2(AP) :WAS A TYPE SPECIFIED?
30 19 0267 1475 : BLSS 20$ :NO, THEN USE DEFAULT
54 0000209A'EF 9E 0269 1476 : MOVAB CVTIME TYPES,R4 :GET ADDRESS OF ITEM TABLE
55 64 9A 0270 1477 15$: MOVZBL (R4),R5 :GET LENGTH OF ITEM
12 13 0273 1478 : BEQL 16$ :BRANCH IF END OF TABLE
55 00 10 BC OC AC 2D 0275 1479 : CMPC5 P2(AP),@P2+4(AP),#0,R5,1(R4) :DOES ITEM MATCH
01 A4 027C :
14 13 027E 1480 : BEQL 17$ :BRANCH IF SO
54 02 A445 9E 0280 1481 : MOVAB 2(R4)[R5],R4 :SKIP TO NEXT TABLE ENTRY
E9 11 0285 1482 : BRB 15$ :
5E 08 C0 0287 1483 16$: ADDL #2*4,SP :RESTORE THE STACK
028A 1484 : STATUS IVKEYW :SET STATUS
58 01 138A 31 0291 1485 : BRW P2 ERROR :REPORT KEYWORD ERROR
01 A445 9A 0294 1486 17$: MOVZBL 1(R4)[R5],R8 :SET THE TYPE
0299 1487 :
0299 1488 :
```

```
0299 1489 : Call LIB$CVT_TIME or LIB$CVT_DTIME to get the time string.
0299 1490 :
55 5E D0 0299 1491 20$: MOVL SP,R5 ;GET ADDRESS OF BUFFER DESCR
7E 7C 029C 1492 CLRQ -(SP) ;ALLOC TIME QUADWORD
53 D4 029E 1493 CLRL R3 ;SET ADDRESS OF INPUT DESCR
04 AC D5 02A0 1494 TSTL P1(AP) ;WAS P1 SPECIFIED?
OD 19 02A3 1495 BLSS 23$ ;NO, THEN SKIP
51 04 AC 7D 02A5 1496 MOVQ P1(AP),R1 ;GET INPUT DESCRIPTOR
FD 54 30 02A9 1497 BSBW DCL$TRIM ;UPCASE AND TRIM IT
04 AC 51 7D 02AC 1498 MOVQ R1,P1(AP) ;RETURN THE DESCRIPTOR
03 11 02B0 1499 BRB 25$ ;
04 AC 7C 02B2 1500 23$: CLRQ P1(AP) ;USE A NULL INPUT DESCRIPTOR
53 04 AC 7E 02B5 1501 25$: MOVQ P1(AP),R3 ;GET INPUT DESCRIPTOR ADDR
55 DD 02B9 1502 PUSHL R5 ;PUSH ADDRESS OF DESCRIPTOR TWICE
55 DD 02BB 1503 PUSHL R5 ;
F8 A5 7F 02BD 1504 PUSHAQ -8(R5) ;PUSH TIME BUFFER ADDR
53 DD 02C0 1505 PUSHL R3 ;PUSH INPUT DESCRIPTOR
02 58 D1 02C2 1506 CML R8,#2 ;IS DELTA TIME REQUESTED?
17 13 02C5 1507 BEQL 27$ ;YES, THEN BRANCH
02C7 1508
00000000'EF 04 FB 02C7 1509 CALLS #4,LIB$CVT_TIME ;PARSE THE TIME SPECIFICATION
24 50 E8 02CE 1510 BLBS R0,30$ ;BRANCH IF SUCCESS
5E 10 C0 02D1 1511 ADDL #16,SP ;RESTORE THE STACK
02D4 1512 STATUS IVATIME ;SET ERROR STATUS
133A 31 02DB 1513 BRW P1_ERROR ;REPORT TIME SYNTAX ERROR
02DE 1514
00000000'EF 04 FB 02DE 1515 27$: CALLS #4,LIB$CVT_DTIME ;PARSE THE TIME SPECIFICATION
OD 50 E8 02E5 1516 BLBS R0,30$ ;BRANCH IF SUCCESS
5E 10 C0 02E8 1517 ADDL #16,SP ;RESTORE THE STACK
02EB 1518 STATUS IVDTIME ;SET ERROR STATUS
1323 31 02F2 1519 BRW P1_ERROR ;REPORT TIME SYNTAX ERROR
02F5 1520
02F5 1521 :
02F5 1522 : Get the item information.
02F5 1523 :
54 000020B8'EF 9E 02F5 1524 30$: MOVAB CVTIME_ITEMS,R4 ;GET ADDRESS OF ITEM TABLE
55 64 9A 02FC 1525 MOVZBL (R4),R5 ;GET LENGTH OF ITEM
54 01 A445 9E 02FF 1526 MOVAB 1(R4)[R5],R4 ;GET ADDRESS OF DATA
7E 6448 3C 0304 1527 MOVZWL (R4)[R8],-(SP) ;GET DEFAULT DATA
14 AC D5 0308 1528 TSTL P3(AP) ;WAS AN ITEM SPECIFIED?
3E 19 030B 1529 BLSS 40$ ;NO, THEN USE DEFAULT
54 000020B8'EF 9E 030D 1530 MOVAB CVTIME_ITEMS,R4 ;GET ADDRESS OF ITEM TABLE
55 64 9A 0314 1531 35$: MOVZBL (R4),R5 ;GET LENGTH OF ITEM
12 13 0317 1532 BEQL 36$ ;BRANCH IF END OF TABLE
55 00 18 BC 14 AC 2D 0319 1533 CMPC5 P3(AP),@P3+4(AP),#0,R5,1(R4) ;DOES ITEM MATCH
01 A4 0320 1534 BEQL 37$ ;BRANCH IF SO
14 13 0322 1535 MOVAB 7(R4)[R5],R4 ;SKIP TO NEXT TABLE ENTRY
54 07 A445 9E 0324 1536 BRB 35$ ;
E9 11 0329 1537 36$: ADDL #20,SP ;RESTORE THE STACK
5E 14 C0 032B 1538 STATUS IVKEYW ;SET STATUS
032E 1539 BRW P3_ERROR ;REPORT KEYWORD ERROR
12EC 31 0335 1540 37$: MOVAB 1(R4)[R5],R4 ;GET ADDRESS OF DATA
54 01 A445 9E 0338 1541 MOVZWL (R4)[R8],(SP) ;SET THE POSITION AND LENGTH
6E 6448 3C 033D 1542 BNEQ 40$ ;IF NONZERO, THEN CONTINUE
08 12 0341 1543 CML R8,#2 ;IS DELTA TIME REQUESTED?
02 58 D1 0343 1544 BEQL 36$ ;YES, THEN ERROR
E3 13 0346 1544
```

```
0081 31 0348 1545 BRW 90$ ;NO, THEN RETURN WEEKDAY
      0348 1546
      0348 1547 ;
      0348 1548 ; Return requested information.
      0348 1549 ;
56 0C AE 7D 0348 1550 40$: MOVQ 12(SP),R6 ;GET TIME STRING DESCRIPTOR
      58 D5 034F 1551 TSTL R8 ;WAS 'COMPARISON' SPECIFIED?
      42 12 0351 1552 BNEQ 60$ ;NO, THEN BRANCH
      0353 1553
      0353 1554 ;
      0353 1555 ; Convert to ASCII comparable time.
      0353 1556 ;
      56 17 D1 0353 1557 CMPL #23,R6 ;IS DAY ONE DIGIT OR TWO?
      05 13 0356 1558 BEQL 50$ ;BRANCH IF TWO
      77 30 90 0358 1559 MOVB #^A'0',-(R7) ;INSERT A ZERO
      56 D6 035B 1560 INCL R6 ;INCREASE TIME LENGTH
54 00002142'EF 9E 035D 1561 50$: MOVAB MONTH_TABLE,R4 ;POINT TO MONTH TABLE
      50 D4 0364 1562 CLRL R0 ;START WITH MONTH #0
      84 03 A7 D1 0366 1563 51$: CMPL 3(R7),(R4)+ ;DOES MONTH MATCH?
      04 13 036A 1564 BEQL 52$ ;BRANCH IF FOUND
      F6 50 0C F2 036C 1565 AOBLS #12,R0,51$ ;LOOP UNTIL END OF TABLE
      53 59 D0 0370 1566 52$: MOVL R9,R3 ;POINT TO SECOND DATE/TIME BUFFER
      83 07 A7 D0 0373 1567 MOVL 7(R7),(R3)+ ;MOVE YEAR NUMBER
      83 2D 90 0377 1568 MOVAB #^A'-',(R3)+ ;MOVE DASH
83 00002172'EF 40 B0 037A 1569 MOVW MONTH_NO_TABLE[R0],(R3)+ ;MOVE MONTH NUMBER
      83 2D 90 0382 1570 MOVAB #^A'-',(R3)+ ;MOVE DASH
      83 67 B0 0385 1571 MOVW (R7),(R3)+ ;MOVE DAY NUMBER
      83 0B A7 7D 0388 1572 MOVQ 11(R7),(R3)+ ;MOVE 12 CHARACTERS OF TIME STRING
      83 13 A7 D0 038C 1573 MOVL 19(R7),(R3)+
      57 59 D0 0390 1574 MOVL R9,R7
      56 D7 0393 1575 DECL R6 ;RESET TIME POINTER
      0395 1576 ;DECREASE TIME LENGTH
      0395 1577 ;
      0395 1578 ; Extract requested field.
      0395 1579 ;
55 6E 08 08 EF 0395 1580 60$: EXTZV #8,#8,(SP),R5 ;GET NUMBER OF DELIMS
      54 6E 9A 039A 1581 MOVZBL (SP),R4 ;GET STARTING DELIM #
      0D 13 039D 1582 BEQL 67$ ;BRANCH IF ZERO
      56 D7 039F 1583 65$: DECL R6 ;DECR STRING SIZE
FEA2 CF 04 87 3A 03A1 1584 LOCC (R7)+,#4,TIME_DELIMS ;IS CHAR A DELIM?
      F6 13 03A7 1585 BEQL 65$ ;NO, THEN SKIP IT
      F3 54 F5 03A9 1586 SOBGTR R4,65$ ;COPY UNTIL NO MORE DELIMS
      52 59 D0 03AC 1587 67$: MOVL R9,R2 ;INIT RESULT DESCR
      53 D4 03AF 1588 CLRL R3 ;
      89 87 90 03B1 1589 70$: MOVAB (R7)+,(R9)+ ;COPY A CHARACTER
      53 D6 03B4 1590 INCL R3 ;INCR THE RESULT LENGTH
      56 D7 03B6 1591 DECL R6 ;DECR STRING SIZE
      08 13 03B8 1592 BEQL 75$ ;DONE IF ZERO
FE89 CF 04 67 3A 03BA 1593 LOCC (R7),#4,TIME_DELIMS ;IS NEXT CHAR A DELIM?
      EF 13 03C0 1594 BEQL 70$ ;NO, THEN MOVE NEXT CHAR
      EC 55 F5 03C2 1595 SOBGTR R5,70$ ;COPY UNTIL NO MORE DELIMS
      51 53 D0 03C5 1596 75$: MOVL R3,R1 ;GET RESULT LENGTH
      SE 14 C0 03C8 1597 ADDL #20,SP ;RESTORE THE STACK
      05 03CB 1598 RSB ;RETURN
      03CC 1599
      03CC 1600 ;
      03CC 1601 ; Return day of week.
```

```
OC AE 04 AE 23C34600 8F 7B 03CC 1602 :  
50 03CC 1603 90$: CDIV #600000000,4(SP),12(SP),R0 ; GET MINUTES SINCE BASE TIME VALUE  
10 AE D4 03D6 1604 CLRL 16(SP) ;  
OC AE OC AE 000005A0 8F 7B 03DA 1605 EDIV #60*24,12(SP),12(SP),R0 ; GET DAYS SINCE BASE TIME  
50 03E4 ;  
OC AE 02 C0 03E5 1605 ADDL #2,12(SP) ; ADJUST FOR BASE DAY BEING WED.  
10 AE D4 03E9 1607 CLRL 16(SP) ;  
50 OC AE OC AE 07 7B 03EC 1608 EDIV #7,12(SP),12(SP),R0 ; GET DAY OF WEEK  
52 0000218C'EF 9E 03F3 1609 MOVAB WEEKDAY_TABLE,R2 ; GET WEEKDAY TABLE ADDR  
50 0A C4 03FA 1610 MULL #10,R0 ; GET WEEKDAY TABLE OFFSET  
52 6240 9E 03FD 1611 MOVAB (R2)[R0],R2 ; GET ASCII WEEKDAY  
51 82 9A 0401 1612 MOVZBL (R2)+,R1 ; GET WEEKDAY DESCR  
5E 14 C0 0404 1613 ADDL #20,SP ; RESTORE THE STACK  
05 0407 1614 RSB ; RETURN
```

```
0408 1616 .SBTTL RETURN DEFAULT DIRECTORY NAME STRING
0408 1617 :+
0408 1618 : FSDIRECTORY - RETURN DEFAULT DIRECTORY NAME STRING
0408 1619 :
0408 1620 : THIS LEXICAL FUNCTION RETURNS THE ADDRESS AND LENGTH OF THE CURRENT DEFAULT
0408 1621 : DIRECTORY STRING.
0408 1622 :
0408 1623 : FSDIRECTORY()
0408 1624 :
0408 1625 : INPUTS:
0408 1626 :
0408 1627 : NONE.
0408 1628 :
0408 1629 : OUTPUTS:
0408 1630 :
0408 1631 : R1 = LENGTH OF DEFAULT DIRECTORY STRING.
0408 1632 : R2 = ADDRESS OF DEFAULT DIRECTORY STRING.
0408 1633 :-
0408 1634 :
0408 1635 FSDIRECTORY:
52 00000000'9F 9E 0408 1636 MOVAB @#PIOSGT_DDSTRING,R2 ;RETURN DEFAULT DIRECTORY STRING
51 82 9A 040F 1637 MOVZBL (R2)+,R1 ;GET ADDRESS OF DEFAULT DIRECTORY STRING
05 0412 1638 RSB ;GET LENGTH OF DEFAULT DIRECTORY STRING
;
```

```
0413 1640 .SBTTL EDIT A STRING
0413 1641 :+
0413 1642 : F$EDIT - EDIT A STRING
0413 1643 :
0413 1644 : THIS LEXICAL FUNCTION PERFORMS THE EDITS A STRING, AS DIRECTED BY AN
0413 1645 : ITEM LIST.
0413 1646 :
0413 1647 : F$EDIT (STRING, ITEM_LIST)
0413 1648 :
0413 1649 : INPUTS:
0413 1650 :
0413 1651 : P1(AP) = DESCRIPTOR OF TEXT STRING
0413 1652 : P2(AP) = ITEM LIST
0413 1653 :
0413 1654 : OUTPUTS:
0413 1655 :
0413 1656 : R1 = LENGTH OF EDITED STRING.
0413 1657 : R2 = ADDRESS OF EDITED STRING.
0413 1658 :-
0413 1659 :
0413 1660 F$EDIT:
54 0C AC 7D 0413 1661 MOVQ P2(AP),R4 ;GET DESCRIPTOR OF COMMA SEPARATED LIST
58 52 54 7D 0417 1662 CLRL R8 ;CLEAR THE FLAGS
00000040 8F C8 0419 1663 10$: MOVQ R4,R2 ;ASSUME NO COMMA IN LIST
65 54 2C 3A 041C 1664 BISL #EDIT_M_EOL,R8 ;ASSUME END OF LIST
58 00000040 8F CA 0423 1665 LOCC #'A',*,R4,(R5) ;LOCATE END OF NEXT KEYWORD
52 54 12 13 0427 1666 BEQL 20$ ;BRANCH IF NOT FOUND, USE ENTIRE STRING
58 00000040 8F CA 0429 1667 BICL #EDIT_M_EOL,R8 ;NOT END OF LIST
52 54 50 C3 0430 1668 SUBL3 R0,R4,R2 ;ISOLATE THE KEYWORD
54 54 70 9E 0434 1669 MOVAB -(R0),R4 ;CHOP THE KEYWORD OFF THE LIST
55 01 A1 9E 0437 1670 MOVAB 1(R1),R5 ;
043B 1671
56 00001E9E'EF 9E 043B 1672 20$: MOVAB EDIT_TABLE,R6 ;GET ADDRESS OF TABLE
57 66 9A 0442 1673 30$: MOVZBL (R6),R7 ;GET LENGTH OF ITEM NAME
7E 36 13 0445 1674 BEQL 90$ ;BRANCH IF END OF TABLE
7E 52 7D 0447 1675 MOVQ R2,-(SP) ;SAVE REGISTERS
00 63 54 7D 044A 1676 MOVQ R4,-(SP)
01 A6 57 2D 044D 1677 CMPC5 R2,(R3),#0,- ;DOES ITEM NAME MATCH?
OD 13 0454 1678 BEQL 50$ ;BRANCH IF ITEM FOUND
54 8E 7D 0456 1679 MOVQ (SP)+,R4 ;RESTORE REGISTERS
52 8E 7D 0459 1681 MOVQ (SP)+,R2
56 02 A647 9E 045C 1682 MOVAB 2(R6)[R7],R6 ;SKIP TO NEXT ENTRY IN TABLE
DF 11 0461 1683 BRB 30$ ;LOOP UNTIL FOUND
54 8E 7D 0463 1684 50$: MOVQ (SP)+,R4 ;RESTORE REGISTERS
52 8E 7D 0466 1685 MOVQ (SP)+,R2
58 01 A647 88 0469 1686 BISB 1(R6)[R7],R8 ;SET OPTION BIT
A7 58 06 E5 046E 1687 BBCC #EDIT_V_EOL,R8,10$ ;PROCESS NEXT KEYWORD IN LIST
0472 1688
51 04 AC 7D 0472 1689 60$: MOVQ P1(AP),R1 ;GET DESCRIPTOR INPUT STRING
50 58 D0 0476 1690 MOVL R8,R0 ;GET EDITING FLAGS
FB84' 30 0479 1691 BSBW DCL$EDIT ;EDIT THE STRING
05 047C 1692 RSB
047D 1693
047D 1694 90$: STATUS IVKEYW ;UNDEFINED ITEM NAME
1197 31 0484 1695 BRW P2_ERROR
```

```
0487 1697 .SBTTL FIND A STRING ELEMENT
0487 1698 :+
0487 1699 : FSELEMENT - FIND A STRING ELEMENT
0487 1700 :
0487 1701 : THIS LEXICAL FUNCTION EXTRACTS A SPECIFIED STRING ELEMENT FROM A STRING.
0487 1702 :
0487 1703 : FSELEMENT (NUMBER, DELIMITER, STRING)
0487 1704 :
0487 1705 : INPUTS:
0487 1706 :
0487 1707 : P1(AP) = ELEMENT NUMBER (BASED AT ZERO)
0487 1708 : P2(AP) = DESCRIPTOR OF SINGLE CHARACTER DELIMITER
0487 1709 : P3(AP) = DESCRIPTOR OF TEXT STRING.
0487 1710 :
0487 1711 : OUTPUTS:
0487 1712 :
0487 1713 : R1 = LENGTH OF EXTRACTED ELEMENT.
0487 1714 : R2 = ADDRESS OF EXTRACTED ELEMENT.
0487 1715 : -
0487 1716 :
0487 1717 FSELEMENT:
53 04 AC D0 0487 1718 MOVL P1(AP),R3 :GET ELEMENT NUMBER
53 53 D5 0488 1719 TSTL R3 :POSITIVE?
53 30 19 048D 1720 BLSS 85$ :NO, THEN ERROR
54 0C AC 7D 048F 1721 MOVQ P2(AP),R4 :GET DELIMITER STRING
50 14 AC 7D 0493 1722 MOVQ P3(AP),R0 :GET TEXT STRING
54 01 D1 0497 1723 CMPL #1,R4 :IS IT EXACTLY ONE CHAR?
54 2D 12 049A 1724 BNEQ 90$ :NO, THEN ERROR
54 50 D6 049C 1725 INCL R0 :SET INITIAL LENGTH
54 51 D7 049E 1726 DECL R1 :SET INITIAL POINTER
54 51 D6 04A0 1727 :
54 50 D7 04A2 1728 10$: INCL R1 :POINT PAST DELIMITER
54 51 D0 04A4 1729 DECL R0 :
61 50 65 3A 04A7 1730 MOVL R1,R2 :REMEMBER START OF ELEMENT
61 07 13 04AB 1731 LOCC (R5),R0,(R1) :LOCATE DELIMITER IN STRING
54 F0 53 F4 04AD 1732 BEQL 80$ :IF NOT FOUND, THEN RETURN DELIMITER
54 51 52 C2 04BC 1733 SOBGEQ R3,10$ :LOOP UNTIL WE FIND THE RIGHT ELEMENT
54 05 04B3 1734 70$: SUBL R2,R1 :CALCULATE LENGTH OF ELEMENT
54 53 D5 04B4 1735 RSB
54 F8 13 04B6 1736 :
54 01 D0 04B8 1737 80$: TSTL R3 :IF LAST ELEMENT
54 52 55 D0 04B8 1738 BEQL 70$ :THEN RETURN IT
54 05 04BE 1739 MOVL #1,R1 :RETURN DELIMITER
54 04BF 1740 MOVL R5,R2 :
54 04BF 1741 RSB
50 00038228 8F D0 04BF 1742 85$: MOVL #CLIS_INVRANGE,R0 :ASSUME RANGE WILL BE INVALID
50 114A 31 04C6 1743 BRW ERROR :SIGNAL ERROR
50 000388FA 8F D0 04C9 1744 :
50 114B 31 04D0 1745 90$: MOVL #CLIS_STRTOOLNG,R0 :DELIMITER IS MORE THAN ONE CHAR
50 04D0 1746 BRW P2_ERROR :SIGNAL ERROR
```

```
04D3 1749 .SBTTL GET DCL INFORMATION
04D3 1750 :+
04D3 1751 : F$ENVIRONMENT - GET DCL INFORMATION
04D3 1752 :
04D3 1753 : THIS LEXICAL FUNCTION RETRIEVES SPECIFIC INFORMATION ABOUT THE DCL PROCESS
04D3 1754 : CONTEXT.
04D3 1755 :
04D3 1756 : F$ENVIRONMENT (ITEM)
04D3 1757 :
04D3 1758 : INPUTS:
04D3 1759 :
04D3 1760 : P1(AP) = DESCRIPTOR OF ITEM
04D3 1761 :
04D3 1762 : OUTPUTS:
04D3 1763 :
04D3 1764 : R1 = LENGTH OF REQUESTED INFORMATION
04D3 1765 : R2 = ADDRESS OF REQUESTED INFORMATION
04D3 1766 : -
04D3 1767 :
04D3 1768 F$ENVIRONMENT:
56 00001ED7'EF 9E 04D3 1769 MOVAB ENVIRON_TABLE,R6 ;GET ADDRESS OF TABLE
7E 52 7D 04DA 1770 MOVQ R2,-(SP) ;SAVE THE SCRATCH DESCR
04DD 1771
57 66 9A 04DD 1772 10$: MOVZBL (R6),R7 ;GET LENGTH OF ITEM NAME
12 13 04E0 1773 BEQL 20$ ;BRANCH IF END OF TABLE
00 08 BC 04 AC 2D 04E2 1774 CMPC5 P1(AP),@P1+4(AP),#0,- ;DOES ITEM NAME MATCH?
01 A6 57 04E8 1775 R7,1(R6)
14 13 04EB 1776 BEQL 90$ ;BRANCH IF ITEM FOUND
56 02 A647 9E 04ED 1777 MOVAB 2(R6)[R7],R6 ;SKIP TO NEXT ENTRY IN TABLE
E9 11 04F2 1778 BRB 10$ ;LOOP UNTIL FOUND
04F4 1779
52 8E 7D 04F4 1780 20$: MOVQ (SP)+,R2 ;RESTORE THE SCRATCH DESCR
04F7 1781 STATUS IVKEYW ;UNDEFINED ITEM NAME
1117 31 04FE 1782 BRW P1_ERROR
52 8E 7D 0501 1783
0501 1784 90$: MOVQ (SP)+,R2 ;RESTORE THE SCRATCH DESCR
0504 1785 CASE 1(R6)[R7],TYPE=B,<- ;CASE ON ITEM
0504 1786 100$,- ;CONTROL
0504 1787 110$,- ;NOCONTROL
0504 1788 200$,- ;DEFAULT
0504 1789 300$,- ;DEPTH
0504 1790 400$,- ;ON SEVERITY
0504 1791 500$,- ;PROMPT
0504 1792 600$,- ;PROMPT CONTROL
0504 1793 700$,- ;PROTECTION
0504 1794 800$,- ;MESSAGE
0504 1795 900$,- ;MAX DEPTH
0504 1796 1000$,- ;ON CONTROL_Y
0504 1797 1100$,- ;PROCEDURE
0504 1798 1200$,- ;KEYPAD_STATE
0504 1799 1300$,- ;VERIFY_PROCEDURE
0504 1800 1400$,- ;VERIFY_IMAGE
0504 1801 1500$,- ;CAPTIVE
0504 1802 1600$,- ;INTERACTIVE
052E 1803 1700$> ;OUTPUT_RATE
052E 1804
052E 1805 :
```

```
052E 1806 : RETURN [NO]CONTROL INFORMATION
052E 1807 :
50 01 D0 052E 1808 100$: MOVL #1,R0 ;TEST IF BIT IS SET
02 11 0531 1809 BRB 120$ ;SKIP
50 03 D4 0533 1810 110$: CLRL R0 ;TEST IF BIT IS CLEAR
52 2D 1A 0535 1811 120$: CMPL #3,R2 ;IS RETURN BUFFER BIG ENOUGH?
56 53 D0 0538 1812 BGTRU 190$ ;BRANCH IF NOT
01 14 ED 053A 1813 MOVL R3,R6 ;COPY ADDRESS OF SCRATCH SPACE
50 00B4 CB 053D 1814 CMPZV #PRC_V_CTRLT,#1,- ;CHECK CTRL/T
04 12 0540 1815 PRC [-OUTOFBAND(R11),R0 ;
83 54 8F 90 0544 1816 BNEQ 130$ ;SKIP IF NOT EQUAL
01 19 ED 0546 1817 MOVB #^A/T/,(R3)+ ;INSERT T IN STRING
50 00B4 CB 054A 1818 130$: CMPZV #PRC_V_CTRLY,#1,- ;CHECK CTRL/Y
0C 12 054D 1819 PRC [-OUTOFBAND(R11),R0 ;
56 53 D1 0551 1820 BNEQ 150$ ;SKIP IF NOT EQUAL
03 13 0553 1821 CMPL R3,R6 ;HAVE WE INSERTED ANYTHING YET?
83 59 8F 90 0556 1822 BEQL 140$ ;NO, THEN SKIP
51 53 56 C3 0558 1823 MOVB #^A/,/(R3)+ ;INSERT A COMMA
52 56 D0 055B 1824 140$: MOVB #^A/Y/,(R3)+ ;INSERT Y IN STRING
05 055F 1825 150$: SUBL3 R6,R3,R1 ;GET LENGTH OF RESULT
0563 1826 MOVL R6,R2 ;GET ADDRESS OF RESULT
0566 1827 RSB ;
0567 1828 ;
10A2 31 0567 1829 190$: STATUS BUFOVF ;SIGNAL BUFFER OVERFLOW ERROR
056E 1830 BRW ERROR
0571 1831 ;
0571 1832 : RETURN DEFAULT
0571 1833 :
7E 52 7D 0571 1834 200$: MOVQ R2,-(SP) ;PUSH SCRATCH AREA DESCRIPTOR
58 5E D0 0574 1835 MOVL SP,R8 ;POINT R8 AT IT
FA86' 30 0577 1836 BSBW DCL$SHOWDEFINT ;GET DEFAULT DIRECTORY SPEC
5E 08 C0 057A 1837 ADDL #8,SP ;RESTORE THE STACK
FA80' 30 057D 1838 BSBW DCL$TRIM ;REMOVE LEADING BLANKS
05 0580 1839 RSB ;
0581 1840 ;
0581 1841 : RETURN DEPTH
0581 1842 :
51 5C AB D0 0581 1843 300$: MOVL PRC_L_INDEPTH(R11),R1 ;GET COMMAND PROCEDURE DEPTH
52 52 D4 0585 1844 CLRL R2 ;RETURN IN INTEGER FORM
05 0587 1845 RSB ;
0588 1846 ;
0588 1847 : RETURN ON_SEVERITY
0588 1848 :
0588 1849 :
0588 1850 :
0588 1851 :
50 6A AB 9A 0588 1852 400$: MOVZBL PRC_W_ONLEVEL(R11),R0 ;GET SEVERITY NUMBER
50 01 8A 058C 1853 BICB #1,R0 ;CLEAR THE CONTROL/Y BIT
52 00002072'EF40 DE 058F 1854 MOVAL SEVERITY_TABLE[R0],R2 ;GET ADDRESS OF ASCII SEVERITY
51 82 9A 0597 1855 MOVZBL (R2)+,R1 ;SET DESCRIPTOR OF SEVERITY
05 059A 1856 RSB ;
059B 1857 ;
059B 1858 : RETURN PROMPT
059B 1859 :
51 00F0 CB 9A 059B 1860 500$: MOVZBL PRC_B_PROMPTLEN(R11),R1 ;GET LENGTH OF PROMPT STRING
51 03 C2 05A0 1861 SUBL #3,R1 ;REMOVE LENGTH OF CR/LF/_
05A0 1862
```

```
52 00F4 CB 9E 05A3 1863      MOVAB  PRC_G_PROMPT(R11),R2      ;GET ADDRESS OF PROMPT STRING
      05 05A8 1864      RSB
      05A9 1865
      05A9 1866      ;
      05A9 1867      ; RETURN PROMPT_CONTROL
      05A9 1868
      05A9 1869 600$:   MOVL  #1,R1      ;ASSUME CR/LF SPECIFIED
02 68 AB 01 00 05AC 1870      BBS  #PRC_V_CARRCNTL,PRC_W_FLAGS(R11),610$  ;SKIP IF SPECIFIED
      51 00 05B1 1871      CLRL  R1      ;RETURN FALSE OTHERWISE
      11CC 31 05B3 1872 610$:   BRW  FORMAT_BOOLEAN      ;RETURN THE VALUE
      05B6 1873
      05B6 1874      ;
      05B6 1875      ; RETURN PROTECTION
      05B6 1876
      05B6 1877 700$:   CLRL  -(SP)      ;WHERE TO RETURN PROTECTION
      7E 7E D4 05B8 1878      MOVL  SP,-(SP)      ;NOTE WHERE PROTECTION IS TO BE PUT
      7E 5E D0 05B8 1879      CLRL  -(SP)      ;DON'T WANT TO SET PROTECTION
00000000'9F 02 FB 05BD 1880      CALLS #2,#SYSS$SETDFPROT      ;GET DEFAULT PROTECTION
      54 8E D0 05C4 1881      MOVL  (SP)+,R4      ;COPY PROTECTION TO USEFUL REG
      110B 31 05C7 1882      BRW  FORMAT_PROT      ;CONVERT INTO ASCII FORM
      05CA 1883
      05CA 1884      ;
      05CA 1885      ; RETURN MESSAGE QUALIFIERS
      05CA 1886
      58 2F 9A 05CA 1887 800$:   MOVZBL #^A'/'',R8      ;STORE FIELD DELIMITER
      56 53 D0 05CD 1888      MOVL  R3,R6      ;COPY ADDRESS OF SCRATCH SPACE
      83 58 90 05D0 1889      MOVB  R8,(R3)+      ;INSERT A LEADING SLASH IN STRING
      05D3 1890
      52 2E D1 05D3 1891      CMPL  #46,R2      ;IS SCRATCH BUFFER BIG ENOUGH?
      8F 1A 05D6 1892      BGTRU  190$      ;NO, THEN ERROR
59 00000000'GF 9A 05D8 1893      MOVZBL G^CTL$GB_M$GMASK,R9      ;GET MESSAGE MASK
      05DF 1894
      05DF 1895      MOVAB  FACILITY,R0      ;GET ADDRESS OF ASCII STRING
50 00002255'EF 9E 05DF 1896      MOVZBL (R0)+,R1      ;GET LENGTH/ADDRESS OF STRING
      51 80 9A 05E6 1897      BBS  #3,R9,810$      ;IS FACILITY SPECIFIED?
      05 59 03 E0 05E9 1898      MOVW  #^A/NO/, (R3)+      ;INSERT NO IN STRING
      83 4F4E 8F B0 05ED 1899 810$:   MOVC3 R1,(R0),(R3)      ;MOVE STRING INTO DESCRIPTOR
      63 60 51 28 05F2 1899      MOVB  R8,(R3)+      ;INSERT DELIMITER
      83 58 90 05F6 1900
      05F9 1901
      50 0000224C'EF 9E 05F9 1902      MOVAB  SEVERITY,R0      ;GET ADDRESS OF ASCII STRING
      51 80 9A 0600 1903      MOVZBL (R0)+,R1      ;GET LENGTH/ADDRESS OF STRING
      05 59 02 E0 0603 1904      BBS  #2,R9,820$      ;IS SEVERITY SPECIFIED?
      83 4F4E 8F B0 0607 1905 820$:   MOVW  #^A/NO/, (R3)+      ;INSERT NO IN STRING
      63 60 51 28 060C 1906      MOVC3 R1,(R0),(R3)      ;MOVE STRING INTO DESCRIPTOR
      83 58 90 0610 1907      MOVB  R8,(R3)+      ;INSERT DELIMITER
      0613 1908
      50 0000223D'EF 9E 0613 1909      MOVAB  IDENT,R0      ;GET ADDRESS OF ASCII STRING
      51 80 9A 061A 1910      MOVZBL (R0)+,R1      ;GET LENGTH/ADDRESS OF STRING
      05 59 01 E0 061D 1911      BBS  #1,R9,830$      ;IS SEVERITY SPECIFIED?
      83 4F4E 8F B0 0621 1912 830$:   MOVW  #^A/NO/, (R3)+      ;INSERT NO IN STRING
      63 60 51 28 0626 1913      MOVC3 R1,(R0),(R3)      ;MOVE STRING INTO DESCRIPTOR
      83 58 90 062A 1914      MOVB  R8,(R3)+      ;INSERT DELIMITER
      062D 1915
      50 00002238'EF 9E 062D 1916      MOVAB  TEXT,R0      ;GET ADDRESS OF ASCII STRING
      51 80 9A 0634 1917      MOVZBL (R0)+,R1      ;GET LENGTH/ADDRESS OF STRING
      05 59 00 E0 0637 1918      BBS  #0,R9,840$      ;IS TEXT SPECIFIED?
      83 4F4E 8F B0 063B 1919      MOVW  #^A/NO/, (R3)+      ;INSERT NO IN STRING
```

