

PPPPPPPPPPPP		AAAAAAAAAA		SSSSSSSSSSSS		CCCCCCCCCCCC		AAAAAAAAAA		LLL
PPPPPPPPPPPP		AAAAAAAAAA		SSSSSSSSSSSS		CCCCCCCCCCCC		AAAAAAAAAA		LLL
PPPPPPPPPPPP		AAAAAAAAAA		SSSSSSSSSSSS		CCCCCCCCCCCC		AAAAAAAAAA		LLL
PPP	PPP	AAA	AAA	SSS		CCC		AAA	AAA	LLL
PPP	PPP	AAA	AAA	SSS		CCC		AAA	AAA	LLL
PPP	PPP	AAA	AAA	SSS		CCC		AAA	AAA	LLL
PPP	PPP	AAA	AAA	SSS		CCC		AAA	AAA	LLL
PPP	PPP	AAA	AAA	SSS		CCC		AAA	AAA	LLL
PPP	PPP	AAA	AAA	SSS		CCC		AAA	AAA	LLL
PPPPPPPPPPPP		AAA	AAA	SSS	SSSSSSSSSS	CCC		AAA	AAA	LLL
PPPPPPPPPPPP		AAA	AAA	SSS	SSSSSSSSSS	CCC		AAA	AAA	LLL
PPPPPPPPPPPP		AAA	AAA	SSS	SSSSSSSSSS	CCC		AAA	AAA	LLL
PPP		AAAAAAAAAAAAAAAA		SSS		CCC		AAAAAAAAAAAAAAAA		LLL
PPP		AAAAAAAAAAAAAAAA		SSS		CCC		AAAAAAAAAAAAAAAA		LLL
PPP		AAAAAAAAAAAAAAAA		SSS		CCC		AAAAAAAAAAAAAAAA		LLL
PPP		AAA	AAA	SSS		CCC		AAA	AAA	LLL
PPP		AAA	AAA	SSS		CCC		AAA	AAA	LLL
PPP		AAA	AAA	SSS		CCC		AAA	AAA	LLL
PPP		AAA	AAA	SSS		CCC		AAA	AAA	LLL
PPP		AAA	AAA	SSS	SSSSSSSSSSSS	CCC	CCCCCCCCCCCC	AAA	AAA	LLLLLLLLLLLLLLLL
PPP		AAA	AAA	SSS	SSSSSSSSSSSS	CCC	CCCCCCCCCCCC	AAA	AAA	LLLLLLLLLLLLLLLL
PPP		AAA	AAA	SSS	SSSSSSSSSSSS	CCC	CCCCCCCCCCCC	AAA	AAA	LLLLLLLLLLLLLLLL

```

PPPPPPPP      AAAAAA      SSSSSSSS      IIIIII      000000      222222
PPPPPPPP      AAAAAA      SSSSSSSS      IIIIII      000000      222222
PP      PP      AA      AA      SS      II      00      00      22      22
PP      PP      AA      AA      SS      II      00      00      22      22
PP      PP      AA      AA      SS      II      00      00      22      22
PP      PP      AA      AA      SS      II      00      00      22      22
PPPPPPPP      AA      AA      SSSSSS      II      00      00      22      22
PPPPPPPP      AA      AA      SSSSSS      II      00      00      22      22
PP      AAAAAAAAAA      SS      II      00      00      22      22
PP      AAAAAAAAAA      SS      II      00      00      22      22
PP      AA      AA      SS      II      00      00      22      22
PP      AA      AA      SSSSSSSS      IIIIII      000000      2222222222      ....
PP      AA      AA      SSSSSSSS      IIIIII      000000      2222222222      ....

```

```

LL      IIIIII      SSSSSSSS
LL      IIIIII      SSSSSSSS
LL      II      SS
LL      II      SS
LL      II      SS
LL      II      SS
LL      II      SSSSSS
LL      II      SSSSSS
LL      II      SS
LL      II      SS
LL      II      SS
LL      II      SS
LLLLLLLLLLLL      IIIIII      SSSSSSSS
LLLLLLLLLLLL      IIIIII      SSSSSSSS

```

```

0000 1 :
0000 2 :*****
0000 3 :*
0000 4 :*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY *
0000 5 :*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS. *
0000 6 :*  ALL RIGHTS RESERVED. *
0000 7 :*
0000 8 :*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED *
0000 9 :*  ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE *
0000 10 :*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER *
0000 11 :*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY *
0000 12 :*  OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY *
0000 13 :*  TRANSFERRED. *
0000 14 :*
0000 15 :*  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE *
0000 16 :*  AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT *
0000 17 :*  CORPORATION. *
0000 18 :*
0000 19 :*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS *
0000 20 :*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL. *
0000 21 :*
0000 22 :*
0000 23 :*****
0000 24 :
0000 25 :      .TITLE  PASSIO INPUT          ; PASCAL RMS Linkage
0000 26 :      .ident  'V04-000'
0000 27 :
0000 28 :*****
0000 29 :*****
0000 30 :**
0000 31 :**          PASCAL RMS LINKAGE FOR VAX-11/780          **
0000 32 :**          =====          **
0000 33 :**
0000 34 :**
0000 35 :**          VERSION V1.0-1 -- OCTOBER 1979          **
0000 36 :**
0000 37 :**  DEVELOPED BY:  COMPUTER SCIENCE DEPARTMENT          **
0000 38 :**                   UNIVERSITY OF WASHINGTON          **
0000 39 :**                   SEATTLE, WA 98195          **
0000 40 :**
0000 41 :**  AUTHORS:      MARK BAILEY, JOHN CHAN, HELLMUT GOLDE          **
0000 42 :**
0000 43 :*****
0000 44 :*****
0000 45 :
0000 46 : Modified to allow input of 31 character scalar values.
0000 47 :           Paul Hohensee 24Jan80
0000 48 :
0000 49 : Modified to check for overflow of integers during a READ
0000 50 :           Susan Azibert 22May80
0000 51 :
0000 52 : Modified to check for overflow of real and double precision numbers
0000 53 :           during a READ
0000 54 :           Susan Azibert 22May80
0000 55 :
0000 56 : Modified to change the setting for PRN_CRLF to <LF> <text> <CR>
0000 57 :           Susan Azibert 16Oct80