```
63 60 51 28 0640 1920 840$:  MOVCL3  R1,(R0),(R3)          ;MOVE STRING INTO DESCRIPTOR
                                0644 1921
51 53 56 C3 0644 1922          SUBCL3  R6,R3,R1          ;GET LENGTH OF RESULT
52 56 D0 0648 1923          MOVL    R6,R2          ;GET ADDRESS OF RESULT
05 064B 1924          RSB
                                064C 1925
                                064C 1926
                                064C 1927          ; RETURN MAX_DEPTH
                                064C 1928
51 00000000'8F D0 064C 1929 900$:  MOVL    #MAX_DEPTH,R1      ;GET MAXIMUM DEPTH OF COMMAND PROCEDURES
52 D4 0653 1930          CLRL    R2          ;RETURN AS INTEGER
05 0655 1931          RSB
                                0656 1932
                                0656 1933
                                0656 1934          ; RETURN ON_CONTROL_Y
                                0656 1935
51 01 D0 0656 1936 1000$:  MOVL    #1,R1          ;ASSUME HANDLER SPECIFIED
00B8 CB D5 0659 1937          TSTL    PRC_L_ONCTLY(R11)      ;IS IT?
02 12 065D 1938          BNEQ    1010$          ;YES, THEN SKIP
51 D4 065F 1939          CLRL    R1          ;NO, THEN RETURN FALSE
111E 31 0661 1940 1010$:  BRW     FORMAT_BOOLEAN          ;RETURN THE VALUE
                                0664 1941
                                0664 1942
                                0664 1943          ; RETURN PROCEDURE
                                0664 1944
50 00BC CB D0 0664 1945 1100$:  MOVL    PRC_L_IDFLNK(R11),R0      ;GET ADDRESS OF IDF BLOCK
52 68 A0 D0 0669 1946          MOVL    IDF_L_FILENAME(R0),R2    ;GET ADDRESS OF ASCIC FILENAME
04 13 066D 1947          BEQL    1110$          ;SKIP IF NONE
51 82 9A 066F 1948          MOVZBL  (R2)+,R1          ;GET LENGTH/ADDRESS OF FILENAME
05 0672 1949          RSB
                                0673 1950
52 51 D4 0673 1951 1110$:  CLRL    R1          ;RETURN NO FILENAME PRESENT
5E D0 0675 1952          MOVL    SP,R2          ;INDICATE NULL STRING
05 0678 1953          RSB
                                0679 1954
                                0679 1955
                                0679 1956          ; RETURN KEYPAD_STATE
                                0679 1957
52 48 AB D0 0679 1958 1200$:  MOVL    PRC_L_CURRKEY(R11),R2      ;GET ADDRESS OF ASCIC STATE
51 82 9A 067D 1959          MOVZBL  (R2)+,R1          ;GET LENGTH/ADDRESS OF STATE
05 0680 1960          RSB
                                0681 1961
                                0681 1962
                                0681 1963          ; RETURN VERIFY_PROCEDURE
                                0681 1964
51 D4 0681 1965 1300$:  CLRL    R1          ;ASSUME NOVERIFY
07 E1 0683 1966          BBC     #PRC_V_VERIFY,-          ;BRANCH IF NOVERIFY
02 68 AB 0685 1967          PRC_B_FLAGS(R11),1320$          ;
51 D6 0688 1968 1310$:  INCL    R1          ;SET TRUE
10F5 31 068A 1969 1320$:  BRW     FORMAT_BOOLEAN          ;RETURN TRUE/FALSE
                                068D 1970
                                068D 1971
                                068D 1972          ; RETURN VERIFY_IMAGE
                                068D 1973
51 D4 068D 1974 1400$:  CLRL    R1          ;ASSUME NOVERIFY
07 E1 068F 1975          BBC     #PRC_V_VERIMAGE,-          ;BRANCH IF NOVERIFY
F5 00AF CB 0691 1976          PRC_B_FLAGS2(R11),1320$          ;
```

```
F1 11 0695 1977 BRB 1310$ ;SET TRUE
      0697 1978
      0697 1979 :
      0697 1980 : RETURN CAPTIVE
      0697 1981 :
50 00000000'GF 9E 0697 1982 1500$: MOVAB G^CTLSAG_CLIDATA,R0 ;GET ADDRESS OF PPD
      51 D4 069E 1983 CLRL R1 ;ASSUME NOT CAPTIVE
      03 E1 06A0 1984 BBC #PPDSV CAPTIVE,- ;BRANCH IF NOT CAPTIVE ACCOUNT
      E5 02 A0 06A2 1985 PPD$W_FLAGS(R0),1320$
      E1 11 06A5 1986 BRB 1310$- ;SET TRUE
      06A7 1987
      06A7 1988 :
      06A7 1989 : RETURN INTERACTIVE
      06A7 1990 :
      51 D4 06A7 1991 1600$: CLRL R1 ;ASSUME NOT INTERACTIVE
      06 E0 06A9 1992 BBS #PRC V MODE,- ;BRANCH IF NOT INTERACTIVE
      DC 68 AB 06AB 1993 PRC Q_FLAGS(R11),1320$
      D8 11 06AE 1994 BRB 1310$- ;SET TRUE
      06B0 1995
      06B0 1996 :
      06B0 1997 : RETURN OUTPUT_RATE
      06B0 1998 :
      7E 52 7D 06B0 1999 1700$: MOVQ R2,-(SP) ;SET UP RETURN DESCR
      06 E1 06B3 2000 BBC #PRC V MODE,- ;BRANCH IF INTERACTIVE
      1B 68 AB 06B5 2001 PRC Q_FLAGS(R11),1710$
      50 5E D0 06B8 2002 MOVL SP,R0 ;GET ADDR OF OUTPUT DESCR
      06BB 2003 SASCTIM_S TIMBUF=(R0),- ;CONVERT BINARY DELTA TIME TO ASCII
      06BB 2004 TIMLEN=(R0),-
      06BB 2005 TIMADR=PRC_Q_FLUSHTIME(R11)
      04 50 E9 06CC 2006 BLBC R0,1710$ ;BR IF ERROR IN TIME
      51 8E 7D 06CF 2007 MOVQ (SP)+,R1 ;GET THE RESULT
      05 06D2 2008 RSB ;RETURN DELTA TIME
      51 8E 7D 06D3 2009 1710$: MOVQ (SP)+,R1 ;RETURN NULL
      51 D4 06D6 2010 CLRL R1
      05 06D8 2011 RSB
```

```
06D9 2013 .SBTTL EXTRACT SUBSTRING FROM STRING
06D9 2014 :+
06D9 2015 : F$EXTRACT - EXTRACT SUBSTRING FROM STRING
06D9 2016 :
06D9 2017 : THIS LEXICAL FUNCTION EXTRACTS A SUBSTRING FROM AN ARGUMENT STRING AND RETURNS
06D9 2018 : ITS DESCRIPTOR.
06D9 2019 :
06D9 2020 : F$EXTRACT(POSITION, SIZE, STRING)
06D9 2021 :
06D9 2022 : INPUTS:
06D9 2023 :
06D9 2024 : P1(AP) = STARTING CHARACTER NUMBER (BASED AT ZERO)
06D9 2025 : P2(AP) = NUMBER OF CHARACTERS
06D9 2026 : P3(AP) = DESCRIPTOR OF TEXT STRING.
06D9 2027 :
06D9 2028 : OUTPUTS:
06D9 2029 :
06D9 2030 : R1 = LENGTH OF EXTRACTED SUBSTRING.
06D9 2031 : R2 = ADDRESS OF EXTRACTED SUBSTRING.
06D9 2032 : -
06D9 2033 :
06D9 2034 F$EXTRACT:
06D9 2035 CLRL R1 ;ASSUME NULL SUBSTRING
14 AC 04 AC 51 D4 06D9 2035 CMPL P1(AP),P3(AP) ;STARTING POSITION BEYOND END OF STRING
06D9 2036 BGEQU 10$ ;IF GTRU YES
51 0C AC 19 1E 06E0 2037 MOVL P2(AP),R1 ;GET NUMBER OF CHARACTERS TO EXTRACT
18 AC 04 AC D0 06E2 2038 ADDL3 P1(AP),P3+4(AP),R2 ;CALCULATE STARTING ADDRESS OF SUBSTRING
14 AC 04 AC C1 06E6 2039 SUBL P1(AP),P3(AP) ;REDUCE APPARENT STRING LENGTH BY POSITION
14 AC 51 C2 06EC 2040 SUBL R1,P3(AP) ;REDUCE STRING BY LENGTH OF SUBSTRING
51 14 AC 04 1E 06F5 2042 BGEQU 10$ ;IF GEQU STRING LENGTH SUFFICIENT
05 06F7 2043 ADDL P3(AP),R1 ;CALCULATE ACTUAL LENGTH OF SUBSTRING
05 06FB 2044 10$: RSB
```

```
06FC 2046 .SBTTL FAO SYSTEM SERVICE
06FC 2047 :+
06FC 2048 : F$FAO - FORMATTED ASCII OUTPUT
06FC 2049 :
06FC 2050 : THIS LEXICAL FUNCTION INVOKES THE $FAO SYSTEM SERVICE WITH
06FC 2051 : EITHER STRINGS OR LONGWORDS AS ARGUMENTS, DEPENDING ON THE
06FC 2052 : TYPE OF THE ARGUMENT.
06FC 2053 :
06FC 2054 : F$FAO(CTRSTR,ARG1,ARG2,...)
06FC 2055 :
06FC 2056 : INPUTS:
06FC 2057 :
06FC 2058 :     NPARM(AP) = NUMBER OF ARGUMENTS SPECIFIED
06FC 2059 :     P1(AP) = FIRST ARGUMENT
06FC 2060 :     P2(AP) = SECOND ARGUMENT
06FC 2061 :     ETC.
06FC 2062 :
06FC 2063 : OUTPUTS:
06FC 2064 :
06FC 2065 :     R1/R2 = RESULTANT STRING FROM $FAO
06FC 2066 : -
06FC 2067 :
06FC 2068 F$FAO:
06FC 2069     MOVQ    R2,-(SP)                ;ADDRESS OF RESULT BUFFER DESCRIPTOR
06FC 2070     MOVL    SP,R2                ;ADDRESS OF BUFFER DESCRIPTOR
06FC 2071     SUBL    #1,NPARM(AP),R0      ;GET NUMBER OF FAO ARGUMENTS
06FC 2072     MOVAQ   P2(AP)[R0],R1        ;START AT LAST ARGUMENT + 1
06FC 2073 10$:    TSTL    -4(R1)          ;BINARY LONGWORD OR STRING DESCRIPTOR?
06FC 2074     BNEQ    15$                ;BRANCH IF STRING
06FC 2075     SUBL    #8,R1              ;SKIP TO NEXT ARGUMENT
06FC 2076     PUSHL   (R1)               ;PUSH LONGWORD BINARY VALUE
06FC 2077     BRB     20$
06FC 2078 15$:    PUSHAQ  -(R1)          ;PUSH ADDRESS OF DESCRIPTOR
06FC 2079 20$:    SOBGTR  R0,10$        ;LOOP THRU ALL FAO ARGUMENTS
06FC 2080     PUSHL   SP                ;ADDRESS OF PARAMETER LIST
06FC 2081     PUSHL   R2                ;ADDRESS OF BUFFER DESCRIPTOR
06FC 2082     PUSHL   R2                ;ADDRESS OF WORD TO RECEIVE LENGTH
06FC 2083     PUSHAQ  P1(AP)            ;ADDRESS OF STRING DESCRIPTOR
06FC 2084     CALLS   #4,G^SYSS$FAOL    ;INVOKE FAO SERVICE
06FC 2085     BLBC    R0,90$            ;BRANCH IF ERROR DETECTED
06FC 2086     MOVL    R2,SP             ;POP FAO ARGUMENTS OFF STACK
06FC 2087     PUPR    #^M<R1,R2>       ;RETRIEVE DESCRIPTOR OF RESULT STRING
06FC 2088 30$:    RSB
06FC 2089
06FC 2090 90$:    ADDL3   #8,R2,SP        ;CLEAN STACK
06FC 2091     BRW     P1_ERROR           ;REPORT ERROR AND DISPLAY P1
```

7E 52 7D 06FC 2069
52 5E DO 06FC 2070
50 6C 01 C3 0702 2071
51 OC AC40 7E 0706 2072
FC A1 D5 070B 2073
07 12 070E 2074
51 08 C2 0710 2075
61 DD 0713 2076
02 11 0715 2077
71 7F 0717 2078
EF 50 F5 0719 2079
5E DD 071C 2080
52 DD 071E 2081
52 DD 0720 2082
04 AC 7F 0722 2083
00000000'GF 04 FB 0725 2084
06 50 E9 072C 2085
5E 52 DO 072F 2086
06 BA 0732 2087
05 0734 2088
0735 2089
5E 52 08 C1 0735 2090
0EDC 31 0739 2091

```
073C 2093 .SBTTL GET FILE ATTRIBUTES
073C 2094 :+
073C 2095 : FSFILE_ATTRIBUTES - GET FILE INFORMATION
073C 2096 :
073C 2097 : THIS LEXICAL FUNCTION RETURNS A SPECIFIED ATTRIBUTE FOR A FILE
073C 2098 : BY USING THE RMS OPEN SYSTEM SERVICE TO OBTAIN THE INFORMATION.
073C 2099 :
073C 2100 : FSFILE_ATTRIBUTES(FILESPEC,ITEM)
073C 2101 :
073C 2102 : INPUTS:
073C 2103 :
073C 2104 : P1(AP) = DESCRIPTOR OF FILE SPECIFICATION
073C 2105 : P2(AP) = DESCRIPTOR OF RMS ITEM NAME
073C 2106 :
073C 2107 : OUTPUTS:
073C 2108 :
073C 2109 : R1/R2 = DESCRIPTOR OF RESULTANT DATA
073C 2110 :-
073C 2111 :
073C 2112 FSFILE:
073C 2113 MOVQ R2,-(SP) ;PUSH DESCRIPTOR OF SCRATCH SPACE
073C 2114 MOVL SP,R9 ;SAVE ADDRESS OF DESCRIPTOR
0742 2115 MOVAB FILE_TABLE,R7 ;GET ADDRESS OF TABLE
0749 2116 10$: MOVZBL 5(R7),R8 ;GET LENGTH OF ITEM NAME
074D 2117 BEQL 15$ ;BRANCH IF END OF TABLE
074F 2118 CMPC5 P2(AP),@P2+4(AP),#0,- ;DOES ITEM NAME MATCH?
0755 2119 R8,6(R7)
0758 2120 BEQL 20$ ;BRANCH IF ITEM FOUND
075A 2121 MOVAB 6(R7)[R8],R7 ;SKIP TO NEXT ENTRY IN TABLE
075F 2122 BRB 10$ ;LOOP UNTIL FOUND
0761 2123 15$: STATUS IVKEYW ;UNDEFINED ITEM NAME
0768 2124 BRW P2 ERROR
076B 2125 20$: MOVAB -FAB$C_BLN(SP),SP ;ALLOCATE FAB ON STACK
076F 2126 MOVL SP,R6 ;GET ADDRESS OF FAB
0772 2127 MOVCS #0,(SP),#0,#FAB$C_BLN,(R6) ;ZERO STORAGE
077A 2128 ASSUME FAB$B_BLN EQ FAB$B_BID+1
077A 2129 MOVW #FAB$C_BID+<FAB$C_BLN@8>,- ;INITIALIZE FAB
077E 2130 FAB$B_BID(R6)
077F 2131 MOVZBL (R7),R0 ;GET BLOCK INDEX
0782 2132 CMPL R0,#1 ;FAB OR NAM BLOCK?
0785 2133 BLSSU 50$ ;BRANCH IF FAB ONLY IS NEEDED
0787 2134 BEQL 40$ ;BRANCH IF NAM BLOCK IS ALSO NEEDED
0789 2135 MOVZWL XAB_INFO-4[R0],R2 ;GET XAB BLOCK ID & LENGTH
0791 2136 ASHL #-8,R2,R1 ;ISOLATE BLOCK LENGTH
0796 2137 SUBL R1,SP ;ALLOCATE SPACE FOR XAB ON STACK
0799 2138 PUSH R2 ;SAVE R2
079B 2139 MOVCS #0,(SP),#0,R1,4(SP) ;ZERO STORAGE
07A2 2140 POPL R2 ;RESTORE R2
07A5 2141 ASSUME XAB$B_BLN EQ XAB$B_COD+1
07A5 2142 MOVW R2,XAB$B_COD(SP) ;STORE XAB BLOCK ID/LENGTH
07AB 2143 MOVL SP,FAB$L_XAB(R6) ;POINT TO XAB BLOCK
07AC 2144 BRB 50$
07AE 2145 40$: MOVAB -NAM$C_BLN-NAM$C_MAXRSS(SP),SP ;ALLOCATE SPACE FOR NAM BLOCK
07B3 2146 MOVCS #0,(SP),#0,#NAM$C_BLN,(SP) ;ZERO STORAGE
07BB 2147 ASSUME NAM$B_BLN EQ NAM$B_BID+1
07BB 2148 MOVW #NAM$C_BID+<NAM$C_BLN@8>,- ;INITIALIZE NAM
07BF 2149 NAM$B_BID(SP)
```

7E	52	7D	073C	2113	MOVQ	R2,-(SP)	;PUSH DESCRIPTOR OF SCRATCH SPACE
59	5E	D0	073F	2114	MOVL	SP,R9	;SAVE ADDRESS OF DESCRIPTOR
57	00001513'EF	9E	0742	2115	MOVAB	FILE_TABLE,R7	;GET ADDRESS OF TABLE
58	05 A7	9A	0749	2116	MOVZBL	5(R7),R8	;GET LENGTH OF ITEM NAME
	12	13	074D	2117	BEQL	15\$	;BRANCH IF END OF TABLE
00	10 BC 0C AC	2D	074F	2118	CMPC5	P2(AP),@P2+4(AP),#0,-	;DOES ITEM NAME MATCH?
	06 A7 58		0755	2119		R8,6(R7)	
	11	13	0758	2120	BEQL	20\$	;BRANCH IF ITEM FOUND
57	06 A748	9E	075A	2121	MOVAB	6(R7)[R8],R7	;SKIP TO NEXT ENTRY IN TABLE
	E8	11	075F	2122	BRB	10\$	;LOOP UNTIL FOUND
			0761	2123	15\$: STATUS	IVKEYW	;UNDEFINED ITEM NAME
	0EB3	31	0768	2124	BRW	P2 ERROR	
	5E B0 AE	9E	076B	2125	20\$: MOVAB	-FAB\$C_BLN(SP),SP	;ALLOCATE FAB ON STACK
66	0050 8F 00	D0	076F	2126	MOVL	SP,R6	;GET ADDRESS OF FAB
	6E 00	2C	0772	2127	MOVCS	#0,(SP),#0,#FAB\$C_BLN,(R6)	;ZERO STORAGE
	5003 8F B0	077A	2128	ASSUME	FAB\$B_BLN EQ FAB\$B_BID+1		
	66	077A	2129	MOVW	#FAB\$C_BID+<FAB\$C_BLN@8>,-		;INITIALIZE FAB
	50 67	077E	2130		FAB\$B_BID(R6)		
	01 50	077F	2131	MOVZBL	(R7),R0		;GET BLOCK INDEX
	47	D1	0782	CMPL	R0,#1		;FAB OR NAM BLOCK?
	25	1F	0785	BLSSU	50\$		;BRANCH IF FAB ONLY IS NEEDED
	52	13	0787	BEQL	40\$		;BRANCH IF NAM BLOCK IS ALSO NEEDED
52	00001505'EF40	3C	0789	2135	MOVZWL	XAB_INFO-4[R0],R2	;GET XAB BLOCK ID & LENGTH
51	52 F8 8F	78	0791	2136	ASHL	8,R2,R1	;ISOLATE BLOCK LENGTH
	5E 51	C2	0796	2137	SUBL	R1,SP	;ALLOCATE SPACE FOR XAB ON STACK
	52	DD	0799	2138	PUSH	R2	;SAVE R2
04	AE 51 00	2C	079B	2139	MOVCS	#0,(SP),#0,R1,4(SP)	;ZERO STORAGE
	6E 00	07A2	2140	POPL	R2		;RESTORE R2
	52 8ED0	07A5	2141	ASSUME	XAB\$B_BLN EQ XAB\$B_COD+1		
	6E 52 B0	07A5	2142	MOVW	R2,XAB\$B_COD(SP)		;STORE XAB BLOCK ID/LENGTH
	24 A6 5E	D0	07AB	2143	MOVL	SP,FAB\$L_XAB(R6)	;POINT TO XAB BLOCK
	20	11	07AC	2144	BRB	50\$	
6E	0060 8F 5E	9E	07AE	2145	40\$: MOVAB	-NAM\$C_BLN-NAM\$C_MAXRSS(SP),SP	;ALLOCATE SPACE FOR NAM BLOCK
	FEA1 CE	2C	07B3	2146	MOVCS	#0,(SP),#0,#NAM\$C_BLN,(SP)	;ZERO STORAGE
	6E 00	07BB	2147	ASSUME	NAM\$B_BLN EQ NAM\$B_BID+1		
	6002 8F B0	07BB	2148	MOVW	#NAM\$C_BID+<NAM\$C_BLN@8>,-		;INITIALIZE NAM
	6E	07BF	2149		NAM\$B_BID(SP)		

```
0A AE FF 8F 90 07C0 2150      MOVB      #NAM$C MAXRSS,NAM$B_ESS(SP)      ;SET SIZE OF EXPANDED BUFFER
0C AE 60 AE 9E 07C5 2151      MOVAB     NAM$C_BLN(SP),NAM$B_ESA(SP)      ;SET ADDRESS OF RESULT BUFFER
28 A6 5E DC 07CA 2152      MOVL      SP,FAB$B_NAM(R6)      ;POINT TO NAM BLOCK
04 A7 0B 91 07CE 2153 50$:   CMPB      #11,4(R7)      ;KNOWN FILE CHECK?
                                BNEQ      55$      ;NO, THEN SKIP
                                SETBIT     NAM$V_NOCONCEAL,-      ;TRANSLATE CONCEALED LOG. NAMES
                                07D4 2155
                                07D4 2156
                                07D8 2157      SETBIT     FAB$V_KFO,FAB$B_FOP(R6)      ;SET KNOWN FILE OPEN
34 A6 04 AC 90 07DD 2158 55$:   MOVB      P1(APT,FAB$B_FNS(R6)      ;SET PRIMARY FILESPEC
2C A6 08 AC DC 07E2 2159      MOVL      P1+4(AP),FAB$B_FNA(R6)      ;
16 A6 02 90 07E7 2160      MOVB      #FAB$M_GET,FAB$B_FAC(R6)      ;READ-ONLY ACCESS
17 A6 4F 8F 90 07EB 2161      MOVB      #FAB$M_GET!FAB$M_PUT!FAB$M_UPD!-      ;ALLOW OTHERS
                                07F0 2162      FAB$M_DEL!FAB$M_OPI,FAB$B_SHR(R6)      ;
                                07F0 2163      SETBIT     FAB$V_PPF,FAB$B_FOP(R6)      ;USE PROCESS I/O SEGMENT
                                07F5 2164      $OPEN      FAB=(R6)      ;OPEN THE FILE TO GET REQUESTED INFO
                                07FE 2165      BLBC      R0,85$      ;IF ERROR, RETURN NULL STRING
                                0801 2166      $CLOSE     FAB=(R6)      ;CLOSE THE FILE AGAIN
55 01 A7 9A 080A 2167      MOVZBL     1(R7),R5      ;GET OFFSET TO FIELD
12 13 080E 2168      BEQL      60$      ;BRANCH IF NONE
55 05 SE C0 0810 2169      ADDL      SP,R5      ;POINT TO FIELD
54 02 A7 9A 0813 2170      MOVZBL     2(R7),R4      ;GET LENGTH OF FIELD
09 13 0817 2171      BEQL      60$      ;BRANCH IF NONE
50 03 A7 9A 0819 2172      MOVZBL     3(R7),R0      ;GET STARTING BIT NUMBER
51 65 54 50 EF 081D 2173      EXTZV     R0,R4,(R5),R1      ;FETCH CONTENTS OF FIELD
37 AF 9F 0822 2174 60$:   PUSHAB     B^90$      ;RETURN AFTER CASE
15 11 0825 2175      BRB      GET_FILE_ITEM      ;GET DATA ITEM
0827 2176
51 0C A6 DC 0827 2177 85$:   MOVL      FAB$B_STV(R6),R1      ;GET SECONDARY STATUS FROM $OPEN
03 13 082B 2178      BEQL      88$      ;BRANCH IF NONE
50 51 DC 082D 2179      MOVL      R1,R0      ;REPORT SECONDARY ACP STATUS, NOT RM
5E 08 A9 9E 0830 2180 88$:   MOVAB     8(R9),SP      ;CLEANUP STACK
ODE1 31 0834 2181      BRW      P1_ERROR      ;REPORT ERROR
5E 08 A9 9E 0837 2182
05 083B 2183 90$:   MOVAB     8(R9),SP      ;DEALLOCATE STORAGE ON STACK
083C 2184      RSB
083C 2185
083C 2186 GET_FILE_ITEM:
083C 2187      CASE      4(R7),TYPE=B,<-      ;CASE ON TYPE OF FIELD
083C 2188      100$,-      ;BINARY
083C 2189      200$,-      ;DATE/TIME
083C 2190      300$,-      ;ASCII STRING
083C 2191      400$,-      ;FILE ID STRING
083C 2192      500$,-      ;FILE ORGANIZATION
083C 2193      600$,-      ;RECORD ATTRIBUTES
083C 2194      700$,-      ;RECORD FORMAT
083C 2195      800$,-      ;OWNER UIC
083C 2196      900$,-      ;PROTECTION
083C 2197      1000$,-      ;EOF - # BLOCKS IN USE
083C 2198      1100$,-      ;BOOLEAN
083C 2199      1200$>      ;KNOWN FILE
0859 2200
0859 2201 ;
0859 2202 ; RETURN BINARY NUMBER
0859 2203 ;
52 D4 0859 2204 100$:   CLRL      R2      ;INDICATE BINARY RESULT
05 085B 2205      RSB
085C 2206 ;
```

```
085C 2207 : RETURN DATE/TIME STRING
085C 2208 :
51 69 7D 085C 2209 200$: SASCTIM_S (R9), (R9), (R5) ; CONVERT TO ASCII DATE/TIME STRING
05 086B 2210 MOVQ -(R9), R1 ; GET DESCRIPTOR OF STRING
086E 2211 RSB
086F 2212 :
086F 2213 : RETURN ASCII STRING
086F 2214 :
51 85 9A 086F 2215 300$: MOVZBL (R5)+, R1 ; GET LENGTH OF STRING
69 51 D1 0872 2216 CMPL R1, (R9) ; ENOUGH ROOM FOR STRING?
OF 1A 0875 2217 BGTRU 310$ ; BRANCH IF NOT
51 DD 0877 2218 PUSHL R1 ; SAVE LENGTH OF STRING
04 B9 65 51 28 0879 2219 MOVC R1, (R5), 24(R9) ; STORE INTO SCRATCH BUFFER
51 8ED0 087E 2220 POPL R1 ; RESTORE LENGTH OF STRING
52 04 A9 D0 0881 2221 MOVL 4(R9), R2 ; GET ADDRESS OF STRING
05 0885 2222 RSB
0886 2223 310$: STATUS BUFOVF ; SIGNAL BUFFER OVERFLOW
0D8E 31 088D 2224 BRW P2_ERROR
0890 2225 :
0890 2226 : RETURN FILE ID NUMBER
0890 2227 :
54 55 D0 0890 2228 400$: MOVL R5, R4 ; PASS ADDRESS OF FILE ID
OEBA 31 0893 2229 BRW FORMAT_FILEID ; FORMAT INTO ASCII STRING
0896 2230 :
0896 2231 : RETURN FILE ORGANIZATION
0896 2232 :
0896 2233 : ASSUME FAB$C_SEQ EQ 0 ; ^X00
0896 2234 : ASSUME FAB$C_REL EQ 16 ; ^X10
0896 2235 : ASSUME FAB$C_IDX EQ 32 ; ^X20
58 44 49 20 4C 45 52 20 51 45 53 0896 2236 505$: .ASCII 'SEQ REL IDX'
08A1 2237 :
51 51 FC 8F 78 08A1 2238 500$: ASHL #-4, R1, R1 ; GET UPPER NIBBLE AS INDEX
52 EC AF 41 DE 08A6 2239 MOVAL 505$(R1), R2 ; SET ADDRESS OF STRING
51 03 D0 08AB 2240 MOVL #3, R1 ; SET LENGTH OF STRING
05 08AE 2241 RSB
08AF 2242 :
08AF 2243 : RETURN RECORD ATTRIBUTES (RAT)
08AF 2244 :
08AF 2245 : ASSUME FAB$M_FTN EQ 1
08AF 2246 : ASSUME FAB$M_CR EQ 2
08AF 2247 : ASSUME FAB$M_PRN EQ 4
4E 54 46 00' 08AF 2248 605$: .ASCII 'FTN'
03 08AF 2249 :
52 43 00' 08B3 2249 .ASCII 'CR'
02 08B3 2250 :
000008B7 08B6 2250 .BLKB 1 ; PAD TO LONGWORD BOUNDARY
4E 52 50 00' 08B7 2251 .ASCII 'PRN'
03 08B7 2252 :
00' 08BB 2252 .ASCII '' ; USE NULL IF FFS FAILS
00 08BB 2253 :
51 65 03 00 EA 08BC 2254 600$: FFS #0, #3, (R5), R1 ; FIND FIRST BIT SET
52 EA AF 41 DE 08C1 2255 MOVAL 605$(R1), R2 ; SET ADDRESS OF STRING
51 82 9A 08C6 2256 MOVZBL (R2)+, R1 ; SET LENGTH OF STRING
05 08C9 2257 RSB
08CA 2258 :
08CA 2259 : RETURN RECORD FORMAT (RFM)
```

```
20 52 41 56 20 58 49 46 20 46 44 55 08CA 2260 ;
4C 4D 54 53 20 4D 54 53 20 43 46 56 08CA 2261 ; ASSUME FAB$C_UDF EQ 0
20 20 20 52 43 4D 54 53 20 20 20 46 08CA 2262 ; ASSUME FAB$C_FIX EQ 1
05 00 05 03 03 03 03 03 08CA 2263 ; ASSUME FAB$C_VAR EQ 2
51 06 D1 08CA 2264 ; ASSUME FAB$C_VFC EQ 3
03 12 D1 08CA 2265 ; ASSUME FAB$C_STM EQ 4
51 07 D1 08CA 2266 ; ASSUME FAB$C_STMLF EQ 5
52 C8 AF41 DE 08CA 2267 ; ASSUME FAB$C_STMCR EQ 6
51 E7 AF41 9A 08CA 2268 705$: .ASCII 'UDF FIX VAR VFC STM STMLF STMCR '
05 0908 2269 707$: .BYTE 3,3,3,3,3,5,0,5
0909 2270 700$: CMPL #6,R1 ;IS RFM ON 4 BYTE OFFSET?
0909 2271 BNEQ 710$ ;YES, SKIP
0909 2272 MOVL #7,R1 ;NO, SET OFFSET
0909 2273 710$: MOVAL 705$(R1),R2 ;SET ADDRESS OF STRING
0909 2274 MOVZBL 707$(R1),R1 ;SET LENGTH OF STRING
0909 2275 RSB
0909 2276 ;
0909 2277 ; RETURN OWNER UIC
0909 2278 ;
54 51 D0 0909 2279 800$: MOVL R1,R4 ;PASS UIC LONGWORD
0D60 31 090C 2280 BRW FORMAT_UIC ;CONVERT INTO ASCII FORM
090F 2281 ;
090F 2282 ; RETURN FILE PROTECTION
090F 2283 ;
090F 2284 ;
54 51 D0 090F 2285 900$: MOVL R1,R4 ;PASS PROTECTION WORD
0DC0 31 0912 2286 BRW FORMAT_PROT ;CONVERT INTO ASCII FORM
0915 2287 ;
0915 2288 ; RETURN # BLOCKS IN USE
0915 2289 ;
0915 2290 ;
04 A5 B5 0915 2291 1000$: TSTW XAB$W_FFB-XAB$L_EBK(R5) ;FIRST FREE BYTE = 0?
02 12 0918 2292 BNEQ 1010$ ;IF NOT, EBK = BLOCKS IN USE
51 D7 091A 2293 DECL R1 ;ELSE DON'T COUNT LAST BLOCK
52 D4 091C 2294 1010$: CLRL R2 ;INDICATE RESULT IS BINARY
05 091E 2295 RSB
091F 2296 ;
091F 2297 ; RETURN BOOLEAN TRUE/FALSE
091F 2298 ;
0E60 31 091F 2299 1100$: BRW FORMAT_BOOLEAN ;RETURN 'TRUE' OR 'FALSE'
0922 2300 ;
0922 2301 ; RETURN BOOLEAN TRUE/FALSE
0922 2302 ;
0922 2303 ;
51 D4 0922 2304 1200$: CLRL R1 ;ASSUME NOT KNOWN
18 A6 D5 0924 2305 TSTL FAB$L_CTX(R6) ;IS IT?
02 13 0927 2306 BEQL 1210$ ;NO, THEN SKIP
51 D6 0929 2307 INCL R1 ;YES, THEN RETURN TRUE
0E54 31 092B 2308 1210$: BRW FORMAT_BOOLEAN ;RETURN 'TRUE' OR 'FALSE'
```

```
092E 2310 .SBTTL GETDVI SYSTEM SERVICE
092E 2311 :+
092E 2312 : FSGETDVI - GET DEVICE/VOLUME INFORMATION
092E 2313 :
092E 2314 : THIS LEXICAL FUNCTION INVOKES THE $GETDVI SYSTEM SERVICE TO
092E 2315 : OBTAIN ANY PIECE OF INFORMATION FROM A GIVEN DEVICE.
092E 2316 :
092E 2317 : FSGETDVI(DEVNAM,ITEM)
092E 2318 :
092E 2319 : INPUTS:
092E 2320 :
092E 2321 : R2/R3 = DESCRIPTOR OF SCRATCH BUFFER
092E 2322 :
092E 2323 : P1(AP) = DESCRIPTOR OF DEVICE NAME
092E 2324 :
092E 2325 : P2(AP) = DESCRIPTOR OF $DVI ITEM NAME
092E 2326 :
092E 2327 : OUTPUTS:
092E 2328 :
092E 2329 : R1/R2 = DESCRIPTOR OF RESULTANT DATA
092E 2330 :-
092E 2331 :
092E 2332 FSGETDVI:
092E 2333 CLRQ -(SP) ;RESERVE IOSB, WILL BE 8(R5)
7E 7E 7C 092E 2333 MOVL R2,-(SP) ;SAVE LENGTH OF SCRATCH BUFFER
7E 52 D0 0930 2334 CLRL -(SP) ;ALLOCATE A LONGWORD FOR DEVCHAR
7E 7E D4 0933 2335 MOVL SP,R5 ;MAKE R5 POINT TO IT
55 5E D0 0935 2336 CLRL -(SP) ;INDICATE BOTTOM OF ITEM LIST
7E 7E D4 0938 2337 PUSHAB -12(SP) ;ADDRESS OF WORD TO RECEIVE DEVCHAR LENGTH
F4 AE 9F 093A 2338 PUSHL R5 ;ADDRESS OF BUFFER TO RECEIVE DEVCHAR
55 DD 093D 2339 PUSHL #<DVIS_DEVCHAR+1>@16+4 ;SECONDARY DEVICE DEVCHAR ITEM CODE
00030004 8F DD 093F 2340 PUSHAB -12(SP) ;ADDRESS OF WORD TO RECEIVE ITEM LENGTH
F4 AE 9F 0945 2341 MOVQ R2,-(SP) ;SAVE DESCRIPTOR OF SCRATCH ITEM BUFFER
7E 52 7D 0948 2342
0948 2343
0948 2344 :
0948 2345 : At this point, the stack looks like below:
0948 2346 :
0948 2347 :
0948 2348 : entry SP=>
0948 2349 :
0948 2350 :
0948 2351 : --- IOSB ---
0948 2352 :
0948 2353 :
0948 2354 : scratch len: R2
0948 2355 :
0948 2356 : R5 => 0 (holds DEVCHAR)
0948 2357 :
0948 2358 : itmlst end: 0
0948 2359 :
0948 2360 : addr of X (retlen)
0948 2361 :
0948 2362 : R5
0948 2363 :
0948 2364 : X: DVIS_DEVCHAR+1 ; len=4
0948 2365 :
0948 2366 : addr of Y (retlen) :
```

```
094B 2367
094B 2368
094B 2369
094B 2370 ; Y:
094B 2371
094B 2372 ; SP =>
094B 2373
094B 2374
094B 2375
56 000C1683'EF 9E 094B 2376 10$: MOVAB GETDVI_TABLE,R6 ;GET ADDRESS OF TABLE
57 56 D0 0952 2377 MOVBL R6,R7 ;GET ADDRESS OF ENTRY
58 04 C0 0953 2378 ADDL #4,R6 ;GET ADDRESS OF ASCII ITEM NAME
58 66 9A 0958 2379 MOVZBL (R6),R8 ;GET LENGTH OF ITEM NAME
70 13 095B 2380 BEQL 80$ ;BRANCH IF END OF TABLE
00 10 BC 0C AC 2D 095D 2381 CMPC5 P2(AP),@P2+4(AP),#0,- ;DOES ITEM NAME MATCH?
01 A6 58 0963 2382 BEQL 20$ ;BRANCH IF ITEM FOUND
56 58 C0 0966 2383 ADDL R8,R6 ;SKIP TO NEXT ENTRY IN TABLE
56 01 C0 0968 2384 ADDL #1,R6 ;BE SURE TO COUNT THE COUNT
E2 11 096E 2385 BRB 10$ ;LOOP UNTIL FOUND
02 AE 67 B0 0970 2386 20$: MOVW (R7),2(SP) ;SET ITEM CODE INTO LIST
50 5E D0 0974 2387 30$: MOVL SP,R0 ;GET ADDRESS OF ITEM LIST
0977 2388 $GETDVI: DEVNAM=P1(AP),- ;GET INFORMATION ON SPECIFIED DEVICE
0977 2389 ITMLST=(R0),-
0977 2390 IOSB=8(R5),-
0977 2391 EFN=#EXE$C_SYSEFN
45 50 E9 0993 2392 BLBC R0,90$ ;BRANCH IF ERROR DETECTED
50 08 A5 3C 0996 2393 MOVZWL 8(R5),R0 ;GET STATUS FROM IOSB
3E 50 E9 099A 2394 BLBC R0,90$ ;BRANCH IF ERROR DETECTED
54 50 D0 099D 2395 MOVL R0,R4 ;SAVE STATUS
1D 65 06 E1 09A0 2396 BBC #DEV$V_SPL,(R5),50$ ;SPOOLED DEVICE?
56 D4 09A4 2397 CLRL R6 ;SET SPOOLED FLAG
0E AE 02 B0 09A6 2398 MOVW #DVIS_DEVCHAR,14(SP) ;GET DEVCHAR OF PRIMARY DEVICE
02 AE 20 B1 09AA 2399 CMPW #DVIS_DEVNAM,2(SP) ;ARE WE GETTING DEVICE NAME?
08 12 09AE 2400 BNEQ 40$ ;NO, THEN NO SPECIAL PROCESSING
04 A7 09 91 09B0 2401 CMPB #9,4(R7) ;ARE WE GETTING SPOOLED DEVICE NAME?
02 12 09B4 2402 BNEQ 40$ ;NO, THEN NO SPECIAL PROCESSING
BC 11 09B6 2403 BRB 30$ ;YES, THEN DON'T INCR ITEM CODE
6E 04 A5 B0 09B8 2404 40$: MOVW 4(R5),(SP) ;RESET RESULT LENGTH
02 AE B6 09BC 2405 INCL 2(SP) ;GET SECONDARY DEVICE DATA
B3 11 09BF 2406 BRB 30$ ;RETRY
02 AE B4 09C1 2407 50$: CLRW 2(SP) ;CLEAR ITEM CODE, LEAVING RETLEN
51 8E 7D 09C4 2408 MOVQ (SP)+,R1 ;GET DESCRIPTOR OF STRING/LONGWORD
SE 55 10 C1 09C7 2409 ADDL3 #16,R5,SP ;RESTORE STACK
33 11 09CB 2410 BRB GETDVI_VALUE ;GET VALUE
09CD 2411 ;
09CD 2412 ; NO SUCH ITEM NAME
09CD 2413 ;
09CD 2414 80$: STATUS IVKEYW ;UNDEFINED ITEM NAME
SE 55 10 C1 09D4 2415 ADDL3 #16,R5,SP ;CLEAN STACK
OC43 31 09D8 2416 BRW P2_ERROR ;REPORT THE ERROR
09DB 2417 ;
09DB 2418 ; ERROR DETECTED BY GETDVI
09DB 2419 ;
09DB 2420 90$: ADDL3 #16,R5,SP ;CLEAN STACK
50 00000000'8F D1 09DF 2421 CMPL #SS$_IVDEVNAM,R0 ;YES, IS NAME INVALID?
09 13 09E6 2422 BEQL 93$ ;YES, THEN CHECK EXISTS BIT
50 00000000'8F D1 09E8 2423 CMPL #SS$_NOSUCHDEV,R0 ;NO, THEN DOES DEVICE EXIST?
```

```
03 A7 0C 12 09EF 2424 BNEQ 95$ ;NO, THEN DON'T CHECK BIT
      FF 8F 91 09F1 2425 93$: CMPB #DEV$V_EXISTS,3(R7) ;DO WE CARE IF IT EXISTS
      05 12 09F6 2426 BNEQ 95$ ;NO, THEN REPORT ERROR
      51 D4 09F8 2427 CLRL R1 ;YES, THEN RETURN FALSE
      OD85 31 09FA 2428 BRW FORMAT_BOOLEAN ;RETURN THE VALUE
      OC18 31 09FD 2429 95$: BRW P1_ERROR ;REPORT THE ERROR
      OA00 2430 ;
      OA00 2431 ; NORMAL PROCESSING FOR LONGWORDS, STRINGS AND HEX VALUES
      OA00 2432 ;
      OA00 2433 GETDVI_VALUE:
      OA00 2434 CASE 2(R7),TYPE=B,<- ;CASE ON TYPE OF FIELD
      OA00 2435 100$,- ;BINARY
      OA00 2436 200$,- ;ASCII STRING
      OA00 2437 300$,- ;OWNER UIC
      OA00 2438 400$,- ;BOOLEAN
      OA00 2439 500$,- ;HEX STRING
      OA00 2440 600$,- ;PROTECTION MASK
      OA00 2441 700$,- ;ACP TYPE
      OA00 2442 800$> ;HEX STRING
      OA15 2443 ;
      OA15 2444 ;
      OA15 2445 ; RETURN BINARY VALUE
      OA15 2446 ;
      51 62 D0 OA15 2447 100$: MOVL (R2),R1 ;GET LONGWORD VALUE
      52 D4 OA18 2448 CLRL R2 ;INDICATE RESULT IS BINARY
      05 OA1A 2449 RSB
      OA1B 2450 ;
      OA1B 2451 ;
      OA1B 2452 ; RETURN ASCII STRING
      OA1B 2453 ;
      02 A7 20 B1 OA1B 2454 200$: CMPW #DVIS_DEVNAM,2(R7) ;ARE WE GETTING DEVICE NAME?
      0C 12 OA1F 2455 BNEQ 210$ ;NO, THEN NO SPECIAL PROCESSING
      04 A7 09 91 OA21 2456 CMPB #9,4(R7) ;ARE WE GETTING SPOOLED DEVICE NAME?
      06 12 OA25 2457 BNEQ 210$ ;NO, THEN NO SPECIAL PROCESSING
      56 D5 OA27 2458 TSTL R6 ;IS DEVICE SPOOLED?
      02 13 OA29 2459 BEQL 210$ ;YES, THEN USE FETCHED NAME
      51 D4 OA2B 2460 CLRL R1 ;NO, THEN RETURN NULL STRING
      05 OA2D 2461 210$: RSB
      OA2E 2462 ;
      OA2E 2463 ;
      OA2E 2464 ; RETURN OWNER UIC
      OA2E 2465 ;
      54 62 D0 OA2E 2466 300$: MOVL (R2),R4 ;PASS UIC LONGWORD
      OC3B 31 OA31 2467 BRW FORMAT_UIC ;CONVERT INTO ASCII FORM
      OA34 2468 ;
      OA34 2469 ;
      OA34 2470 ; RETURN BOOLEAN TRUE/FALSE
      OA34 2471 ;
      51 62 9A OA34 2472 400$: MOVZBL (R2),R1 ; FETCH THE RETURNED VALUE
      03 A7 FF 8F 91 OA37 2473 CMPB #DEV$V_EXISTS,3(R7) ; ARE WE CHECKING FOR DEVICE EXISTS?
      03 12 OA3C 2474 BNEQ 410$ ; NEQ MEANS WE AREN'T
      51 01 D0 OA3E 2475 MOVL #1,R1 ; IT ALWAYS EXISTS (AT THIS POINT)
      OD3E 31 OA41 2476 410$: BRW FORMAT_BOOLEAN ; RETURN "TRUE" OR "FALSE"
      OA44 2477 ;
      OA44 2478 ;
      OA44 2479 ;
      OA44 2480 ; RETURN HEX STRING
```

```

      50 62 D0 0A44 2481 :
      E4 13 0A44 2482 500$: MOVL (R2),R0 ;PASS HEX LONGWORD AS ARGUMENT
00000000'EF 17 0A47 2483 BEQL 210$ ;IF ZERO, RETURN NULL STRING
      0A49 2484 JMP DCL$CBTA_HEX ;CONVERT HEX TO STRING AND EXIT
      0A4F 2485
      0A4F 2486 :
      0A4F 2487 : RETURN PROTECTION STRING
      0A4F 2488 :
      54 62 D0 0A4F 2489 600$: MOVL (R2),R4 ;PASS PROTECTION MASK
      0C80 31 0A52 2490 BRW FORMAT_PROT ;CONVERT INTO ASCII FORM
      0A55 2491
      0A55 2492 :
      0A55 2493 : RETURN ACP TYPE
      0A55 2494 :
      52 50 62 D0 0A55 2495 700$: MOVL (R2),R0 ;GET ACP TYPE NUMBER
      00001E5E'EF40 7E 0A58 2496 MOVAQ ACP_TABLE[R0],R2 ;GET ADDRESS OF ASCII ACP TYPE
      51 82 9A 0A60 2497 MOVZBL (R2)+,R1 ;SET DESCRIPTOR OF ACP TYPE
      05 0A63 2498 RSB
      0A64 2499
      0A64 2500 :
      0A64 2501 : RETURN HEX STRING
      0A64 2502 :
      51 D5 0A64 2503 800$: TSTL R1 ;IS LENGTH ZERO?
      06 13 0A66 2504 BEQL 810$ ;YES, RETURN NULL STRING
00000000'EF 17 0A68 2505 JMP DCL$CBTA_HEXSTR ;CONVERT HEX TO STRING AND EXIT
      05 0A6E 2506 810$: RSB ;
```

0A6F	2508	.SBTTL GETJPI SYSTEM SERVICE								
0A6F	2509	:*								
0A6F	2510	F\$GETJPI - GET JOB/PROCESS INFORMATION								
0A6F	2511									
0A6F	2512	THIS LEXICAL FUNCTION INVOKES THE \$GETJPI SYSTEM SERVICE TO								
0A6F	2513	OBTAIN ANY PIECE OF INFORMATION FROM A GIVEN PROCESS.								
0A6F	2514									
0A6F	2515	F\$GETJPI(PID,ITEM)								
0A6F	2516									
0A6F	2517	INPUTS:								
0A6F	2518									
0A6F	2519	P1(AP) = DESCRIPTOR OF PID STRING OF DESIRED PROCESS,								
0A6F	2520	NULL OR 0 FOR CURRENT PROCESS								
0A6F	2521	P2(AP) = DESCRIPTOR OF \$JPI ITEM NAME								
0A6F	2522									
0A6F	2523	OUTPUTS:								
0A6F	2524									
0A6F	2525	R1/R2 = DESCRIPTOR OF RESULTANT DATA								
0A6F	2526	:-								
0A6F	2527									
0A6F	2528	F\$GETJPI:								
56	00000100	30	0A6F	2529	BSBW	DCL\$MARK	:MARK POSITION IN EXPANSION BUFFER			
	7E	7C	0A72	2530	CLRQ	-(SP)	:RESERVE IOSB			
	7E	7C	0A74	2531	CLRQ	-(SP)	:CLEAR LENGTH AND ITEM ID OF LAST ITEM			
			0A76	2532			:AND ALLOCATE SPACE FOR PID LONGWORD			
	F4	AE	9F	0A76	2533	PUSHAB	-12(SP)	:ADDRESS OF WORD TO RECEIVE LENGTH		
	7E	52	7D	0A79	2534	MOVQ	R2,-(SP)	:SAVE DESCRIPTOR OF SCRATCH BUFFER		
	00000100	'EF	9E	0A7C	2535	MOVAB	GETJPI_TABLE,R6	:GET ADDRESS OF TABLE		
	57	86	D0	0A83	2536	10\$:	MOVL	(R6)+,R7	:GET JPI CODE(0:15),FLAGS(16:23),	
				0A86	2537				:ITEM STRING LENGTH(24:31)	
		03	12	0A86	2538	BNEQ	15\$		:SKIP IF NOT AT END OF TABLE	
		0092	31	0A88	2539	BRW	80\$		:ERROR IF END OF TABLE	
00	58	FF	A6	9A	0A8B	2540	15\$:	MOVZBL	-1(R6),R8	:GET LENGTH OF ITEM NAME
10	BC	0C	AC	2D	0A8F	2541		CMPC5	P2(AP),@P2+4(AP),#0,-	:DOES ITEM NAME MATCH?
	66	58			0A95	2542			R8,(R6)	:
		05	13	0A97	2543	BEQL	20\$			:BRANCH IF ITEM FOUND
	56	58	C0	0A99	2544	ADDL	R8,R6			:SKIP TO NEXT ENTRY IN TABLE
		E5	11	0A9C	2545	BRB	10\$			:LOOP UNTIL FOUND
	02	AE	57	B0	0A9E	2546	20\$:	MOVW	R7,2(SP)	:SET ITEM CODE INTO LIST
	52	04	AC	7D	0AA2	2547		MOVQ	P1(AP),R2	:GET DESCRIPTOR OF PID STRING
		51	00	9A	0AA6	2548		MOVZBL	#PRC_K_HEX,R1	:SET RADIX FOR CONVERSION
		F554	'	30	0AA9	2549		BSBW	DCL\$CNVASCBIN	:CONVERT HEX STRING TO BINARY IN R1
		62	12	0AAC	2550	BNEQ	70\$			:BRANCH IF CONVERSION ERROR
	10	AE	51	D0	0AAE	2551		MOVL	R1,16(SP)	:STORE PID LONGWORD
	50	5E	D0	0AB2	2552	MOVL	SP,R0			:GET ADDRESS OF ITEM LIST
				0AB5	2553			\$GETJPIW	S PIDADR=16(R0),-	:GET INFORMATION ON SPECIFIED PROCESS
				0AB5	2554				ITMLST=(R0),-	:
				0AB5	2555				IOSB=20(R0),-	:
				0AB5	2556				EFN=#EXESC_SYSEFN	:
	50	53	50	E9	0ACE	2557		BLBC	R0,90\$	:BRANCH IF ERROR DETECTED
		14	AE	3C	0AD1	2558		MOVZWL	20(SP),R0	:GET STATUS FROM IOSB
		4C	50	E9	0AD5	2559		BLBC	R0,90\$	:BRANCH IF ERROR DETECTED
		02	AE	B4	0AD8	2560		CLRW	2(SP)	:CLEAR ITEM CODE, LEAVING RETLEN
			06	BA	0ADB	2561		POPR	#*M<R1,R2>	:GET DESCRIPTOR OF STRING/LONGWORD
										:AND CLEAN STACK
	5E	14	C0	0ADD	2562	ADDL	#5*4,SP			:ASKING FOR MODE?
0322	8F	57	B1	0AE0	2563	CMPL	R7,#JPI\$_MODE			:IF NOT, THEN SKIP
		03	12	0AE5	2564	BNEQ	30\$			

```
0206 8F 00E0 31 0AE7 2565 BRW STS MODE ;IF SO, CHECK IF USER WANTED 'MODE'
      57 B1 0AEA 2566 30$: CMPW R7,#JPI$_LOGINTIM ;ASKING FOR LOGIN TIME?
      03 12 0AEF 2567 BNEQ 40$ ;IF NOT, THEN SKIP
0306 8F 00A6 31 0AF1 2568 BRW GETJPI LOGINTIM ;IF SO, FORMAT AS DATE/TIME
      57 B1 0AF4 2569 40$: CMPW R7,#JPI$_STATE ;ASKING FOR STATE?
      03 12 0AF9 2570 BNEQ 50$ ;IF NOT, THEN SKIP
0304 8F 00B6 31 0AFB 2571 BRW GETJPI STATE ;IF SO, RETURN STATE KEYWORD
      57 B1 0AFE 2572 50$: CMPW R7,#JPI$_UIC ;ASKING FOR UIC?
      03 12 0B03 2573 BNEQ 60$ ;IF NOT, THEN SKIP
57 57 08 00B8 31 0B05 2574 BRW GETJPI_UIC ;IF SO, FORMAT UIC SPECIALLY
      10 EF 0B08 2575 60$: EXTZV #16,#8,R7,R7 ;EXTRACT ITEM TYPE FIELD
      002B 31 0B0D 2576 BRW GETJPI_VALUE ;GET VALUE
      0B10 2577 :
      0B10 2578 : ERROR CONVERTING PID STRING TO A BINARY LONGWORD
      0B10 2579 :
      0B10 2580 70$: STATUS EXPSYN ;EXPRESSION SYNTAX ERROR
      5E 1C C0 0B17 2581 ADDL #7*4,SP ;CLEAN STACK
      0AFB 31 0B1A 2582 75$: BRW P1_ERROR ;REPORT ERROR
      0B1D 2583 :
      0B1D 2584 : NO SUCH ITEM NAME
      0B1D 2585 :
      0B1D 2586 80$: STATUS IVKEYW ;UNDEFINED ITEM NAME
      0B24 2587 :
      0B24 2588 : ERROR DETECTED BY GETJPI
      0B24 2589 :
0000'8F 5E 1C C0 0B24 2590 90$: ADDL #7*4,SP ;CLEAN STACK
      50 B1 0B27 2591 CMPW R0,#SS$_SUSPENDED ;SUSPENDED ERROR?
      0A 13 0B2C 2592 BEQL 95$ ;RETURN NULL/ZERO IF SUSPENDED
0000'8F 50 B1 0B2E 2593 CMPW R0,#SS$_NONEXPR ;NONEXISTANT PROCESS?
      E5 13 0B33 2594 BEQL 75$ ;REPORT P1 ERROR
      0AE6 31 0B35 2595 BRW P2_ERROR ;REPORT ERROR
      51 D4 0B38 2596 95$: CLRL R1 ;RETURN NULL STRING
      05 0B3A 2597 RSB
      0B3B 2598 :
      0B3B 2599 : NORMAL PROCESSING FOR LONGWORDS, STRINGS AND HEX VALUES
      0B3B 2600 :
      0B3B 2601 GETJPI_VALUE:
      0B3B 2602 CASE R7,TYPE=B,<- ;CASE ON ITEM TYPE
      0B3B 2603 10$,- ;BINARY
      0B3B 2604 20$,- ;STRING
      0B3B 2605 30$,- ;HEX LONGWORD STRING
      0B3B 2606 40$,- ;PRIVILEGE LIST
      0B3B 2607 60$,- ;BITVAL
      0B3B 2608 70$> ;HEX STRING
      0B4B 2609 :
      0B4B 2610 : RETURN BINARY VALUE
      0B4B 2611 :
      51 62 D0 0B4B 2612 10$: MOVL (R2),R1 ;GET LONGWORD VALUE
      52 D4 0B4E 2613 CLRL R2 ;INDICATE RESULT IS BINARY
      05 0B50 2614 20$: RSB
      0B51 2615 :
      0B51 2616 : RETURN HEX STRING
      0B51 2617 :
      0B51 2618 :
      50 51 D4 0B51 2619 30$: CLRL R1 ;ASSUME RETURNING NULL STRING
      62 D0 0B53 2620 MOVL (R2),R0 ;PASS LONGWORD PID AS ARGUMENT
      F8 13 0B56 2621 BEQL 20$ ;IF ZERO, RETURN NULL STRING
```

```
00000000'EF 17 0B58 2622 JMP DCL$CBTA_HEX ;CONVERT PID TO STRING AND EXIT
0B5E 2623
0B5E 2624
0B5E 2625 ; RETURN PRIVILEGE LIST
0B5E 2626
7E 62 7D 0B5E 2627 40$: MOVQ (R2),-(SP) ;PUSH PRIVILEGE MASK ON STACK
56 5E D0 0B61 2628 MOVL SP,R6 ;GET ADDRESS OF MASK
03 66 51 D4 0B64 2629 CLRL R1 ;START WITH PRIV #0
0959 30 0B66 2630 45$: BBC R1,(R6),50$ ;SKIP IF PRIV IS NOT PRESENT
F1 51 00000040 8F F2 0B6D 2631 50$: BSBW ADD_PRIV_STATE ;APPEND EACH KEYWORD TO STRING
F488' 30 0B75 2632 AOBLS #64-R1,45$ ;LOOP FOR EACH PRIV IN MASK
F48A CA D0 0B78 2633 BSBW DCL$MARKEDTOKEN ;GET DESCRIPTOR OF STRING
F486 CA D0 0B7C 2634 MOVL WRK_L_MARKPTR(R10),- ;RESTORE POSITION IN EXPANSION BUFFER
51 D5 0B7F 2635 WRK_L_EXPANDPTR(R10)
02 13 0B81 2636 TSTL R1 ;WERE NO PRIVS FOUND?
51 D7 0B83 2637 BEQL 55$ ;YES, THEN NO TRAILING COMMA
5E 08 C0 0B85 2638 DECL R1 ;REMOVE TRAILING COMMA
05 0B88 2639 55$: ADDL #8,SP ;RESTORE STACK
0B89 2640
0B89 2641
0B89 2642 ; RETURN BOOLEAN
0B89 2643
0B89 2644
51 62 9A 0B89 2645 60$: MOVZBL (R2),R1 ;FETCH THE VALUE
0BF3 31 0B8C 2646 BRW FORMAT_BOOLEAN ;RETURN "TRUE" OR "FALSE"
0B8F 2647
0B8F 2648 ; RETURN HEX STRING
0B8F 2649
51 D5 0B8F 2650 70$: TSTL R1 ;IS LENGTH ZERO?
06 13 0B91 2651 BEQL 75$ ;YES, RETURN NULL STRING
00000000'EF 17 0B93 2652 JMP DCL$CBTA_HEXSTR ;CONVERT HEX TO STRING AND EXIT
05 0B99 2653 75$: RSB
0B9A 2654
0B9A 2655
0B9A 2656 ; SPECIAL PROCESSING FOR "LOGINTIM"
0B9A 2657
0B9A 2658
0B9A 2659 GETJPI_LOGINTIM:
52 DD 0B9A 2660 PUSHL R2 ;ADDRESS OF RESULT BUFFER
17 DD 0B9C 2661 PUSHL #23 ;RESULT BUFFER LENGTH
50 5E D0 0B9E 2662 MOVL SP,R0
0BA1 2663 $ASCTIM_S TIMBUF=(R0),-
0BA1 2664 TIMADR=(R2)
51 8E 7D 0BB0 2665 MOVQ (SP)+,R1 ;PICK UP RESULTANT STRING DESCRIPTOR
05 0BB3 2666 RSB
0BB4 2667
0BB4 2668 ; SPECIAL PROCESSING FOR "STATE"
0BB4 2669
0BB4 2670
0BB4 2671 GETJPI_STATE:
50 62 D0 0BB4 2672 MOVL (R2),R0 ;GET STATE NUMBER
52 00001491'EF40 7E 0BB7 2673 MOVQ STATE_TABLE[R0],R2 ;GET ADDRESS OF ASCII STATE NAME
51 82 9A 0BBF 2674 MOVZBL (R2)+,R1 ;SET DESCRIPTOR OF STATE NAME
05 0BC2 2675 RSB
0BC3 2676
0BC3 2677
0BC3 2678 ; SPECIAL PROCESSING FOR "UIC"
```

```

      OBC3 2679 :
      OBC3 2680 GETJPI_UIC:
54    62    D0 OBC3 2681      MOVL      (R2),R4      ;GET UIC LONGWORD
      OAA6 30 OBC6 2682      BSBW      FORMAT_UIC    ;FORMAT UIC INTO ASCII STRING
      05 OBC9 2683      RSB
      OBCA 2684
      OBCA 2685 :
      OBCA 2686 : WE JUST GOT JPIS_STS - CHECK IF USER WANTS 'MODE'
      OBCA 2687 :
      OBCA 2688 STS_MODE:
      54    62    D0 OBCA 2689      MOVL      (R2),R4      ;GET MODE LONGWORD
      53    D4 OBCD 2690      CLRL      R3              ;SET LOOP INDEX
52    000021F3'EF 9E OBCF 2691      MOVAB     MODE_TABLE,R2    ;GET ADDRESS OF MODE TABLE
      51    82    9A OBD6 2692 25$:  MOVZBL   (R2)+,R1        ;GET LENGTH OF MODE
      0E    13 OBD9 2693      BEQL      40$             ;BRANCH IF END OF TABLE
      53    54    D1 OBD8 2694      CMPL     R4,R3          ;IS THIS THE ONE WE WANT?
      09    13 OBDE 2695      BEQL      40$             ;YES, THEN RETURN THE STRING
52    01 A142 9E OBE0 2696      MOVAB     1(R1)(R2),R2    ;SKIP TO NEXT TABLE ENTRY
      53    D6 OBE5 2697      INCL      R3              ;INCR LOOP INDEX
      ED    11 OBE7 2698      BRB       25$
      05 OBE9 2699 40$:  RSB
                        :
```

```

OBEA 2701      .SBTTL  GETSYI SYSTEM SERVICE
OBEA 2702      :+
OBEA 2703      : F$GETSYI - GET SYSTEM INFORMATION
OBEA 2704      :
OBEA 2705      : THIS LEXICAL FUNCTION INVOKES THE $GETSYI SYSTEM SERVICE TO
OBEA 2706      : OBTAIN ANY PIECE OF INFORMATION FOR THE CURRENT SYSTEM.
OBEA 2707      :
OBEA 2708      : F$GETSYI(ITEM,NODE)
OBEA 2709      :
OBEA 2710      : INPUTS:
OBEA 2711      :
OBEA 2712      : P1(AP) = DESCRIPTOR OF $SYI ITEM NAME
OBEA 2713      : P2(AP) = DESCRIPTOR OF NODE NAME OF DESIRED SYSTEM,
OBEA 2714      : NULL OR 0 FOR CURRENT SYSTEM
OBEA 2715      :
OBEA 2716      : OUTPUTS:
OBEA 2717      :
OBEA 2718      : R1/R2 = DESCRIPTOR OF RESULTANT DATA
OBEA 2719      : -
OBEA 2720      :
OBEA 2721      : F$GETSYI:
OBEA 2722      : CLRQ      -(SP)      ;RESERVE IOSB
OBEA 2723      : CLRQ      -(SP)      ;RESERVE CSID AND CLEAR LENGTH AND ITEM ID 0
OBEA 2724      : PUSHAB    -12(SP)      ;ADDRESS OF WORD TO RECEIVE LENGTH
OBEA 2725      : MOVQ      R2,-(SP)      ;SAVE DESCRIPTOR OF SCRATCH BUFFER
OBEA 2726      : MOVAB     GETSYI TABLE,R6 ;GET ADDRESS OF TABLE
OBEA 2727      : MOVL      (R6)+,R7      ;GET SYI CODE(0:15),TYPE(16:23),
OBEA 2728      : 10$:      ; ITEM STRING LENGTH(24:31)
OBEA 2729      : BNEQ      15$      ;BRANCH IF NOT END OF TABLE
OBEA 2730      : BRW       80$      ;BRANCH IF END OF TABLE
OBEA 2731      : MOVZBL    -1(R6),R8      ;GET LENGTH OF ITEM NAME
OBEA 2732      : CMPC5     P1(AP),@P1+4(AP),#0,- ;DOES ITEM NAME MATCH?
OBEA 2733      : R8,(R6)      ;
OBEA 2734      : BEQL      20$      ;BRANCH IF ITEM FOUND
OBEA 2735      : ADDL      R8,R6      ;SKIP TO NEXT ENTRY IN TABLE
OBEA 2736      : BRB       10$      ;LOOP UNTIL FOUND
OBEA 2737      : MOVW      R7,2(SP)      ;SET ITEM CODE INTO LIST
OBEA 2738      : TSTL      P2(AP)      ;P2 SPECIFIED?
OBEA 2739      : BLSS      25$      ;BRANCH IF NOT
OBEA 2740      : MOVAQ     P2(AP),16(SP) ;GET ADDRESS OF NODE DESCRIPTOR
OBEA 2741      : MOVL      SP,R0      ;GET ADDRESS OF ITEM LIST
OBEA 2742      : 25$:      $GETSYIW S ITMLST=(R0),- ;GET INFORMATION
OBEA 2743      :      NODENAME=@16(R0),-
OBEA 2744      :      IOSB=20(R0),-
OBEA 2745      :      EFN=#EXESC_ $YSEFN
OBEA 2746      : BLBC      R0,90$      ;BRANCH IF ERROR DETECTED
OBEA 2747      : MOVZWL    20(SP),R0      ;GET STATUS FROM IOSB
OBEA 2748      : BLBC      R0,90$      ;BRANCH IF ERROR DETECTED
OBEA 2749      : CLRW      2(SP)      ;CLEAR ITEM CODE, LEAVING RETLEN
OBEA 2750      : POPR      #^M<R1,R2> ;GET DESCRIPTOR OF STRING/LONGWORD
OBEA 2751      : ADDL      #5*4,SP ;AND CLEAN STACK
OBEA 2752      :
OBEA 2753      : EXTZV     #16,#8,R7,R7 ;GET ITEM TYPE
OBEA 2754      :
OBEA 2755      : CASE      R7,TYPE=B,<- ;CASE ON ITEM TYPE
OBEA 2756      : 91$,-      ;BINARY
OBEA 2757      : 40$,-      ;STRING

```

```

          OC57 2758          30$,-          :HEX LONGWORD STRING
          OC57 2759          91$,-          :PRIVILEGE LIST
          OC57 2760          96$,-          :BITVAL
          OC57 2761          30$,-          :HEX STRING
          OC57 2762          100$>         :STDIM
          OC69 2763
51 62 9A OC69 2764 96$: MOVZBL (R2),R1      :FETCH THE VALUE
          OB13 31 OC6C 2765          BRW    FORMAT_BOOLEAN :RETURN 'TRUE' OR 'FALSE'
51 62 D0 OC6F 2766 91$: MOVL (R2),RT      :GET LONGWORD VALUE
          52 D4 OC72 2767          CLRL    R2              :INDICATE RESULT IS BINARY
          OA 11 OC74 2768          BRB     40$             :
          51 D5 OC76 2769 30$: TSTL R1      :IS LENGTH ZERO?
          06 13 OC78 2770          BEQL    40$             :YES, RETURN NULL STRING
00000000'EF 17 OC7A 2771          JMP     DCL$CBTA_HEXSTR   :CONVERT HEX TO STRING AND EXIT
          05 OC80 2772 40$: RSB              :RETURN
          OC81 2773
          OC81 2774 100$: SUBL3 WRK_L_EXPANDPTR(R10),-      :CALCULATE SIZE OF BUFFER
50 F486 CA C3 OC85 2775          WRK_L_CHARPTR(R10),R0    :
          F48E CA D1 OC89 2776          CMPL    R0,#23      :IS BUFFER LARGE ENOUGH?
          17 50 1F OC8C 2777          BLSSU    110$         :NO, THEN ERROR
          F486 CA DD OC8E 2778          PUSHL   WRK_L_EXPANDPTR(R10) :CREATE DESCRIPTOR OF SCRATCH BUFFER
          50 DD OC92 2779          PUSHL    R0              :
          50 5E D0 OC94 2780          MOVL     SP,R0        :GET ADDRESS OF BUFFER DESCRIPTOR
          OC97 2781          $ASCTIM_S TIMLEN=(R0),-      :CONVERT QUADWORD TIME TO ASCII
          OC97 2782          TIMBUF=(R0),-
          OC97 2783          TIMADR=(R2),-
          51 8E 7D OCA6 2784          MOVQ    (SP)+,R1      :
          05 OCA9 2785          RSB              :
          OCAA 2786
          095F 31 OCAA 2787 110$: BRW    BUFOVF           :SIGNAL BUFFER OVERFLOW ERROR
          OCAD 2788
          OCAD 2789 : NO SUCH ITEM NAME
          OCAD 2790 :
          OCAD 2791 :
          OCAD 2792 80$: STATUS IVKEYW          :UNDEFINED ITEM NAME
          OCB4 2793
          OCB4 2794 :
          OCB4 2795 : ERROR DETECTED BY GETSYI
          OCB4 2796 :
          5E 1C C0 OCB4 2797 90$: ADDL #7*4,SP          :CLEAN STACK
50 00000000'8F D1 OCB7 2798          CMPL    #SS$_NOSUCHNODE,R0 :CHECK FOR NON-EXISTENT NODE
          OC 12 OCBE 2799          BNEQ     92$            :NEQ MEANS IT WASN'T THAT ERROR
          57 10CF 8F B1 OCC0 2800          CMPW   #SYIS_CLUSTER_MEMBER,R7 :ARE WE CHECKING FOR EXISTENCE?
          08 12 JCC5 2801          BNEQ     93$            :NEQ MEANS WE ARE NOT
          51 D4 OCC7 2802          CLRL    R1              :SET 'FALSE'
          OAB6 31 OCC9 2803          BRW     FORMAT_BOOLEAN :RETURN 'FALSE' TO THE USER
          0949 31 OCCC 2804 92$: BRW     P1_ERROR        :REPORT ERROR
          094C 31 OCCF 2805 93$: BRW     P2_ERROR        :REPORT ERROR
```