```

```
0000 58 :  
0000 59 : Modified to force a READLN on read of a string if EOLN is true.  
0000 60 : Old behavior was to force a READLN only if both EOLN was true  
0000 61 : and last read was a string read. If the last read was not a string  
0000 62 : read, a null string (all blanks) was read and the file left  
0000 63 : positioned at EOLN.  
0000 64 : Paul Hohensee 19Jan81  
0000 65 :  
0000 66 : Correct PASSREADSCAL so it accepts capital Z.  
0000 67 : Paul Hohensee 20Feb81  
0000 68 :  
0000 69 : 7. Change PASSCNV_IN_DEFG to an integer-valued function  
0000 70 :  
0000 71 : 8. 11-Aug-81 Paul Hohensee Change to general addressing of external routines  
0000 72 :  
0000 73 : *****  
0000 74 : *****  
0000 75 : **  
0000 76 : **  
0000 77 : ** SECTION 2 **  
0000 78 : **  
0000 79 : ** INPUT PROCEDURES **  
0000 80 : **  
0000 81 : **  
0000 82 : *****  
0000 83 : *****
```

0000 85 : For any file variable the following storage is assumed:

0000 86	:		
0000 87	:		
0000 88	:	FSB:	-----
0000 89	:		POINTER
0000 90	:		-----
0000 91	:		STATUS WORD
0000 92	:		-----
0000 93	:		LAST
0000 94	:		-----
0000 95	:		LINELIMIT
0000 96	:		-----
0000 97	:		LINECOUNT
0000 98	:		-----
0000 99	:		RECORD NUMBER
0000 100	:		-----
0000 101	:	RAB:	44(HEX) BYTES
0000 102	:		.
0000 103	:		.
0000 104	:		.
0000 105	:		-----
0000 106	:	FAB:	50(HEX) BYTES
0000 107	:		.
0000 108	:		.
0000 109	:		.
0000 110	:		-----
0000 111	:		
0000 112	:	NAM.	38(HEX) BYTES
0000 113	:		.
0000 114	:		.
0000 115	:		.
0000 116	:		-----
0000 117	:		

NOTE: The NAM block is allocated for the PASCAL logical files 'INPUT' and 'OUTPUT' only.

0000 118 :  
0000 119 : MACRO OPTIONS  
0000 120 :  
0000 121 : .DSABL GBL ; No undefined references  
0000 122 : .ENABL FPT ; Rounded arithmetic  
0000 123 :

0000 124 : External references  
0000 125 :  
0000 126 : .EXTRN PASSREADOK  
0000 127 : .EXTRN PASSACTUALGET  
0000 128 : .EXTRN PASSSTATUSUPDAT  
0000 129 : .EXTRN PASSBLANK R3  
0000 130 : .EXTRN PASSIOERROR  
0000 131 :  
0000 132 : .EXTRN FOR\$CNV\_IN\_DEFG  
0000 133 : .EXTRN OTS\$CVT\_TI\_L  
0000 134 :

0000 135 : Provide definitions of system values  
0000 136 :  
0000 137 : \$DSCDEF ; String descriptor definitions  
0000 138 : \$FABDEF  
0000 139 : \$RABDEF  
0000 140 :

0000 141 : PASCAL compiler constants

```

0000 142 :
0000 143 : NOTE: The constants below with the names 'PASSC_XXXXX' are
0000 144 : used in the PASCAL compiler with the names 'XXXXX'. If the
0000 145 : values in the compiler are altered then the values below
0000 146 : must be altered accordingly.
0000 147 :
0000 148 : PASSC_DFLTRECSI = 257; ; default buffer size
0000 149 : PASSC_NIL = 0 ; NIL pointer
0000 150 : PASSC_TRUE = 1 ; TRUE
0000 151 : PASSC_FALSE = 0 ; FALSE
0000 152 : PASSC_NOCARR = 0 ; no carriage control
0000 153 : PASSC_CARRIAGE = 1 ; FORTRAN carriage control
0000 154 : PASSC_LIST = 2 ; LIST carriage control
0000 155 : PASSC_PRN = 3 ; PRN carriage control
0000 156 :
0000 157 : PRN carriage control constants
0000 158 :
0000 159 : PRN_CRLF = ^X8D01 ; PRN carriage control constant
0000 160 : ; for <LF> <text> <CR>
0000 161 : PRN_NULL = ^X0000 ; PRN carriage control constant
0000 162 : ; for no carriage control
0000 163 :
0000 164 : File status block constants
0000 165 :
00000018 0000 166 : FSB$C_BLN = ^X18 ; FSB block length
0000 167 : FSB$V_OPEN = 5
00000002 0000 168 : FSB$V_EOF = 1
0000 169 : FSB$V_EOLN = 2
0000 170 : FSB$V_GET = 3
0000 171 : FSB$V_TXT = 4 ; textfile flag
0000 172 : FSB$V_RDLN = 0 ; last access READLN
0000 173 : FSB$V_DIR = 6 ; direct access flag
0000 174 : FSB$V_PUT = 7
0000 175 : FSB$V_INT = 8 ; internal flag
0000 176 : FSB$V_PRMT = 9 ; prompt flag
0000 177 : FSB$V_OUTPUT = 10 ; OUTPUT file flag
00000008 0000 178 : FSB$V_ACTIN = 11 ; actual input flag
0000 179 : FSB$V_DELZ = 30 ; delete file if empty
0000 180 : FSB$V_INC = 31 ; included file flag
0000 181 : FSB$B_CC = 6 ; carriage control byte offset
0000 182 : FSB$M_OPEN = ^X0020
0000 183 : FSB$M_EOF = ^X0002
0000 184 : FSB$M_EOLN = ^X0004
0000 185 : FSB$M_GET = ^X0008
0000 186 : FSB$M_PRMT = ^X0200
0000 187 : FSB$M_PUT = ^X00000080
0000 188 : FSB$M_TXT = ^X0010
0000 189 : FSB$M_RDLN = ^X0001
0000 190 : FSB$M_DIR = ^X00000040
0000 191 : FSB$M_INT = ^X00000100
0000 192 : FSB$M_OUTPUT = ^X0400
0000 193 : FSB$M_ACTIN = ^X0800
0000 194 : FSB$M_DELZ = ^X40000000
0000 195 : FSB$M_INC = ^X80000000
0000 196 : FSB$S_CNT = 16 ; line count (textfiles)
0000 197 : FSB$S_INC = 20 ; %INCLUDE block address
0000 198 : FSB$S_LIM = 12 ; linelimit