```

OCD2 2807 .SBTTL CONVERT IDENTIFIER
OCD2 2808 :+
OCD2 2809 : FSIDENTIFIER - CONVERT IDENTIFIER
OCD2 2810 :
OCD2 2811 : THIS FUNCTION CONVERTS IDENTIFIERS BETWEEN STRINGS AND NUMBERS.
OCD2 2812 :
OCD2 2813 : FSIDENTIFIER (IDENTIFIER, DIRECTION)
OCD2 2814 :
OCD2 2815 : INPUTS:
OCD2 2816 :
OCD2 2817 : P1(AP) = IDENTIFIER NAME OR NUMBER
OCD2 2818 : P2(AP) = 'NUMBER' OR 'NAME'
OCD2 2819 :
OCD2 2820 : OUTPUTS:
OCD2 2821 :
OCD2 2822 : R1/R2 = DESCRIPTOR OF RESULTANT IDENTIFIER
OCD2 2823 :-
OCD2 2824 :
OCD2 2825 FSIDENTIFIER:
56 000021D2'EF 9E OCD2 2826 MOVAB IDENT_TABLE,R6 ;GET ADDRESS OF TABLE
57 86 9A OCD2 2827 10$: MOVZBL (R6)+,R7 ;GET LENGTH OF ITEM NAME
5F 13 OCD2 2828 BEQL 95$ ;ERROR IF ZERO
00 10 BC 0C AC 2D OCD2 2829 CMPC5 P2(AP),@P2+4(AP),#0,- ;DOES ITEM NAME MATCH?
66 57 2D OCD2 2830 R7,(R6) ;
07 13 OCD2 2831 BEQL 20$ ;BRANCH IF ITEM FOUND
56 57 C0 OCD2 2832 ADDL R7,R6 ;SKIP TO NEXT ENTRY IN TABLE
56 56 D6 OCD2 2833 INCL R6 ;
EA 11 OCD2 2834 BRB 10$ ;LOOP UNTIL FOUND
6647 95 OCD2 2835 20$: TSTB (R6)[R7] ;'NAME TO NUMBER'?
2A 13 OCD2 2836 BEQL 50$ ;YES, THEN BRANCH
OCD2 2837
OCD2 2838 PUSHL WRK_L_EXPANDPTR(R10) ;CREATE DESCRIPTOR OF SCRATCH BUFFER
7E F486 CA DD OCD2 2839 SUBL3 WRK_L_EXPANDPTR(R10),-
F486 CA C3 OCD2 2840 WRK_L_CHARPTR(R10),-(SP) ;
F48E CA D0 OCD2 2841 MOVL SP,R0 ;
50 5E D0 OCD2 2842 $IDTOASC S ID=P1(AP),- ;CONVERT NUMBER TO NAME
OCD2 2843 NAMLEN=(R0),-
OCD2 2844 NAMBUF=(R0) ;
51 8E 7D OCD2 2845 MOVQ (SP)+,R1 ;GET RESULT DESCRIPTOR
1D 50 E9 OCD2 2846 BLBC R0,90$ ;BRANCH IF ERROR
OCD2 2847 RSB ;RETURN
OCD2 2848
OCD2 2849 50$: CLRL -(SP) ;ALLOCATE A RETURN BUFFER
50 5E D0 OCD2 2850 MOVL SP,R0 ;
OCD2 2851 $ASCTOID S NAME=P1(AP),- ;CONVERT NAME TO NUMBER
OCD2 2852 ID=(R0) ;
51 8E D0 OCD2 2853 MOVL (SP)+,R1 ;GET RESULT
52 D4 OCD2 2854 CLRL R2 ;
01 50 E9 OCD2 2855 BLBC R0,90$ ;BRANCH IF ERROR
OCD2 2856 RSB ;RETURN
OCD2 2857
OCD2 2858
OCD2 2859 90$: CLRL R1 ;RETURN THE NULL STRING
OCD2 2860 RSB ;
OCD2 2861 95$: STATUS IVKEYW ;REPORT KEYWORD ERROR
08D7 31 OCD2 2862 BRW P2_ERROR ;

```

```
OD47 2864 .SBTTL CONVERT TO INTEGER
OD47 2865 :+
OD47 2866 : F$INTEGER - CONVERT TO INTEGER
OD47 2867 :
OD47 2868 : THIS FUNCTION CONVERTS ANY VALUE TO AN INTEGER
OD47 2869 :
OD47 2870 : F$INTEGER(EXPRESSION)
OD47 2871 :
OD47 2872 : INPUTS:
OD47 2873 :
OD47 2874 : P1(AP) = EXPRESSION RESULT
OD47 2875 :
OD47 2876 : OUTPUTS:
OD47 2877 :
OD47 2878 : R1 = INTEGER
OD47 2879 : R2 = 0 (TO INDICATE RESULT IS BINARY RATHER THAN ASCII)
OD47 2880 :-
OD47 2881 :
OD47 2882 F$INTEGER:
51 04 AC 7D OD47 2883 MOVQ P1(AP),R1 ;GET EXPRESSION VALUE
00000000'EF 16 OD48 2884 JSB DCL$CVT_BINARY ;CONVERT STRING TO BINARY
01 50 E9 OD51 2885 BLBC R0,90$ ;BRANCH IF ERROR
05 OD54 2886 RSB
08C0 31 OD55 2887 90$: BRW P1_ERROR ;REPORT ERROR AND DISPLAY P1
```

```
0D58 2889 .SBTTL RETURN LENGTH OF STRING
0D58 2890 :+
0D58 2891 : F$LENGTH - RETURN LENGTH OF STRING
0D58 2892 :
0D58 2893 : THIS LEXICAL FUNCTION RETURNS THE LENGTH OF ITS ARGUMENT AS A DECIMAL STRING.
0D58 2894 :
0D58 2895 : F$LENGTH(String)
0D58 2896 :
0D58 2897 : INPUTS:
0D58 2898 :
0D58 2899 : P1(AP) = DESCRIPTOR OF STRING.
0D58 2900 :
0D58 2901 : OUTPUTS:
0D58 2902 :
0D58 2903 : R1 = STRING LENGTH
0D58 2904 : R2 = 0 TO INDICATE RESULT IS BINARY
0D58 2905 :-
0D58 2906
0D58 2907 F$LENGTH:
51 04 AC D0 0D58 2908 MOVL P1(AP),R1 ;RETURN LENGTH OF STRING
52 D4 0D5C 2909 CLRL R2 ;GET VALUE TO BE CONVERTED
05 0D5E 2910 RSB ;INDICATE BINARY RESULT
```

```
0D5F 2912 .SBTTL LOCATE SUBSTRING IN STRING
0D5F 2913 :+
0D5F 2914 : FSLOCATE - LOCATE SUBSTRING IN STRING
0D5F 2915 :
0D5F 2916 : THIS LEXICAL FUNCTION RETURNS THE STARTING POSITION OF A SUBSTRING IN ANOTHER
0D5F 2917 : STRING. IF THE SUBSTRING IS NOT CONTAINED IN THE SPECIFIED STRING, THEN THE
0D5F 2918 : LENGTH OF THE SPECIFIED STRING IS RETURNED.
0D5F 2919 :
0D5F 2920 : FSLOCATE(SUBSTRING, STRING)
0D5F 2921 :
0D5F 2922 : INPUTS:
0D5F 2923 :
0D5F 2924 : P1(AP) = DESCRIPTOR OF SUBSTRING.
0D5F 2925 : P2(AP) = DESCRIPTOR OF STRING TO BE SEARCHED.
0D5F 2926 :
0D5F 2927 : OUTPUTS:
0D5F 2928 :
0D5F 2929 : R1 = NUMBER INDICATING POSITION WITHIN STRING
0D5F 2930 : R2 = 0 TO INDICATE RESULT IS BINARY
0D5F 2931 :-
0D5F 2932
0D5F 2933 FSLOCATE:
0D5F 2934 MATCHC P1(AP),@P1+4(AP),P2(AP),@P2+4(AP) ;LOCATE SUBSTRING IN STRING
0D5F 2935 BNEQ 10$ ;SEARCH FOR SUBSTRING MATCH
0D5F 2936 SUBL P1(AP),R3 ;IF NEQ NO MATCH FOUND
0D5F 2937 10$: SUBL3 P2+4(AP),R3,R1 ;CALCULATE STARTING ADDRESS OF SUBSTRING
0D5F 2938 CLRL R2 ;CALCULATE STARTING POSITION OF SUBSTRING
0D5F 2939 RSB ;MARK RESULT IS BINARY
```

10 BC	0C AC	08 BC	04 AC	39	0D5F 2934
			04	12	0D68 2935
		53	04 AC	C2	0D6A 2936
	51	53	10 AC	C3	0D6E 2937
			52	D4	0D73 2938
				05	0D75 2939

```
0D76 2941 .SBTTL LOGICAL NAME TRANSLATION
0D76 2942 :+
0D76 2943 : FSLOGICAL - LOGICAL NAME TRANSLATION
0D76 2944 :
0D76 2945 : THIS LEXICAL FUNCTION OBTAINS THE TRANSLATION FOR A SPECIFIED LOGICAL NAME AND
0D76 2946 : RETURN THE EQUIVALENCE NAME.
0D76 2947 :
0D76 2948 : FSLOGICAL(LOGICAL_NAME)
0D76 2949 :
0D76 2950 : INPUTS:
0D76 2951 :
0D76 2952 : P1(AP) = DESCRIPTOR OF LOGICAL NAME STRING.
0D76 2953 :
0D76 2954 : OUTPUTS:
0D76 2955 :
0D76 2956 : R1 = LENGTH OF EQUIVALENCE STRING.
0D76 2957 : R2 = ADDRESS OF EQUIVALENCE STRING.
0D76 2958 :-
0D76 2959 :
0D76 2960 FSLOGICAL:
0D76 2961 MOVAB P1(AP),R0 ;LOGICAL NAME TRANSLATION
0D7A 2962 MOVQ R2,-(SP) ;GET ADDRESS OF LOGICAL NAME DESCRIPTOR
0D7D 2963 MOVQ R2,-(SP) ;SAVE EQUIVALENCE NAME DESCRIPTOR
0D80 2964 MOVL SP,R1 ;SET EQUIVALENCE NAME DESCRIPTOR
0D83 2965 $TRNLOG_S(R0),(R1),(R1) ;SET ADDRESS OF EQUIVALENCE NAME DESCRIPTOR
0D96 2966 MOVQ -(SP)+,R1 ;TRANSLATE LOGICAL NAME
0D99 2967 BLBC R0,90$ ;RETRIEVE EQUIVALENCE NAME PARAMETERS
0D9C 2968 CMPW #SS$_NOTRAN,R0 ;IF LBC TRANSLATION FAILURE
0DA1 2969 BEQL 10$ ;DID TRANSLATION ACTUALLY OCCUR?
0DA3 2970 ADDL #8,SP ;IF EQL, THEN NO
0DA6 2971 BRB 20$ ;RESTORE STACK
0DA8 2972 : ;EXAMINE RESULT
0DA8 2973 : TRY AGAIN WITH UPCASED LOGICAL NAME STRING.
0DA8 2974 :
0DA8 2975 10$: MOVQ P1(AP),R1 ;CREATE LOGICAL NAME DESCRIPTOR
0DAC 2976 BSBW DCL$UPCASE ;UPCASE THE LOGICAL NAME STRING
0DAF 2977 MOVQ R1,-(SP) ;PUT THE LOGICAL NAME DESC ON THE STACK
0DB2 2978 MOVL SP,R0 ;SET ADDRESS OFF LOGICAL NAME DESCRIPTOR
0DB5 2979 MOVAB 8(SP),R1 ;SET ADDRESS OF EQUIVALENCE NAME DESCRIPTOR
0DB9 2980 $TRNLOG_S(R0),(R1),(R1) ;TRANSLATE LOGICAL NAME
0DCC 2981 ADDL #8,SP ;REMOVE THE LOGICAL NAME DESCRIPTOR
0DCF 2982 MOVQ -(SP)+,R1 ;RETRIEVE EQUIVALENCE NAME PARAMETERS
0DD2 2983 BLBC R0,90$ ;IF LBC TRANSLATION FAILURE
0DD5 2984 CMPW #SS$_NOTRAN,R0 ;DID TRANSLATION ACTUALLY OCCUR?
0DDA 2985 BEQL 30$ ;IF EQL, THEN NO
0DDC 2986 :
0DDC 2987 : STRIP ESCAPE SEQUENCE OFF EQUIVALENCE NAME STRING.
0DDC 2988 :
0DDC 2989 20$: TSTL R1 ;ZERO LENGTH VALUE?
0DDE 2990 BEQL 40$ ;IF EQL YES
0DE0 2991 CMPB #27,(R2) ;FIRST CHARACTER ESCAPE?
0DE3 2992 BNEQ 40$ ;IF NEQ NO
0DE5 2993 ADDL #4,R2 ;POINT PAST ESCAPE SEQUENCE
0DE8 2994 SUBL #4,R1 ;REDUCE LENGTH OF EQUIVALENCE STRING
0DEB 2995 BGEQ 40$ ;IF GEQ OKAY
0DED 2996 30$: CLRL R1 ;SET TO RETURN NULL STRING
0DEF 2997 40$: RSB ;
```

50	04	AC	9E	0D76	2961	MOVAB	P1(AP),R0	;LOGICAL NAME TRANSLATION
	7E	52	7D	0D7A	2962	MOVQ	R2,-(SP)	;GET ADDRESS OF LOGICAL NAME DESCRIPTOR
	7E	52	7D	0D7D	2963	MOVQ	R2,-(SP)	;SAVE EQUIVALENCE NAME DESCRIPTOR
	51	5E	D0	0D80	2964	MOVL	SP,R1	;SET EQUIVALENCE NAME DESCRIPTOR
				0D83	2965	\$TRNLOG_S	(R0),(R1),(R1)	;SET ADDRESS OF EQUIVALENCE NAME DESCRIPTOR
	51	8E	7D	0D96	2966	MOVQ	-(SP)+,R1	;TRANSLATE LOGICAL NAME
	54	50	E9	0D99	2967	BLBC	R0,90\$	;RETRIEVE EQUIVALENCE NAME PARAMETERS
50	0000	8F	B1	0D9C	2968	CMPW	#SS\$_NOTRAN,R0	;IF LBC TRANSLATION FAILURE
		05	13	0DA1	2969	BEQL	10\$	;DID TRANSLATION ACTUALLY OCCUR?
	5E	08	C0	0DA3	2970	ADDL	#8,SP	;IF EQL, THEN NO
		34	11	0DA6	2971	BRB	20\$	;RESTORE STACK
				0DA8	2972	:	:	;EXAMINE RESULT
				0DA8	2973	:	TRY AGAIN WITH UPCASED LOGICAL NAME STRING.	
				0DA8	2974	:	:	
51	04	AC	7D	0DA8	2975	10\$: MOVQ	P1(AP),R1	;CREATE LOGICAL NAME DESCRIPTOR
	F251		30	0DAC	2976	BSBW	DCL\$UPCASE	;UPCASE THE LOGICAL NAME STRING
	7E	51	7D	0DAF	2977	MOVQ	R1,-(SP)	;PUT THE LOGICAL NAME DESC ON THE STACK
	50	5E	D0	0DB2	2978	MOVL	SP,R0	;SET ADDRESS OFF LOGICAL NAME DESCRIPTOR
51	08	AE	9E	0DB5	2979	MOVAB	8(SP),R1	;SET ADDRESS OF EQUIVALENCE NAME DESCRIPTOR
				0DB9	2980	\$TRNLOG_S	(R0),(R1),(R1)	;TRANSLATE LOGICAL NAME
	5E	08	C0	0DCC	2981	ADDL	#8,SP	;REMOVE THE LOGICAL NAME DESCRIPTOR
	51	8E	7D	0DCF	2982	MOVQ	-(SP)+,R1	;RETRIEVE EQUIVALENCE NAME PARAMETERS
	1B	50	E9	0DD2	2983	BLBC	R0,90\$	;IF LBC TRANSLATION FAILURE
50	0000	8F	B1	0DD5	2984	CMPW	#SS\$_NOTRAN,R0	;DID TRANSLATION ACTUALLY OCCUR?
		11	13	0DDA	2985	BEQL	30\$	;IF EQL, THEN NO
				0DDC	2986	:	:	
				0DDC	2987	:	STRIP ESCAPE SEQUENCE OFF EQUIVALENCE NAME STRING.	
				0DDC	2988	:	:	
	51	D5	0DDE	2989	20\$: TSTL	R1		;ZERO LENGTH VALUE?
	0F	13	0DDE	2990	BEQL	40\$		;IF EQL YES
62	1B	91	0DE0	2991	CMPB	#27,(R2)		;FIRST CHARACTER ESCAPE?
	0A	12	0DE3	2992	BNEQ	40\$		;IF NEQ NO
52	04	C0	0DE5	2993	ADDL	#4,R2		;POINT PAST ESCAPE SEQUENCE
51	04	C2	0DE8	2994	SUBL	#4,R1		;REDUCE LENGTH OF EQUIVALENCE STRING
	02	18	0DEB	2995	BGEQ	40\$		;IF GEQ OKAY
	51	D4	0DED	2996	30\$: CLRL	R1		;SET TO RETURN NULL STRING
		05	0DEF	2997	40\$: RSB			;

LEXICON
V04-000

1 11
- LEXICAL FUNCTION PARSE AND EXECUTION R 16-SEP-1984 00:00:36 VAX/VMS Macro V04-00
LOGICAL NAME TRANSLATION 4-SEP-1984 23:41:41 [DCL.SRC]LEXICON.MAR;1

Page 65
(25)

```
ODFO 2998
ODFO 2999 :
ODFO 3000 : LOGICAL NAME TRANSLATION FAILURE
ODFO 3001 :
ODFO 3002 :
0825 31 ODF0 3003 90$: BRW P1_ERROR ;REPORT ERROR AND DISPLAY P1
```

```
ODF3 3005 .SBTTL LOGICAL NAME TRANSLATION
ODF3 3006 :+
ODF3 3007 : F$TRNLNM - LOGICAL NAME TRANSLATION
ODF3 3008 :
ODF3 3009 : THIS LEXICAL FUNCTION OBTAINS THE TRANSLATION FOR A SPECIFIED LOGICAL NAME AND
ODF3 3010 : RETURN THE EQUIVALENCE NAME OR THE REQUESTED ATTRIBUTE.
ODF3 3011 :
ODF3 3012 : F$TRNLNM (NAME, TABLE, INDEX, MODE, ATTR, ITEM)
ODF3 3013 :
ODF3 3014 : INPUTS:
ODF3 3015 :
ODF3 3016 : P1(AP) = DESCRIPTOR OF LOGICAL NAME STRING.
ODF3 3017 : P2(AP) = DESCRIPTOR OF NAME OF LOGICAL NAME TABLE TO SEARCH.
ODF3 3018 : P3(AP) = INDEX OF THE DESIRED TRANSLATION.
ODF3 3019 : P4(AP) = DESCRIPTOR OF ACCESS MODE OF TRANSLATION.
ODF3 3020 : P5(AP) = DESCRIPTOR OF TYPE OF TRANSLATION.
ODF3 3021 : P6(AP) = DESCRIPTOR OF REQUESTED ITEM.
ODF3 3022 :
ODF3 3023 : OUTPUTS:
ODF3 3024 :
ODF3 3025 : R1 = LENGTH OF EQUIVALENCE STRING.
ODF3 3026 : R2 = ADDRESS OF EQUIVALENCE STRING.
ODF3 3027 : -
ODF3 3028 :
ODF3 3029 F$TRNLNM: ;LOGICAL NAME TRANSLATION
ODF3 3030 :
ODF3 3031 :++
ODF3 3032 : Item list created on stack:
ODF3 3033 :
ODF3 3034 :
ODF3 3035 : 48(R9) : Translation attributes :
ODF3 3036 :
ODF3 3037 : 44(R9) : Attribute bit number :
ODF3 3038 :
ODF3 3039 : 40(R9) : Access mode :
ODF3 3040 :
ODF3 3041 : Table name descriptor :
ODF3 3042 :
ODF3 3043 : 32(R9) :
ODF3 3044 :
ODF3 3045 : 28(R9) : Translation index :
ODF3 3046 :
ODF3 3047 : 24(R9) : End of item list :
ODF3 3048 :
ODF3 3049 : 20(R9) : 12(R9) :
ODF3 3050 :
ODF3 3051 : 16(R9) : Scratch buffer descr :
ODF3 3052 :
ODF3 3053 : 12(R9) : lnm$_string! :
ODF3 3054 :
ODF3 3055 : 8(R9) : 0 :
ODF3 3056 :
ODF3 3057 : 4(R9) : 28(R9) :
ODF3 3058 :
ODF3 3059 : (R9) : lnm$_index! 4 :
ODF3 3060 :
ODF3 3061 : --
```



```
55 00 20 BC 1C AC 2D 0E90 3118 CMPC5 P4(AP),@P4+4(AP),#0,R5,1(R4) ;DOES ACCESS MODE MATCH
      01 A4 0E97
      07 13 0E99 3119 BEQL 27$ ;BRANCH IF ACCESS MODE FOUND
      54 02 A445 9E 0E9B 3120 MOVAB 2(R4)[R5],R4 ;SKIP TO NEXT TABLE ENTRY
      E9 11 0EA0 3121 BRB 25$
      28 A9 01 A445 9A 0EA2 3122 27$: MOVZBL 1(R4)[R5],40(R9) ;SET THE ACCESS MODE
      0EA8 3123
      0EA8 3124 :
      0EA8 3125 : Set the table name.
      0EA8 3126
      0C AC D5 0EA8 3127 30$: TSTL P2(AP) ;WAS TABLE SPECIFIED?
      05 19 0EAB 3128 BLSS 40$ ;NO, THEN USE DEFAULT
      20 A9 0C AC 7D 0EAD 3129 MOVQ P2(AP),32(R9) ;SET SPECIFIED TABLE NAME
      0EB2 3130
      0EB2 3131 :
      0EB2 3132 : Set the tranlation attribute.
      0EB2 3133
      24 AC D5 0EB2 3134 40$: TSTL P5(AP) ;WAS AN ATTR SPECIFIED?
      27 19 0EB5 3135 BLSS 48$ ;NO, THEN USE DEFAULT
      54 00002029'EF 9E 0EB7 3136 MOVAB LOGATTR_TABLE,R4 ;GET ADDRESS OF ATTR TABLE
      55 64 9A 0EBE 3137 45$: MOVZBL (R4),R5 ;GET LENGTH OF ATTR NAME
      24 13 0EC1 3138 BEQL 95$ ;BRANCH IF END OF TABLE
55 00 28 BC 24 AC 2D 0EC3 3139 CMPC5 P5(AP),@P5+4(AP),#0,R5,1(R4) ;DOES ATTR NAME MATCH
      01 A4 0ECA
      07 13 0ECC 3140 BEQL 47$ ;BRANCH IF ATTR FOUND
      54 05 A445 9E 0ECE 3141 MOVAB 5(R4)[R5],R4 ;SKIP TO NEXT TABLE ENTRY
      E9 11 0ED3 3142 BRB 45$
      50 01 A445 9E 0ED5 3143 47$: MOVAB 1(R4)[R5],R0 ;GET ADDR OF ATTRIBUTE CODE
      30 A9 60 D0 0EDA 3144 MOVL (R0),48(R9) ;SET THE ATTRIBUTE
      0036 31 0EDE 3145 48$: BRW 50$
      0EE1 3146
      0EE1 3147 :
      0EE1 3148 : Signal all errors.
      0EE1 3149
      52 1C AC 9E 0EE1 3150 94$: MOVAB P4(AP),R2
      0A 11 0EE5 3151 BRB 961$
      52 24 AC 9E 0EE7 3152 95$: MOVAB P5(AP),R2
      04 11 0EEB 3153 BRB 961$
      52 2C AC 9E 0EED 3154 96$: MOVAB P6(AP),R2
      0EF1 3155 961$: STATUS IVKEYW
      5E 34 A9 9E 0EF8 3156 MOVAB 52(R9),SP
      073B 31 0EFC 3157 BRW PN_ERROR
      0EFF 3158
      5E 34 A9 9E 0EFF 3159 97$: MOVAB 52(R9),SP
      0706 31 0F03 3160 BRW BUFOVF
      0F06 3161
      50 0000'8F B1 0F06 3162 90$: CMPW #SS$_BUFFEROVF,R0
      F2 13 0F0B 3163 BEQL 97$
      5E 34 A9 9E 0F0D 3164 MOVAB 52(R9),SP
      06FF 31 0F11 3165 BRW ERROR
      0088 31 0F14 3166
      0F14 3167 51$: BRW 80$
      0F17 3168
      0F17 3169 :
      0F17 3170 : Perform the translation.
      0F17 3171
      0F17 3172 50$: $TRNLNM_S ATTR=48(R9),-
```

;TRANSLATE THE LOGICAL NAME

LEXICON
V04-000

N 11
- LEXICAL FUNCTION PARSE AND EXECUTION R 16-SEP-1984 00:00:36
LOGICAL NAME TRANSLATION 4-SEP-1984 23:41:41

VAX/VMS Macro V04-00
[DCL.SRC]LEXICON.MAR;1

Page 70
(26)

	0D	12	0F95	3230		BNEQ	85\$		; IF NEQ NO
52	04	C0	0F97	3231		ADDL	#4,R2		; POINT PAST ESCAPE SEQUENCE
51	04	C2	0F9A	3232		SUBL	#4,R1		; REDUCE LENGTH OF EQUIVALENCE STRING
	05	18	0F9D	3233		BGEQ	85\$		; IF GEQ OKAY
	51	D4	0F9F	3234	80\$:	CLRL	R1		; SET TO RETURN NULL STRING
52	5E	D0	0FA1	3235		MOVL	SP,R2		;
5E	34	A9	9E	0FA4	3236	85\$:	MOVAB	52(R9),SP	; RESTORE THE STACK
		05	0FA8	3237		RSB			;

```

OFA9 3239      .SBTTL  GET MESSAGE TEXT
OFA9 3240      :+
OFA9 3241      : F$MESSAGE - GET MESSAGE TEXT
OFA9 3242      :
OFA9 3243      : THIS LEXICAL FUNCTION OBTAINS THE MESSAGE TEXT ASSOCIATED WITH A MESSAGE CODE.
OFA9 3244      :
OFA9 3245      :     F$MESSAGE(CODE)
OFA9 3246      :
OFA9 3247      : INPUTS:
OFA9 3248      :
OFA9 3249      :     P1(AP) = MESSAGE CODE
OFA9 3250      :
OFA9 3251      : OUTPUTS:
OFA9 3252      :
OFA9 3253      :     R1 = LENGTH OF MESSAGE TEXT STRING.
OFA9 3254      :     R2 = ADDRESS OF MESSAGE TEXT STRING.
OFA9 3255      : -
OFA9 3256      :
OFA9 3257      F$MESSAGE:
7E 52 7D OFA9 3258      MOVQ    R2,-(SP)      ;GET MESSAGE TEXT
50 5E D0 OFAC 3259      MOVL    SP,R0        ;SET MESSAGE BUFFER DESCRIPTOR
OFAF 3260      $GETMSG_S P1(AP),(R0),(R0),#15 ;SET ADDRESS OF MESSAGE BUFFER DESCRIPTOR
51 8E 7D OFC1 3261      MOVQ    -(SP)+,R1    ;GET MESSAGE TEXT
05 05 OFC4 3262      RSB                    ;RETRIEVE MESSAGE TEXT PARAMETERS
```

```

OFC5 3264 .SBTTL RETURN JOB MODE
OFC5 3265 :+
OFC5 3266 : F$MODE - RETURN JOB MODE
OFC5 3267 :
OFC5 3268 : THIS LEXICAL FUNCTION RETURNS A STRING THAT IS EITHER 'INTERACTIVE' OR 'BATCH'
OFC5 3269 : DEPENDENT ON THE JOB MODE.
OFC5 3270 :
OFC5 3271 : F$MODE()
OFC5 3272 :
OFC5 3273 : INPUTS:
OFC5 3274 :
OFC5 3275 : NONE.
OFC5 3276 :
OFC5 3277 : OUTPUTS:
OFC5 3278 :
OFC5 3279 : R1 = LENGTH OF JOB MODE DESCRIPTION STRING.
OFC5 3280 : R2 = ADDRESS OF JOB MODE DESCRIPTION STRING.
OFC5 3281 : -
OFC5 3282 :
OFC5 3283 F$MODE:
OFC5 3284 CLRQ -(SP)
OFC5 3285 CLRQ -(SP)
OFC5 3286 PUSHAB -2*4(SP)
OFC5 3287 PUSHBL #JPI$_MODE@16+4
OFC5 3288 MOVL SP,R0
OFC5 3289 $GETJPIW S ITMLST=(R0),-
OFC5 3290 IOSB=16(R0),-
OFC5 3291 EFN=#EXESC_SYSEFN
OFC5 3292 BLBC R0,50$
OFC5 3293 MOVZWL 16(SP),R0
OFC5 3294 BLBC R0,50$
OFC5 3295
OFC5 3296 CLRL R3
OFC5 3297 MOVAB MODE_TABLE,R2
OFC5 3298 MOVZBL (R2)+,R1
OFC5 3299 BEQL 40$
OFC5 3300 CMPL (SP),R3
OFC5 3301 BEQL 40$
OFC5 3302 MOVAB 1(R1)[R2],R2
OFC5 3303 INCL R3
OFC5 3304 BRB 25$
OFC5 3305
OFC5 3306 40$: ADDL #6*4,SP
OFC5 3307 RSB
OFC5 3308
OFC5 3309 50$: ADDL #6*4,SP
OFC5 3310 BRW ERROR

;RETURN JOB MODE
;RESERVE IOSB
;MARK END OF LIST, NO RETLEN
;SET BUFFER ADDRESS
;REQUEST MODE, SET BUFFER LENGTH
;SET ADDRESS OF TIME LIST
;GET MODE LONGWORD
;
;IF LBC SERVICE FAILURE
;GET STATUS FROM IOSB
;IF LBC SERVICE FAILURE
;
;SET LOOP INDEX
;GET ADDRESS OF MODE TABLE
;GET LENGTH OF MODE
;BRANCH IF END OF TABLE
;IS THIS THE ONE WE WANT?
;YES, THEN RETURN THE STRING
;SKIP TO NEXT TABLE ENTRY
;INCR LOOP INDEX
;
;DEALLOCATE GETJPI ITEM LIST AND IOSB
;
;CLEAN STACK
;REPORT ERROR IN FUNCTION
```

```
101D 3312 .SBTTL PARSE A FILE SPECIFICATION
101D 3313 :+
101D 3314 : F$PARSE - PARSE A FILE SPECIFICATION
101D 3315 :
101D 3316 : THIS FUNCTION PERFORMS A $PARSE SYSTEM SERVICE RETURNING THE
101D 3317 : ENTIRE (OR OPTIONALLY ONLY PARTS OF THE) EXPANDED FILESPEC.
101D 3318 :
101D 3319 : INPUTS:
101D 3320 :
101D 3321 : P1(AP) = PRIMARY FILESPEC
101D 3322 : P2(AP) = DEFAULT FILESPEC
101D 3323 : P3(AP) = RELATED FILESPEC (FOR TEMPORARY STICKYNESS)
101D 3324 : P4(AP) = NAME OF A FILESPEC FIELD
101D 3325 : IF NOT SPECIFIED, THE ENTIRE FILESPEC IS RETURNED.
101D 3326 : P5(AP) = TYPE OF PARSE (ITEM LIST)
101D 3327 :
101D 3328 : P2,P3,P4, AND P5 ARE ALL OPTIONAL (INDICATED BY A -1 IN THE FIRST LONGWORD)
101D 3329 :
101D 3330 : OUTPUTS:
101D 3331 :
101D 3332 : R1/R2 = DESCRIPTOR OF EXPANDED FILE SPECIFICATION
101D 3333 : -
101D 3334 :
101D 3335 : F$PARSE:
101D 3336 : MOVQ R2,-(SP) ;PUSH DESCRIPTOR OF SCRATCH BUFFER
1020 3337 : MOVL SP,R9 ;SAVE ADDRESS OF DESCRIPTOR
1023 3338 : MOVAB -NAM$C_BLN-FAB$C_BLN-NAM$C_MAXRSS-NAM$C_BLN(SP),SP ;ALLOCATE STORAGE
1028 3339 : MOVCS #0,(SP),#0,#FAB$C_BLN+NAM$C_BLN+NAM$C_MAXRSS+NAM$C_BLN,(SP) ;ZERO ST
1030 3340 : MOVL SP,R7 ;GET ADDRESS OF NAM
1033 3341 : MOVAB NAM$C_BLN(R7),R6 ;GET ADDRESS OF FAB
1037 3342 : ASSUME FAB$B_BLN EQ FAB$B_BID+1
1037 3343 : MOVW #FAB$C_BID+<FAB$C_BLN@8>,FAB$B_BID(R6) ;INITIALIZE FAB
103C 3344 : ASSUME NAM$B_BLN EQ NAM$B_BID+1
103C 3345 : MOVW #NAM$C_BID+<NAM$C_BLN@8>,NAM$B_BID(R7) ;INITIALIZE NAM
1041 3346 :
1041 3347 : VALIDATE PARSE TYPE PARAMETER LIST, IF ANY
1041 3348 :
1041 3349 : TSTL P5(AP) ;PARSE TYPE SPECIFIED?
1044 3350 : BLSS 18$ ;IF NOT, SKIP BIT SETTING
1046 3351 :
1046 3352 : MOVL R8,-(SP) ;SAVE R8,R10,R11
1049 3353 : MOVQ R10,-(SP)
104C 3354 : MOVQ P5(AP),R4 ;R4/R5 = ITEM LIST DESCRIPTOR
1050 3355 : CLRB R8 ;ASSUME ONLY 1 ITEM IN LIST
1052 3356 :
1052 3357 5$: MOVQ R4,R2 ;SETUP FOR TABLE CHECK IN CASE 1 ITEM
1055 3358 : LOCC #^A/,/,R4,(R5) ;SEARCH ITEM LIST FOR A COMMA
1059 3359 : BEQL 6$ ;CHECK TABLE NOW IF ONLY 1 ITEM IN LIST
105B 3360 :
105B 3361 : INCB R8 ;INDICATE AT LEAST ONE MORE ITEM IN LIST
105D 3362 : SUBL3 R0,R4,R2 ;CALCULATE LENGTH OF THIS ITEM
1061 3363 : MOVAB -(R0),R4 ;STRIP CURRENT ITEM FROM LIST
1064 3364 : MOVAB 1(R1),R5
1068 3365 :
1068 3366 6$: MOVAB PARSE_TYPES,R10 ;R10 = ADDR. OF PARSE TYPES TABLE
106F 3367 7$: MOVZBL (R10),R11 ;R11 = LENGTH OF THIS PARSE TYPE
1072 3368 : BEQL 10$ ;HAVE INVALID KEY WORD IF AT TABLE END
```

6E	020F	8F	00	5E	FDF1	CE	9E	1023	3338	
			56	6E	00	2C	1028	3339		
				57	5E	D0	1030	3340		
				60	A7	9E	1033	3341		
							1037	3342		
			66	5003	8F	B0	1037	3343		
							103C	3344		
			67	6002	8F	B0	103C	3345		
							1041	3346		
							1041	3347		
							1041	3348		
				24	AC	D5	1041	3349		
				67		19	1044	3350		
							1046	3351		
				7E	58	D0	1046	3352		
				7E	5A	7D	1049	3353		
			54	24	AC	7D	104C	3354		
					58	94	1050	3355		
							1052	3356		
				52	54	7D	1052	3357	5\$:	
			65	54	2C	3A	1055	3358		
					0D	13	1059	3359		
							105B	3360		
					58	96	105B	3361		
			52	54	50	C3	105D	3362		
				54	70	9E	1061	3363		
			55	01	A1	9E	1064	3364		
							1068	3365		
5A	00001669	'EF	9E	1068	3366	6\$:	MOVAB	PARSE_TYPES,R10		
	5B	6A	9A	106F	3367	7\$:	MOVZBL	(R10),R11		
	16	13	1072	3368			BEQL	10\$		

00	7E	52	7D	1074	3369					
	63	52	2D	1074	3370	MOVQ	R2,-(SP)	:	SAVE ITEM LENGTH AND ADDR.	
01	AA	5B		1077	3371	CMPC5	R2,(R3),#0,-	:	CHECK CURR. ITEM AGAINST TAB. ENTRY	
		1A	13	107B	3372		R11,1(R10)	:		
				107E	3373	BEQL	12\$	:	IF MATCH, SET BIT IN NAM\$B_NOP FLAG	
				1080	3374			:		
5A	52	8E	7D	1080	3375	MOVQ	(SP)+,R2	:	RESTORE ITEM LENG. AND ADDR.	
	02	AA4B	9E	1083	3376	MOVAB	2(R10)[R11],R10	:	R10 = ADDR. OF NEXT TABLE ENTRY	
		E5	11	1088	3377	BRB	7\$	:	CHECK ITEM AGAINST NEXT TABLE ENTRY	
				108A	3378			:		
	5A	8E	7D	108A	3379	10\$:	MOVQ	(SP)+,R10	:	RESTORE WORK REGISTERS
	58	8E	D0	108D	3380		MOVL	(SP)+,R8	:	
				1090	3381		STATUS	IVKEYW	:	INVALID KEYWORD
		0596	31	1097	3382		BRW	P5_ERROR	:	GOTO ERROR HANDLER
				109A	3383				:	
02	52	8E	7D	109A	3384	12\$:	MOVQ	(SP)+,R2	:	HAVE MATCH. FIRST RESTORE ITEM REG.
A7	01	AA4B	88	109D	3385		BISB	1(R10)[R11],NAM\$B_NOP(R7)	:	SET CORRESPONDING BIT
	AB	58	E4	10A3	3386		BBSC	#0,R8,5\$	:	CHECK NEXT ITEM IN LIST(IF ANY LEFT)
				10A7	3387				:	
	5A	8E	7D	10A7	3388		MOVQ	(SP)+,R10	:	RESTORE REGISTERS
	58	8E	D0	10AA	3389		MOVL	(SP)+,R8	:	
				10AD	3390	:			:	
				10AD	3391	:			:	
				10AD	3392	:			:	
				10AD	3393	18\$:	TSTL	P4(AP)	:	FIELD SPECIFIED?
	1C	AC	D5	10AD	3393		BLSS	19\$	:	IF NOT, RETURN ENTIRE EXPANDED STRING
54	00001634	'EF	9E	10B0	3394		MOVAB	PARSE_F:FLDS,R4	:	GET ADDRESS OF TABLE
	55	64	9A	10B9	3396	300\$:	MOVZBL	(R4),R5	:	GET LENGTH OF ITEM NAME
		12	13	10BC	3397		BEQL	350\$	:	BRANCH IF END OF TABLE
00	20	BC	1C	10BE	3398		CMPC5	P4(AP),@P4+4(AP),#0,-	:	DOES ITEM NAME MATCH?
	01	A4	55	10C4	3399			R5,1(R4)	:	
			13	10C7	3400		BEQL	380\$	:	BRANCH IF ITEM FOUND
	54	03	A445	9E	10C9		MOVAB	3(R4)[R5],R4	:	SKIP TO NEXT ENTRY IN TABLE
		E9	11	10CE	3402		BRB	300\$	:	LOOP UNTIL FOUND
				10D0	3403	350\$:	STATUS	IVKEYW	:	UNDEFINED ITEM NAME
		0550	31	10D7	3404		BRW	P4_ERROR	:	
	54	83	9A	10DA	3405	380\$:	MOVZBL	(R3)+,R4	:	GET OFFSET TO FIELD LENGTH
	55	83	9A	10DD	3406		MOVZBL	(R3)+,R5	:	GET OFFSET TO FIELD ADDRESS
				10E0	3407	:			:	
				10E0	3408	:			:	
				10E0	3409	:			:	
	28	A6	57	D0	10E0	19\$:	MOVL	R7,FAB\$L_NAM(R6)	:	SET POINTER TO NAM BLOCK
					10E4		SETBIT	FAB\$V_PPF,FAB\$L_FOP(R6)	:	USE PROCESS I/O SEGMENT (SINCE THERE
					10E9				:	MAY NOT BE ANY IMAGE I/O SEGMENT)
34	A6	04	AC	90	10E9		MOVB	P1(AP),FAB\$B_FNS(R6)	:	STORE PRIMARY NAME STRING INTO FAB
2C	A6	08	AC	D0	10EE		MOVL	P1+4(AP),FAB\$B_FNA(R6)	:	
		0C	AC	D5	10F3		TSTL	P2(AP)	:	P2 SPECIFIED?
			19	10F6	3416		BLSS	20\$	:	BRANCH IF NOT
35	A6	0C	AC	90	10F8		MOVB	P2(AP),FAB\$B_DNS(R6)	:	STORE DEFAULT NAME STRING INTO FAB
30	A6	10	AC	D0	10FD		MOVL	P2+4(AP),FAB\$B_DNA(R6)	:	
		14	AC	D5	1102	20\$:	TSTL	P3(AP)	:	P3 SPECIFIED?
		18	19	1105	3420		BLSS	25\$	:	BRANCH IF NOT
50	014F	C6	9E	1107	3421		MOVAB	FAB\$B_BLN+NAM\$C_MAXRSS(R6),R0	:	ADDRESS OF RELATED NAM
				110C	3422		ASSUME	NAM\$B_BLN ~ NAM\$B_BID+1	:	
60	6002	8F	80	110C	3423		MOVW	#NAM\$C_BID+NAM\$C_BLN@8>,NAM\$B_BID(R0)	:	INITIALIZE RELATED NAM
03	A0	14	AC	90	1111		MOVB	P3(AP),NAM\$B_RSL(R0)	:	INITIALIZE RELATED FILESPEC
04	A0	18	AC	D0	1116		MOVL	P3+4(AP),NAM\$B_RSA(R0)	:	

10	A7	50	D0	111B	3426		MOVL	R0,NAM\$L_RLF(R7)	:POINT TO RELATED NAM BLOCK	
0A	A7	FF	8F	90	111F	3427	25\$:	MOVW	#NAM\$C_MAXRSS,NAM\$B_ESS(R7)	:SET SIZE OF EXPANDED BUFFER
0C	A7	50	A6	9E	1124	3428		MOVAB	FAB\$C_BLN(R6),NAM\$B_ESA(R7)	:SET ADDRESS OF RESULT BUFFER
					1129	3429		\$PARSE	FAB=(R6)	:SETUP EXPANDED NAME STRING, NAM FIELDS
					1132	3430				: (SUCH AS FID,DID,DVI,ESL)
	03	50	E8	1132	3431			BLBS	R0,28\$	:BRANCH IF OK
	0B	A7	94	1135	3432			CLRB	NAM\$B_ESL(R7)	:RETURN NULL ON ANY PARSE ERROR
	51	0B	9A	1138	3433	28\$:		MOVZBL	#NAM\$B_ESL,R1	:ASSUME USING ENTIRE EXPANDED STRING
	52	0C	9A	113B	3434			MOVZBL	#NAM\$B_ESA,R2	
	1C	AC	D5	113E	3435			TSTL	P4(AP)	:FIELD SPECIFIED?
		03	19	1141	3436			BLSS	40\$	:IF NOT, RETURN ENTIRE EXPANDED STRING
	51	54	7D	1143	3437			MOVQ	R4,R1	:GET OFFSET TO FIELD LENGTH, ADDRESS
	51	67	41	9A	1146	3438	40\$:	MOVZBL	(R7)[R1],R1	:GET LENGTH OF EXPANDED STRING
	52	57	C0	114A	3439			ADDL	R7,R2	:GET ADDRESS OF EXPANDED STRING ADDRESS
	69	51	D1	114D	3440			CMPL	R1,(R9)	:ENOUGH ROOM FOR STRING?
		2C	1A	1150	3441			BGTRU	60\$	:BRANCH IF NOT
		51	DD	1152	3442			PUSHL	R1	:SAVE REGISTERS
04	B9	00	B2	51	28	1154		MOVC	R1,@(R2),@4(R9)	:COPY INTO SCRATCH BUFFER
				53	8E	D0		POPL	R3	:SAVE LENGTH OF EXPANDED STRING
		30	A7	D5	115D	3445		TSTL	NAM\$B_WCC(R7)	:DID RMS SAVE ANY INTERNAL CONTEXT?
		10	13	1160	3446			BEQL	50\$	:BRANCH IF NOT
					1162	3447		ASSUME	FAB\$B_DNS EQ FAB\$B_FNS+1	
		34	A6	B4	1162	3448		CLRW	FAB\$B_FNS(R6)	:SET PRIMARY/DEFAULT FILESPECS TO NULL
					1165	3449		SETBIT	NAM\$V_SYNCHK,NAM\$B_NOP(R7)	:SET SYNTAX ONLY CHECK FLAG
					1169	3450		\$PARSE	FAB=(R6)	:TERMINATE WILDCARD SEQUENCE TO GET RMS
					1172	3451				:TO DEALLOCATE INTERNAL CONTEXT STORAGE
	51	53	D0	1172	3452	50\$:		MOVL	R3,R1	:GET LENGTH OF RESULTANT STRING
	52	04	A9	D0	1175	3453		MOVL	4(R9),R2	:SET ADDRESS OF STRING
	5E	08	A9	9E	1179	3454		MOVAB	8(R9),SP	:DEALLOCATE STORAGE
			05	117D	3455			RSB		
					117E	3456				
		04	8B	31	117E	3457	60\$:	BRW	BUFOVF	:REPORT BUFFER OVERFLOW

```
1181 3459 .SBTTL GET NEXT PROCESS ID
1181 3460 :+
1181 3461 : F$PID - GET NEXT PROCESS ID
1181 3462 :
1181 3463 : THIS FUNCTION PERFORMS A WILDCARD GETJPI FUNCTION TO RETURN THE
1181 3464 : NEXT PROCESS PID IN SEQUENCE, GIVEN THE WILDCARD CONTEXT.
1181 3465 :
1181 3466 : F$PID(CONTEXT_SYMBOL)
1181 3467 :
1181 3468 : INPUTS:
1181 3469 :
1181 3470 : P1(AP) = DESCRIPTOR OF SYMBOL NAME
1181 3471 :
1181 3472 : OUTPUTS:
1181 3473 :
1181 3474 : R1/R2 = DESCRIPTOR OF PID CONVERTED TO AN ASCII STRING
1181 3475 :-
1181 3476 :
1181 3477 F$PID:
1181 3478 CLRQ -(SP) ;RESERVE IOSB
1183 3479 CLRQ -(SP) ;CLEAR LAST ITEM AND RETLEN ADDRESS
1185 3480 PUSHAB -8(SP) ;ADDRESS TO RECEIVE PID
1188 3481 PUSHL #JPI$ PID@16+4 ;SET ITEM CODE AND BUFLN
118E 3482 MNEGL #1,-(SP) ;ASSUME CONTEXT LONGWORD OF -1 (BEGIN)
1191 3483 MOVQ P1(AP),R1 ;GET DESCRIPTOR OF SYMBOL NAME
1195 3484 BSBW DCL$SEARCH ;FIND SYMBOL IN SYMBOL TABLE
1198 3485 BLBC R0,10$ ;IF NOT FOUND, SEARCH FROM BEGINNING
119B 3486 TSTL R2 ;STRING VALUE?
119D 3487 BNEQ 10$ ;IF SO, START SEARCH AT BEGINNING
119F 3488 TSTL R1 ;TOP BIT MUST BE SET FOR VALID PID CTX
11A1 3489 BGEQ 10$ ;IF NOT OK, START SEARCH AT BEGINNING
11A3 3490 MOVL R1,(SP) ;STORE CONTEXT LONGWORD ON STACK
11A6 3491 10$: MOVL SP,R0 ;GET ADDRESS OF CONTEXT/ITEM LIST
11A9 3492 $GETJPIW S$PIDADR=(R0),- ;GET NEXT PID IN SEQUENCE
11A9 3493 ITMLST=4(R0),-
11A9 3494 IOSB=20(RU),-
11A9 3495 EFN=#EXESC_SYSEFN
11C2 3496 BLBC R0,15$ ;BRANCH IF ERROR
11C5 3497 MOVZWL 20(SP),R0 ;FETCH STATUS FROM IOSB
11C9 3498 BLBS R0,20$ ;BRANCH IF NO ERROR
11CC 3499 15$: CMPW R0,#SS$_NOPRIV ;NO PRIVILEGE TO GET INFO?
11D1 3500 BEQL 10$ ;IF SO, TRY NEXT PROCESS
11D3 3501 CMPW R0,#SS$_NOMOREPROC ;NO MORE PROCESSES?
11D8 3502 BNEQ 80$ ;IF NOT, REPORT ERROR
11DA 3503 CLRQ (SP) ;RETURN NULL PID ON THIS INVOCATION
11DC 3504 20$: POPL R1 ;GET NEW CONTEXT LONGWORD
11DF 3505 MOVQ P1(AP),R3 ;SET DESCRIPTOR OF SYMBOL NAME
11E3 3506 MOVAB PRC_Q_LOCAL(R11),R5 ;SET ADDRESS OF SYMBOL TABLE LISTHEAD
11E7 3507 MOVL #SYM_R_BINARY,R0 ;SET TYPE IS BINARY LONGWORD
11EA 3508 DISABLE ;DISABLE CONTROL/Y AST'S
11F0 3509 BSBW DCL$ALLOCSYM ;REPLACE VALUE OF SYMBOL
11F3 3510 ENABLE ;RE-ENABLE CONTROL/Y AST'S
11F5 3511 POPL R0 ;GET PID OF NEXT PROCESS
11F8 3512 ADDL #5*4,SP ;AND CLEAN STACK
11FB 3513 MOVL SP,R2 ;MARK VALUE IS A STRING
11FE 3514 MOVL R0,R1 ;END OF WILDCARD SEARCH?
1201 3515 BEQL 30$ ;IF SO, RETURN WITH A NULL STRING
```

LEXICON
V04-000

H 12
- LEXICAL FUNCTION PARSE AND EXECUTION R 16-SEP-1984 00:00:36 VAX/VMS Macro V04-00
GET NEXT PROCESS ID 4-SEP-1984 23:41:41 [DCL.SRC]LEXICON.MAR;1

Page 77
(30)

00000000'EF	16	1203	3516		JSB	DCL\$CBTA_HEX	:CONVERT PID TO STRING AND EXIT
	05	1209	3517	30\$:	RSB		
		120A	3518				
5E 1C	C0	120A	3519	80\$:	ADDL	#7*4,SP	:CLEAN STACK
0408	31	120D	3520		BRW	P1_ERROR	:REPORT ERROR AND DISPLAY P1

```
1210 3522 .SBTTL CHECK IF SPECIFIED PRIVS ARE SET OR CLEAR
1210 3523
1210 3524 :+ F$PRIVILEGE - CHECK IF SPECIFIED PRIVILEGES ARE SET OR CLEAR
1210 3525
1210 3526 : THIS LEXICAL FUNCTION CAN BE USED TO DETERMINE WHETHER THE CURRENT PROCESS
1210 3527 : HAS A GIVEN SET OF PRIVILEGES ON OR OFF, DEPENDING ON THE KEYWORDS GIVEN.
1210 3528 : THE ARGUMENT IS A COMMA SEPARATED LIST OF PRIVILEGE KEYWORDS ABOUT WHICH
1210 3529 : THE CALLER IS INTERESTED. IF THE PREFIX 'NO' APPEARS ON THE KEYWORD, THEN
1210 3530 : THE FUNCTION CHECKS IF THE PRIVILEGE IS OFF. OTHERWISE, THE FUNCTION CHECKS
1210 3531 : THAT THE PRIVILEGE IS ON. THE FINAL VALUE OF THE FUNCTION IS THE 'AND' OF
1210 3532 : ALL OF THE PRIVILEGE CHECKS - THAT IS, IT RETURNS TRUE IF ALL THE PRIVILEGES
1210 3533 : ARE SET AS SPECIFIED.
1210 3534
1210 3535 F$PRIVILEGE(PRVSTATES)
1210 3536
1210 3537 INPUTS:
1210 3538
1210 3539 P1(AP) = DESCRIPTOR OF A COMMA SEPARATED LIST OF PRIVILEGE KEYWORDS.
1210 3540
1210 3541 OUTPUTS:
1210 3542
1210 3543 R1/R2 = BOOLEAN TRUE OR FALSE STRING
1210 3544 :-
1210 3545
1210 3546 F$PRIVILEGE:
1210 3547 MOVL R3,R6 :GET ADDRESS OF SCRATCH QUADWORD
1213 3548 $SETPRV S PRVPRV=(R6),- :GET CURRENT PERMANENT PRIVILEGES
1213 3549 -PRMFLG=#1
1222 3550 MOVQ P1(AP),R4 :GET DESCRIPTOR OF COMMA SEPARATED LIST
1226 3551 10$: MOVQ R4,R2 :ASSUME NO COMMA IN LIST
1229 3552 LOCC #^A',',R4,(R5) :LOCATE END OF NEXT KEYWORD
122D 3553 BEQL 20$ :BRANCH IF NOT FOUND, USE ENTIRE STRING
122F 3554 SUBL3 R0,R4,R2 :ISOLATE NEXT KEYWORD
1233 3555 20$: MOVAB -(R0),R4 :CHOP KEYWORD OFF STRING
1236 3556 MOVAB 1(R1),R5
123A 3557 MOVQ R2,-(SP)
123D 3558 PUSHAL -(SP)
123F 3559 PUSHAB 8(SP)
1242 3560 CALLS #2,G^PRV$PRIVBIT :PUSH DESCRIPTOR OF KEYWORD
1249 3561 POPR #^M<R1,R2,R3> :PUSH ADDRESS OF WORD TO GET BIT NUMBER
124B 3562 BLBC R0,70$ :PUSH ADDRESS OF DESCRIPTOR
124E 3563 CMPW (R3),#^A'NO' :LOOKUP KEYWORD AND GET PRIV. BIT #
1253 3564 BEQL 30$ :CLEANUP STACK
1255 3565 BBC R1,(R6),50$ :IF ERROR, SIGNAL IT
1259 3566 BRB 40$ :IS KEYWORD NEGATED?
125B 3567 30$: BBS R1,(R6),50$ :IF SO, CHECK IF PRIVILEGE OFF
125F 3568 40$: TSTL R4 :BRANCH IF PRIVILEGE NOT ON
1261 3569 BGTR 10$ :BRANCH IF PRIVILEGE NOT OFF
1263 3570 MOVL #1,R1 :ANY MORE KEYWORDS LEFT?
1266 3571 BRB 60$ :IF NOT, LOOP UNTIL DONE
1268 3572
1268 3573 50$: CLRL R1 :RETURN SUCCESS
126A 3574 60$: BRW FORMAT_BOOLEAN :RETURN FAILURE
126D 3575
126D 3576 70$: CMPL #PRV$_NOTUNQ,R0 :RETURN 'TRUE' OR 'FALSE'
1274 3577 BNEQ 75$ :AMBIGUOUS KEYWORD?
1276 3578 STATUS ABKEYW :IF NOT, EXIT WITH INVALID KEYWORD
:SET AMBIGUOUS KEYWORD ERROR
```

LEXICON
V04-000

J 12
- LEXICAL FUNCTION PARSE AND EXECUTION R 16-SEP-1984 00:00:36 VAX/VMS Macro V04-00
CHECK IF SPECIFIED PRIVS ARE SET OR CLEA 4-SEP-1984 23:41:41 [DCL.SRC]LEXICON.MAR;1

Page 79
(31)

07	11	127D	3579	BRB	80\$	
		127F	3580	STATUS	IVKEYW	
038F	31	1286	3581	BRW	P1_ERROR	
			80\$:			;SET INVALID KEYWORD
			80\$:			;REPORT ERROR

```
1289 3583 .SBTTL RETURN PROCESS NAME
1289 3584 :+
1289 3585 : F$PROCESS - RETURN PROCESS NAME
1289 3586 :
1289 3587 : THIS LEXICAL FUNCTION OBTAINS THE CURRENT PROCESS NAME AND RETURN A DESCRIPTOR
1289 3588 : FOR IT.
1289 3589 :
1289 3590 : F$PROCESS()
1289 3591 :
1289 3592 : INPUTS:
1289 3593 :
1289 3594 : R2 = LENGTH OF SCRATCH BUFFER.
1289 3595 : R3 = ADDRESS OF SCRATCH BUFFER.
1289 3596 :
1289 3597 : OUTPUTS:
1289 3598 :
1289 3599 : R1 = LENGTH OF PROCESS NAME.
1289 3600 : R2 = ADDRESS OF PROCESS NAME.
1289 3601 :-
1289 3602 :
1289 3603 F$PROCESS:
1289 3604 CMPL #15,R2 ;RETURN PROCESS NAME
1289 3605 BGTRU 20$ ;BUFFER BIG ENOUGH TO HOLD PROCESS NAME?
1289 3606 CLRQ -(SP) ;BRANCH IF NOT
1289 3607 CLRL -(SP) ;RESERVE IOSB
1289 3608 PUSHAB -12(SP) ;CLEAR ITEM ID OF LAST ITEM
1289 3609 MOVQ R2,-(SP) ;PUSH ADDRESS TO STORE LENGTH
1289 3610 MOVW #JPI$PRCNAM,2(SP) ;PUSH OUTPUT BUFFER DESCRIPTOR
1289 3611 MOVL SP,R0 ;SET PROCESS NAME IDENTIFICATION
1289 3612 $GETJPIW S ITMLST=(R0),- ;SET ADDRESS OF ITEM LIST
1289 3613 IOSB=16(R0),- ;GET PROCESS NAME
1289 3614 EFN=#EXE$C_SYSEFN
1289 3615 BLBC R0,10$ ;IF LBC SERVICE FAILURE
1289 3616 MOVZWL 16(SP),R0 ;GET STATUS FROM IOSB
1289 3617 10$: MOVQ (SP)+,R1 ;RETRIEVE PROCESS NAME PARAMETERS
1289 3618 MOVZWL R1,R1 ;CLEAR ITEM ID
1289 3619 ADDL #4*4,SP ;CLEAN OFF STACK
1289 3620 BLBC R0,30$ ;BRANCH IF ERROR
1289 3621 RSB
1289 3622 20$: BRW BUFOVF ;REPORT BUFFER OVERFLOW
1289 3623 30$: BRW ERROR ;REPORT ERROR IN R0
```

02 AE 52 OF D1 1289 3604
3F 1A 128C 3605
7E 7C 128E 3606
7E D4 1290 3607
F4 AE 9F 1292 3608
7E 52 7D 1295 3609
031C 8F B0 1298 3610
50 5E D0 129E 3611
12A1 3612
12A1 3613
12A1 3614
50 04 50 E9 1289 3615
10 AE 3C 128C 3616
51 8E 7D 12C0 3617
51 51 3C 12C3 3618
5E 10 C0 12C6 3619
04 50 E9 12C9 3620
033C 31 12CC 3621
0340 31 12CD 3622
12D0 3623

```
12D3 3625 .SBTTL GET NEXT WILDCARD FILESPEC
12D3 3626 :+
12D3 3627 : FS$SEARCH - GET NEXT WILDCARD FILESPEC
12D3 3628 :
12D3 3629 : THIS FUNCTION INVOKES THE $SEARCH SYSTEM SERVICE TO RETURN THE NEXT
12D3 3630 : FILESPEC IN SEQUENCE, GIVEN THE WILDCARD CONTEXT. IF THE EXPANDED
12D3 3631 : NAME STRING IS NOT WILDCARD, THE EXPANDED NAME STRING IS CONVERTED
12D3 3632 : INTO A RESULTANT NAME STRING.
12D3 3633 :
12D3 3634 : FS$SEARCH(ENM, STREAM_ID)
12D3 3635 :
12D3 3636 : INPUTS:
12D3 3637 :
12D3 3638 : P1(AP) = DESCRIPTOR OF EXPANDED NAME STRING
12D3 3639 : P2(AP) = OPTIONAL BINARY STREAM IDENTIFIER, DEFAULTS TO 0
12D3 3640 :
12D3 3641 : OUTPUTS:
12D3 3642 :
12D3 3643 : R1/R2 = DESCRIPTOR OF RESULTANT FILESPEC
12D3 3644 : -
12D3 3645 :
12D3 3646 : FS$SEARCH:
12D3 3647 :
12D3 3648 : SEARCH THE LIST OF SEARCH CONTEXT BLOCKS FOR THE REQUESTED STREAM.
12D3 3649 : THE CONTEXT IS MATCHED BASED ON A USER SPECIFIED 32-BIT BINARY NUMBER.
12D3 3650 :
12D3 3651 : EACH CONTEXT BLOCK HAS THE FOLLOWING FORMAT:
12D3 3652 :
12D3 3653 : LONGWORD POINTER TO NEXT BLOCK, 0 MEANS END OF LIST
12D3 3654 : LENGTH OF THIS BLOCK IN BYTES
12D3 3655 : FAB BLOCK (FAB$L_CTX CONTAINS THE USER STREAM IDENTIFIER, P2)
12D3 3656 : NAM BLOCK
12D3 3657 : EXPANDED NAME STRING BUFFER
12D3 3658 : RESULTANT NAME STRING BUFFER
12D3 3659 : PRIMARY FILESPEC BUFFER
12D3 3660 :
12D3 3661 : ON OUTPUT FROM THIS PARAGRAPH:
12D3 3662 :
12D3 3663 : R6 = ADDRESS OF CURRENT CONTEXT BLOCK, IF FOUND
12D3 3664 : R7 = ADDRESS OF PREVIOUS CONTEXT BLOCK IN LINKED LIST
12D3 3665 :
12D3 3666 : MOVL PRC_L_STACKPT(R11),R0 ;GET ADDRESS OF CURRENT INDIRECT FRAME
12D8 3667 : MOVAB IDF_L_SEARCHCTX(R0),R7 ;GET ADDRESS OF LISTHEAD
12DC 3668 : MOVL (R7),R6 ;GET ADDRESS OF NEXT CONTEXT BLOCK
12DF 3669 : BEQL 20$ ;BRANCH IF CONTEXT NOT FOUND
12E1 3670 : CMPL P2(AP),FAB$L_CTX+8(R6) ;DOES THE STREAM IDENTIFIER MATCH?
12E6 3671 : BEQL 15$ ;BRANCH IF CONTEXT FOUND
12E8 3672 : MOVL R6,R7 ;SKIP TO NEXT BLOCK IN LIST
12EB 3673 : BRB 10$ ;AND CONTINUE SEARCHING
12ED 3674 :
12ED 3675 : CONTEXT BLOCK FOUND IN LIST. IF THE INPUT FILESPEC HAS CHANGED,
12ED 3676 : THEN RESTART THIS STREAM WITH NEW FILESPEC.
12ED 3677 :
12ED 3678 : 15$: MOVAB 8(R6),R5 ;SET ADDRESS OF FAB
12F1 3679 : MOVAB FAB$C_BLN(R5),R4 ;SET ADDRESS OF NAM
12F5 3680 : MOVZBL FAB$B_FNS(R5),R0 ;GET SIZE OF PREVIOUS FILESPEC
12F9 3681 : TSTL P1(AP) ;NULL STRING?
```

50	00A0	CB	D0	12D3	3666			
57	64	A0	9E	12D8	3667			
	56	67	D0	12DC	3668	10\$:		
		2E	13	12DF	3669			
20	A6	0C	D1	12E1	3670			
		05	13	12E6	3671			
	57	56	D0	12E8	3672			
		EF	11	12EB	3673			
				12ED	3674			
				12ED	3675			
				12ED	3676			
				12ED	3677			
55	08	A6	9E	12ED	3678	15\$:		
54	50	A5	9E	12F1	3679			
50	34	A5	9A	12F5	3680			
	04	AC	D5	12F9	3681			

```
04 AC 00 2C B5 0E 13 12FC 3682 BEQL 17$ :YES, THEN CLEAN UP AND EXIT
08 BC 2D 12FE 3683 CMPCS R0,@FAB$L_FNA(R5),#0,P1(AP),@P1+4(AP);SAME FILESPEC?
51 12 1307 3684 BNEQ 28$ :IF NOT, RESTART SEARCH SEQUENCE
0089 31 1309 3685 BRW 30$ :ELSE, CONTINUE WHERE WE LEFT OFF
00A8 31 130C 3686 17$: BRW 50$ :CLEAN UP AND EXIT
130F 3687
130F 3688
130F 3689 : THE CONTEXT WAS NOT FOUND AMONG THE EXISTING CONTEXT BLOCKS FOR THIS
130F 3690 : INDIRECT LEVEL. CREATE A NEW CONTEXT BLOCK BY PARSING THE INPUT
130F 3691 : FILESPEC, P1, AND INSERTING THE CONTEXT BLOCK INTO THE LIST.
130F 3692
51 03B5 8F 3C 130F 3693 20$: MOVZWL #8+FAB$C_BLN+NAM$C_BLN+<3*NAM$C_MAXRSS>,R1 ;SET SIZE OF BLOCK
ECE9 30 1314 3694 BSBW DCL$ALLDYNMEM ;ALLOCATE CONTEXT BLOCK
OA 50 E8 1317 3695 BLBS R0,25$ ;BRANCH IF OK
02F4 31 131A 3696 STATUS SYMOVF ;SYMBOL TABLE OVERFLOW
56 52 D0 1324 3697 BRW P1_ERROR ;REPORT ERROR
66 67 D0 1327 3698 25$: MOVL R2,R6 ;SAVE ADDRESS OF ALLOCATED BLOCK
67 56 D0 132A 3700 MOVL (R7),(R6) ;INSERT INTO LINKED LIST
04 A6 51 D0 132D 3701 MOVL R6,(R7)
0080 3F 00 6E 00 2C 1331 3702 MOVL R1,4(R6) ;STORE SIZE IN NEWLY ALLOCATED BLOCK
55 08 A6 9E 133A 3703 MOVCS #0,(SP),#0,#FAB$C_BLN+NAM$C_BLN,8(R6) ;ZERO REST OF BLOCK
54 50 A5 9E 133E 3704 MOVAB 8(R6),R5 ;SET ADDRESS OF FAB
65 5003 8F B0 1342 3705 MOVAB FAB$C_BLN(R5),R4 ;SET ADDRESS OF NAM
64 6002 8F B0 1347 3706 ASSUME FAB$B_BLN EQ FAB$B_BID+1
28 A5 54 D0 1347 3707 MOVW #FAB$C_BID+<FAB$C_BLN@8>,FAB$B_BID(R5) ;INITIALIZE FAB
6002 8F B0 1347 3708 ASSUME NAM$B_BLN EQ NAM$B_BID+1
28 A5 54 D0 1347 3708 MOVW #NAM$C_BID+<NAM$C_BLN@8>,NAM$B_BID(R4) ;INITIALIZE NAM
134C 3709 MOVL R4,FAB$L_NAM(R5) ;SET POINTER TO NAM BLOCK
1350 3710 SETBIT FAB$V_PPF,FAB$L_FOP(R5) ;USE PROCESS I/O SEGMENT (SINCE THERE
1355 3711 : MAY NOT BE ANY IMAGE I/O SEGMENT)
18 A5 0C AC D0 1355 3712 MOVL P2(AP),FAB$L_CTX(R5) ;SAVE USER SEARCH STREAM IDENTIFIER
34 A5 04 AC 90 135A 3713 28$: MOVAB P1(AP),FAB$B_FNS(R5) ;STORE PRIMARY NAME STRING INTO FAB
2C A5 025E C4 9E 135F 3714 MOVAB NAM$C_BLN+<2*NAM$C_MAXRSS>(R4),FAB$L_FNA(R5)
00FF 8F 00 08 BC 04 AC 2C 1365 3715 PUSHR #*M<R4,R5> ;SAVE REGISTERS
2C B5 30 BB 1367 3716 MOVC5 P1(AP),@P1+4(AP),#0,#NAM$C_MAXRSS,@FAB$L_FNA(R5)
1370
1372 3717 POPR #*M<R4,R5> ;RESTORE REGISTERS
0A A4 FF 8F 90 1374 3718 MOVAB #NAM$C_MAXRSS,NAM$B_ESS(R4) ;SET DESCRIPTOR OF ESA BUFFER
0C A4 60 A4 9E 1379 3719 MOVAB NAM$C_BLN(R4),NAM$L_ESA(R4)
02 A4 FF 8F 90 137E 3720 MOVAB #NAM$C_MAXRSS,NAM$B_RSS(R4) ;SET DESCRIPTOR OF RESULT BUFFER
015F C4 9E 1383 3721 MOVAB NAM$C_BLN+NAM$C_MAXRSS(R4),- ;FOR $SEARCH (DONE HERE SINCE IT
04 A4 1387 3722 NAM$L_RSA(R4) ;DOESN'T CHANGE DURING ENTIRE SEQUENCE)
1389 3723 $PARSE FAB=(R5) ;SETUP EXPANDED NAME STRING, NAM FIELDS
22 50 E9 1392 3724 BLBC R0,50$ ;(SUCH AS FID,DID,DVI,ESL)
1395 3725
1395 3726 :
1395 3727 : SEARCH CONTEXT BLOCK IS NOW ASSOCIATED WITH THIS STREAM (IT WAS EITHER
1395 3728 : FOUND IN THE LIST, OR CREATED). IF THE INPUT FILESPEC HAS CHANGED,
1395 3729 : THEN START OVER WITH NEW FILESPEC. OTHERWISE, SEARCH FOR THE NEXT
1395 3730 : FILESPEC IN SEQUENCE.
1395 3731
1395 3732 30$: $SEARCH FAB=(R5) ;SEARCH FOR NEXT FILESPEC
139E 3733 BLBC R0,50$ ;BRANCH IF ERROR DETECTED
51 03 A4 9A 13A1 3734 MOVZBL NAM$B_RSL(R4),R1 ;GET LENGTH OF RESULTANT STRING
52 04 A4 D0 13A5 3735 MOVL NAM$L_RSA(R4),R2 ;GET ADDRESS OF RESULTANT STRING
08 34 A4 08 E0 13A9 3736 BBS #NAM$V_WILDCARD,NAM$L_FNB(R4),40$ ;IF WILDCARD, LEAVE CONTEXT
```