```

```

00000008 0000 199      FSB$L_LST = 8           ; last word offset
          0000 200 :   FSB$L_PFSB = 20       ; related file FSB for prompting
          0000 201 :   ; for INPUT, has address of OUTPUT FSB
          0000 202 :   ; for OUTPUT, has address of INPUT FSB
          0000 203 :   ; (shares storage with include address
          0000 204 :   ; and direct access record
          0000 205 :   ; buffer address
          0000 206 :   FSB$L_REC = 20       ; record buffer address for
          0000 207 :   ; direct access (shares storage
          0000 208 :   ; with include address and related
          0000 209 :   ; file FSB)
00000004 0000 210      FSB$L_STA = 4       ; status word offset
          0000 211 :
          0000 212 : Character constants
          0000 213 :
          0000 214 :   TAB = ^X09
00000020 0000 215 :   SPACE = ^X20
00000024 0000 216 :   DOLLAR = ^X24
          0000 217 :   FORMFEED = ^XC
          0000 218 :   STAR = ^X2A
0000002B 0000 219 :   PLUS = ^X2B
0000002D 0000 220 :   MINUS = ^X2D
0000002E 0000 221 :   POINT = ^X2E
00000030 0000 222 :   ZERO = ^X30
          0000 223 :   ONE = ^X31
00000038 0000 224 :   EIGHT = ^X38
00000039 0000 225 :   NINE = ^X39
00000041 0000 226 :   AA = ^X41
00000044 0000 227 :   DD = ^X44
00000045 0000 228 :   EE = ^X45
0000005A 0000 229 :   ZZ = ^X5A
0000005F 0000 230 :   UNDERSCORE = ^X5F
00000061 0000 231 :   AA_SMALL = ^X61
0000007A 0000 232 :   ZZ_SMALL = ^X7A
          0000 233 :
0CCCCCCC 0000 234 :   MAX10 = 214748364
80000000 0000 235 :   MAXNEG = ^X80000000
          0000 236 :
          0000 237 : .PSECT _PASSCODE,          PIC,EXE,SHR,NOWRT
          0000 238 :
          0000 239 : *****
          0000 240 : *
          0000 241 : *   PASSGETBIN   *
          0000 242 : *
          0000 243 : *****
          0000 244 :
          0000 245 : Gets the next record from a (binary) file
          0000 246 :
          0000 247 : Argument offsets
          0000 248 :
          0000 249 :
          0000 250 :   AP           ; number of arguments (1)
00000004 0000 250 :   FSB_DISP = 04 ; FSB address
          0000 251 :
          0000 252 : .ENTRY PASSGETBIN,^M<R7>
          0002 253 : ADDL3 FSB_DISP(AP),#FSB$C_BLN,R7 ; R7 = address of RAB
          0007 254 : CALLG (APT,G^PASS$READOK
          000E 255 : MOVVB #RAB$C_SEQ,RAB$B_RAC(R7); make sure sequential

```

```

57 18 04 AC C1 0002
00000000 GF 6C FA 0007
1E A7 00 90 000E

```

```

00000000'GF 6C FA 0012 256
              04 0019 257
              001A 258
              001A 259
0000001A 260
              001A 261
              001A 262
              001A 263
              001A 264
              001A 265
              001A 266
              001A 267
              001A 268
              001A 269
              001A 270
              001A 271
              001A 272
              001A 273
00000004 001A 274
              001A 275
0040 001A 276
00000000'GF 6C FA 001C 277
           56 04 AC D0 0023 278
           66 D6 0027 279
00000000'GF 6C FA 0029 280
              04 0030 281
              0031 282
              0031 283
00000031 0031 284
              0031 285
              0031 286
              0031 287
              0031 288
              0031 289
              0031 290
              0031 291
              0031 292
              0031 293
              0031 294
              0031 295
              0031 296
              0031 297
00000004 0031 298
              0031 299
0040 0031 300
           56 04 AC D0 0033 301
00000000'GF 6C FA 0037 302
           66 08 A6 D0 003E 303
           66 D6 0042 304
00000000'GF 6C FA 0044 305
              04 004B 306
              004C 307
              004C 308
0000004C 004C 309
              004C 310
              004C 311
              004C 312

```

```

CALLG (AP),G^PASSACTUALGET ; get for call to GET
RET

.PSECT _PASSCODE, PIC,EXE,SHR,NOWRT
*****
* PASSGETTXT *
*****

Advances the file pointer and sets the status word as required
for textfiles.

Argument offsets
AP ; number of arguments (1)
FSB_DISP = 04 ; FSB address

.ENTRY PASSGETTXT,^M<R6>
CALLG (AP),G^PASSREADOK ; R6 = address of FSB
MOVL FSB_DISP(AP),R6
INCL (R6)
CALLG (AP),G^PASSSTATUSUPDAT ; update status word
RET

.PSECT _PASSCODE, PIC,EXE,SHR,NOWRT
*****
* PASSREADLN *
*****

Positions the pointer to the last character of the line, clears
the EOLN flag, and sets the RDLN flag.

Argument offsets
AP ; number of arguments (1)
FSB_DISP = 04 ; FSB address

.ENTRY PASSREADLN,^M<R6>
MOVL FSB_DISP(AP),R6 ; R6 = address of FSB
CALLG (AP),G^PASSREADOK
MOVL FSB$L_LST(R6),(R6) ; set pointer to LAST + 1
INCL (R6)
CALLG (AP),G^PASSSTATUSUPDAT
RET

.PSECT _PASSCODE, PIC,EXE,SHR,NOWRT
*****
*

```



```

004C 313 : * PASS$READCHAR *
004C 314 : * *
004C 315 : *****
004C 316 :
004C 317 : Argument offsets
004C 318 :
004C 319 : AP ; number of arguments (1)
00000004 004C 320 FSB_DISP = 04 ; FSB address
00000008 004C 321 VAR_DISP = 08 ; variable address
004C 322 :
0040 004C 323 .ENTRY PASS$READCHAR,^M<R6>
56 04 AC D0 004E 324 MOVL FSB_DISP(AP),R6 ; R6 = address of FSB
00000000'GF 56 DD 0052 325 PUSHL R6
01 FB 0054 326 CALLS #1,G^PASS$READOK
005B 327 :
005B 328 : Store the character and increment pointer
005B 329 :
08 BC 00 B6 90 005B 330 MOVB @(R6),@VAR_DISP(AP)
66 D6 0060 331 INCL (R6)
00000000'GF 56 DD 0062 332 PUSHL R6
01 FB 0064 333 CALLS #1,G^PASS$STATUSUPDAT
04 006B 334 RET
006C 335 :
006C 336 :
0000 006C 337 .PSECT _PASS$CODE, PIC,EXE,SHR,NOWRT
006C 338 :
006C 339 : *****
006C 340 : *
006C 341 : * PASS$READSTR *
006C 342 : *
006C 343 : *****
006C 344 :
006C 345 : Argument offsets
006C 346 :
006C 347 :
00000004 006C 348 AP ; number of arguments (3)
00000008 006C 349 FSB_DISP = 04 ; FSB address
0000000C 006C 350 STR_DISP = 08 ; string address
006C 351 LEN_DISP = 12 ; string length (by value)
007C 006C 352 .ENTRY PASS$READSTR,^M<R2,R3,R4,R5,R6>
56 04 AC D0 006E 353 MOVL FSB_DISP(AP),R6 ; R6 = address of FSB
09 04 A6 02 E1 0072 354 BBC #FSB$V_EOLN,FSB$L_STA(R6),100$; if EOLN = TRUE,
00000031'GF 56 DD 0077 355 PUSHL R6 ; go to next line
01 FB 0079 356 CALLS #1,G^PASS$READLN
0080 357 100$:
0080 358 PUSHL R6
00000000'GF 01 FB 0082 359 CALLS #1,G^PASS$READOK
50 08 A6 66 C3 0089 360 SUBL3 (R6),FSB$L_LST(R6),R0 ; R0 = remaining length
OC AC 20 00 B6 50 2C 008E 361 MOVCS R0,@(R6),#SPACE,LEN_DISP(AP),@STR_DISP(AP)
08 BC 0095 362 :
66 51 D0 0097 362 MOVL R1,(R6) ; store new pointer
56 DD 009A 363 PUSHL R6
00000000'GF 01 FB 009C 364 CALLS #1,G^PASS$STATUSUPDAT
04 00A3 365 RET
00A4 366 :
00A4 367 :
000000A4 368 .PSECT _PASS$CODE, PIC,EXE,SHR,NOWRT

```



```

00'00'00'00'00'00'00'00'00'00'00'00'00'00'00'00' 0124 420 .BYTE 0C^X7F]
00'00'00'00'00'00'00'00'00'00'00'00'00'00'00'00' 0130
00'00'00'00'00'00'00'00'00'00'00'00'00'00'00'00' 013C
00'00'00'00'00'00'00'00'00'00'00'00'00'00'00'00' 0148
00'00'00'00'00'00'00'00'00'00'00'00'00'00'00'00' 0154
00'00'00'00'00'00'00'00'00'00'00'00'00'00'00'00' 0160
00'00'00'00'00'00'00'00'00'00'00'00'00'00'00'00' 016C
00'00'00'00'00'00'00'00'00'00'00'00'00'00'00'00' 0178
00'00'00'00'00'00'00'00'00'00'00'00'00'00'00'00' 0184
00'00'00'00'00'00'00'00'00'00'00'00'00'00'00'00' 0190
00'00'00'00'00'00'00'00'00'00'00'00'00'00'00'00' 019C
01A3 421
01FC 01A3 422 .ENTRY PASS$READSCAL,^M<R2,R3,R4,R5,R6,R7,R8>
56 04 AC D0 01A5 423 MOVL FSB_DISP(AP),R6 ; R6 = address of FSB
52 56 D0 01A9 424 MOVL R6,R2 ; for PASS$BLANK_R3
00000000'GF 16 01AC 425 JSB G^PASS$BLANK_R3 ; skip leading blanks
; returns next address in R1
01B2 426
01B2 427
01B2 428 ; Check if first character is a letter
01B2 429
41 8F 61 91 01B2 430 CMPB (R1),#AA
7A 19 01B6 431 BLSS 900$ ; error
5A 8F 61 91 01B8 432 CMPB (R1),#ZZ
0C 15 01BC 433 BLEQ 110$ ; ok
7A 8F 61 91 01BE 434 CMPB (R1),#ZZ_SMALL
6E 14 01C2 435 BGTR 900$ ; error
61 8F 61 91 01C4 436 CMPB (R1),#AA_SMALL
68 19 01C8 437 BLSS 900$ ; error
01CA 438
01CA 439 ; Ok, lets read and translate the string
01CA 440
01CA 441 110$:
50 08 A6 51 C3 01CA 442 SUBL3 R1,FSB$L_LST(R6),R0
50 D6 01CF 443 INCL R0 ; R0 = # of characters left in line
5E 50 C2 01D1 444 SUBL2 R0,SP ; make room for translated string
; on stack
50 FEC9 CF 00 61 50 2F 01D4 445 MOVTUC R0,(R1),#0,SCALTRANSTABLE,R0,(SP)
6E 01DC 446
01DD 447 ; Update the FSB
01DD 448
01DD 449
66 51 D0 01DD 450 MOVL R1,(R6) ; update pointer
00000000'GF 56 DD 01E0 451 PUSHL R6
01  FB 01E2 452 CALLS #1,G^PASS$STATUSUPDAT
01E9 453
01E9 454 ; Try to find a match and store the value
01E9 455
55 5E C2 01E9 456 SUBL2 SP,R5 ; R5 = # of characters translated
1F 55 D1 01EC 457 CMPL R5,#MAXNAM ; compare only first maxnam bytes
03 15 01EF 458 BLEQ 115$
55 1F D0 01F1 459 MOVL #MAXNAM,R5
54 10 AC 20 C5 01F4 460 115$:
MULL3 #NAMELEN,MAX_DISP(AP),R4; R4 = table offset of current string
58 D4 01F9 461 CLRL R8 ; R8 will equal 1 if unique initial substring
01FB 463
6E 55 20 0C BC44 20 2D 01FB 464 120$:
CMPC5 #NAMELEN,@NAM_DISP(AP)[R4],#SPACE,R5,(SP)
15 13 0203 465 BEQL 199$ ; exit, found an exact match, search no fur