```
7E 51 7D 13AE 3737      MOVQ   R1,-(SP)      ;SAVE RESULTANT STRING
    OC 10 13B1 3738      BSBB    DELCTX      ;DEALLOCATE THE CONTEXT BLOCK
51  8E 7D 13B3 3739      MOVQ   (SP)+,R1     ;RESTORE RESULTANT STRING
    05 13B6 3740 40$:    RSB
    13B7 3741
    13B7 3742
    13B7 3743 : AN ERROR HAS BEEN DURING EITHER THE $PARSE OR $SEARCH FUNCTIONS.
    13B7 3744 : IN ANY CASE, TERMINATE THIS SEARCH SEQUENCE BY REMOVING THE CONTEXT
    13B7 3745 : BLOCK FROM THE LIST. RETURN A NULL STRING. DO NOT REPORT THE ERROR,
    13B7 3746 : BUT SET THE STATUS.
    13B7 3747
    06 10 13B7 3748 50$:  BSBB    DELCTX      ;DEALLOCATE THE CONTEXT BLOCK
52  51 D4 13B9 3749      CLRL    R1           ;RETURN NULL STRING
    SE D0 13BB 3750      MOVL    SP,R2
    05 13BE 3751      RSB
    13BF 3752
    13BF 3753 :
    13BF 3754 : DEALLOCATE THE CONTEXT BLOCK
    13BF 3755 :
    13BF 3756 : R5 = ADDRESS OF FAB
    13BF 3757 : R6 = POINTER TO CONTEXT BLOCK
    13BF 3758 : R7 = POINTER TO PREVIOUS BLOCK IN LIST
    13BF 3759 :
    13BF 3760 : R0-R3 DESTROYED.
    13BF 3761
    34 A5 0000228D'EF 90 13BF 3762 DELCTX: MOV B NLA0,FAB$B FNS(R5) ;SET NULL DEVICE NAME
    2C A5 0000228E'EF 9E 13C7 3763      MOVAB NLA0+1,FAB$L FNA(R5)
    50 28 A5 D0 13CF 3764      MOVL    FAB$L_NAM(R5),R0 ;GET NAMBLK PTR
    04 13 13D3 3765      BEQL    10$ ;IF THERE IS ONE
    13D5 3766      SETBIT NAMS$V SYNCHK,NAMS$B_NOP(R0) ;SET SYNTAX ONLY CHECK
    67 66 D0 13D9 3767 10$: $PARSE FAB=(R5) ;TERMINATE SEARCH SEQUENCE
    50 56 D0 13E2 3768      MOVL    (R6),(R7) ;REMOVE FROM LINKED LIST
    51 04 A6 D0 13E8 3769      MOVL    R6,R0 ;SET ADDRESS OF BLOCK TO DEALLOCATE
    EC11' 30 13EC 3771      MOVL    4(R6),R1 ;SET SIZE OF BLOCK TO DEALLOCATE
    05 13EF 3772      BSBB    DCL$DEADYNMEM ;DEALLOCATE CONTEXT BLOCK
    RSB
```

```
13F0 3774 .SBTTL SET/CLEAR SPECIFIED PRIVILEGES
13F0 3775 :+
13F0 3776 : FSSETPRV - SET OR CLEAR THE SPECIFIED LIST OF PRIVILEGES
13F0 3777 :
13F0 3778 : THIS FUNCTION SETS OR CLEARS (DEPENDING ON WHETHER A 'NO' IS PREFIXED
13F0 3779 : TO THE PRIVILEGE KEYWORD) THE SPECIFIED LIST OF PRIVILEGES IN THE PROCESS
13F0 3780 : PERMANENT PRIVILEGE MASK. THE PREVIOUS STATES OF THE LISTED PRIVILEGES
13F0 3781 : ARE RETURNED AS THE VALUE OF THE FUNCTION. NOTE THAT ONLY THE PREVIOUS
13F0 3782 : STATES OF THE PRIVILEGES LISTED IN THE ARGUMENT ARE RETURNED, TO AVOID
13F0 3783 : DEALING WITH STRINGS OF EXCESSIVE LENGTH.
13F0 3784 :
13F0 3785 : FSSETPRV(PRVSTATES)
13F0 3786 :
13F0 3787 : INPUTS:
13F0 3788 :
13F0 3789 : P1(AP) = DESCRIPTOR OF A COMMA SEPARATED LIST OF PRIVILEGE KEYWORDS.
13F0 3790 :
13F0 3791 : OUTPUTS:
13F0 3792 :
13F0 3793 : R1/R2 = DESCRIPTOR OF COMMA SEPARATED LIST OF KEYWORDS INDICATING
13F0 3794 : THE PREVIOUS STATES OF THESE PRIVILEGES.
13F0 3795 :-
13F0 3796 :
13F0 3797 FSSETPRV:
13F0 3798 MOVAQ -(SP),R6 ;GET ADDRESS OF SCRATCH QUADWORD
13F3 3799 BSBW DCL$MARK ;MARK POSITION IN EXPANSION BUFFER
13F6 3800 $SETPRV_S PRVPRV=(R6),- ;GET CURRENT PERMANENT PRIVILEGES
13F6 3801 PRMFLG=#1
13F0 3802
13F0 3803 10$: MOVQ P1(AP),R4 ;GET DESCRIPTOR OF COMMA SEPARATED LIST
13F0 3804 MOVQ R4,R2 ;ASSUME NO COMMA IN LIST
13F0 3805 LOCC #^A',',R4,(R5) ;LOCATE END OF NEXT KEYWORD
13F0 3806 BEQL 20$ ;BRANCH IF NOT FOUND, USE ENTIRE STRING
13F0 3807 SUBL3 R0,R4,R2 ;ISOLATE NEXT KEYWORD
13F0 3808 MOVAB -(R0),R4 ;CHOP KEYWORD OFF STRING
13F0 3809 MOVAB 1(R1),R5
13F0 3810 PUSHR #^M<R2,R3> ;SAVE REGISTERS
13F0 3811 CMPC5 #3,W^NOALL+2,#0,R2,(R3) ;IS KEYWORD = 'ALL'?
13F0 3812 BEQL 22$ ;BRANCH IF SO
13F0 3813 CMPC5 #5,W^NOALL,#0,R2,(R3) ;IS KEYWORD = 'NOALL'?
13F0 3814 POPR #^M<R2,R3> ;RESTORE REGISTERS
13F0 3815 BNEQ 30$ ;BRANCH IF NOT
13F0 3816 CLRL R1 ;START WITH PRIVILEGE #0
13F0 3817 BSBW ADD PRIV STATE ;APPEND PRIVILEGE KEYWORD TO STRING
13F0 3818 AOBLS #64,R1,25$ ;FOR EACH PRIVILEGE IN MASK
13F0 3819 BRB 40$
13F0 3820 30$: MOVQ R2,-(SP) ;PUSH DESCRIPTOR OF KEYWORD
13F0 3821 PUSHAL -(SP) ;ADDRESS OF LONGWORD TO RECEIVE BIT #
13F0 3822 PUSHAB 8(SP) ;SET ADDRESS OF KEYWORD
13F0 3823 CALLS #2,G^PRV$PRIVBIT ;LOOKUP KEYWORD
13F0 3824 POPR #^M<R1,R2,R3> ;AND GET PRIVILEGE BIT NUMBER IN R1
13F0 3825 BLBC R0,51$ ;BRANCH IF ERROR DETECTED
13F0 3826 BSBB ADD PRIV STATE ;APPEND PRIVILEGE KEYWORD TO STRING
13F0 3827 MOVQ R2,-(SP) ;PUSH DESCRIPTOR OF KEYWORD
13F0 3828 PUSHL R6 ;SET ADDRESS OF PRIVILEGE MASK
13F0 3829 PUSHAB 4(SP) ;SET ADDRESS OF KEYWORD
13F0 3830 CALLS #2+2,G^PRV$SETPRIV ;ENABLE/DISABLE SPECIFIED PRIVILEGE
13F0 3831 BLBC R0,50$ ;BRANCH IF ERROR DETECTED
```

56 7E 7E 13F0 3798 MOVAQ -(SP),R6 ;GET ADDRESS OF SCRATCH QUADWORD
ECO A' 30 13F3 3799 BSBW DCL\$MARK ;MARK POSITION IN EXPANSION BUFFER
54 04 AC 7D 1405 3802 10\$: MOVQ P1(AP),R4 ;GET DESCRIPTOR OF COMMA SEPARATED LIST
52 54 7D 1409 3803 MOVQ R4,R2 ;ASSUME NO COMMA IN LIST
65 54 2C 3A 140C 3804 LOCC #^A',',R4,(R5) ;LOCATE END OF NEXT KEYWORD
54 04 13 1410 3805 BEQL 20\$;BRANCH IF NOT FOUND, USE ENTIRE STRING
52 54 50 C3 1412 3806 SUBL3 R0,R4,R2 ;ISOLATE NEXT KEYWORD
54 70 9E 1416 3807 MOVAB -(R0),R4 ;CHOP KEYWORD OFF STRING
55 01 A1 9E 1419 3808 MOVAB 1(R1),R5
63 52 00 2260'CF 03 2D 141F 3809 PUSHR #^M<R2,R3> ;SAVE REGISTERS
08 13 1427 3811 CMPC5 #3,W^NOALL+2,#0,R2,(R3) ;IS KEYWORD = 'ALL'?
63 52 00 225E'CF 05 2D 1429 3812 BEQL 22\$;BRANCH IF SO
0C BA 1431 3813 CMPC5 #5,W^NOALL,#0,R2,(R3) ;IS KEYWORD = 'NOALL'?
0F 12 1433 3814 POPR #^M<R2,R3> ;RESTORE REGISTERS
51 D4 1435 3815 BNEQ 30\$;BRANCH IF NOT
008C 30 1437 3816 CLRL R1 ;START WITH PRIVILEGE #0
F5 51 00000040 8F F2 143A 3817 BSBW ADD PRIV STATE ;APPEND PRIVILEGE KEYWORD TO STRING
16 11 1442 3818 AOBLS #64,R1,25\$;FOR EACH PRIVILEGE IN MASK
7E 52 7D 1444 3819 BRB 40\$
7E DF 1447 3820 30\$: MOVQ R2,-(SP) ;PUSH DESCRIPTOR OF KEYWORD
08 AE 9F 1449 3821 PUSHAL -(SP) ;ADDRESS OF LONGWORD TO RECEIVE BIT #
00000000'GF 02 FB 144C 3822 PUSHAB 8(SP) ;SET ADDRESS OF KEYWORD
0E BA 1453 3823 CALLS #2,G^PRV\$PRIVBIT ;LOOKUP KEYWORD
11 50 E9 1455 3824 POPR #^M<R1,R2,R3> ;AND GET PRIVILEGE BIT NUMBER IN R1
6C 10 1458 3825 BLBC R0,51\$;BRANCH IF ERROR DETECTED
7E 52 7D 145A 3826 BSBB ADD PRIV STATE ;APPEND PRIVILEGE KEYWORD TO STRING
56 DD 145D 3827 MOVQ R2,-(SP) ;PUSH DESCRIPTOR OF KEYWORD
04 AE 9F 145F 3828 PUSHL R6 ;SET ADDRESS OF PRIVILEGE MASK
00000000'GF 04 FB 1462 3829 PUSHAB 4(SP) ;SET ADDRESS OF KEYWORD
3B 50 E9 1469 3830 CALLS #2+2,G^PRV\$SETPRIV ;ENABLE/DISABLE SPECIFIED PRIVILEGE
51\$ BLBC R0,50\$;BRANCH IF ERROR DETECTED

Address	Disassembly	Comment
54	D5 146C 3831	TSTL R4 ;ANY MORE KEYWORDS?
99	14 146E 3832	BGTR 10\$;IF SO, LOOP UNTIL DONE
7E 01	CE 1470 3833	MNEGL #1,-(SP) ;CREATE PRIVILEGE MASK WITH ALL
7E 01	CE 1473 3834	MNEGL #1,-(SP) ;THE PRIVILEGE BITS SET
52 5E	DO 1476 3835	MOVL SP,R2 ;GET ADDRESS OF IT
	1479 3836	\$SETPRV_S PRVADR=(R2),- ;DISABLE ALL PRIVILEGES
	1479 3837	PRMFLG=#1,-
	1479 3838	ENBFLG=#0
	1488 3839	\$SETPRV_S PRVADR=(R6),- ;ENABLE NEW PRIVILEGES
	1488 3840	PRMFLG=#1,-
	1488 3841	ENBFLG=#1
SE 10	CO 1497 3842	ADDL #16,SP ;CLEANUP STACK
EB63'	30 149A 3843	BSBW DCL\$MARKEDTOKEN ;GET DESCRIPTOR OF STRING
F486 CA F48A CA	DO 149D 3844	MOVL WRK_L_MARKPTR(R10),WRK_L_EXPANDPTR(R10) ;RESTORE POSITION IN EXPANSI
51	D7 14A4 3845	DECL R1 ;DO NOT RETURN TRAILING COMMA
	05 14A6 3846	RSB
	14A7 3847	
50 00000000'8F	D1 14A7 3848 50\$:	CMPL #PRVS_NOTUNQ,R0 ;AMBIGUOUS KEYWORD?
09	12 14AE 3849	BNEQ 55\$;IF NOT, EXIT WITH INVALID KEYWORD
	14B0 3850	STATUS ABKEYW ;SET AMBIGUOUS KEYWORD ERROR
07	11 14B7 3851	BRB 80\$
	14B9 3852 55\$:	STATUS IVKEYW ;SET INVALID KEYWORD
SE 08	CO 14C0 3853 80\$:	ADDL #8,SP ;CLEANUP STACK
0152	31 14C3 3854	BRW P1_ERROR ;REPORT ERROR

```
14C6 3856 .SBTTL APPEND PRIVILEGE STATE TO STRING
14C6 3857 :---
14C6 3858 :ADD_PRIV_STATE - APPEND PRIVILEGE STATE (IN KEYWORD FORM) TO STRING.
14C6 3859 :
14C6 3860 :THIS ROUTINE APPENDS THE PRIVILEGE KEYWORD ASSOCIATED WITH A GIVEN
14C6 3861 :PRIVILEGE BIT AND MASK TO THE CURRENT STRING BEING ASSEMBLED IN THE
14C6 3862 :EXPANSION BUFFER. IF THE PRIVILEGE BIT NUMBER DOES NOT CORRESPOND
14C6 3863 :TO A KEYWORD, NOTHING IS APPENDED TO THE STRING AND NO ERROR IS RETURNED.
14C6 3864 :
14C6 3865 :INPUTS:
14C6 3866 :
14C6 3867 :R1 = PRIVILEGE BIT NUMBER
14C6 3868 :R6 = ADDRESS OF PRIVILEGE MASK
14C6 3869 :
14C6 3870 :OUTPUTS:
14C6 3871 :
14C6 3872 :THE KEYWORD (NEGATED IF THE PRIVILEGE IS OFF) IS APPENDED TO THE
14C6 3873 :STRING.
14C6 3874 :
14C6 3875 :R0 DESTROYED.
14C6 3876 :---
14C6 3877 :
14C6 3878 :ADD_PRIV_STATE:
14C6 3879 :PUSHL R1 ;SAVE PRIVILEGE BIT NUMBER
14C6 3880 :PUSHL WRK_L_EXPANDPTR(R10) ;SAVE CURRENT POSITION IN BUFFER
14C6 3881 :BBS R1,7(R6),30$ ;BRANCH IF PRIVILEGE WAS PREVIOUSLY SET
14C6 3882 :MOVZBL #^A^N^,R0 ;INSERT AN 'N' INTO THE BUFFER
14C6 3883 :BSBW DCL$PUTCHAR
14C6 3884 :MOVZBL #^A^O^,R0 ;INSERT AN 'O' INTO THE BUFFER
14C6 3885 :BSBW DCL$PUTCHAR
14C6 3886 30$: PUSHL WRK_L_EXPANDPTR(R10) ;CREATE DESCRIPTOR OF EXPANSION BUFFER
14C6 3887 :SUBL3 (SP),WRK_L_CHARPTR(R10),-(SP) ;SCRATCH AREA LEFT
14C6 3888 :PUSHL SP ;ADDRESS OF WORD TO GET RETLEN
14C6 3889 :PUSHAB 4(SP) ;ADDRESS OF BUFFER DESCRIPTOR
14C6 3890 :PUSHL R1 ;SET PRIVILEGE BIT NUMBER
14C6 3891 :CALLS #3,G^PRV$KEYWORD ;GET KEYWORD ASSOCIATED WITH BIT NUMBER
14C6 3892 :ADDL3 (SP)+,(SP)+,R1 ;POINT TO STRING END+1
14C6 3893 :BLBC R0,50$ ;IF NO SUCH PRIVILEGE, APPEND NOTHING
14C6 3894 :ADDL #4,SP ;THROW AWAY SAVED EXPANSION POINTER
14C6 3895 :MOVL R1,WRK_L_EXPANDPTR(R10) ;INSERT INTO EXPANSION BUFFER
14C6 3896 :MOVZBL #^A^,R0 ;INSERT AN '^' INTO THE BUFFER
14C6 3897 :BSBW DCL$PUTCHAR
14C6 3898 90$: POPL R1 ;RESTORE PRIVILEGE BIT NUMBER
14C6 3899 :RSB
14C6 3900 :
14C6 3901 50$: POPL WRK_L_EXPANDPTR(R10) ;BACKUP POSITION IN BUFFER
14C6 3902 :BRB 90$ ;AND EXIT WITHOUT APPENDING ANYTHING
```

51	DD	14C6	3879
F486 CA	DD	14C8	3880
OE 66 51	EO	14CC	3881
50 4E 8F	9A	14D0	3882
EB29'	30	14D4	3883
50 4F 8F	9A	14D7	3884
EB22'	30	14DB	3885
F486 CA	DD	14DE	3886
7E F48E CA	6E C3	14E2	3887
5E	DD	14E8	3888
04 AE	9F	14EA	3889
51	DD	14ED	3890
00000000'GF	03 FB	14EF	3891
51 8E	8E C1	14F6	3892
12 50	E9	14FA	3893
5E 04	C0	14FD	3894
F486 CA	51 D0	1500	3895
50 2C	9A	1505	3896
EAF5'	30	1508	3897
51 8ED0	150B	3898	90\$:
05	150E	3899	
	150F	3900	
F486 CA	8ED0	150F	3901
F5 11	1514	3902	50\$:

```
1516 3904 .SBTTL CONVERT EXPRESSION TO ASCII STRING
1516 3905 :+
1516 3906 : F$STRING - CONVERT TO ASCII STRING
1516 3907 :
1516 3908 : THIS FUNCTION CONVERTS ANY EXPRESSION INTO AN ASCII STRING. IF THE
1516 3909 : EXPRESSION IS A BINARY VALUE, IT IS CONVERTED INTO DECIMAL ASCII.
1516 3910 :
1516 3911 : F$STRING(EXPRESSION)
1516 3912 :
1516 3913 : INPUTS:
1516 3914 :
1516 3915 : P1(AP) = EXPRESSION VALUE
1516 3916 :
1516 3917 : OUTPUTS:
1516 3918 :
1516 3919 : R1/R2 = DESCRIPTOR OF VALUE STRING
1516 3920 :-
1516 3921 :
1516 3922 F$STRING:
51 04 AC 7D 1516 3923 MOVQ P1(AP),R1 ;GET EXPRESSION VALUE
00000000'EF 16 151A 3924 JSB DCL$CVT_STRING ;CONVERT TO ASCII STRING
01 50 E9 1520 3925 BLBC R0,90$ ;BRANCH IF ERROR DETECTED
00F1 05 1523 3926 RSB
31 1524 3927 90$: BRW P1_ERROR ;REPORT ERROR AND DISPLAY P1
```

```
1527 3929 .SBTTL RETURN CURRENT TIME AND DATE
1527 3930 :+
1527 3931 : F$TIME - RETURN CURRENT TIME AND DATE
1527 3932 :
1527 3933 : THIS LEXICAL FUNCTION OBTAINS THE CURRENT TIME AND DATE AND RETURNS A STRING
1527 3934 : DESCRIPTOR FOR IT.
1527 3935 :
1527 3936 : F$TIME()
1527 3937 :
1527 3938 : INPUTS:
1527 3939 :
1527 3940 : NONE.
1527 3941 :
1527 3942 : OUTPUTS:
1527 3943 :
1527 3944 : R1 = LENGTH OF TIME AND DATE STRING.
1527 3945 : R2 = ADDRESS OF TIME AND DATE STRING.
1527 3946 :-
1527 3947 :
1527 3948 F$TIME:
7E 52 7D 1527 3949 MOVQ R2, -(SP) ;RETURN TIME AND DATE
6E 17 D1 152A 3950 CMPL #23, (SP) ;BUILD TIME BUFFER DESCRIPTOR
19 1A 152D 3951 BGTRU 50$ ;TIME BUFFER LARGE ENOUGH?
50 5E D0 152F 3952 MOVL SP, R0 ;IF GTRU NO
51 8E 7D 1532 3953 $ASCTIM_S (R0), (R0) ;SET ADDRESS OF TIME BUFFER DESCRIPTOR
04 50 E9 1541 3954 MOVQ -(SP)+, R1 ;GET CURRENT TIME
05 1544 3955 BLBC R0, 60$ ;RETRIEVE TIME BUFFER PARAMETERS
1547 3956 RSB ;IF LBC CONVERSION FAILURE
1548 3957 :
00C1 31 1548 3958 50$: BRW BUFOVF ;REPORT BUFFER OVERFLOW
00C5 31 154B 3959 60$: BRW ERROR ;REPORT SYSTEM SERVICE ERROR
```

```
154E 3961 .SBTTL GET SYMBOL TYPE
154E 3962 :+
154E 3963 : FSTYPE - GET SYMBOL TYPE
154E 3964 :
154E 3965 : THIS FUNCTION RETURNS NULL, 'INTEGER', OR 'STRING' CHARACTER STRINGS
154E 3966 : DEPENDING ON WHETHER THE SPECIFIED SYMBOL IS UNDEFINED, EQUATED TO A
154E 3967 : NUMERIC STRING OR AN INTEGER, OR EQUATED TO A NON-NUMERIC STRING.
154E 3968 :
154E 3969 : FSTYPE (SYMBOL)
154E 3970 :
154E 3971 : INPUTS:
154E 3972 :
154E 3973 : P1(AP) = DESCRIPTOR OF SYMBOL NAME
154E 3974 :
154E 3975 : OUTPUTS:
154E 3976 :
154E 3977 : R1/R2 = DESCRIPTOR OF ASCII STRING DESCRIBING THE SYMBOL TYPE
154E 3978 : -
154E 3979 :
154E 3980 FSTYPE:
154E 3981 MOVQ R2, -(SP) ;SAVE BUFFER DESCRIPTOR
154E 3982 MOVQ P1(AP), R1 ;GET DESCRIPTOR OF SYMBOL NAME
154E 3983 BSBW DCL$SEARCH ;FIND SYMBOL IN SYMBOL TABLE
154E 3984 BLBC R0, 20$ ;IF NOT FOUND, RETURN NULL
154E 3985 TSTL R2 ;STRING DESCRIPTOR?
154E 3986 BEQL 10$ ;NO, THEN BRANCH
154E 3987 TSTL R1 ;NULL STRING?
154E 3988 BEQL 5$ ;YES, THEN BRANCH
154E 3989 CMPL R1, (SP) ;BUFFER LARGE ENOUGH?
154E 3990 BGTRU 90$ ;IF GTRU NO
154E 3991 MOVL R1, (SP) ;COPY SYMBOL LENGTH
154E 3992 MOVC3 R1, (R2), 24(SP) ;COPY THE SYMBOL VALUE
154E 3993 MOVQ (SP), R2 ;MOVE THE DESCRIPTOR INTO R2/R3
154E 3994 MOVL #PRC_K_DEC, R1 ;SET CONVERSION RADIX
154E 3995 BSBW DCL$CNVASCBIN ;CONVERT THE STRING TO DECIMAL
154E 3996 BEQL 10$ ;IF SUCCESSFUL, THEN INTEGER
154E 3997 5$: MOVAB STRING, R2 ;IF NOT, THEN RETURN 'STRING'
154E 3998 MOVZBL (R2)+, R1
154E 3999 BRB 20$
154E 4000 10$: MOVAB INTEGER, R2 ;RETURN 'INTEGER'
154E 4001 MOVZBL (R2)+, R1
154E 4002 20$: ADDL #8, SP ;RESTORE THE STACK
154E 4003 RSB
154E 4004
154E 4005 90$: ADDL #8, SP ;RESTORE THE STACK
154E 4006 BRW BUF$VF ;REPORT BUFFER OVERFLOW
```

51	7E	52	7D	154E	3981	MOVQ	R2, -(SP)	;SAVE BUFFER DESCRIPTOR
	04	AC	7D	1551	3982	MOVQ	P1(AP), R1	;GET DESCRIPTOR OF SYMBOL NAME
	EA	A8	30	1555	3983	BSBW	DCL\$SEARCH	;FIND SYMBOL IN SYMBOL TABLE
	36	50	E9	1558	3984	BLBC	R0, 20\$	;IF NOT FOUND, RETURN NULL
		52	D5	155B	3985	TSTL	R2	;STRING DESCRIPTOR?
		28	13	155D	3986	BEQL	10\$	;NO, THEN BRANCH
		51	D5	155F	3987	TSTL	R1	;NULL STRING?
		18	13	1561	3988	BEQL	5\$	;YES, THEN BRANCH
	6E	51	D1	1563	3989	CMPL	R1, (SP)	;BUFFER LARGE ENOUGH?
		2D	1A	1566	3990	BGTRU	90\$	;IF GTRU NO
04	BE	6E	51	1568	3991	MOVL	R1, (SP)	;COPY SYMBOL LENGTH
		62	51	156B	3992	MOVC3	R1, (R2), 24(SP)	;COPY THE SYMBOL VALUE
		52	6E	1570	3993	MOVQ	(SP), R2	;MOVE THE DESCRIPTOR INTO R2/R3
		51	01	1573	3994	MOVL	#PRC_K_DEC, R1	;SET CONVERSION RADIX
		EA	87	1576	3995	BSBW	DCL\$CNVASCBIN	;CONVERT THE STRING TO DECIMAL
		0C	13	1579	3996	BEQL	10\$	;IF SUCCESSFUL, THEN INTEGER
52		0000226E	9E	157B	3997	5\$: MOVAB	STRING, R2	;IF NOT, THEN RETURN 'STRING'
		51	82	1582	3998	MOVZBL	(R2)+, R1	
		0A	11	1585	3999	BRB	20\$	
52		00002275	9E	1587	4000	10\$: MOVAB	INTEGER, R2	;RETURN 'INTEGER'
		51	82	158E	4001	MOVZBL	(R2)+, R1	
		5E	08	1591	4002	20\$: ADDL	#8, SP	;RESTORE THE STACK
			05	1594	4003	RSB		
		5E	08	1595	4004			
		0071	31	1595	4005	90\$: ADDL	#8, SP	;RESTORE THE STACK
				1598	4006	BRW	BUF\$VF	;REPORT BUFFER OVERFLOW

```
159B 4008 .SBTTL RETURN USER IDENTIFICATION CODE
159B 4009 :+
159B 4010 : FSUSER - RETURN USER IDENTIFICATION CODE
159B 4011 :
159B 4012 : THIS LEXICAL FUNCTION CONVERTS THE USER IDENTIFICATION CODE TO AN
159B 4013 : ASCII STRING AND RETURNS A DESCRIPTOR FOR IT.
159B 4014 :
159B 4015 : FSUSER()
159B 4016 :
159B 4017 : INPUTS:
159B 4018 :
159B 4019 : NONE.
159B 4020 :
159B 4021 : OUTPUTS:
159B 4022 :
159B 4023 : R1 = LENGTH OF USER IDENTIFICATION CODE.
159B 4024 : R2 = ADDRESS OF USER IDENTIFICATION CODE.
159B 4025 :-
159B 4026 :
159B 4027 FSUSER:
159B 4028 CLRQ -(SP) ;RETURN USER IDENTIFICATION CODE
159B 4029 CLRQ -(SP) ;RESERVE IOSB
159B 4030 PUSHAB -8(SP) ;CLEAR LAST ITEM AND LENGTH ADDRESS
159B 4031 PUSHL #JPI$_UIC@16+4 ;PUSH ADDRESS TO STORE USER IDENTIFICATION
159B 4032 MOVL SP,R0 ;SET PARAMETER ID AND LENGTH
159B 4033 $GETJPIW S ITMLST=(R0),- ;SET ADDRESS OF TIME LIST
159B 4034 IOSB=16(R0),- ;GET USER IDENTIFICATION CODE
159B 4035 EFN=#EXESC_$YSEFN ;
159B 4036 BLBC R0,20$ ;IF LBC SERVICE FAILURE
159B 4037 MOVZWL 16(SP),R0 ;GET STATUS FROM IOSB
159B 4038 BLBC R0,20$ ;IF LBC SERVICE FAILURE
159B 4039 MOVL (SP),R4 ;GET UIC
159B 4040 BSBW FORMAT UIC ;FORMAT THE UIC
159B 4041 ADDL #6*4,SP ;REMOVE ITEM LIST AND IOSB FROM STACK
159B 4042 RSB ;
159B 4043 15D7 4043
159B 4044 20$: ADDL #6*4,SP ;REMOVE ITEM LIST AND IOSB FROM STACK
159B 4045 BRB ERROR ;
```

03040004 7E 7C 159B 4028
50 5E DD 15A2 4031
F8 AE 9F 159F 4030
8F DD 15A8 4032
5E DO 15AB 4033
15AB 4034
15AB 4035
50 11 50 E9 15C3 4036
10 AE 3C 15C6 4037
0A 50 E9 15CA 4038
54 6E DO 15CD 4039
009C 30 15D0 4040
5E 18 C0 15D3 4041
05 15D6 4042
15D7 4043
5E 18 C0 15D7 4044
37 11 15DA 4045

```
15DC 4047 .SBTTL RETURN VERIFY MODE
15DC 4048 :+
15DC 4049 : FSVERIFY - RETURN VERIFY MODE
15DC 4050 :
15DC 4051 : THIS LEXICAL FUNCTION RETURNS A BOOLEAN VARIABLE THAT DESCRIBES THE CURRENT
15DC 4052 : STATE OF THE VERIFY MODE AND, IN ADDITION, WILL SET OR CLEAR VERIFICATION
15DC 4053 : IF A NON-NULL ARGUMENT IS SPECIFIED.
15DC 4054 :
15DC 4055 : FSVERIFY([NEW_VERIFY_MODE [, NEW_IMAGE_VERIFY_MODE]])
15DC 4056 :
15DC 4057 : INPUTS:
15DC 4058 :
15DC 4059 : P1(AP) = OPTIONAL NEW VERIFY MODE (1 TO SET, 0 TO CLEAR)
15DC 4060 : (-1 IF UNSPECIFIED)
15DC 4061 : P2(AP) = OPTIONAL NEW IMAGE VERIFY MODE (1 TO SET, 0 TO CLEAR)
15DC 4062 : (-1 IF UNSPECIFIED)
15DC 4063 :
15DC 4064 : OUTPUTS:
15DC 4065 :
15DC 4066 : R1 = BOOLEAN RESULT
15DC 4067 : R2 = 0 TO INDICATE RESULT IS BINARY
15DC 4068 :-
15DC 4069
15DC 4070 FSVERIFY: ;RETURN VERIFY MODE
15DC 4071 MOVL #1,R8 ;ASSUME VERIFY IS CURRENTLY ON
02 68 AB 58 01 D0 15DC 4072 BBS #PRC_V_VERIFY,PRC_W_FLAGS(R11),10$ ;BR IF ON
15DC 4073 CLRL R8 ;MARK VERIFY TURNED OFF
15DC 4074 10$: TSTL P1(AP) ;A NON-NULL ARGUMENT?
15DC 4075 BLSS 90$ ;NOPE, IT'S NULL, NO CHANGES
15DC 4076 CLRBIT PRC_V_VERIFY,PRC_W_FLAGS(R11) ;TURN OFF VERIFICATION
05 04 AC E9 15DC 4077 BLBC P1(AP),20$ ;LEAVE VERIFY OFF IF LBC
15DC 4078 SETBIT PRC_V_VERIFY,PRC_W_FLAGS(R11) ;ELSE TURN ON VERIFICATION
56 0C AC D0 15DC 4079 20$: MOVL P2(AP),R6 ;A NON-NULL ARGUMENT?
15DC 4080 BGEQ 30$ ;BRANCH IF NON-NULL
56 04 AC D0 15DC 4081 MOVL P1(AP),R6 ;ELSE USE FIRST ARGUMENT
15DC 4082 30$: BSBW DCL$SETVERIFY_IMAGE ;SET IMAGE VERIFICATION
51 58 D0 1606 4083 90$: MOVL R8,R1 ;RETURN BOOLEAN RESULT
15DC 4084 CLRL R2 ;INDICATE RESULT IS BINARY
15DC 4085 RSB ;
```