```

```

52 B5 0205 466 TSTW R2 ; did we match full input string?
05 12 0207 467 BNEQ 125$
58 D6 0209 468 INCL R8 ; R8 := 1 if first initial substring match
57 54 D0 020B 469 MOVL R4,R7 ; preserve offset
54 20 C2 020E 470 125$:
E8 18 0211 471 SUBL2 #NAMELEN,R4 ; R4 = offset of next string to try
58 D7 0213 472 BGEQ 120$
18 12 0215 473 DECL R8 ; no exact match, was there a unique initial
54 57 D0 0217 474 BNEQ 900$ ; NEQ: no, error
021A 475 MOVL R7,R4 ; yes, set up table offset
021A 476
021A 477 : Store value and exit
021A 478
021A 479 199$:
00000100 8F 54 20 C6 021A 480 DIVL2 #NAMELEN,R4 ; convert offset to index
10 AC D1 021D 481 CML MAX_DISP(AP),#256 ; store byte or word?
06 14 0225 482 BGTR 201$
08 BC 54 90 0227 483 MOV B R4,@SCA_DISP(AP) ; store byte
04 11 022B 484 BRB 202$
08 BC 54 B0 022D 485 201$:
MOVW R4,@SCA_DISP(AP) ; store word
04 0231 486 202$:
RET
04 0231 487
0232 488 : No match found, input conversion error
0232 489
0232 490
0232 491
0232 492 900$:
7E 8394 8F 3C 0232 493 MOVZWL #^X8394,-(SP)
7E 0090 C6 9A 0237 494 MOVZBL <FSB$C_BLN+RAB$C_BLN+FAB$B_FNS>(R6),-(SP)
0088 C6 DD 023C 495 PUSHL <FSB$C_BLN+RAB$C_BLN+FAB$L_FNA>(R6)
00000000'GF 03 FB 0240 496 CALLS #3,G^PASSIOERROR
0247 497
0247 498
0000 0247 499 .PSECT _PASSCODE, PIC,EXE,SHR,NOWRT
0247 500
0247 501 *****
0247 502 *
0247 503 * PASS$READINT *
0247 504 *
0247 505 *****
0247 506
0247 507 : Argument offsets
0247 508
0247 509
00000004 0247 510 AP ; number of arguments
00000008 0247 511 FSB_DISP = 04 ; FSB address
; Descriptor offsets
0247 512 VAR_DISP = 08 ; variable address
FFFFF4 0247 513 RESULT = -12 ; offset of result
FFFFF8 0247 514 LENGTH = -8 ; offset of length
FFFFFA 0247 515 CLASS = -6 ; offset of class and type
FFFFFC 0247 516 ADDR = -4 ; offset of address
0247 517
005C 0247 518 .ENTRY PASS$READINT,^M<R2,R3,R4,R6>
56 04 AC D0 0249 519 MOVL FSB_DISP(AP),R6 ; R6 = address of FSB
52 56 D0 024D 520 MOVL R6,R2 ; R2 needed for PASS$BLANK_R3
00000000'GF 16 0250 521 JSB G^PASS$BLANK_R3 ; skip leading blanks
0256 522 ; returns R1 as address of byte

```

B  
C  
C  
D  
E  
F  
G  
H  
I  
J  
K  
L  
M  
N  
O  
P  
Q  
R  
S  
T  
U  
V  
W  
X  
Y  
Z  
[  
\  
]  
^  
\_  
`  
a  
b  
c  
d  
e  
f  
g  
h  
i  
j  
k  
l  
m  
n  
o  
p  
q  
r  
s  
t  
u  
v  
w  
x  
y  
z  
{  
|  
}  
~  
?  
@  
#  
\$  
%  
&  
'  
(  
)  
\*  
,  
-  
.  
:  
;  
=

```

50 D4 0256 523 CLRL R0 ; set counter
52 D4 0258 524 CLRL R2 ; set sum register
54 D4 025A 525 CLRL R4 ; clear extract register
53 01 D0 025C 526 MOVL #1,R3 ; set sign flag
51 D0 025F 527 PUSHL R1 ; store address of string in descriptor
2B 61 91 0261 528 CMPB (R1),#PLUS ; plus?
06 12 0264 529 BNEQU 100$
50 D6 0266 530 INCL R0
51 D6 0268 531 INCL R1
10 11 026A 532 BRB 110$
026C 533 100$: ; minus?
2D 61 91 026C 534 CMPB (R1),#MINUS
0B 12 026F 535 BNEQU 110$
53 FFFFFFFF 8F D0 0271 536 MOVL #-1,R3 ; set sign flag
50 D6 0278 537 INCL R0
51 D6 027A 538 INCL R1
027C 539 110$: ; process integer
54 61 30 83 027C 540 SUBB3 #ZERO,(R1),R4 ; R4 = integer value of digit
30 19 0280 541 BLSS 120$
39 61 91 0282 542 CMPB (R1),#NINE
2B 14 0285 543 BGTR 120$ ; branch if not digit
OCCCCCCC 8F 52 D1 0287 544 CMPL R2,#MAX10 ; check for out of range
16 19 028E 545 BLSS 111$
4D 14 0290 546 BGTR 900$
38 61 91 0292 547 CMPB (R1),#EIGHT
48 14 0295 548 BGTR 900$
0D 19 0297 549 BLSS 111$
53 D5 0299 550 TSTL R3 ; check for largest negative
42 18 029B 551 BGEQ 900$
52 80000000 8F D0 029D 552 MOVL #MAXNEG,R2
06 11 02A4 553 BRB 112$
02A6 554 111$:
52 0A C4 02A6 555 MULL2 #10,R2
52 54 C0 02A9 556 ADDL2 R4,R2 ; R2 = new sum
02AC 557 112$:
50 D6 02AC 558 INCL R0 ; increment counter
51 D6 02AE 559 INCL R1 ; increment address
CA 11 02B0 560 BRB 110$ ; loop if more digits
02B2 561 120$: ; read until not digit
50 D5 02B2 562 TSTL R0 ; test for no digits read
29 13 02B4 563 BEQL 900$ ; conversion error
0A 50 D1 02B6 564 CMPL R0,#10 ; check for excess digits
24 14 02B9 565 BGTR 900$
50 DD 02BB 566 PUSHL R0 ; store length of descriptor
7E D4 02BD 567 CLRL -(SP) ; clear a longword for the result
5E DD 02BF 568 PUSHL SP ; pass the address to store the result
00000000'GF F8 AD 9F 02C1 569 PUSHAB LENGTH(FP) ; pass the address of the descriptor
11 02  FB 02C4 570 CALLS #2,G^OTS$CVT_TI_L ; call conversion routine
08 BC 8E D0 02CE 571 BLBC R0,900$ ; if error, output message
66 51 D0 02D2 572 MOVL (SP)+,@VAR_DISP(AP) ; store integer
56 DD 02D5 573 MOVL R1,(R6) ; restore pointer address
00000000'GF 01 FB 02D7 574 PUSHL R6
04 02DE 575 CALLS #1,G^PAS$STATUSUPDAT ; update status block
02DF 576 RET
02DF 577 ;
02DF 578 ; No match found, input conversion error
02DF 579 ;

```

```

7E 8394 8F 3C 02DF 580 900$:
7E 0090 C6 9A 02E4 581 MOVZWL #^X8394,-(SP)
00000000'GF 03 FB 02E9 582 MOVZBL <FSB$C_BLN+RAB$C_BLN+FAB$B_FNS>(R6),-(SP)
00000000'GF 03 FB 02ED 583 PUSHL <FSB$C_BLN+RAB$C_BLN+FAB$L_FNA>(R6)
02F4 584 CALLS #3,G^PASSIOERROR
02F4 585
02F4 586
000002F4 587 .PSECT _PASSCODE, PIC,EXE,SHR,NOVRT
02F4 588
02F4 589 *****
02F4 590 *
02F4 591 * PASSREADREAL *
02F4 592 *
02F4 593 *****
02F4 594
02F4 595 Argument offsets
02F4 596
02F4 597
00000004 02F4 598 AP ; number of arguments (2)
00000008 02F4 599 FSB_DISP = 04 ; FSB address
02F4 600 VAR_DISP = 08 ; variable address
02F4 601
001C 02F4 602 .ENTRY PASSREADREAL,^M<R2,R3,R4>
52 04 AC D0 02F6 603 MOVL FSB_DISP(AP),R2 ; R2 = address of FSB
00000000'GF 16 02FA 604 JSB G^PASS$BLANK_R3 ; skip leading blanks
50 51 D0 0300 605 MOVL R1,R0 ; returns located byte in R1
2B 61 91 0303 606 ; save starting address
04 12 0306 607 CMPB (R1),#PLUS ; check for plus
51 D6 0308 608 BNEQ 210$
07 11 030A 609 INCL R1
20 61 91 030C 610 BRB 220$
02 12 030F 611 210$: CMPB (R1),#MINUS ; check for minus if not plus
51 D6 0311 612 BNEQ 220$
0313 613 INCL R1
30 61 91 0313 614 220$: CMPB (R1),#ZERO ; count integer part
09 19 0316 615 BLSS 230$
39 61 91 0318 616 CMPB (R1),#NINE
04 14 031B 617 BGTR 230$
51 D6 031D 618 INCL R1
F2 11 031F 619 BRB 220$ ; loop
2E 61 91 0321 620 230$: CMPB (R1),#POINT ; count decimal point
10 12 0324 621 BNEQ 250$
51 D6 0326 622 INCL R1
30 61 91 0328 623 240$: CMPB (R1),#ZERO ; count decimal part
09 19 032B 624 RLSS 250$
39 61 91 032D 625 CMPB (R1),#NINE
04 14 0330 626 BGTR 250$
51 D6 0332 627 INCL R1
F2 11 0334 628 BRB 240$ ; loop
45 8F 61 91 0336 629 250$: CMPB (R1),#EE ; check for 'E'
06 13 033A 630 beql 251$
65 8F 61 91 033C 631 cmpb (r1),#^a'e'
632
633
634
635
636

```

```

2A 12 0340 637      BNEQ 280$      ; done if no exponent
      0342 638      251$: INCL R1      ; found exponent
51 D6 0342 639
      0344 640      ; check sign
2B 61 91 0344 641      CMPB (R1),#PLUS
      04 12 0347 642      BNEQ 260$
      51 D6 0349 643      INCL R1
      77 11 034B 644      BRB 270$
      034D 645      260$:      ; check minus if not plus
2D 61 91 034D 646      CMPB (R1),#MINUS
      02 12 0350 647      BNEQ 270$
      51 D6 0352 648      INCL R1
      0354 649      270$:      ; two digit exponent
30 61 91 0354 650      CMPB (R1),#ZERO
      13 19 0357 651      BLSS 280$
39 61 91 0359 652      CMPB (R1),#NINE
      0E 14 035C 653      BGTR 280$
      51 D6 035E 654      INCL R1
30 61 91 0360 655      CMPB (R1),#ZERO
      07 19 0363 656      BLSS 280$
39 61 91 0365 657      CMPB (R1),#NINE
      02 14 0368 658      BGTR 280$
      51 D6 036A 659      INCL R1
      036C 660      280$:      ; finished with number
53 51 50 C3 036C 661      SUBL3 R0,R1,R3      ; R3 = length
      04 BC 51 D0 0370 662      MOVL R1,@FSB_DISP(AP) ; update file pointer
      53 D5 0374 663      TSTL R3
      25 13 0376 664      BEQL 900$      ; branch if conversion error
      0378 665      ;
      0378 666      ; Make room for value on stack and convert input
      0378 667      ;
      5E 08 C2 0378 668      SUBL2 #8,SP
      54 5E D0 037B 669      MOVL SP,R4      ; R4 = address of double result
      53 DD 037E 670      PUSHL R3      ; length
      54 DD 0380 671      PUSHL R4      ; value address
      50 DD 0382 672      PUSHL R0      ; string address
00000473'GF 03 FB 0384 673      CALLS #3,G^PASS$CNV_IN_DEFG
      OF 50 E9 038B 674      BLBC R0,900$      ; branch if error
      08 BC 64 76 038E 675      CVTDF (R4),@VAR_DISP(AP) ; store read number
      04 AC DD 0392 676      PUSHL FSB_DISP(AP)
00000000'GF 01 FB 0395 677      CALLS #1,G^PASS$STATUSUPDAT ; update status block
      04 039C 678      RET
      039D 679      ;
      039D 680      ; Input conversion error
      039D 681      ;
      039D 682      900$:
7E 8394 8F 3C 039D 683      MOVZWL #^X8394,-(SP)
7E 0090 C2 9A 03A2 684      MOVZBL <FSB$C_BLN+RAB$C_BLN+FAB$B_FNS>(R2),-(SP)
      0088 C2 DD 03A7 685      PUSHL <FSB$C_BLN+RAB$C_BLN+FAB$L_FNA>(R2)
00000000'GF 03 FB 03AB 686      CALLS #3,G^PASS$IOERROR
      03B2 687      ;
      03B2 688      ;
0000 03B2 689      .PSECT _PASS$CODE,      PIC,EXE,SHR,NOWRT
      03B2 690      ;
      03B2 691      ; *****
      03B2 692      ; * PASS$READDOUB *
      03B2 693      ;