```
160C 4087 .SBTTL ERROR RETURNS
160C 4088
160C 4089 :
160C 4090 : BUFFER OVERFLOW
160C 4091 :
160C 4092 :
160C 4093 BUFOVF: STATUS BUFOVF ;SET BUFFER OVERFLOW STATUS
E9EA' 30 1613 4094 ERROR: BSBW DCL$MARK ;SET FOR NO SEGMENT DISPLAY
39 11 1616 4095 BRB DCL$PARSERR
1618 4096
1618 4097 :
1618 4098 : SIGNAL ERROR WITH AN ARGUMENT AS THE ERROR SEGMENT
1618 4099 :
1618 4100 :
1618 4101 P1_ERROR:
52 04 AC 9E 1618 4102 MOVAB P1(AP),R2 ;USE P1 AS ERROR SEGMENT
1C 11 161C 4103 BRB PN_ERROR
161E 4104 P2_ERROR:
52 0C AC 9E 161E 4105 MOVAB P2(AP),R2 ;USE P2 AS ERROR SEGMENT
16 11 1622 4106 BRB PN_ERROR
1624 4107 P3_ERROR:
52 14 AC 9E 1624 4108 MOVAB P3(AP),R2 ;USE P3 AS ERROR SEGMENT
10 11 1628 4109 BRB PN_ERROR
162A 4110 P4_ERROR:
52 1C AC 9E 162A 4111 MOVAB P4(AP),R2 ;USE P4 AS ERROR SEGMENT
0A 11 162E 4112 BRB PN_ERROR
1630 4113 P5_ERROR:
52 24 AC 9E 1630 4114 MOVAB P5(AP),R2 ;USE P5 AS ERROR SEGMENT
04 11 1634 4115 BRB PN_ERROR
1636 4116 P6_ERROR:
52 2C AC 9E 1636 4117 MOVAB P6(AP),R2 ;USE P6 AS ERROR SEGMENT
163A 4118
163A 4119 PN_ERROR:
OF FO AA 51 D4 163A 4120 CLRL R1 ;PRESET TO RETURN NULL STRING
F48A CA 04 A2 E0 163C 4121 BBS #WRK_V COMMENT,WRK_W FLAGS(R10),30$ ;RETURN NULL IF IN COMMENT
F486 CA 04 A2 62 C1 1641 4122 MOVL 4(R2),WRK_L MARKPTR(R10) ;SET START OF ERROR SEGMENT
01 11 1647 4123 ADDL3 (R2),4(R2),WRK_L_EXPANDPTR(R10) ;SET END OF ERROR SEGMENT
05 164E 4124 BRB DCL$PARSERR ;REPORT ERROR
1650 4125 30$: RSB ;RETURN WITH FUNCTION=NULL/ZERO
1651 4126
1651 4127 :
1651 4128 : ERROR EXIT FROM LEXICAL FUNCTION STRING EXPANSION
1651 4129 :
1651 4130 :
1651 4131 DCL$PARSERR::
E9AC' 30 1651 4132 BSBW DCL$ERRORMSG ;OUTPUT ERROR MESSAGE
50 DD 1654 4133 PUSHL R0 ;SAVE FINAL STATUS
00000000'EF 16 1656 4134 JSB DCL$FLUSH ;FLUSH COMMAND
50 8ED0 165C 4135 POPL R0 ;RETRIEVE FINAL STATUS
00000000'EF 16 165F 4136 JSB DCL$SET_STATUS ;SET COMMAND STATUS
00000000'EF 17 1665 4137 JMP DCL$RESTART ;
```

```
166B 4139 .SBTTL CONVERT UIC TO ASCII STRING
166B 4140 :+
166B 4141 :+ FORMAT_UIC - CONVERT BINARY UIC TO ASCII STRING
166B 4142 :+
166B 4143 :+ CONVERTS A BINARY UIC LONGWORD INTO AN ASCII STRING
166B 4144 :+
166B 4145 :+ INPUTS:
166B 4146 :+
166B 4147 :+ R4 = UIC LONGWORD
166B 4148 :+
166B 4149 :+ OUTPUTS:
166B 4150 :+
166B 4151 :+ R1/R2 = DESCRIPTOR OF ASCII STRING
166B 4152 :+
166B 4153 :+
166B 4154 :+
49 25 21 00' 166B 4155 UIC: .ASCIC '!%I'
03 166B
166F 4156
166F 4157 FORMAT_UIC:
51 F9 AF 9E 166F 4158 MOVAB UIC,R1 ;GET ADDRESS OF ASCIC FAO STRING
50 81 9A 1673 4159 MOVZBL (R1)+,R0 ;CREATE DESCRIPTOR OF FAO STRING
7E 50 7D 1676 4160 MOVQ R0,-(SP) ;PUSH ONTO STACK
F486 CA DD 1679 4161 PUSHL WRK_L_EXPANDPTR(R10) ;CREATE DESCRIPTOR OF SCRATCH BUFFER
F486 CA C3 167D 4162 SUBL3 WRK_L_EXPANDPTR(R10),-
7E F48E CA 1681 4163 WRK_L_CHARPTR(R10),-(SP)
50 5E D0 1685 4164 MOVL SP,R0
1688 4165 $FAO_S CTRSTR=8(R0),- ;FORMAT INTO ASCII STRING
1688 4166 OUTBUF=(R0),- ;ADDRESS OF RESULT BUFFER
1688 4167 OUTLEN=(R0),- ;WORD TO RECEIVE RESULT LENGTH
1688 4168 P1=R4 ;UIC NUMBER
1E BA 1698 4169 POPR #^M<R1,R2,R3,R4> ;GET DESCRIPTOR OF STRING
169A 4170 ;AND CLEAN STACK
05 169A 4171 RSB
```

```
169B 4173 .SBTTL CONVERT PROTECTION MASK TO ASCII STRING
169B 4174 ;+
169B 4175 ; FORMAT_PROT - CONVERT BINARY PROTECTION MASK TO ASCII STRING
169B 4176 ;
169B 4177 ; CONVERTS A BINARY PROTECTION MASK WORD INTO AN ASCII STRING
169B 4178 ;
169B 4179 ; INPUTS:
169B 4180 ;
169B 4181 ; R4 = PROTECTION WORD
169B 4182 ; R10 = BASE ADDRESS OF COMMAND WORK AREA
169B 4183 ;
169B 4184 ; OUTPUTS:
169B 4185 ;
169B 4186 ; R1/R2 = DESCRIPTOR OF ASCII STRING
169B 4187 ;
169B 4188 ; REGISTER USAGE:
169B 4189 ;
169B 4190 ; R0,R1,R2,R3,R7,R8
169B 4191 ; -
169B 4192 ;
169B 4193 NULL: .BYTE 0
3D 00 169C 4194 EQUAL: .ASCIC /=/
01 169C
44 45 57 52 169E 4195 ACCESS: .ASCII /RWED/
16A2 4196 PROTECTMSG:
44 41 21 43 41 21 4D 45 54 53 59 53 16A2 4197 .ASCII /SYSTEM!AC!AD, OWNER!AC!AD, GROUP!AC!AD, WORLD!AC!AD/
41 21 43 41 21 52 45 4E 57 4F 20 2C 16AE
21 43 41 21 50 55 4F 52 47 20 2C 44 16BA
43 41 21 44 4C 52 4F 57 20 2C 44 41 16C6
44 41 21 16D2
16D5 4198 PROTECTEND:
16D5 4199
16D5 4200 FORMAT_PROT:
53 5E D0 16D5 4201 MOVL SP,R3 ;START A SCRATCH AREA ON THE STACK
5E 30 C2 16D8 4202 SUBL #4*12,SP ;ALLOCATE FAO ARGUMENT DESCRIPTORS
;AND ASCII ADDRESS DESCRIPTORS
58 0C 9A 16DB 4203
7E D4 16DE 4204
73 7C 16E0 4205 MOVZBL #12,R8 ;SET OUTER LOOP INDEX
73 B6 AF 9E 16E2 4206 10$: CLRL -(SP) ;ALLOCATE SPACE FOR FAO ARG BUFFER
51 54 04 58 EF 16E6 4207 CLRQ -(R3) ;ZERO FAO ARG DESCRIPTOR
52 51 D2 16E8 4208 MOVAB NULL, -(R3) ;INIT ASCII POINTER, ASSUME NO ACCESS
52 0F D3 16EB 4209 EXTZV R8,#4,R4,R1 ;EXTRACT ONE PROTECTION FIELD
1F 13 16EE 4210 MCOML R1,R2 ;COMPLEMENT IT
63 A6 AF 9E 16F1 4211 BITL #^XF,R2 ;ALL ACCESS DENIED?
57 D4 16F3 4212 BEQL 40$ ;IF EQL, YES
08 A3 6E 9E 16F7 4213 MOVAB EQUAL,(R3) ;SET POINTER TO ASCII =
04 A3 D4 16F9 4214 CLRL R7 ;CLEAR INNER LOOP INDEX
1700 4215 MOVAB (SP),8(R3) ;PUT ADDR OF BUFFER IN DESCRIPTOR
1700 4216 CLRL 4(R3) ;CLEAR BUFFER LENGTH
0A 51 57 E0 1700 4217
04 B34E 96 AF47 90 1700 4218 20$: BBS R7,R1,30$ ;IF SET, ACCESS DENIED
EE 57 04 F2 1704 4219 MOVAB ACCESS[R7],24(R3)[SP] ;MOVE ACCESS CHARACTER INTO BUFFER
FFC5 58 FC 8F 00 9D 170B 4220 INCL 4(R3) ;INCREMENT BUFFER LENGTH
1712 4221 30$: AOBLS #4,R7,20$ ;CHECKED ALL FOUR BITS?
1719 4222
1719 4223 40$: ACBB #0,#-4,R8,10$ ;CHECKED ALL FOUR MODES?
1719 4224
```

LEXICON
V04-000

M 13
- LEXICAL FUNCTION PARSE AND EXECUTION R 16-SEP-1984 00:00:36 VAX/VMS Macro V04-00
CONVERT PROTECTION MASK TO ASCII STRING 4-SEP-1984 23:41:41 [DCL.SRC]LEXICON.MAR;1

Page 95
(43)

86	AF	9F	1719	4225	PUSHAB	PROTECTMSG	;PUSH DESCRIPTOR FOR CONTROL STRING
	33	DD	171C	4226	PUSHL	#PROTECTEND-PROTECTMSG	
F486	CA	DD	171E	4227	PUSHL	WRK_L_EXPANDPTR(R10)	;PUSH DESCRIPTOR TO RECEIVE FAO OUTPUT
	37	DD	1722	4228	PUSHL	#PROTECTEND-PROTECTMSG+4	
50	5E	DD	1724	4229	MOVL	SP,R0	;SAVE POINTER TO TOP OF STACK
			1727	4230	\$FAOL_S	CTRSTR=8(R0),-	;GENERATE FORMATTED ASCII STRING
			1727	4231		OUTBUF=(R0),-	
			1727	4232		OUTLEN=(R0),-	
			1727	4233		PRMLST=(R3)	
			1737	4234			
	51	6E	7D	1737	MOVQ	(SP),R1	;MOVE OUTPUT DESCRIPTOR INTO R1/R2
SE	00000050	8F	CO	173A	ADDL	#20*4,SP	;RESTORE STACK
			05	1741	RSB		
				4237			

```
1742 4239 .SBTTL CONVERT FILE ID TO ASCII STRING
1742 4240 ;+
1742 4241 ; FORMAT_FILEID - CONVERT FILE ID TO ASCII STRING
1742 4242 ;
1742 4243 ; INPUTS:
1742 4244 ;
1742 4245 ; R4 = ADDRESS OF 3-WORD FILE ID
1742 4246 ;
1742 4247 ; OUTPUTS:
1742 4248 ;
1742 4249 ; R1/R2 = DESCRIPTOR OF ASCII STRING
1742 4250 ; -
1742 4251 ;
55 21 2C 57 55 21 2C 57 55 21 28 00' 1742 4252 FILEID: .ASCIC '(!UW,!UW,!UB)'
29 42 174E
0D 1742
1750 4253
1750 4254 FORMAT_FILEID:
1750 4255 MOVAB FILEID,R1 ;GET ADDRESS OF ASCIC FAO STRING
1754 4256 MOVZBL (R1)+,R0 ;CREATE DESCRIPTOR OF FAO STRING
1757 4257 MOVQ R0,-(SP) ;PUSH ONTO STACK
175A 4258 PUSHL WRK_L_EXPANDPTR(R10) ;CREATE DESCRIPTOR OF SCRATCH BUFFER
175E 4259 SUBL3 WRK_L_EXPANDPTR(R10),WRK_L_CHARPTR(R10),-(SP)
1766 4260 MOVL SP,R0
1769 4261 $FAO_S CTRSTR=8(R0),- ;FORMAT INTO ASCII STRING
1769 4262 OUTBUF=(R0),- ;ADDRESS OF RESULT BUFFER
1769 4263 OUTLEN=(R0),- ;WORD TO RECEIVE RESULT LENGTH
1769 4264 P1=(R4),- ;FILE NUMBER
1769 4265 P2=2(R4),- ;SEQUENCE NUMBER
1769 4266 P3=4(R4) ;RELATIVE VOLUME NUMBER
1E BA 177F 4267 POPR #^M<R1,R2,R3,R4> ;GET DESCRIPTOR OF STRING
1781 4268 ;AND CLEAN STACK
05 1781 4269 RSB
```