```

```

03B2 694 : *
03B2 695 : .....
03B2 696 :
03B2 697 : Argument offsets
03B2 698 :
03B2 699 : AP ; number of arguments (2)
00000004 03B2 700 FSB_DISP = 04 ; FSB address
00000008 03B2 701 VAR_DISP = 08 ; variable address
03B2 702 :
000C 03B2 703 .ENTRY PASS$READDOUB,^M<R2,R3>
52 04 AC 00 03B4 704 MOVL FSB_DISP(AP),R2 ; R2 = address of FSB
00000000 GF 16 03B8 705 JSB G^PASS$BLANK_R3 ; skip leading blanks
50 51 D0 03BE 706 ; R1 = located address
03BE 707 MOVL R1,R0 ; save starting address
03C1 708 :
03C1 709 : Check for plus
03C1 710 :
2B 61 91 03C1 711 CMPB (R1),#PLUS
04 12 03C4 712 BNEQ 210$
51 D6 03C6 713 INCL R1
07 11 03C8 714 BRB 220$
210$: ; check for minus if not plus
2D 61 91 03CA 715 CMPB (R1),#MINUS
02 12 03CD 716 BNEQ 220$
51 D6 03CF 717 INCL R1
220$: ; count integer part
30 61 91 03D1 719 CMPB (R1),#ZERO
09 19 03D4 721 BLSS 230$
39 61 91 03D6 722 CMPB (R1),#NINE
04 14 03D9 723 BGTR 230$
51 D6 03DB 724 INCL R1
F2 11 03DD 725 BRB 220$
230$: ; loop
2E 61 91 03DF 726 CMPB (R1),#POINT
10 12 03E2 727 BNEQ 250$
51 D6 03E4 728 INCL R1
240$: ; count decimal part
30 61 91 03E6 730 CMPB (R1),#ZERO
09 19 03E9 732 BLSS 250$
39 61 91 03EB 733 CMPB (R1),#NINE
04 14 03EE 734 BGTR 250$
51 D6 03F0 735 INCL R1
F2 11 03F2 736 BRB 240$
250$: ; loop
44 8F 61 91 03F4 737 CMPB (R1),#DD ; check for 'D' or 'E'
12 13 03F8 738 BEQL 251$
64 8F 61 91 03FA 739 cmpb (r1),#^a'd'
0C 13 03FE 741 beql 251$
45 8F 61 91 0400 742 CMPB (R1),#EE
06 13 0404 743 beql 251$
65 8F 61 91 0406 744 cmpb (r1),#^a'e'
2A 12 040A 745 BNEQ 280$
; done if no exponent
; found exponent
51 D6 040C 746 INCL R1
2B 61 91 040E 747 ; check sign
04 12 0411 750 CMPB (R1),#PLUS
BNEQ 260$

```



```

51 D6 0413 751 INCL R1
07 11 0415 752 BRB 270$
2D 61 01 0417 753 260$: ; check minus if not plus
02 12 041A 754 CMPB (R1),#MINUS
51 D5 041C 755 BNEQ 270$
01 041E 756 INCL R1
30 61 91 041E 757 270$: ; two digit exponent
13 19 0421 758 CMPB (R1),#ZERO
39 61 91 0423 759 BLSS 280$
0E 14 0426 760 CMPB (R1),#NINE
51 D6 0428 761 BGTR 280$
30 61 91 042A 762 INCL R1
07 19 042D 763 CMPB (R1),#ZERO
39 61 91 042F 764 BLSS 280$
02 14 0432 765 CMPB (R1),#NINE
51 D6 0434 766 BGTR 280$
53 51 50 C3 0436 767 INCL R1
04 BC 51 D0 043A 768 280$: ; finished with number
53 D5 043E 769 SUBL3 R0,R1,R3 ; R3 = length
1C 13 0440 770 MOVL R1,@FSB_DISP(AP) ; update file pointer
0442 771 TSTL R3
0442 772 BEQL 900$ ; branch if conversion error
0442 773 :
0442 774 : Convert input
0442 775 :
53 DD 0442 776 PUSHL R3 ; length
08 AC DD 0444 777 PUSHL VAR_DISP(AP) ; variable address
00000473'GF 50 DD 0447 778 PUSHL R0 ; string address
08 03 FB 0449 779 CALLS #3,G^PASS$CNV_IN_DEFG
04 50 E9 0450 780 BLBC R0,900$ ; branch if error
00000000'GF 04 AC DD 0453 781 PUSHL FSB_DISP(AP)
01 01 FB 0456 782 CALLS #1,G^PASS$STATUSUPDAT ; update status block
04 04 045D 783 RET
045E 784 :
045E 785 : Input conversion error
045E 786 :
045E 787 :
7E 8394 8F 3C 045E 788 900$: MOVZWL #^X8394,-(SP)
7E 0090 C2 9A 0463 789 MOVZBL <FSB$C_BLN+RAB$C_BLN+FAB$B_FNS>(R2),-(SP)
0088 C2 DD 0468 790 PUSHL <FSB$C_BLN+RAB$C_BLN+FAB$L_FNA>(R2)
00000000'GF 03 FB 046C 791 CALLS #3,G^PASS$IOERROR
0473 792 :
0473 793 :
0000 0473 794 .PSECT _PASS$CODE, PIC,EXE,SHR,NOWRT
0473 795 :
0473 796 : *****
0473 797 : *
0473 798 : * PASS$CNV_IN_DEFG *
0473 799 : *
0473 800 : *****
0473 801 :
0473 802 : Converts a character string representing a real or double precision
0473 803 : value into a double precision value
0473 804 :
0473 805 : Argument offsets
0473 806 :
0473 8J7 : AP ; number of arguments (3)

```

```
00000004 0473 808      BUF_DISP = 04      ; buffer address
00000008 0473 809      VAR_DISP = 08      ; variable address (of quadword)
0000000C 0473 810      LEN_DISP = 12     ; string length (by value)
           0473 811      :
0000      0473 812      : .ENTRY PASSCNV_IN_DEFG,^M<>
           0475 813      :
           0475 814      : Make room for descriptor on stack
           0475 815      :
           0475 816      :
04 A1 5E 08 C2 0475 816      SUBL2 #DSC$C_S_BLN,SP
           51 SE DO 0478 817      MOVL SP,R1      ; R1 = address of descriptor
04 A1 04 AC DO 047B 818      MOVL BUF_DISP(AP),DSC$A_POINTER(R1); string address
           61 0C AC BO 0480 819      MOVW LEN_DISP(AP),DSC$W_LENGTH(R1); string length
           0484 820      :
           0484 821      : Convert the value
           0484 822      :
           00 DD 0484 823      PUSHL #0      ; zero digits in fraction
           08 AC DD 0486 824      PUSHL VAR_DISP(AP) ; variable address
00000000'GF 51 DD 0489 825      PUSHL R1      ; descriptor address
           03 FB 048B 826      CALLS #3,G^FOR$CNV_IN_DEFG
           04 0492 827      RET
           0493 828      :
           0493 829      :
           0493 830      :
           0493 831      : .END
```

AA	=	00000041		
AA_SMALL	=	00000061		
ADDR	=	FFFFFFFFC		
BUF_DISP	=	00000004		
CLASS	=	FFFFFFFA		
DD	=	00000044		
DOLLAR	=	00000024		
DSCBA_POINTER	=	00000004		
DSCBC_S_BLN	=	00000008		
DSCBW_LENGTH	=	00C00000		
EE	=	00000045		
EIGHT	=	00000038		
FABS_B_FNS	=	00000034		
FABS_L_FNA	=	0000002C		
FORSCNV_IN_DEFG	*****		X	00
FSBSC_BLN	=	00000018		
FSBSL_LST	=	00000008		
FSBSL_STA	=	00000004		
FSBSV_ACTIN	=	0000000B		
FSBSV_EOLN	=	00000002		
FSB_DISP	=	00000004		
LENGTH	=	FFFFFFFFP		
LEN_DISP	=	0000000L		
MAXTO	=	0CCCCCCC		
MAXNAM	=	0000001F		
MAXNEG	=	80000000		
MAX_DISP	=	00000010		
MINOS	=	0000002D		
NAMELEN	=	00000020		
NAM_DISP	=	0000000C		
NINE	=	00000039		
OTSCVT_T1_L	*****		X	00
PASSACTOALGET	*****		X	00
PASSBLANK_R3	*****		X	00
PASSCNV_IN_DEFG	00000473	RG		02
PASSGETBIN	00000000	RG		02
PASSGETTXT	0000001A	RG		02
PASSIOERROR	*****		X	00
PASSREADCHAR	0000004C	RG		02
PASSREADDOUB	000003B2	RG		02
PASSREADINT	00000247	RG		02
PASSREADLN	00000031	RG		02
PASSREADOK	*****		X	00
PASSREADREAL	000002F4	RG		02
PASSREADSCAL	000001A3	RG		02
PASSREADSTR	0000006C	RG		02
PASSSTATUSUPDAT	*****		X	00
PLUS	=	0000002B		
POINT	=	0000002E		
RABS_B_RAC	=	0000001E		
RABS_C_BLN	=	00000044		
RABS_C_SEQ	=	00000000		
RESULT	=	FFFFFFF4		
SCALTRANSTABLE	000000A4	R		02
SCA_DISP	=	00000008		
SPACE	=	00000020		
STR_DISP	=	00000008		

UNDERSCORE	=	0000005F
VAR_DISP	=	00000008
ZERO	=	00000030
ZZ	=	0000005A
ZZ_SMALL	=	0000007A

-----  
! 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	00000000 ( 0.)	01 ( 1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
_PASSCODE	00000493 ( 1171.)	02 ( 2.)	PIC USR CON REL LCL SHR EXE RD NOWRT NOVEC BYTE

-----  
! Performance indicators .  
-----

Phase	Page faults	CPU Time	Elapsed Time
Initialization	32	00:00:00.08	00:00:00.64
Command processing	106	00:00:00.47	00:00:02.21
Pass 1	206	00:00:05.52	00:00:11.91
Symbol table sort	0	00:00:00.60	00:00:00.61
Pass 2	153	00:00:02.10	00:00:04.85
Symbol table output	8	00:00:00.07	00:00:00.09
Psect synopsis output	2	00:00:00.03	00:00:00.03
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	510	00:00:08.90	00:00:20.36

The working set limit was 1200 pages.  
34227 bytes (67 pages) of virtual memory were used to buffer the intermediate code.  
There were 30 pages of symbol table space allocated to hold 484 non-local and 35 local symbols.  
831 source lines were read in Pass 1, producing 40 object records in Pass 2.  
10 pages of virtual memory were used to define 9 macros.

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

Macro library name	Macros defined
_\$255\$DUA28:[SYSLIB]STARLET.MLB;2	6

497 GETS were required to define 6 macros.

There were no errors, warnings or information messages.

MACRO/DISABLE=TRACE/LIS=LIS\$:PASIO2/OBJ=OBJ\$:PASIO2 MSRC\$:PASIO2/UPDATE=(ENH\$:PASIO2)

The image displays a grid of 100 terminal window screenshots, arranged in 10 rows and 10 columns. Each window shows a different system utility or data list, with the following titles visible:

- Row 1: PASDEF LIS, PAS102 LIS
- Row 2: STATUS LIS, UNKNOWN LIS
- Row 3: SHUTDOWN LIS
- Row 4: SHARE LIS, PAS101 LIS
- Row 5: PAS103 LIS
- Row 6: PASCAL
- Row 7: PASLINE LIS
- Row 8: TIMESTAMP LIS

Each screenshot contains text-based data, including headers, lists of entries, and status information, typical of a VAX/VMS environment.