```
1782 4271 .SBTTL CONVERT BINARY TO BOOLEAN STRING
1782 4272 :+
1782 4273 :+ FORMAT_BOOLEAN - CONVERT BINARY TO ASCII BOOLEAN STRING
1782 4274 :+
1782 4275 :+ INPUTS:
1782 4276 :+
1782 4277 :+ R1 = STATUS (ONLY BOTTOM BIT CHECKED)
1782 4278 :+
1782 4279 :+ OUTPUTS:
1782 4280 :+
1782 4281 :+ R1/R2 = DESCRIPTOR OF ASCII STRING (TRUE OR FALSE)
1782 4282 :+
1782 4283 :+
1782 4284 FORMAT_BOOLEAN:
52 00002263'EF 9E 1782 4285 MOVAB FALSE,R2 ;ASSUME FALSE
   07 51 E9 1789 4286 BLBC R1,10$ ;BRANCH IF FALSE
52 00002269'EF 9E 178C 4287 MOVAB TRUE,R2 ;SET TRUE
   51 82 9A 1793 4288 10$: MOVZBL (R2)+,R1 ;SET DESCRIPTOR OF BOOLEAN STRING
   05 1796 4289 RSB
1797 4290
1797 4291 .END
```

LEXICON
Symbol table- LEXICAL FUNCTION PARSE AND EXECUTION R 16-SEP-1984 00:00:36 VAX/VMS Macro V04-00
4-SEP-1984 23:41:41 [DCL.SRC]LEXICON.MAR;1Page 98
(45)

\$\$TMP1	= 00000001			DCL\$DEADYNMEM	*****	X	03
\$\$TMP2	= 00000065			DCL\$DISABLE	*****	X	03
\$\$T1	= 00000001			DCL\$EDIT	*****	X	03
\$\$T2	= 00000006			DCL\$ERRORMSG	*****	X	03
\$ARG\$	= 00000044			DCL\$EXPRESS	*****	X	03
\$BITS	= 00000008			DCL\$FLUSH	*****	X	03
\$XX\$	= 00000004			DCL\$GETOKEN	*****	X	03
\$ZZ\$	= 00000004			DCL\$LEXIF	00000000	RG	03
ACCESS	0000169E	R	03	DCL\$MARK	*****	X	03
ACCESS TABLE	0000204C	R	02	DCL\$MARKEDTOKEN	*****	X	03
ACPSV_READCHK	= 00000000	G		DCL\$MOVCHAR	*****	X	03
ACPSV_SWAPGRP	= 00000001	G		DCL\$PARSERR	00001651	RG	03
ACPSV_SWAPMAG	= 00000003	G		DCL\$PUTCHAR	*****	X	03
ACPSV_SWAPPRV	= 00000002	G		DCL\$RESTART	*****	X	03
ACPSV_SWAPSYS	= 00000000	G		DCL\$SEARCH	*****	X	03
ACPSV_WRITECHK	= 00000001	G		DCL\$SETNBLK	*****	X	03
ACP TABLE	00001E5E	R	02	DCL\$SETVERIFY IMAGE	*****	X	03
ADD_PRIV STATE	000014C6	R	03	DCL\$SET STATUS	*****	X	03
ARG_C_ANY	= 00000003			DCL\$SHOWDEFINT	*****	X	03
ARG_C_BINARY	= 00000000			DCL\$SQUEEZE	*****	X	03
ARG_C_ENTSIZE	= 00000008			DCL\$STRIM	*****	X	03
ARG_C_STRING	= 00000001			DCL\$UPCASE	*****	X	03
ARG_C_SYMREF	= 00000002			DELCTX	000013BF	R	03
ARG_S_TYPE	= 00000002			DEV\$V_EXISTS	= FFFFFFFF		
ARG_V_KEYWORD	= 00000003			DEV\$V_SPL	= 00000006		
ARG_V_OPTIONAL	= 00000002			DISPATCH	000001BB	R	03
ARG_V_TYPE	= 00000000			DVISC_ACP_F11V1	= 00000001		
BATCH	00002218	R	02	DVISC_ACP_F11V2	= 00000002		
BIN	= 00000000			DVISC_ACP_JNL	= 00000006		
BIT	= 00000004			DVISC_ACP_MTA	= 00000003		
BIT...	= 00000001			DVISC_ACP_NET	= 00000004		
BUFOVF	0000160C	R	03	DVISC_ACP_REM	= 00000005		
CLIS_ABFNAM	= 000381B8			DVISC_ACPPTD	= 00000040		
CLIS_ABKEYW	= 00038010			DVISC_ACPTYPE	= 00000042		
CLIS_ARGREQ	= 00038268			DVISC_ALL	= 0000006C		
CLIS_BUFOVF	= 00038C18			DVISC_ALLDEVNAM	= 000000EC		
CLIS_EXPSYN	= 00038038			DVISC_ALLOCLASS	= 000000F2		
CLIS_INVRANGE	= 00038228			DVISC_ALT_HOST_AVAIL	= 000000F4		
CLIS_IVATIME	= 00038290			DVISC_ALT_HOST_NAME	= 000000F6		
CLIS_IVDIME	= 00038298			DVISC_ALT_HOST_TYPE	= 000000F8		
CLIS_IVFNAM	= 000381C0			DVISC_AVL	= 00000062		
CLIS_IVKEYW	= 00038060			DVISC_CCL	= 00000048		
CLIS_STRTOOLNG	= 000388FA			DVISC_CLUSTER	= 0000003A		
CLIS_SYMDEL	= 00038130			DVISC_CONCEALED	= 00000044		
CLIS_SYHOVF	= 00038138			DVISC_CYLINDERS	= 00000028		
CTLSAG_CLIDATA	*****	X	03	DVISC_DEVBUSIZ	= 00000008		
CTLSGB_MSGMASK	*****	X	03	DVISC_DEVCHAR	= 00000002		
CVTIME_ITEMS	000020B8	R	02	DVISC_DEVCHAR2	= 000000E6		
CVTIME_TYPES	0000209A	R	02	DVISC_DEVCLASS	= 00000004		
DCL\$ALCDYNMEM	*****	X	03	DVISC_DEVDEPEND	= 0000000A		
DCL\$ALLOCSYM	*****	X	03	DVISC_DEVDEPEND2	= 0000001C		
DCL\$CBTA_DEC	*****	X	03	DVISC_DEVLOCKNAM	= 000000F0		
DCL\$CBTA_HEX	*****	X	03	DVISC_DEVNAM	= 00000020		
DCL\$CBTA_HEXSTR	*****	X	03	DVISC_DEVSTS	= 000000E4		
DCL\$CNVASCBIN	*****	X	03	DVISC_DEVTTYPE	= 00000006		
DCL\$CVT_BINARY	*****	X	03	DVISC_DIR	= 0000004C		
DCL\$CVT_STRING	*****	X	03	DVISC_DMT	= 00000068		

LEXICON
Symbol table

D 14

- LEXICAL FUNCTION PARSE AND EXECUTION R 16-SEP-1984 00:00:36 VAX/VMS Macro V04-00
4-SEP-1984 23:41:41 [DCL.SRC]LEXICON.MAR;1

Page 99
(45)

DVIS_DUA	= 0000005C
DVIS_ELG	= 0000006A
DVIS_ERRCNT	= 00000014
DVIS_FOD	= 0000005A
DVIS_FOR	= 0000006E
DVIS_FREEBLOCKS	= 0000002A
DVIS_FULLDEVNAM	= 000000E8
DVIS_GEN	= 00000060
DVIS_HOST_AVAIL	= 000000FA
DVIS_HOST_COUNT	= 000000FC
DVIS_HOST_NAME	= 000000FE
DVIS_HOST_TYPE	= 00000100
DVIS_IDV	= 00000072
DVIS_LOCKID	= 000000EA
DVIS_LOGVOLNAM	= 0000002C
DVIS_MAXBLOCK	= 0000001A
DVIS_MAXFILES	= 0000003C
DVIS_MBX	= 00000066
DVIS_MEDIA_ID	= 0000011A
DVIS_MEDIA_NAME	= 00000116
DVIS_MEDIA_TYPE	= 00000118
DVIS_MNT	= 00000064
DVIS_MOUNTCNT	= 00000038
DVIS_NET	= 00000058
DVIS_NEXTDEVNAM	= 00000034
DVIS_ODV	= 00000074
DVIS_OPCNT	= 00000016
DVIS_OPR	= 00000054
DVIS_OWNUIC	= 00000010
DVIS_PID	= 0000000E
DVIS_RCK	= 0000007A
DVIS_RCT	= 00000056
DVIS_REC	= 00000046
DVIS_RECSIZ	= 00000018
DVIS_REFCNT	= 0000001E
DVIS_REMOTE_DEVICE	= 00000102
DVIS_RND	= 00000076
DVIS_ROOTDEVNAM	= 00000032
DVIS_RTM	= 00000078
DVIS_SDI	= 0000004E
DVIS_SECTORS	= 00000024
DVIS_SERIALNUM	= 0000003E
DVIS_SERVED_DEVICE	= 00000104
DVIS_SHDW_CATCHUP_COPYING	= 00000106
DVIS_SHDW_MASTER	= 00000108
DVIS_SHDW_MASTER_NAME	= 0000010A
DVIS_SHDW_MEMBER	= 0000010C
DVIS_SHDW_MERGE_COPYING	= 0000010E
DVIS_SHDW_NEXT_MBR_NAME	= 00000110
DVIS_SHDW_SPARE_BIT_1	= 0000011C
DVIS_SHDW_SPARE_BIT_2	= 0000011E
DVIS_SHDW_SPARE_INTEGER_1	= 00000124
DVIS_SHDW_SPARE_INTEGER_2	= 00000126
DVIS_SHDW_SPARE_STRING_1	= 00000120
DVIS_SHDW_SPARE_STRING_2	= 00000122
DVIS_SHR	= 0000005E
DVIS_SPL	= 00000052

DVIS_SQD	= 00000050
DVIS_STS	= 000000E2
DVIS_SWL	= 00000070
DVIS_TRACKS	= 00000026
DVIS_TRANSCNT	= 00000036
DVIS_TRM	= 0000004A
DVIS_TT_ALTYPEAHD	= 000000B8
DVIS_TT_ANSICRT	= 000000D6
DVIS_TT_APP_KEYPAD	= 000000D2
DVIS_TT_AUTOBAUD	= 000000AE
DVIS_TT_AVO	= 000000DC
DVIS_TT_BLOCK	= 000000DA
DVIS_TT_BRDCSTMBX	= 000000B4
DVIS_TT_CRFILL	= 00000092
DVIS_TT_DCL_MAILBX	= 000000BC
DVIS_TT_DECCRT	= 000000E0
DVIS_TT_DECCRT2	= 00000114
DVIS_TT_DIALUP	= 000000C4
DVIS_TT_DISCONNECT	= 000000C8
DVIS_TT_DMA	= 000000B6
DVIS_TT_DRCS	= 000000CE
DVIS_TT_EDIT	= 000000DE
DVIS_TT_EDITING	= 000000BE
DVIS_TT_EIGHTBIT	= 0000009A
DVIS_TT_ESCAPE	= 00000084
DVIS_TT_FALLBACK	= 000000C2
DVIS_TT_HALFDUP	= 000000A4
DVIS_TT_HANGUP	= 000000B0
DVIS_TT_HOSTSYNC	= 00000086
DVIS_TT_INSERT	= 000000C0
DVIS_TT_LFFILL	= 00000094
DVIS_TT_LOCALECHO	= 000000AC
DVIS_TT_LOWER	= 0000008C
DVIS_TT_MBXDSABL	= 0000009C
DVIS_TT_MECHFORM	= 000000A2
DVIS_TT_MECHTAB	= 0000008E
DVIS_TT_MODEM	= 000000A6
DVIS_TT_MODHANGUP	= 000000B2
DVIS_TT_NOBRDCST	= 0000009E
DVIS_TT_NOECHO	= 00000080
DVIS_TT_NOTYPEAHD	= 00000082
DVIS_TT_OPER	= 000000AB
DVIS_TT_PAGE	= 000000AA
DVIS_TT_PASSALL	= 0000007E
DVIS_TT_PASTHRU	= 000000CA
DVIS_TT_PHYDEVNAM	= 00000112
DVIS_TT_PRINTER	= 000000D0
DVIS_TT_READSYNC	= 000000A0
DVIS_TT_REGIS	= 000000D8
DVIS_TT_REMOTE	= 00000098
DVIS_TT_SCOPE	= 00000096
DVIS_TT_SCRIPT	= 0000008A
DVIS_TT_SECURE	= 000000C6
DVIS_TT_SETSPEED	= 000000BA
DVIS_TT_SIXEL	= 000000CC
DVIS_TT_SYSPWD	= 000000D4
DVIS_TT_TTSYNC	= 00000088

LEXICON
Symbol table

E 14
- LEXICAL FUNCTION PARSE AND EXECUTION R 16-SEP-1984 00:00:36 VAX/VMS Macro V04-C0
4-SEP-1984 23:41:41 [DCL.SRC]LEXICON.MAR;1

Page 100
(45)

DVIS TT WRAP	=	00000090			F\$PARSE	0000101D	R	03
DVIS UNIT	=	0000000C			F\$PID	00001181	R	03
DVIS VOLCOUNT	=	00000030			F\$PRIVILEGE	00001210	R	03
DVIS VOLNAM	=	00000022			F\$PROCESS	00001289	R	03
DVIS VOLNUMBER	=	0000002E			F\$SEARCH	000012D3	R	03
DVIS VOLSETMEM	=	000000EE			F\$SETPRV	000013F0	R	03
DVIS VPROT	=	00000012			F\$STRING	00001516	R	03
DVIS WCK	=	0000007C			F\$TIME	00001527	R	03
EDIT B FLAGS		00000000			F\$TRNLNM	00000DF3	R	03
EDIT M COLLAPSE	=	00000004			F\$TYPE	0000154E	R	03
EDIT M COMPRESS	=	00000002			F\$USER	0000159B	R	03
EDIT M EOL	=	00000040			F\$VERIFY	000015DC	R	03
EDIT M LOWERCASE	=	00000020			FABS\$ BID	=	00000000	
EDIT M TRIM	=	00000008			FABS\$ BKS	=	0000003E	
EDIT M UNCOMMENT	=	00000001			FABS\$ BLN	=	00000001	
EDIT M UPCASE	=	00000010			FABS\$ DNS	=	00000035	
EDIT TABLE		00001E9E	R	02	FABS\$ FAC	=	00000016	
EDIT V COLLAPSE	=	00000002			FABS\$ FNS	=	00000034	
EDIT V COMPRESS	=	00000001			FA \$B FSZ	=	0000003F	
EDIT V EOL	=	00000006			FABS\$ ORG	=	0000001D	
EDIT V LOWERCASE	=	00000005			FABS\$ RAT	=	0000001E	
EDIT V TRIM	=	00000003			FABS\$ RFM	=	0000001F	
EDIT V UNCOMMENT	=	00000000			FABS\$ SHR	=	00000017	
EDIT V UPCASE	=	00000004			FAB\$C BID	=	00000003	
ENVIRON TABLE		00001ED7	R	02	FAB\$C BLN	=	00000050	
EQUAL		0000169C	R	03	FAB\$C FIX	=	00000001	
ERROR		00001613	R	03	FAB\$C IDX	=	00000020	
EXESC SYSEFN		*****	X	03	FAB\$C REL	=	00000010	
EXESV BRK DISUSER	=	00C00003			FAB\$C SEQ	=	00000000	
EXESV BRK TERM	=	00000002			FAB\$C STM	=	00000004	
EXESV CLASS PROT	=	00000000			FAB\$C STMCR	=	00000006	
EXESV DISMOOMSG	=	00000001	G		FAB\$C STMLF	=	00000005	
EXESV MOUNTMSG	=	00000000	G		FAB\$C UDF	=	00000000	
EXESV OPAO	=	00000000	G		FAB\$C VAR	=	00000002	
EXESV REBLDSYSD	=	00000001			FAB\$C VFC	=	00000003	
EXESV WRITESYSPARAMS	=	00000001			FAB\$C ALQ	=	00000010	
EXESV XQP RESIDENT	=	00000000			FAB\$C CTX	=	00000018	
F\$CV\$INTEGER		00000202	R	03	FAB\$C DNA	=	00000030	
F\$CV\$TIME		0000024D	R	03	FAB\$C FNA	=	0000002C	
F\$CV\$JINTEGER		00000207	R	03	FAB\$C FOP	=	00000004	
F\$DIRECTORY		00000408	R	03	FAB\$C MRN	=	00000038	
F\$EDIT		00000413	R	03	FAB\$C NAM	=	00000028	
F\$ELEMENT		00000487	R	03	FAB\$C STV	=	0000000C	
F\$ENVIRONMENT		000004D3	R	03	FAB\$C XAB	=	00000024	
F\$EXTRACT		000006D9	R	03	FAB\$M CR	=	00000002	
F\$FAO		000006FC	R	03	FAB\$M DEL	=	00000004	
F\$FILE		0000073C	R	03	FAB\$M FTN	=	00000001	
F\$GETDVI		0000092E	R	03	FAB\$M GET	=	00000002	
F\$GETJPI		00000A6F	R	03	FAB\$M PRN	=	00000004	
F\$GETSYI		00000BEA	R	03	FAB\$M PUT	=	00000001	
F\$IDENTIFIER		00000CD2	R	03	FAB\$M UPD	=	00000008	
F\$INTEGER		00000D47	R	03	FAB\$M UPI	=	00000040	
F\$LENGTH		00000D58	R	03	FAB\$V CBT	=	00000015	
F\$LOCATE		00000D5F	R	03	FAB\$V CTG	=	00000014	
F\$LOGICAL		00000D76	R	03	FAB\$V KFO	=	0000001E	
F\$MESSAGE		00000FA9	R	03	FAB\$V PPF	=	00000012	
F\$MODE		00000FC5	R	03	FAB\$V RCK	=	00000017	

LEXICON
Symbol table

F 14
- LEXICAL FUNCTION PARSE AND EXECUTION R 16-SEP-1984 00:00:36 VAX/VMS Macro V04-00
4-SEP-1984 23:41:41 [DCL.SRC]LEXICON.MAR;1

Page 101
(45)

FABSV_WCK	= 00000009			JPI\$_AUTHPRIV	= 00000412
FABSW_BLS	= 0000003C			JPI\$_BIOCNT	= 0000030F
FABSW_DEQ	= 00000014			JPI\$_BIOLM	= 00000310
FABSW_MRS	= 00000036			JPI\$_BUFIO	= 0000040C
FACILITY	00002255	R	02	JPI\$_BYTCNT	= 00000311
FALSE	00002263	R	02	JPI\$_BYTLM	= 0000031A
FILEID	00001742	R	03	JPI\$_CLNAME	= 0000020A
FILE_TABLE	00001513	R	02	JPI\$-CPULIM	= 0000040D
FORMAT_BOOLEAN	00001782	R	03	JPI\$-CPUTIM	= 00000407
FORMAT_FILEID	00001750	R	03	JPI\$-CREPRC_FLAGS	= 0000020C
FORMAT_PROT	000016D5	R	03	JPI\$-CURPRIV	= 00000400
FORMAT_UIC	0000166F	R	03	JPI\$-DFPFC	= 00000406
GETDVI_TABLE	00001683	R	02	JPI\$-DFWSCNT	= 00000403
GETDVI_VALUE	00000A00	R	03	JPI\$-DIOCNT	= 00000312
GETJPI_LOGINTIM	00000B9A	R	03	JPI\$-DIOLM	= 00000313
GETJPI_STATE	00000B84	R	03	JPI\$-DIRIO	= 0000040B
GETJPI_TABLE	00000100	R	02	JPI\$-EFCS	= 00000317
GETJPI_UIC	00000BC3	R	03	JPI\$-EFCU	= 00000318
GETJPI_VALUE	00000B3B	R	03	JPI\$-EFWM	= 00000316
GETSYISW	= 00000000			JPI\$-ENQCNT	= 0000031F
GETSYI_TABLE	00000437	R	02	JPI\$-ENQLM	= 00000320
GET_FILE_ITEM	0000087C	R	03	JPI\$-EXCVEC	= 00000100
HEX	= 00000002			JPI\$-FILCNT	= 00000314
HEXSTR	= 00000005			JPI\$-FILLM	= 0000040F
IDENT	0000223D	R	02	JPI\$-FINALEXC	= 00000101
IDENT_TABLE	000021D2	R	02	JPI\$-FREPOVA	= 00000404
IDF_B_OUTFLAGS	00000038			JPI\$-FREPIVA	= 00000405
IDF_C_LENGTH	00000074			JPI\$-FREPTCNT	= 00000415
IDF_K_LENGTH	00000074			JPI\$-GPGCNT	= 0000030C
IDF_L_FILENAME	00000068			JPI\$-GRP	= 00000308
IDF_L_INPRABCTX	0000000C			JPI\$-IMAGECOUNT	= 0000041A
IDF_L_LNK	00000000			JPI\$-IMAGNAME	= 00000207
IDF_L_ONCTLY	00000060			JPI\$-IMAGPRIV	= 00000413
IDF_L_ONERROR	00000008			JPI\$-JOBPRCNT	= 0000031E
IDF_L_OUTRABCTX	00000024			JPI\$-JOBTYPE	= 00000323
IDF_L_SEARCHCTX	00000064			JPI\$-LOGINTIM	= 00000206
IDF_Q_LABEL	00000018			JPI\$-MASTER_PID	= 00000325
IDF_Q_LOCAL	00000010			JPI\$-MAXDETACH	= 0000020E
IDF_T_INPDVI	0000003C			JPI\$-MAXJOBS	= 0000020F
IDF_T_OUTDVI	00000028			JPI\$-MEM	= 00000307
IDF_W_FLAG	0000005E			JPI\$-MODE	= 00000322
IDF_W_INPDID	00000052			JPI\$-MSGMASK	= 00000277
IDF_W_INPFID	0000004C			JPI\$-OWNER	= 00000373
IDF_W_INPIFI	00000004			JPI\$-PAGEFLTS	= 0000040A
IDF_W_INPRFA	00000058			JPI\$-PAGFILCNT	= 00000414
IDF_W_ONLEVEL	00000006			JPI\$-PAGFILLOC	= 00000419
IDF_W_OUTIFI	00000020			JPI\$-PGFLQUOTA	= 0000040E
IDF_W_OUTISI	00000022			JPI\$-PHDFLAGS	= 0000041B
INTEGER	00002275	R	02	JPI\$-PID	= 00000319
INTERACTIVE	0000221E	R	02	JPI\$-PPGCNT	= 0000030D
JPI\$_ACCOUNT	= 00000203			JPI\$-PRCNT	= 0000031B
JPI\$-APTCNT	= 0000030A			JPI\$-PRCLM	= 00000408
JPI\$-ASTACT	= 00000300			JPI\$-PRCNAM	= 0000031C
JPI\$-ASTCNT	= 0000030E			JPI\$-PRI	= 00000302
JPI\$-ASTEN	= 00000301			JPI\$-PRIB	= 00000309
JPI\$-ASTLM	= 00000409			JPI\$-PROCPRIV	= 00000204
JPI\$-AUTHPRI	= 00000418			JPI\$-PROC_INDEX	= 00000324

LEXICON
Symbol table

G 14

- LEXICAL FUNCTION PARSE AND EXECUTION R 16-SEP-1984 00:00:36 VAX/VMS Macro V04-00
4-SEP-1984 23:41:41 [DCL.SRC]LEXICON.MAR;1

Page 102
(45)

JPI\$-SHRFILLM	=	00000210		
JPI\$-SITESPEC	=	00000208		
JPI\$-STATE	=	00000306		
JPI\$-STS	=	00000305		
JPI\$-SWPFILLOC	=	00000321		
JPI\$-TABLNAME	=	00000208		
JPI\$-TERMINAL	=	0000031D		
JPI\$-TMBU	=	00000308		
JPI\$-TQCNT	=	00000315		
JPI\$-TQLM	=	00000410		
JPI\$-UAF_FLAGS	=	0000020D		
JPI\$-UIC	=	00000304		
JPI\$-USERNAME	=	00000202		
JPI\$-VIRTPEAK	=	00000200		
JPI\$-VOLUMES	=	00000205		
JPI\$-WSAUTH	=	00000401		
JPI\$-WSAUTHEXT	=	00000417		
JPI\$-WSEXTENT	=	00000416		
JPI\$-WSPEAK	=	00000201		
JPI\$-WSQUOTA	=	00000402		
JPI\$-WSSIZE	=	00000411		
L	=	00000003		
LIBSCVT-DTIME	*****		X	03
LIBSCVT-TIME	*****		X	03
LNMSM-CASE BLIND	=	02000000		
LNMSV-CONCEALED	=	00000008		
LNMSV-CONFINE	=	00000001		
LNMSV-CRELOG	=	00000002		
LNMSV-NO ALIAS	=	00000000		
LNMSV-TABLE	=	00000003		
LNMSV-TERMINAL	=	00000009		
LNMS-ACMODE	=	00000006		
LNMS-ATTRIBUTES	=	00000003		
LNMS-INDEX	=	00000001		
LNMS-LENGTH	=	00000005		
LNMS-MAX INDEX	=	00000007		
LNMS-STRING	=	00000002		
LNMS-TABLE	=	00000004		
LOGATTR-TABLE	=	00002029	R	02
LOGICAL-TABLE	=	00001FA8	R	02
LXFUNTABLE	=	00000000	R	02
LXFUNTABLEN	=	00000100		
MAX-ARGS	=	00000010		
MAX-DEPTH	*****		X	03
MODE-TABLE	=	000021F3	R	02
MONTH-NO TABLE	=	00002172	R	02
MONTH-TABLE	=	00002142	R	02
NAMSB-BID	=	00000000		
NAMSB-BLN	=	00000001		
NAMSB-DEV	=	00000039		
NAMSB-DIR	=	0000003A		
NAMSB-ESL	=	0000000B		
NAMSB-ESS	=	0000000A		
NAMSB-NAME	=	0000003B		
NAMSB-NODE	=	00000038		
NAMSB-NOP	=	00000008		
NAMSB-RSL	=	00000003		

NAMSB-RSS	=	00000002		
NAMSB-TYPE	=	0000003C		
NAMSB-VER	=	0000003D		
NAMSC-BID	=	00000002		
NAMSC-BLN	=	00000060		
NAMSC-MAXRSS	=	000000FF		
NAMSL-DEV	=	00000044		
NAMSL-DIR	=	00000048		
NAMSL-ESA	=	0000000C		
NAMSL-FNB	=	00000034		
NAMSL-NAME	=	0000004C		
NAMSL-NODE	=	00000040		
NAMSL-RLF	=	00000010		
NAMSL-RSA	=	00000004		
NAMSL-TYPE	=	00000050		
NAMSL-VER	=	00000054		
NAMSL-WCC	=	00000030		
NAMSM-NOCONCEAL	=	00000010		
NAMSM-SYNCHK	=	00000008		
NAMST-DVI	=	00000014		
NAMSV-NOCONCEAL	=	00000004		
NAMSV-SYNCHK	=	00000003		
NAMSV-WILDCARD	=	00000008		
NAMSW-DID	=	0000002A		
NAMSW-FID	=	00000024		
NARGS	=	00000002		
NETWORK	=	0000222A	R	02
NLA0	=	0000228D	R	02
NOALL	=	0000225E	R	02
NPARM	=	00000000		
NULL	=	0000169B	R	03
OTHER	=	00002232	R	02
OUTLEN	=	00000004		
P1	=	00000004		
P1_ERROR	=	00001618	R	03
P2	=	0000000C		
P2_ERROR	=	0000161E	R	03
P3	=	00000014		
P3_ERROR	=	00001624	R	03
P4	=	0000001C		
P4_ERROR	=	0000162A	R	03
P5	=	00000024		
P5_ERROR	=	00001630	R	03
P6	=	0000002C		
P6_ERROR	=	00001636	R	03
PARSE-FIELDS	=	00001634	R	02
PARSE-TYPES	=	00001669	R	02
PIOSGT-DDSTRING	*****		X	03
PN_ERROR	=	0000163A	R	03
PPDSB-NPROCS	=	0000001C		
PPDSC-LENGTH	=	00000168		
PPDSK-LENGTH	=	00000168		
PPDSL-INPDEV	=	00000044		
PPDSL-LGI	=	00000014		
PPDSL-LSTSTATUS	=	00000018		
PPDSL-OUTDEV	=	00000064		
PPDSL-PRC	=	00000008		

LEXICON
Symbol table

H 14
- LEXICAL FUNCTION PARSE AND EXECUTION R 16-SEP-1984 00:00:36 VAX/VMS Macro V04-00
4-SEP-1984 23:41:41 [DCL.SRC]LEXICON.MAR;1

Page 103
(45)

PPDSQ_CLIREG 00000004
PPDSQ_CLISYMTBL 0000000C
PPDST_FILENAME 00000068
PPDST_INPDVI 00000028
PPDST_OUTDVI 00000048
PPDSV_CAPTIVE = 00000003
PPDSW_FLAGS 00000002
PPDSW_INPCHAN 0000001E
PPDSW_INPDID 0000003E
PPDSW_INPFID 00000038
PPDSW_INPFI 00000020
PPDSW_INPISI 00000022
PPDSW_OUTDID 0000005E
PPDSW_OUTFID 00000058
PPDSW_OUTIFI 00000024
PPDSW_OUTISI 00000026
PPDSW_SIZE 00000000
PRC_B_CONTINUE 000000F3
PRC_B_DEFRADIX 000000AE
PRC_B_EXMDEPMOD 000000AD
PRC_B_EXMDEPWID 000000AC
PRC_B_EXONLYL 0000012D
PRC_B_FLAGS2 000000AF
PRC_B_IMGFLAG 00000078
PRC_B_OUTFLAGS 0000012C
PRC_B_PROMPTLEN 000000F0
PRC_C_LENGTH 00000534
PRC_G_COMMANDS 00000133
PRC_G_PROMPT 000000F4
PRC_K_DEC = 00000001
PRC_K_HEX = 00000000
PRC_K_LENGTH 00000534
PRC_L_CURRKEY 00000048
PRC_L_EXMDEPADR 000000A8
PRC_L_EXTARG 00000094
PRC_L_EXTBLK 0000008C
PRC_L_EXTCOD 0000009C
PRC_L_EXTHND 00000090
PRC_L_EXTPRM 00000098
PRC_L_IDFLNK 0000008C
PRC_L_IMGACTSTS 00000080
PRC_L_INDCLOCK 0000007C
PRC_L_INDEPTH 0000005C
PRC_L_INDFAB 0000001C
PRC_L_INDINPRAB 00000014
PRC_L_INDOUTRAB 00000018
PRC_L_INPRAB 00000008
PRC_L_LASTKEY 0000004C
PRC_L_LSTSTATUS 000000B0
PRC_L_ONCTLY 000000B8
PRC_L_ONERROR 0000006C
PRC_L_OUTOFBAND 00000084
PRC_L_OUTRAB 0000000C
PRC_L_OUTRABCTX 00000118
PRC_L_PPFLIST 00000070
PRC_L_RECALLPTR 0000012F
PRC_L_RESTART 00000058

PRC_L_SAVAP 00000000
PRC_L_SAVFP 00000004
PRC_L_SEVERITY 00000050
PRC_L_SPWN 000000C0
PRC_L_STACKLM 000000A4
PRC_L_STACKPT 000000A0
PRC_L_STATUS 00000054
PRC_L_STS 00000084
PRC_L_STV 00000088
PRC_L_SYMBOL 00000060
PRC_L_TMBX 00000074
PRC_L_TRMLIST 00000010
PRC_Q_ALLOCREG 00000020
PRC_Q_COMMAND 000000E0
PRC_Q_FLUSHTIME 000000D0
PRC_Q_GLOBAL 00000028
PRC_Q_IMAGENAME 000000D8
PRC_Q_KEYPAD 00000040
PRC_Q_LABEL 00000030
PRC_Q_LOCAL 00000038
PRC_Q_SAVEPRIV 000000E8
PRC_T_OUTDVI 0000011C
PRC_V_CARRCTL = 00000000
PRC_V_CTRLT = 00000014
PRC_V_CTRLY = 00000019
PRC_V_MODE = 00000006
PRC_V_VERIFY = 00000007
PRC_V_VERIMAGE = 00000007
PRC_W_ASTIOSB 000000C6
PRC_W_ASTRETN 000000C8
PRC_W_ASTSTATUS 000000C4
PRC_W_ATTMBX 0000007A
PRC_W_FLAGS 00000068
PRC_W_INPCHAN 00000064
PRC_W_ONLEVEL 0000006A
PRC_W_OUTIFI 00000114
PRC_W_OUTISI 00000116
PRC_W_OUTMBXCHN 000000CA
PRC_W_OUTMBXREF 000000CE
PRC_W_OUTMBXSIZ 000000CC
PRC_W_PMPTCTRL 000000F1
PRC_W_WAITIOSB 00000066
PROTECTEND 000016D5 R 03
PROTECTMSG 000016A2 R 03
PRV = 00000003
PRV\$KEYWORD ***** X 03
PRV\$PRIVBIT ***** X 03
PRV\$SETPRIV ***** X 03
PRV\$NOTUNQ ***** X 03
PSL\$C_USER = 00000003
S = 00000006
SEVERITY 0000224C R 02
SEVERITY TABLE 00002072 R 02
SGNSV_LOADCHKPRT = 00000001 G
SGNSV_LOADERAPAT = 00000000 G
SGNSV_LOADMTACCESS = 00000002 G
SIZ... = 00000001

LEXICON
Symbol table

1 14

- LEXICAL FUNCTION PARSE AND EXECUTION R 16-SEP-1984 00:00:36 VAX/VMS Macro V04-00
4-SEP-1984 23:41:41 [DCL.SRC]LEXICON.MAR;1

Page 104
(45)

SS\$_BUFFEROVF	*****	X	03	SYIS_CLUSTER_NODES	= 000010CA
SS\$_IVDEVNAM	*****	X	03	SYIS_CLUSTER_QUORUM	= 000010CC
SS\$_NOLOGNAM	*****	X	03	SYIS_CLUSTER_VOTES	= 000010CB
SS\$_NOMOREPROC	*****	X	03	SYIS_CONCEAL_DEVICES	= 00002012
SS\$_NONEXPR	*****	X	03	SYIS_CPU	= 00002000
SS\$_NOPRIV	*****	X	03	SYIS_CRDENABLE	= 00C02002
SS\$_NOSUCHDEV	*****	X	03	SYIS_CTLIMGLIM	= 00001027
SS\$_NOSUCHNODE	*****	X	03	SYIS_CTLPAGES	= 00001026
SS\$_NOTRAN	*****	X	03	SYIS_DEADLOCK_WAIT	= 0000105E
SS\$_SUSPENDED	*****	X	03	SYIS_DECIMAL_EMULATED	= 0000201F
STATE TABLE	00001491	R	02	SYIS_DEFBMBXBUFQUO	= 00001050
STDIM	= 00000006			SYIS_DEFBMBXMXMSG	= 00001051
STR	= 00000001			SYIS_DEFBMBXNUMMSG	= 00001052
STRING	0000226E	R	02	SYIS_DEFPRI	= 000010B7
STSSK_ERROR	= 00000002			SYIS_DEFQUEPRI	= 000010E2
STSSK_SEVERE	= 00000004			SYIS_DISK_QUORUM	= 000010DC
STSSK_WARNING	= 00000000			SYIS_DISMOUMSG	= 00002015
STS_MODE	00000BCA	R	03	SYIS_DLCKEXTRASTK	= 00001011
SYIS_ACP_BASEPRIO	= 000010B5			SYIS_DORMANTWAIT	= 000010F1
SYIS_ACP_DATACHECK	= 000010B4			SYIS_DUMPBUG	= 00002003
SYIS_ACP_DINDXCACHE	= 00001101			SYIS_D_FLOAT_EMULATED	= 00002020
SYIS_ACP_DIRCACHE	= 000010AA			SYIS_EXTRACPD	= 0000104C
SYIS_ACP_EXTCACHE	= 000010AD			SYIS_EXUSRSTK	= 0000101B
SYIS_ACP_EXTLIMIT	= 000010AE			SYIS_FREEGOAL	= 00001054
SYIS_ACP_FIDCACHE	= 000010AC			SYIS_FREELIM	= 00001053
SYIS_ACP_HDRCACHE	= 000010A9			SYIS_F_FLOAT_EMULATED	= 00002021
SYIS_ACP_MAPCACHE	= 000010A8			SYIS_GBLPAGES	= 00001007
SYIS_ACP_MAXREAD	= 000010B1			SYIS_GBLPAGFIL	= 00001008
SYIS_ACP_MULTIPLE	= 00002005			SYIS_GBLSECTIONS	= 00001006
SYIS_ACP_QUOCACHE	= 000010AF			SYIS_GROWLIM	= 00001055
SYIS_ACP_REBLDSYSD	= 00002029			SYIS_G_FLOAT_EMULATED	= 00002022
SYIS_ACP_SHARE	= 0000200C			SYIS_H_FLOAT_EMULATED	= 00002023
SYIS_ACP_SWAPFLGS	= 000010B6			SYIS_IJOB LIM	= 000010B8
SYIS_ACP_SYSACC	= 000010B0			SYIS_IMGIOCNT	= 00001028
SYIS_ACP_WINDOW	= 000010B2			SYIS_INTSTKPAGES	= 00001010
SYIS_ACP_WORKSET	= 000010AB			SYIS_IOTA	= 0000103D
SYIS_ACP_WRITEBACK	= 000010B3			SYIS_IRPCOUNT	= 00001013
SYIS_ACP_XQP_RES	= 00002025			SYIS_IRPCOUNTV	= 00001014
SYIS_ALLOCLASS	= 000010E5			SYIS_KFILSTCNT	= 00001005
SYIS_ARCHFLAG	= 000010DA			SYIS_LAMAPREGS	= 00001059
SYIS_AWSMIN	= 00001038			SYIS_LGI_BRK_DISUSER	= 00002028
SYIS_AWSTIME	= 00001039			SYIS_LGI_BRK_LIM	= 000010E8
SYIS_BALSETCNT	= 00001012			SYIS_LGI_BRK_TERM	= 00002027
SYIS_BJOB LIM	= 000010B9			SYIS_LGI_BRK_TMO	= 000010E9
SYIS_BOOTTIME	= 000010BF			SYIS_LGI_HID_TIM	= 000010EA
SYIS_BORROWLIM	= 00001056			SYIS_LGI_PWD_TMO	= 000010EB
SYIS_BUGCHECKFATAL	= 00002004			SYIS_LGI_RETRY_LIM	= 000010E6
SYIS_BUGREBOOT	= 00002001			SYIS_LGI_RETRY_TMO	= 000010E7
SYIS_CHANNELCNT	= 00001024			SYIS_LNMPHASHTBL	= 00001072
SYIS_CHARACTER_EMULATED	= 0000201E			SYIS_LNMHASHTBL	= 00001071
SYIS_CJFLOAD	= 00002019			SYIS_LOADCHKPRT	= 00002017
SYIS_CJFSYSRUJ	= 0000201A			SYIS_LOADERAPT	= 00002016
SYIS_CLASS_PROT	= 0000201D			SYIS_LOADMTACCESS	= 00002024
SYIS_CLISYMTBL	= 0000105B			SYIS_LOCKDIRWT	= 000010EF
SYIS_CLUSTER_FSYSID	= 000010CD			SYIS_LOCKIDTBL	= 0000105C
SYIS_CLUSTER_FTIME	= 000010CE			SYIS_LOCKIDTBL_MAX	= 000010C0
SYIS_CLUSTER_MEMBER	= 000010CF			SYIS_LOCKRETRY	= 00001057

LEXICON
Symbol table

J 14
- LEXICAL FUNCTION PARSE AND EXECUTION R 16-SEP-1984 00:00:36 VAX/VMS Macro V04-00
4-SEP-1984 23:41:41 [DCL.SRC]LEXICON.MAR;1

Page 105
(45)

SYIS_LONGWAIT = 0000103E
SYIS_LRPCOUNT = 0000101C
SYIS_LRPCOUNTV = 0000101D
SYIS_LRPMIN = 0000101F
SYIS_LRPSIZE = 0000101E
SYIS_MAXBUF = 0000104F
SYIS_MAXPROCESSCNT = 00001009
SYIS_MAXQUEPRI = 000010E3
SYIS_MAXSYSGROUP = 0000104D
SYIS_MINWSCNT = 0000100C
SYIS_MOUNTMSG = 00002014
SYIS_MPW_HILIMIT = 0000102B
SYIS_MPW_LOLIMIT = 0000102C
SYIS_MPW_PPIO = 0000102D
SYIS_MPW_THRESH = 0000102F
SYIS_MPW_WAITLIMIT = 00001030
SYIS_MPW_WRTCLUSTER = 0000102A
SYIS_MVTIMEOUT = 0000104E
SYIS_NJOBLIM = 0000108A
SYIS_NOAUTOCONFIG = 00002006
SYIS_NOCLOCK = 00002007
SYIS_NOCLUSTER = 00002008
SYIS_NODENAME = 000010D9
SYIS_NODE_AREA = 0000201B
SYIS_NODE_CSID = 000010D0
SYIS_NODE_HWTYPE = 000010D7
SYIS_NODE_HWVERS = 000010D8
SYIS_NODE_NUMBER = 0000201C
SYIS_NODE_QUORUM = 000010D2
SYIS_NODE_SWINCARN = 000010D4
SYIS_NODE_SWTYPE = 000010D5
SYIS_NODE_SWVERS = 000010D6
SYIS_NODE_SYSTEMID = 000010D3
SYIS_NODE_VOTES = 000010D1
SYIS_NPAGEDYN = 00001016
SYIS_NPAGEVIR = 00001017
SYIS_PAGEDYN = 00001018
SYIS_PAGEFILE_FREE = 000010F4
SYIS_PAGEFILE_PAGE = 000010F2
SYIS_PAGFILCNT = 0000100D
SYIS_PAGTBLPFC = 00001003
SYIS_PAMAXPORT = 000010E0
SYIS_PANOPOLL = 000010F8
SYIS_PANUMPOLL = 0000106C
SYIS_PAPOLLINTERVAL = 0000106D
SYIS_PAPOOLINTERVAL = 0000106E
SYIS_PASANITY = 000010E1
SYIS_PASTDGBUF = 0000106B
SYIS_PASTIMOUT = 0000106A
SYIS_PE1 = 000010F9
SYIS_PE2 = 000010FA
SYIS_PE3 = 000010FB
SYIS_PE4 = 000010FC
SYIS_PE5 = 000010FD
SYIS_PE6 = 000010FE
SYIS_PFCDEFAULT = 00001002
SYIS_PFRATH = 00001034

SYIS_PFRATL = 00001033
SYIS_PFRATS = 00001035
SYIS_PHYSICALPAGES = 00001032
SYIS_PIOFAGES = 00001025
SYIS_PIXSCAN = 0000100A
SYIS_POOLPAGING = 00002009
SYIS_PQL_DASTLM = 0000108E
SYIS_PQL_DBIOLM = 00001090
SYIS_PQL_DBYTLM = 00001092
SYIS_PQL_DCPULM = 00001094
SYIS_PQL_DDIOLM = 00001096
SYIS_PQL_DENQLM = 000010A6
SYIS_PQL_DFILLM = 00001098
SYIS_PQL_DJTQUOTA = 000010EC
SYIS_PQL_DPGFLQUOTA = 0000109A
SYIS_PQL_DPRCLM = 0000109C
SYIS_PQL_DTQELM = 0000109E
SYIS_PQL_DWSDEFAULT = 000010A0
SYIS_PQL_DWSEXTENT = 000010A4
SYIS_PQL_DWSQUOTA = 000010A2
SYIS_PQL_MASTLM = 0000108F
SYIS_PQL_MBIOLM = 00001091
SYIS_PQL_MBYTLM = 00001093
SYIS_PQL_MCPULM = 00001095
SYIS_PQL_MDIOLM = 00001097
SYIS_PQL_MENQLM = 000010A7
SYIS_PQL_MFILLM = 00001099
SYIS_PQL_MJTQUOTA = 000010ED
SYIS_PQL_MPGFLQUOTA = 0000109B
SYIS_PQL_MPRCLM = 0000109D
SYIS_PQL_MTQELM = 0000109F
SYIS_PQL_MWSDEFAULT = 000010A1
SYIS_PQL_MWSEXTENT = 000010A5
SYIS_PQL_MWSQUOTA = 000010A3
SYIS_PRCPOLINTERVAL = 00001069
SYIS_PROCSECTCNT = 0000100B
SYIS_QDSKINTERVAL = 000010E4
SYIS_QDSKVOTES = 000010F0
SYIS_QUANTUM = 00001029
SYIS_QUORUM = 000010BC
SYIS_REALTIME_SPTS = 0000105A
SYIS_RECIXINTERVAL = 000010BE
SYIS_RESALLOC = 00002010
SYIS_RESHASHTBL = 0000105D
SYIS_RJOBLIM = 000010BB
SYIS_RMS_DFMBC = 00001084
SYIS_RMS_DFMBSHSH = 0000108A
SYIS_RMS_DFMBSHSH = 00001089
SYIS_RMS_DFMBSHSH = 00001088
SYIS_RMS_DFMBSHSH = 00001085
SYIS_RMS_DFMBSHSH = 00001086
SYIS_RMS_DFMBSHSH = 00001087
SYIS_RMS_DFMBSHSH = 00001100
SYIS_RMS_EXTEND_SIZE = 0000108C
SYIS_RMS_FILEPROT = 0000108D
SYIS_RMS_GBLBUFQUO = 000010FF
SYIS_RMS_PROLOGUE = 0000108B

LEXICON
Symbol table

K 14
- LEXICAL FUNCTION PARSE AND EXECUTION R 16-SEP-1984 00:30:36 VAX/VMS Macro V04-00
4-SEP-1984 23:41:41 [DCI.SRC]LEXICON.MAR;1

Page 106
(45)

SYIS_SAVEDUMP = 00002013
SYIS_SBIERRENABLE = 0000200A
SYIS_SCSBUFFCNT = 0000105F
SYIS_SCSCONN CNT = 00001060
SYIS_SCSFLOWCUSH = 00001064
SYIS_SCSMAXDG = 00001062
SYIS_SCSMAXMSG = 00001063
SYIS_SCSNODE = 00001067
SYIS_SCSRESPCNT = 00001061
SYIS_SCSSYSTEMID = 00001065
SYIS_SCSSYSTEMIDH = 00001066
SYIS_SCS_EXISTS = 000010DB
SYIS_SETTIME = 0000200B
SYIS_SID = 00001001
SYIS_SPTREQ = 0000101A
SYIS_SRPCOUNT = 00001020
SYIS_SRPCOUNTV = 00001021
SYIS_SRP MIN = 00001023
SYIS_SRP SIZE = 00001022
SYIS_SSINHIBIT = 00002011
SYIS_STARTUP_P1 = 000010C2
SYIS_STARTUP_P2 = 000010C3
SYIS_STARTUP_P3 = 000010C4
SYIS_STARTUP_P4 = 000010C5
SYIS_STARTUP_P5 = 000010C6
SYIS_STARTUP_P6 = 000010C7
SYIS_STARTUP_P7 = 000010C8
SYIS_STARTUP_P8 = 000010C9
SYIS_SWAPFILE_FREE = 000010F5
SYIS_SWAPFILE_PAGE = 000010F3
SYIS_SWPALLOC INC = 0000103C
SYIS_SWPFAIL = 0000103F
SYIS_SWPFI LCNT = 0000100E
SYIS_SWPOUTPGCNT = 0000103B
SYIS_SWPRATE = 0000103A
SYIS_SWP_PRIO = 0000102E
SYIS_SYSMWCNT = 0000100F
SYIS_SYSPAGING = 0000200D
SYIS_SYSPFC = 00001004
SYIS_TAILORED = 000010C1
SYIS_TBSKIPWSL = 00001031
SYIS_TIMEPROMPTWAIT = 0000106F
SYIS_TTY_ALTALARM = 0000107D
SYIS_TTY_ALTYPAHD = 0000107C
SYIS_TTY_AUTOCHAR = 000010F7
SYIS_TTY_BUF = 00001078
SYIS_TTY_CLASSNAME = 00001081
SYIS_TTY_DEFCHAR = 00001079
SYIS_TTY_DEFCHAR2 = 0000107A
SYIS_TTY_DEFPORT = 00001083
SYIS_TTY_DIALTYPE = 00001074
SYIS_TTY_DMASIZE = 0000107E
SYIS_TTY_OWNER = 00001080
SYIS_TTY_PARITY = 00001077
SYIS_TTY_PROT = 0000107F
SYIS_TTY_RSPEED = 00001076
SYIS_TTY_SCANDelta = 00001073

SYIS_TTY_SILOTIME = 00001082
SYIS_TTY_SPEED = 00001075
SYIS_TTY_TIMEOUT = 000010F6
SYIS_TTY_TYPAHDSZ = 0000107B
SYIS_UAFALTERNATE = 0000200E
SYIS_UDABURSTRATE = 00001070
SYIS_USER3 = 0000104A
SYIS_USER4 = 0000104B
SYIS_USERD1 = 00001048
SYIS_USERD2 = 00001049
SYIS_VAXCLUSTER = 000010EE
SYIS_VERSION = 00001000
SYIS_VIRTUALPAGECNT = 00001019
SYIS_VMS5 = 00001044
SYIS_VMS6 = 00001045
SYIS_VMS7 = 00001046
SYIS_VMS8 = 00001047
SYIS_VMSD1 = 00001040
SYIS_VMSD2 = 00001041
SYIS_VMSD3 = 00001042
SYIS_VMSD4 = 00001043
SYIS_VOTES = 000010BD
SYIS_WRITABLESYS = 0000200F
SYIS_WRITESSPARAMS = 00002026
SYIS_WSDC = 00001037
SYIS_WSINC = 00001036
SYIS_WSMAX = 00001015
SYIS_WS_OPAO = 0000202A
SYIS_XFMAXRATE = 00001058
SYM_B_FLAGS = 0000000B
SYM_B_NONUNIQUE = 0000000B
SYM_B_TYPE = 0000000A
SYM_K_BINARY = 00000002
SYM_L_BL = 00000004
SYM_L_FL = 00000000
SYM_T_SYMBOL = 0000000C
SYM_W_SIZE = 00000008
SYSSASCTIM ***** GX 03
SYSSASCTOID ***** GX 03
SYSSCLOSE ***** GX 03
SYSSFAO ***** X 03
SYSSFAOL ***** GX 03
SYSSGETDVIW ***** GX 03
SYSSGETJPIW ***** GX 03
SYSSGETMSG ***** GX 03
SYSSGETSYIW ***** GX 03
SYSSIDTOASC ***** GX 03
SYSSOPEN ***** GX 03
SYSSPARSE ***** GX 03
SYSSSEARCH ***** GX 03
SYSSSETDFPROT ***** X 03
SYSSSETPRV ***** GX 03
SYSSTRNLNM ***** GX 03
SYSSTRNLOG ***** GX 03
TABDEF 0000227D R 02
TEXT 00002238 R 02
TIME_DELIMS 00000249 R 03

LEXICON
Symbol table

L 14

- LEXICAL FUNCTION PARSE AND EXECUTION R 16-SEP-1984 00:00:36 VAX/VMS Macro V04-00
4-SEP-1984 23:41:41 [DCL.SRC]LEXICON.MAR;1

Page 107
(45)

TRUE	00002269	R	02	XABSL_EBK	= 00000010	
UIC	00001668	R	03	XABSQ_BDT	= 00000024	
WEEKDAY_TABLE	0000218C	R	02	XABSQ_CDT	= 00000014	
WRK_B_CMDOPT	FFFFFFFFC3			XABSQ_EDT	= 0000001C	
WRK_B_MAXPARM	FFFFFFFFD0			XABSQ_RDT	= 0000000C	
WRK_B_MINPARM	FFFFFFFFD1			XABSW_FFB	= 00000014	
WRK_B_PARMCNT	FFFFFFFFCE			XABSW_GRP	= 0000000E	
WRK_B_PARMSUM	FFFFFFFFCF			XABSW_MBM	= 0000000C	
WRK_B_RECALLCNT	FFFFFFFFC5			XABSW_PRO	= 00000008	
WRK_B_VALLEV	FFFFFFFFC4			XABSW_PVN	= 0000000A	
WRK_B_VERBTYP	FFFFFFFFC2			XABSW_RVN	= 00000008	
WRK_C_LENGTH	FFFFFF486			XAB_INFO	00001509	R 02
WRK_G_BUFFER	FFFFFF492			_\$\$_	= 000000EF	
WRK_G_INPBUF	FFFFFF896					
WRK_G_RESULT	FFFFFF9B6					
WRK_K_LENGTH	FFFFFF486					
WRK_L_CHARPTR	FFFFFF48E					
WRK_L_DISALLOW	FFFFFFE6					
WRK_L_ERRORRTN	FFFFFF9AE					
WRK_L_EXPANDPTR	FFFFFF486					
WRK_L_IMAGE	FFFFFFE2					
WRK_L_MARKPTR	FFFFFF48A					
WRK_L_PAROUT	FFFFFFFD2					
WRK_L_PMPADDR	FFFFFF9A2					
WRK_L_PROMPTRN	FFFFFF9A6					
WRK_L_PROPTR	FFFFFFC6					
WRK_L_QUABLK	FFFFFFCA					
WRK_L_READRTN	FFFFFF9AA					
WRK_L_RECALLPTR	FFFFFFEA					
WRK_L_RSLEND	FFFFFFB6					
WRK_L_RSLNXT	FFFFFFBA					
WRK_L_SAVAP	FFFFFFF8					
WRK_L_SAVFP	FFFFFFFC					
WRK_L_SAVSP	FFFFFFF4					
WRK_L_SIGNALRTN	FFFFFFD6					
WRK_L_SPECRTN	FFFFFF9B2					
WRK_L_TAB_VEC	FFFFFFDE					
WRK_L_VERB	FFFFFFBE					
WRK_V_COMMENT	= 0000000C					
WRK_W_FLAGS	FFFFFFF0					
WRK_W_FLAGS2	FFFFFFF2					
WRK_W_IMGCHAN	FFFFFFEE					
WRK_W_PMPLEN	FFFFFF99E					
XABSB_BLN	= 00000001					
XABSB_COD	= 00000000					
XABSB_NOA	= 00000008					
XABSB_NOK	= 00000009					
XABSC_ALL	= 00000014					
XABSC_ALLLEN	= 00000020					
XABSC_DAT	= 00000012					
XABSC_DATLEN	= 0000002C					
XABSC_FHC	= 0000001D					
XABSC_FHCLEN	= 0000002C					
XABSC_PRO	= 00000013					
XABSC_PROLEN	= 00000058					
XABSC_SUM	= 00000016					
XABSC_SUMLEN	= 0000000C					

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

PSECT name	Allocation	PSECT No.	Attributes
ABS	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
\$ABSS	FFFFFFFFC (0.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
DCL\$ZDATA	00002294 (8852.)	02 (2.)	NOPIC USR CON REL LCL NOSHR EXE RD NOWRT NOVEC BYTE
DCL\$ZCODE	00001797 (6039.)	03 (3.)	NOPIC USR CON REL LCL NOSHR EXE RD NOWRT NOVEC BYTE

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

Phase	Page faults	CPU Time	Elapsed Time
Initialization	15	00:00:00.04	00:00:01.87
Command processing	96	00:00:00.69	00:00:07.52
Pass 1	2302	00:01:56.74	00:05:37.29
Symbol table sort	0	00:00:03.35	00:00:08.09
Pass 2	776	00:00:25.46	00:01:17.72
Symbol table output	0	00:00:00.78	00:00:02.97
Psect synopsis output	2	00:00:00.02	00:00:00.33
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	3191	00:02:27.09	00:07:15.79

The working set limit was 3750 pages.
632877 bytes (1237 pages) of virtual memory were used to buffer the intermediate code.
There were 120 pages of symbol table space allocated to hold 2003 non-local and 328 local symbols.
4291 source lines were read in Pass 1, producing 78 object records in Pass 2.
242 pages of virtual memory were used to define 89 macros.

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

Macro library name	Macros defined
\$255\$DUA28:[SYSLIB]SYSBLDMLB.MLB;1	7
\$255\$DUA28:[DCL.OBJ]DCL.MLB;1	13
\$255\$DUA28:[SYS.OBJ]LIB.MLB;1	2
\$255\$DUA28:[SYSLIB]STARLET.MLB;2	41
TOTALS (all libraries)	63

5663 GETS were required to define 63 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS\$:LEXICON/OBJ=OBJ\$:LEXICON MSRC\$:LEXICON/UPDATE=(ENH\$:LEXICON)+EXECML\$/LIB+LIB\$:DCL/LIB+SYSS\$LIBRARY:SYSBLDMLB/LIB

0071 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

INQUIRE
LIS

LEXICON
LIS

KEYPAD
LIS

LOGICAL
LIS